CLICKONCE AND YOU’RE IN:
When .appref-ms abuse is operating as intended
ClickOnce and you’re in

- Speaker Introduction
- The End of the Golden Age
- Compendium of ClickOnce
- Aquaman, King of the Phish
- Sleeper Cells: C2 Management
- A Little Help for my Friends
- Closing Questions
SPEAKER INTRODUCTION
Speaker Introduction – @0xF4B0

- William Joseph Burke IV
- CISA Red Team Lead
  - Provide red team ops for whole federal sector
- 15 Years across intelligence & cyber fields
  - Military, Private, and Public sectors
- Linguistics, NetAdmin, SysAdmin, Operations
- Adjunct Graduate Professor, Marymount University
- OSCP, GXPN, GPEN, GCIH, GWAPT, eCPPT, CORIII, CNDA, CEH, Sec+, CISSP
THE END OF THE GOLDEN AGE
The End of the Golden Age

- Initial access via phishing used to be simple
  - Batch script Object Linking & Embedding (OLE) in Word documents
  - Basic scripts delivered via HTTP Application (HTA)
  - ...Pretty much everything Windows 7

- As time ticked through the hourglass, difficulty increased
  - Windows Defender / Antimalware Scan Interface (AMSI)
  - Permitted filetypes for execution via OLE’s are increasingly limited
  - Additional layers of barriers preventing entry
  - ...Pretty much everything Windows 10
The End of the Golden Age

- The resulting operational need:
  - Additional methods of code execution
  - Delivery as either an attachment, hyperlink, or OLE

- Capabilities needed to work in the following environments:
  - Native, fully patched Windows 10 with Defender enabled
  - Native, fully patched Windows 7 with third party anti-virus enabled
  - Cooperate with the Cobalt Strike Command & Control (C2) platform

- Focus of research was on delivery and execution of code
So, where to begin? Let’s take a journey…

OLE delivery was the first item of interest
- Inspired by .SettingContent-ms research by Matt Nelson (@enigma0x3)
- Cross-referenced native Win 10 executable formats to the OLE blacklist
- Research available executable formats for delivery potential
- This resulted in a preliminary list which was narrowed down to:

  .appref-ms  .appx  .cat  .webpnp  .wcx

- These filetypes were individually researched for potential manipulation
The End of the Golden Age

<table>
<thead>
<tr>
<th>File name extension</th>
<th>File type</th>
</tr>
</thead>
<tbody>
<tr>
<td>.ade</td>
<td>Access Project Extension (Microsoft)</td>
</tr>
<tr>
<td>.adp</td>
<td>Access Project (Microsoft)</td>
</tr>
<tr>
<td>.app</td>
<td>Executable Application</td>
</tr>
<tr>
<td>.appcontent-ms</td>
<td>Application Content</td>
</tr>
<tr>
<td>.application</td>
<td>Application Manifest</td>
</tr>
<tr>
<td>.asp</td>
<td>Active Server Page</td>
</tr>
<tr>
<td>.bas</td>
<td>BASIC Source Code</td>
</tr>
<tr>
<td>.bat</td>
<td>Batch Processing script</td>
</tr>
<tr>
<td>.cer</td>
<td>Internet Security Certificate File</td>
</tr>
</tbody>
</table>

Choose default apps by file type:

- .appcontent-ms: Application Content
- .appinstaller: APPINSTALLER File
- .application: Application Manifest
- .appref-ms: Application Reference
- .appx: APPX File
- .appxbundle: APPXBUNDLE File
- .aps: APS File
- .arc: ARC File
- .ari: ARI File
- .arj: WinRAR archive

Choose a default:
The End of the Golden Age

- So, where to begin? Let’s take a journey…

- OLE delivery was the first item of interest
  - Inspired by .SettingContent-ms research by Matt Nelson (@enigma0x3)
  - Cross-referenced native Win 10 executable formats to the OLE blacklist
  - Research available executable formats for delivery potential
  - This resulted in a preliminary list which was narrowed down to:

    | .appref-ms | .appx | .cat | .webpnp | .wcx |

- These filetypes were further researched for potential manipulation
While researching the extension formats, this caught my eye. You had my curiosity, but now you have my attention.
A New Light Arises

- No prior research on .appref-ms abuse was discovered
- ClickOnce is the application that runs the .appref-ms filetype
- Now, some great prior research on ClickOnce was discovered
  - Ryan Gandrud, NETSPI 2015
  - Justin Warner (@sixdub), 2015
  - @Bohops, 2018
- However, their research focused on a very different aspect
- Ultimately obtained execution while “operating as intended”
COMPRENDIUM OF CLICKONCE
Intended use of ClickOnce follows this type of path:

- An application is developed in C# within Visual Studio
- The application is published either to a share or remote server

Published as “Online only” or “Online or Offline” access

(This becomes important later)
A ClickOnce Summary

- The root folder of the published directory could contain:
  - `publish.htm` - landing page for the application
  - `.application` file – Initiates web installation for the application
  - `setup.exe`, raw installer for the program
  - ”Application Files” folder – Stores the app version, manifest, & deploy files
A ClickOnce Summary

- It’s incredibly simple for the end user to install the program:
  - Launch the .application link through a web browser, execution is automated
  - Launch or install the application from the publish.htm page and run it

These methods will always install the latest version of the app
A ClickOnce Summary

- Installation differs between "Online only" and "Online & Offline"
  - Established when the application is deployed

- Online Only: Drops files to temp directory and runs
- Online or Offline: “Installs” the program and runs

[Image: Application Run - Security Warning]

[Image: Application Install - Security Warning]
And that’s it! Simple way for a dev to get their app deployed

(So simple Google uses ClickOnce to install chrome via IE)
But what about .appref-ms?

- In both “Online only” and “Online or Offline” availability:
  - Files are dropped to the following directory -
    C:\Users\<username>\AppData\Local\Apps\2.0\<Random String>

- In Online Only deployment the application is ran a single time

- In ”Online or Offline” availability some additional work is done
  - Two major actions are performed as the “installation”
  - A registry key is added under
    HKCU\Software\Microsoft\Windows\CurrentVersion\Uninstall
  - An application reference file (!) is installed under the user’s start menu
But what about .apprefs-ms?

- The application reference (.appref-ms) file runs the application

- If the .appref-ms file is executed, it will:
  - Check the deployment site to see if there is an update
  - Will download any required or missing files and run the application
  - If the developer mandates the latest version, it will force an update
  - So if we send an .appref-ms file in an e-mail…
AQUAMAN, KING OF THE PHISH
“Application” Deployment

- **Pre-Deployment requirements**
  - C# code that bypasses your defensive mechanism of choice upon execution
  - Code signing certificate (if deploying externally)
  - A method to clean up files / stop the IOC’s below:

- **Methods for removing the following should be in your C# code:**
  - Reg Key:
    
    \`HKCU\Software\Microsoft\Windows\CurrentVersion\Uninstall\<key>\`
  - Directory & Files:

    \`C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\<Application name>\<Application Files>\`

- **Example code is provided in the white paper**
“Application” Deployment

- With the code in place, it’s time to deploy
- Publishing options sets your deployment configurations
  - For .appref-ms use, “available offline as well” must be selected
  - Any version number can be arbitrarily set
The options section opens up some additional configurations:
- Here you can mandate when and how the application checks for updates.
- You can also specify a minimum required version.
- If the current version does not match the required minimum, it will force install.
Two ways to generate your own .appref-ms file:

- Test on a host you own and copy the .appref-ms from the startup folder
- A better option would be to create your own!
  - Saves time on continuous testing
  - (Especially if using self-cleaning deployment)

An .appref-ms file consists of the following in a single line

- URL_to_App#<name>, Culture, Public Key Token, Processor Architecture
- This information is in the “Assembly Identity” section of the .application file

```
name="PatchMgr.application" version="1.0.0.0" publicKeyToken="6ff5ee14e8b3a058" language="neutral" processorArchitecture="msil"
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Token</th>
<th>Culture</th>
<th>Arch</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
“Application” Deployment

- You can put the information into a text file as a single line
- The file **must** be saved with UTF-16 LE encoding
  - Shout out to Alex Feinberg’s blog post from 2014
- Save as an .appref-ms file and it is ready to go!

- At this point we are ready to move to delivery
You’re .appref-ms’ing with me

- Lazy mode: It could be attached directly to an e-mail
  - the .appref-ms file isn’t flagged as malicious once attached or on download
You’re .appref-ms’ing with me

Totally innocuous OLE
You’re .appref-ms’ing with me

1) Word document delivered

2) .appref-ms opened
You’re .appref-ms’ing with me

3) Once install is clicked…

4) Gondor calls for aid! We’ve got beacons

<table>
<thead>
<tr>
<th>external</th>
<th>internal</th>
<th>user</th>
<th>computer</th>
<th>note</th>
<th>pid</th>
<th>last</th>
</tr>
</thead>
<tbody>
<tr>
<td>173.54.167.128</td>
<td>172.16.202.183</td>
<td>devtest</td>
<td>DESKTOP-9RCLIP6</td>
<td>DevTest2_Host</td>
<td>2384</td>
<td>9s</td>
</tr>
</tbody>
</table>
You’re .appref-ms’ing with me
Lateral Movement Rundown

- Lateral movement could be obtained by combining .appref-ms deployment with other capabilities

- Example 1: If you can move files remotely you could push the .appref-ms file to the remote user’s startup folder

- Example 2: Run the application with psexec
  - Call the .application deployment link by invoking dfshim.dll with rundll32
  - Not appref-ms specific - can also be used in online only deployments

- The user will still need to approve the initial installation
  - Social engineering still at play – name your application accordingly
  - Once installed the user will no longer be prompted for execution or updates
SLEEPER CELLS: C2 MGMT
C2 Management

= C2 Communication

= Periodic .appref-ms execution
C2 Management

= C2 Denied

= Periodic .appref-ms execution

(Your average operator)
C2 Management

= C2 has ceased to be, it is no more

= Periodic .appref-ms execution

= Update pushed to server

(Your average operator)
C2 Management

= Update pulled from server on next Ex

(Your average operator)
C2 Management

= C2 communication reestablished!

(Your average operator)
C2 Management

- Operational requirements may necessitate “lifelines”
  - Compromised hosts in an environment that can be utilized as backdoors

- By using ClickOnce’s update management capability, you can:
  - Have non-malicious code running on a remote host
  - Use an .appref-ms file to run on a schedule, startup, etc.
  - When it runs it will check for an update
  - If you lose access to your environment - push a malicious update!
  - The next time it checks in, if updates are forced it will run your malicious code
  - Can also be used to create logic bombs – Maybe a future talk?
C2 Management - Demo
A LITTLE HELP
(FOR MY FRIENDS)
IOCs & Defensive actions

- Can be difficult to detect .appref-ms as it is “Living off the Land”
  - Blocking .appref-ms execution may or may not be an option
  - Activity within the AppData folder is not atypical

- Monitor registry key modification
  - Addition and potential deletion of keys in the Uninstall tree

- Train end users to report odd activity
  - Odd installation prompts
  - ClickOnce execution sequence

- Continued efforts on post-execution detection
BLACKHAT TAKEAWAYS
3 Takeaways:

- .appref-ms is a versatile addition to any offensive toolbelt

- Tinker - Be curious about “outside the box” applicability

- Defender awareness of .appref-ms malicious activity
CLOSING QUESTIONS?