Modern Initial Access and Evasion Tactics
Red Teamer’s Delight

Mariusz Banach
Red Team Operator at ING Tech Poland

@mariuszbit, github/mgeeky
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

Security Certs holder
  • 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
  » Command & Control
  » Exfiltration
Disclaimer

» Initial Access & Evasion tactics 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 :<

» This talk shares my insights based on engagements delivered with following security stacks:
  » MDE, MS Defender For Endpoint + ATP
  » MDO, MS Defender For Office365
  » MDI, MS Defender For Identity
  » McAfee AV
  » CrowdStrike Falcon EDR
  » Palo Alto Proxy
  » BlueCoat Proxy
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

**Bonkers tricks:**

» Ride-to-Left-Override-Like-Its-90s

> “This E-mail was scanned. [...] No Spam detected. Links are safe to open.”
» Get familiar with state-of-the-art Detections

• Here we reverse-engineer 20+ MS Defender for Office365 Anti-Spam rules

Anti-Spam Rules ReverseEngineered

```python
'3510500006' : logger.colored('(SPAM) Message contained embedded image. ', 'red'),

'52087058' : 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"
'16253004' : '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://metz-sche-ipsom.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.
'564544004' : '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 </> tag in body.
'57855001' : 'HTML mail body contained underline </> tag.',

# message with html,head,body and body containing simple text with no b/i/u formatting.
'579154003' : 'HTML mail body contained text, but no text formatting (<b>, <i>, <u>) was present',

# This is a strong signal. Mails without </a> doesn't have this rule.
'166802' : 'HTML mail body contained URL </a> link.',

# Message contained <a href="https://something.com/file.html?parameter=value" - GET parameter with va
'21651005' : 'Mail body contained </a> tag with URL containing GET parameter: ex. href="https://foo.bar

# Message contained <a href="https://something.com/file.html?parameter=another.com/website" - GET parameter with value, being a URL to another website
'45080440002' : '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

  » Number of GET params, their names & values DO MATTER
Phishing

» Apply Phishing e-mail HTML Linting

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

(1) Test: Embedded Images
DESCRIPTION:
Embedded images can increase Spam Confidence Level (SCL) in Office365 by 4 points. Embedded images are those with <img src="data:image/png;base64,\" />. They should be avoided.

CONTEXT:
<img src="data:image/png;base64,ivB0Rw6wGGoAAAANSU...AAAEFTKzSjQmC" style="width:121px; height:32px"/>

ANALYSIS:
- Found 1 <img> tags with embedded image (data:image/png;base64,ivB0Rw6wK). Embedded images increase Office365 SCL (Spam) level by 4 points!

(2) Test: Images without ALT

(6) Test: <a href="..."> URL pointed to an executable file
Message contained <a> tags with href="..." links pointing to a file with dangerous extension (such as .exe)

CONTEXT:
<a href="https://report.z13.web.core.windows.n...r:#f2f2f2">Gelöschte Dateien überprüfen</a>/</span></a>
Phishing

» Email Sending Strategy - MS Defender for Office365 cools down a sender upon 4-5th mail

» Throttling is absofreakinglutely crucial

» What works nice for MDO:

  » GoPhish -> EC2 587/tcp Socat Redirector -> Gsuite -> Target
INITIAL ACCESS
Initial Access

Phish to Persist

» instead of Phish to Access (Matt Hand @SpecterOps)
» Strive for delayed & elonged 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, VBS - VBScript
  » JSE, JS - JScript
  » HTA - HTML Application
  » XSL - XML
  » WSF - Windows Script File
    » Language agnostic file format
    » Allows multiple scripts ("jobs") and combination of languages within a single file

» Mostly very-well detected
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

CrowdStrike Falcon combines some of the most advanced features with a very intuitive user interface. The latter provides a self and the machine’s state during an attack through process tracking.
Clever use of shortcut files

Still a popular threat, especially in Phishing campaigns

**lnk - Link**

Often detected
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”
Initial Access »

Typical Vectors - COM Scriptlets

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

» Used to instantiate COM objects
  » via Regsvr32
  » via GetObject

» Can be detected
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
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
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

Project_Open()?

Not detected
Rise of Containerized Malware

» Malware-in-Archive

» Malware-in-Document

» Can effectively smuggle back-in Blocked File Formats
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.

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.
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

<table>
<thead>
<tr>
<th>Format</th>
<th>MOTW?</th>
<th>Off the shelf Windows support?</th>
<th>Elevation required?</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZIP</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>MOTW stripped only on manual files extraction</td>
</tr>
<tr>
<td>7zip*</td>
<td>Partially</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ISO</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IMG</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PLF</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Depends on Javascript support in PDF reader</td>
</tr>
<tr>
<td>CAB</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Requires few additional clicks on victim-side</td>
</tr>
<tr>
<td>VHD</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>This script currently can't make directories</td>
</tr>
<tr>
<td>VHDX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>This script currently can't make directories</td>
</tr>
</tbody>
</table>
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

```javascript
this.exportDataObject({ cName: "Foo.xls", nLaunch: 2 })
```
The Beauty of HTML Smuggling

Web filter inspects content
- File extension?
  - .html
- Mime type?
  - text/html
- Malicious content?
  - No, only JavaScript

Browser processes HTML/JS
- JS decodes payload
- JS creates blob locally
- JS clicks anchor
- Browser saves payload to disk
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

HTML5 download attribute

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

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

```javascript
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

» Below list of extensions that pose actual risk:

<table>
<thead>
<tr>
<th>Word</th>
<th>Publisher</th>
<th>MS Project</th>
<th>WSH, COM, HTML</th>
<th>Executables</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. docm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. doc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. docx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. dot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. dotm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. rtf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. xls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. xlsx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. xlam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. xlsm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. xla</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. xlt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13. xltm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14. slk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. chm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16. scf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17. url</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18. csproj</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Exotics</td>
</tr>
<tr>
<td>19. pub</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20. ppa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21. ppam</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22. pptm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23. ppdm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24. pot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25. potm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26. pps</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27. pptx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28. vdw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29. vsd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30. vsdm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31. vss</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32. vssm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33. vst</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34. library-ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35. settingscontent-ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36. mpd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37. mpp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38. mpt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39. mpw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40. mpx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41. vbs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42. vbe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43. hta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44. sct</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45. vsf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46. wsc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47. xsl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48. xml</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49. xbe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50. js</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51. jse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52. html</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53. zip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54. 7z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55. iso</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56. img</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57. cab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58. pdf</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59. vhd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60. vhdx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61. exe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62. scr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63. cpl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64. xll</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65. bat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66. ps1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67. cmd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68. sh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69. lnk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70. chm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Evasion In-Depth

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquire Infrastructure</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Compromise Account</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Compromise Infrastructure</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Develop Capabilities</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Establish Accounts</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Obtain Capabilities</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Stage Capabilities</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Supply Chain</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Compromise</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Shared Relationship</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Valid Accounts</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Intrusion Compromise</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Exploit Public-Facing Application</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>External Remote Services</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Access Integrity</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Container Administration</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Command and Scripting Interpreter</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Bypass Jobging</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Boot or Login</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Authentication</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Malware Implementation</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Malware Execution</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Intrusion Integrity</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Malware Implementation</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Malware Execution</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
<tr>
<td>Intrusion Integrity</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
<td>(9)</td>
</tr>
</tbody>
</table>
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
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)
Evasion In-Depth »

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
**Evasion In-Depth**

**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
**Evasion**

**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
  - ShellBrowserWindow COM (spawns targets as explorer.exe descendants)
  - 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
Evasion In-Depth »

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

» PoshC2 – may work just fine

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

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.
Evasion In-Depth

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
Evasion In-Depth

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!

---

- 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 Teams

- Inject into:
  - DOS Stub
  - Additional PE Section
  - Manifest
  - Version Info
  - PE Checksum, Timestamp
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

Pssst. ScareCrow does Signaturing very well!
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 elonged 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 computer name,
- From a disk serial number.
Evasion In-Depth »

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
Evasion In-Depth »

**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

```
beacon> bofnet_init
beacon> bofnet_load seatbelt
beacon> bofnet_executeassembly seatbelt osinfo
```

```
[*] Attempting to execute BOFNET Bofs.Jobs.JobList
[+] [05/17 14:35:23] host called home, sent: 8120 bytes
[+] received output:
```
```csharp
- [ 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:  
+ [  6] Type: ExecuteAssembly, Active: True, Output: False ( 0 bytes), Args:  
```

```
[.] Machine connected to Azure AD:
  Tenant ID : 
  Tenant Name : 
  Device Name : 
  OS Version : 10.0.19042.867
  User Email : 
[.] Primary Refresh Token extraction:
  Nonce : AwAABZAAACGz.BAD0_y6Rhs7uIDmQfCBe6aCwP4iaUVMe70dKJ5p8gAeyJnbGcI0iJIUc1ILIsICJrZGZwVmdyIjoyM
  Target : https://login.microsoftonline.com/login.srf
  Cookie : x-mms-RefreshTokenCredential
  PRT : 
```
Evasion In-Depth

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
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
Evasion In-Depth »

Delivery - Bullet Points

» My personal Bonnie & Clyde:
  » 2022, still HTML Smuggling + Macro-Enabled Office document = MacOS – VBA to JXA -> but then heavily sandboxed

» Secret Sauce lies in VBA poetry

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

» Smuggling must include self-defence logic

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

» VBA Purging lowers detection potential

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

» Among different VBA Strategies – File Droppers, DotNetToJS, XSL TransformNode are killing machines
Initial Access - Bullet Points

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

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

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

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

» **Develop Custom Shellcode Loader**
  » If you ask me - I’m a purist - C/C++ is the optimal language choice.
    » Rust/Go/C# add their own specific nuances, I don’t buy them for MalDev
    » Nim looks promising though
  
  » Embed shellcodes in Proxy DLL loaders
  » Utilize DLL Side-Loading as your execution entry point (Teams’ version.dll is convenient)
  » Direct Syscalls or intelligent Unhooking, AMSI + ETW evasion, delayed execution are MUST HAVE
  » Remote-Process Injection is a tough one to get it right, prefer operating Inline/Inprocess

» **Malware Development CI/CD Pipeline**
  » Develop -> pass through daisy-chained obfuscations -> Backdoor legitimate PE -> Watermark -> Sign It.
C2 – Bullet Points

» Egress Through HTTPS – Highly Trafficked Servers Only
  » Serverless Redirectors,
  » Domain Fronting via CDN,
  » Legitimate services – Github, Slack, MS Teams, Asana

» Forget DNS, ICMP, IRC
  » We’re no longer in mid-90s – robust NIPS/NIDS and ML-based signaturing outrules exotic protocols

» Offensive Deep Packet Inspection
  » Closely examine Inbound requests and decide if they originate from your Implants/Infra
  » If not, kill them at spot – TCP RESET/Redirect/404
  » RedWarden-style:
    » Rev-PTR inspection
    » WHOIS, IP Geo
    » HTTP Headers
    » Alignment to expected Malleable contract
Q & A

Questions? 😊

@mariuszbit / mb@binary-offensive.com

https://mgeeky.tech

https://github.com/mgeeky