ATTACK SIMULATION GROUP

Purple Team Workshop: Lessons from Purple Teaming

CSA
ASG/CSEC

ASG Introduction

ATTACK
SIMULATION
GROUP

Enterprise

OT

Comms

Agenda

Time	Activity
5 mins	Purple Teaming Framework
15 mins	Lessons Learnt from Purple Teaming
40 mins	Practical Purple Teaming – Detection Engineering

Learning Objectives

- Understand how ASG conducts purple team exercises (PpTX)
- Identify challenges from the PpTX and how to address them
- Using detection engineering to create better rulesets

ATTACK
SIMULATION
GROUP

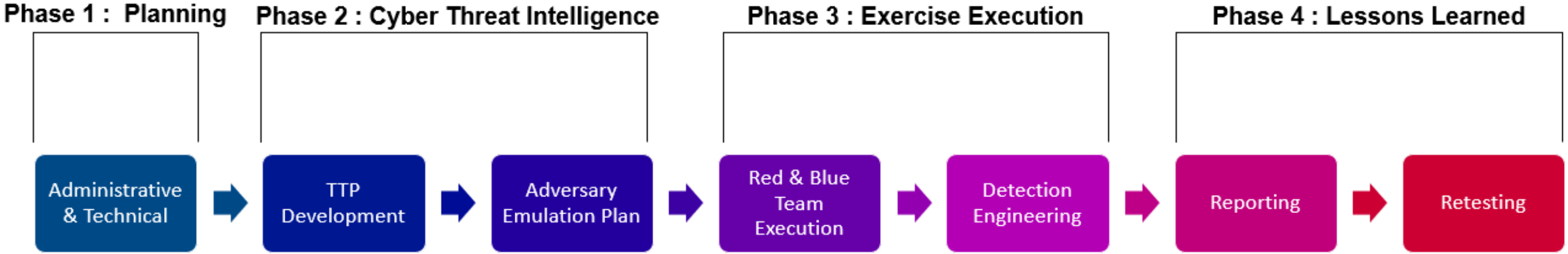
Conduct of Purple Teaming

Purple Teaming

Purple teaming is a collaborative approach between both teams with one shared goal

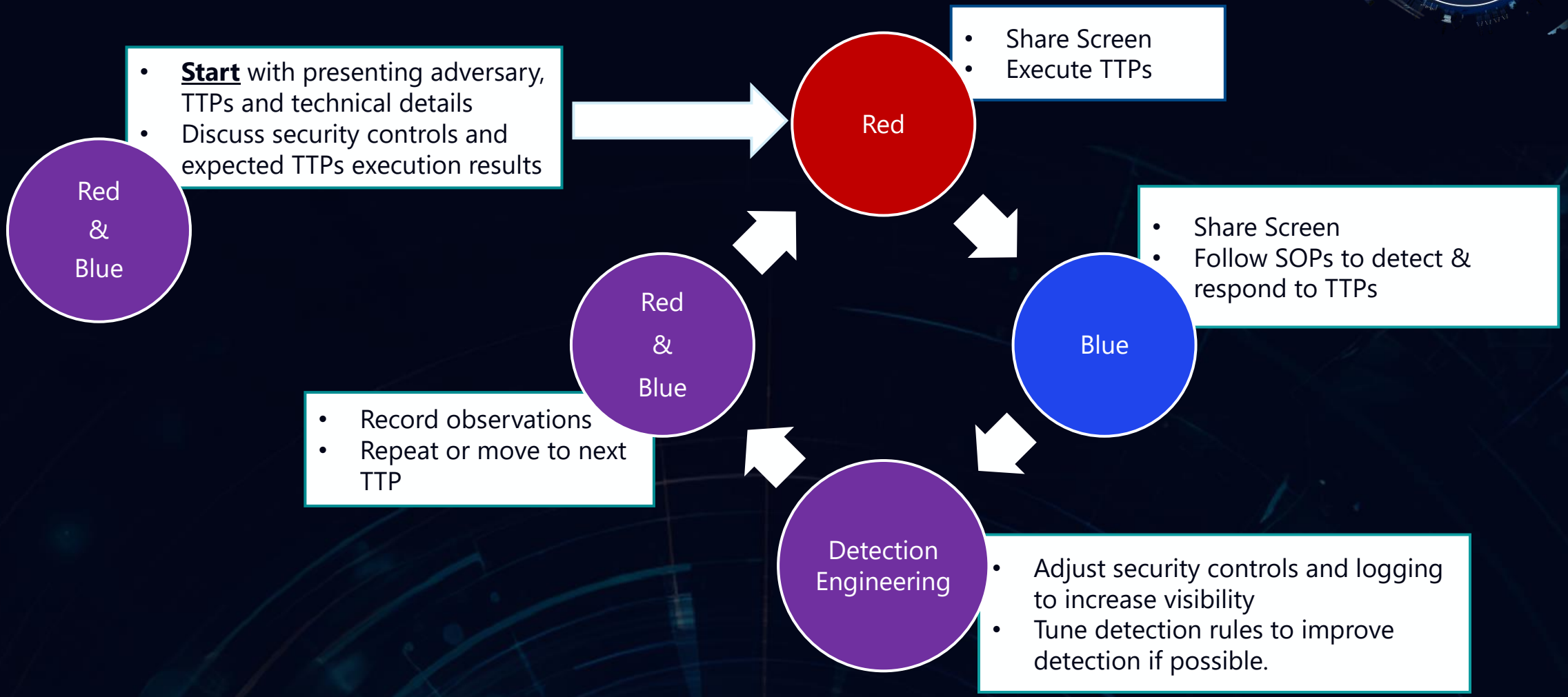
Purple Team Exercise Framework

ASG adapted the methodology defined under the “**Purple Team Exercise Framework**” (PTEF)*



* SCYTHE’s PTEF is one of the earliest and most targeted framework specifically designed for purple teaming

Purple Teaming Exercise Flow



ATTACK
SIMULATION
GROUP

Lessons Learnt

Overview

01

Planning

Determine the goals, scenarios and target systems

02

Cyber Threat Intel

Develop Adversary Emulation Plan (AEP) using a threat-informed approach relevant to target organisation

03

Execution

Execute the AEP, detect and respond to the TTPs

04

Closure

Documenting of results, reporting and retesting

Planning

Scoping

- Target System
  - Exercise should be done on *production* system or exact replica
  - UAT systems might not have the same configurations or detection/monitoring
  - Target system should have full detection and monitoring capabilities
- Approach
  - Assume-breach; Red Team have a foothold in the network and launches attack from there, based on the adversary emulation plan
  - Foothold (VM, or actual laptop) to have typical detection tools deployed
- Timeline
  - Depending on the environment and execution complexity
  - Planning: 2-3 weeks
  - Adversary Emulation Plan: 2 weeks
  - Execution: 1-2 weeks

Planning

Exercise Red Team (ERT)

To identify, understand and execute TTPs, ensuring reliable and consistent emulation

- Red Team Manager
- Red Team Members

Exercise White Team (EWT)

To oversee the development, execution, review and/or approval of the exercise

- Exercise Coordinator (Lead point of contact for exercise)
- CISO/CCSO
- System Owners

Exercise Blue Team (EBT)

To detect TTPs, identify evidence and perform detection engineering to increase visibility

- Blue Team Manager
- MSSP
- SOC
- Incident Responders
- Threat Hunters
- Digital Forensics

Cyber Threat Intelligence (CTI)

<u>Tactic, Techniques and Procedures (TTP) Development</u>

- TTPs chosen should be based on real-world threat actors
  - Resources are limited; Threat-informed approach prioritises resource allocation
  - More useful if threat actor is known to target your organisations sector
  - E.g. Blackcat, Lockbit, Volt Typhoon
- Emulation of TTPs
  - Modify the TTPs based on system configuration
    - E.g. CTI indicates that threat actor performs a registry edit on specific Windows 2019 Servers while your system do not have any Windows 2019 Servers
  - Realistic changes at the Procedural level while keeping the Tactic and Techniques relevant

Cyber Threat Intelligence (CTI)

Adversary Emulation Plan (AEP)

- During the review, for each TTP, the EBT will list down the expected outcome and observables
  - Expected Outcome: Prevented/Alerted/Logged
  - Expected Observables: E.g. AV Quarantine
  - Compare initial assumption versus exercise results
- TTPs defanged to prevent damaging hosts and network of the target system
- EWT reviews the TTPs provided in the AEP and approve the TTPs based on risk assessment and relevance

TTPs	Tactic	Technique	Procedure	Expected Outcome	Expected Observables	Risk Assessment for TTP	Approval by White Team (Y/N)
EICAR test string	Execution	T1204.002 User Execution: Malicious File	Open notepad, copy EICAR test string and save the file as eicar.txt	Prevented	AV quarantined eicar.txt	Low	Y

Execution

- Pre-Exercise Workshop
  - Briefing to all participants on the conduct and expectations of the Purple Team Exercise
  - It is useful to conduct small-scale execution of some TTPs. One example is using EICAR string
  - Get the working groups to experience actual-day events, defences working as expected
  - This also serves as a connectivity test to ensure all systems are go

EICAR test string

Scenario

This scenario involves a user saving a text file (.txt) that simulates a benign virus signature, specifically the European Institute for Computer Antivirus Research (EICAR) test string. The EICAR test string is a standardized sequence of characters that antivirus programs use to simulate a virus detection without using real malware. This scenario deals with the saving of a text file that will be detected by most antivirus software as a test virus.

Outcome

If the antivirus software detects and alerts upon opening the EICAR test string within the text file, the outcome is "Prevented." If the file is opened without detection, the outcome is "Not Prevented."

Prerequisites

Interactive user access on target workstation

Tactic

Execution

Technique

T1204.002 User Execution: Malicious File

Procedure

Open notepad, copy EICAR test string and save the file as `eicar.txt`

Clean up

Delete `eicar.txt`

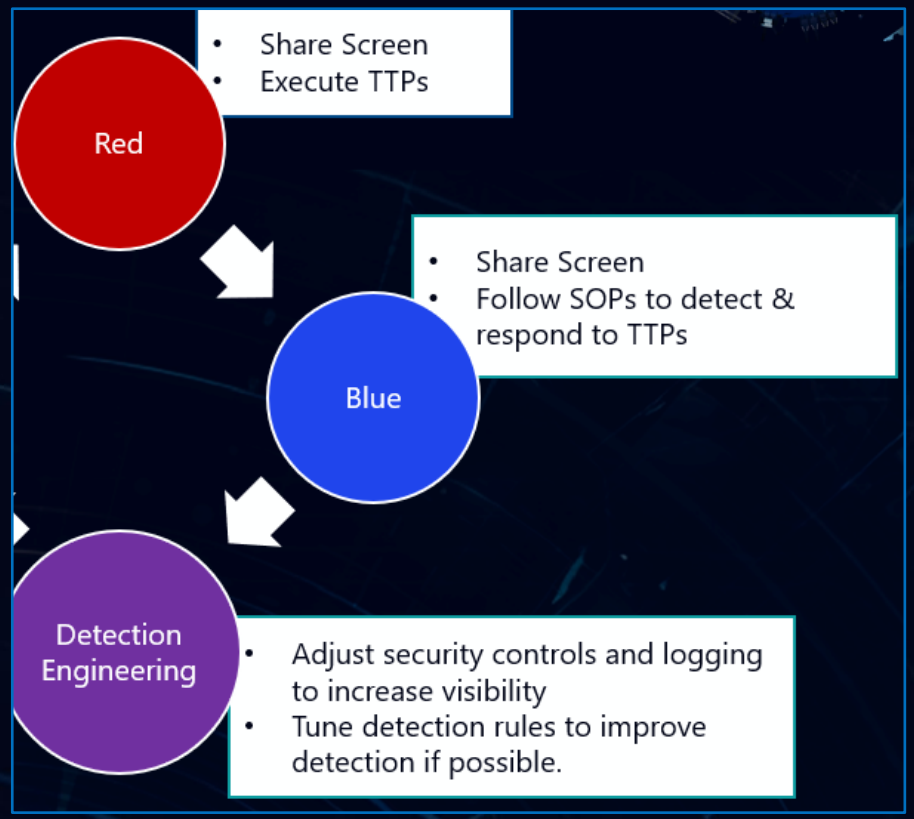
Detection

Monitor for antivirus alerts and logs indicating the detection of the EICAR test string within a text file

Execution

Actual Execution

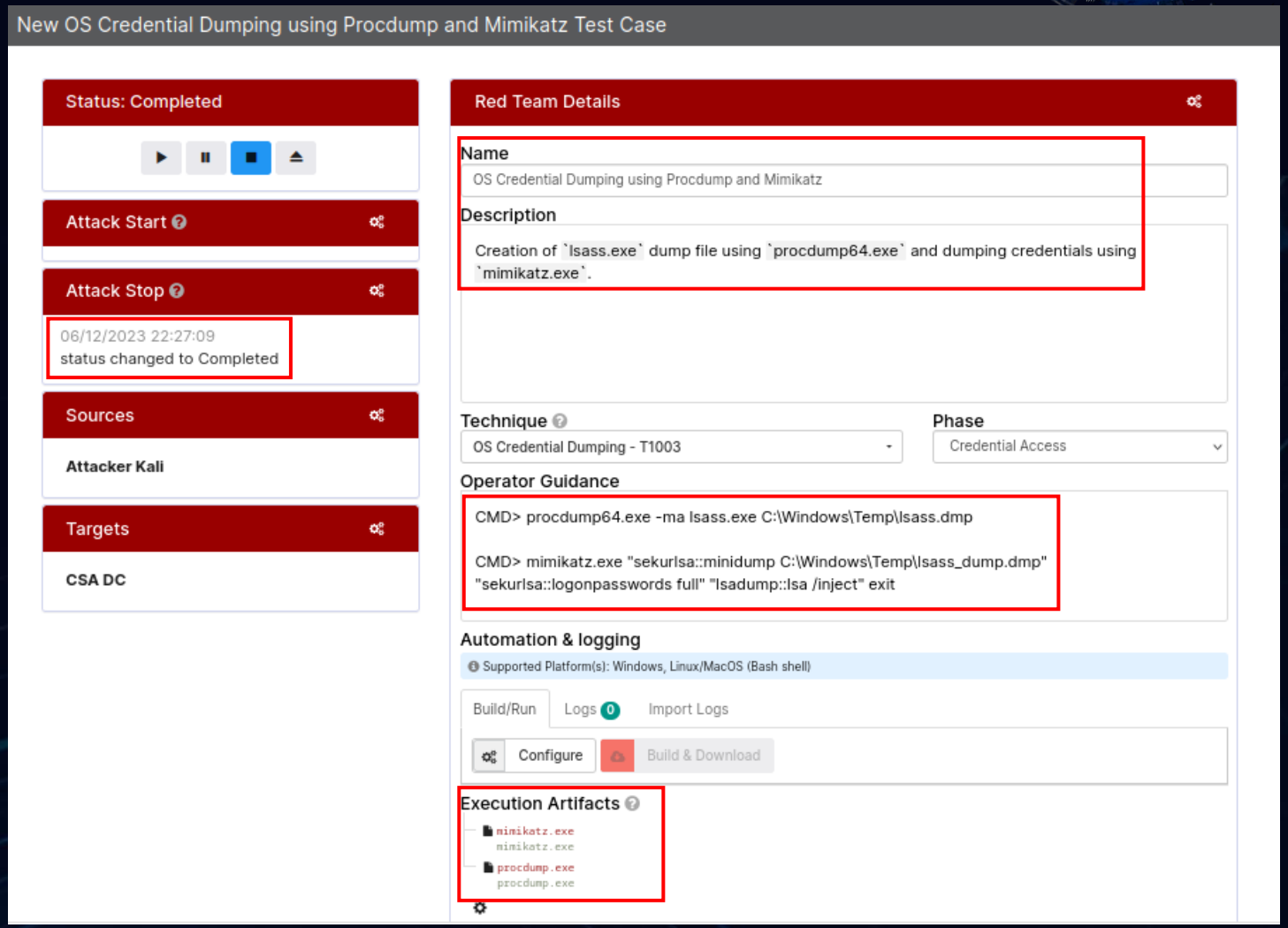
- Exercise cadence
  - Number of TTPs per day
  - Give enough time for all teams, most importantly EBT to investigate and perform detection engineering
- Increased value when both Red and Blue teams are openly sharing their actions
  - Easier to understand how the attack is carried out and how it is detected
- Some organizations are not comfortable with sharing screens due to privileged data that could be displayed
  - This lowers the synergy between ERT and EBT
  - Such issues to be flagged out earlier in planning stage and figure ways to mitigate this



Closure

Documentation and Samples

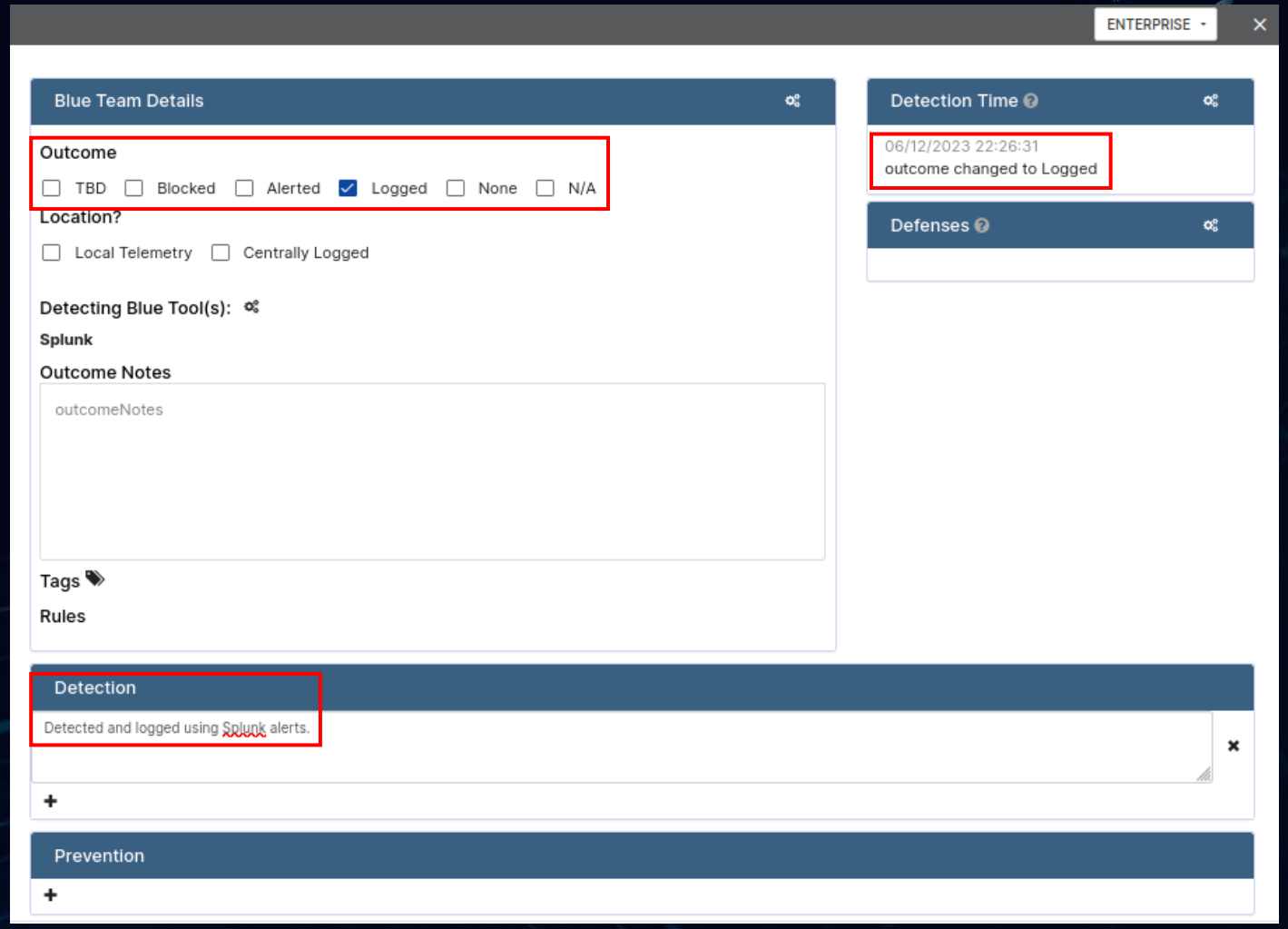
- Use tools to record down the results of the attacks
- Capture the exact commands run, artifacts generated etc



Closure

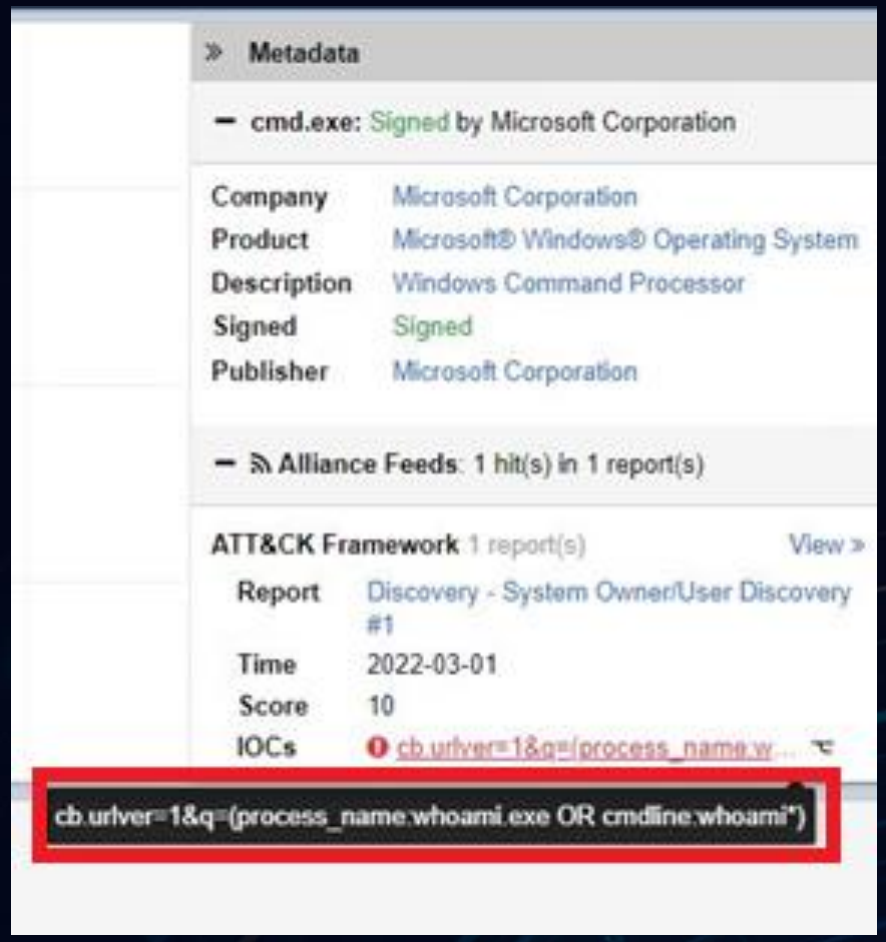
Documentation and Samples

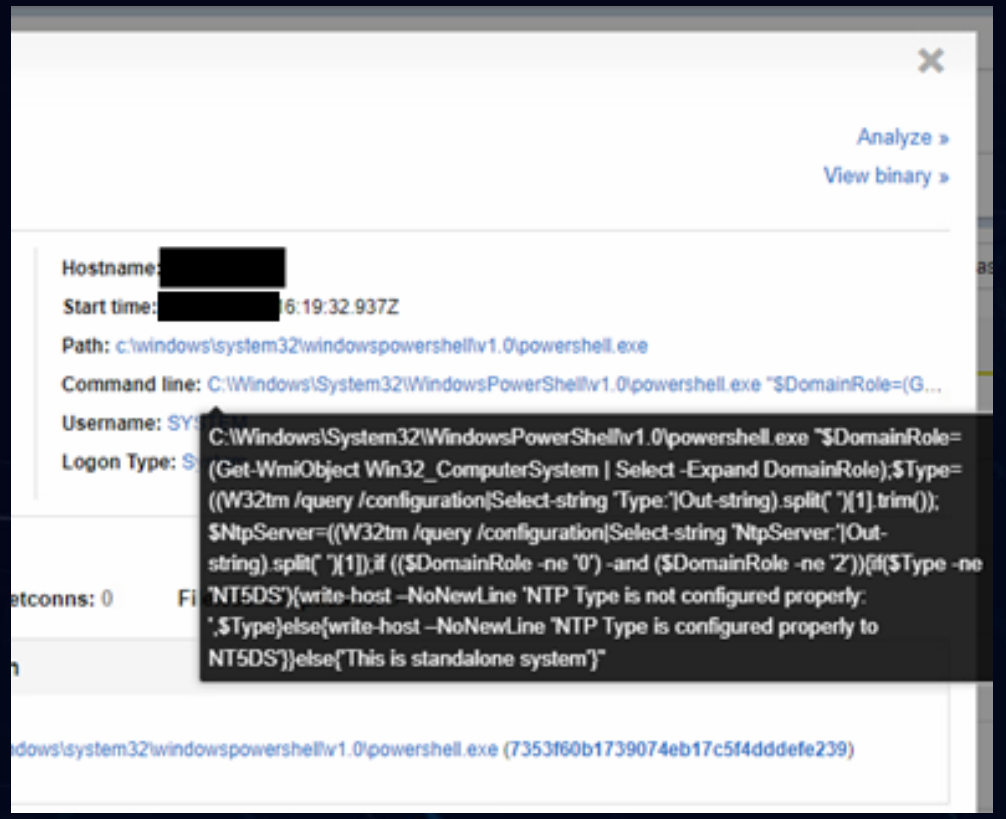
- Use tools to record down the detection/alerts used for each TTP
- Record outcome, where and how it was detected
- Provide screenshots of the detection rule



Closure

- Example screenshot of a TTP that was alerted and logged





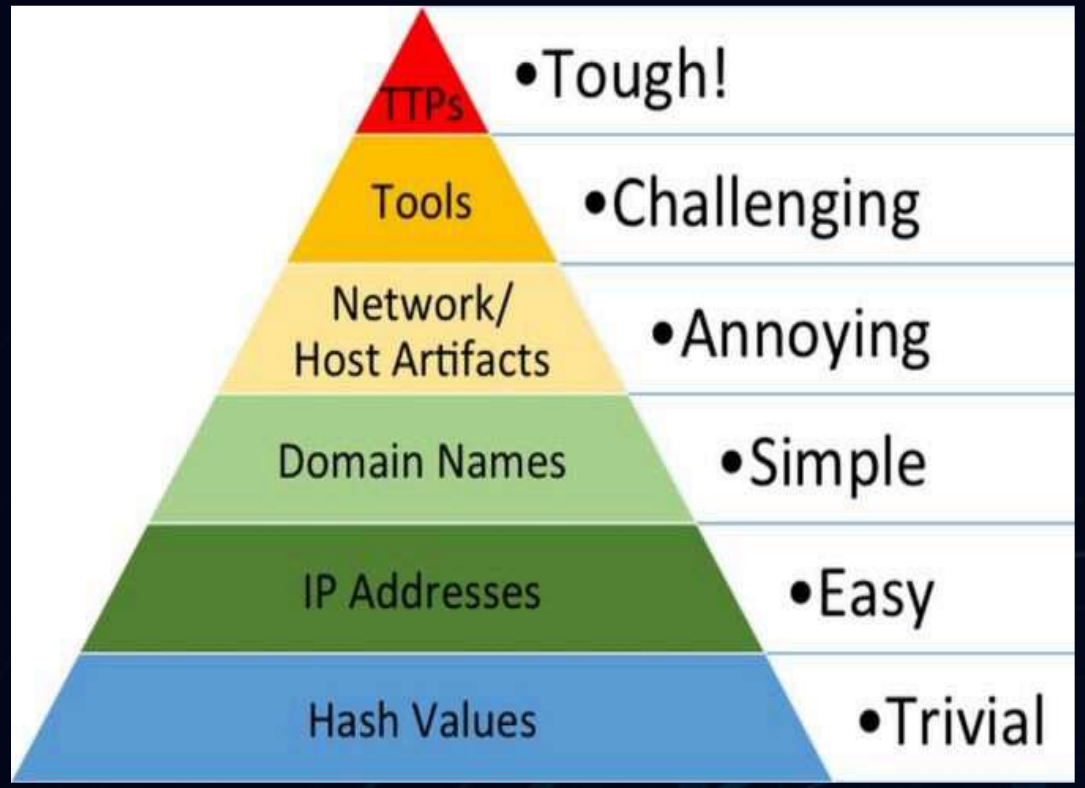
Lessons Learnt - Conclusion

- Planning
  - Target systems should have fully operational detection mechanism
- CTI
  - Threat-informed defense, with realistic emulation plan in mind
- Execution
  - Proactive sharing between EBT and ERT results in better detection engineering process
- Closure
  - Clear documentation can help in future Purple Team exercises to further improve detection capabilities

ATTACK
SIMULATION
GROUP

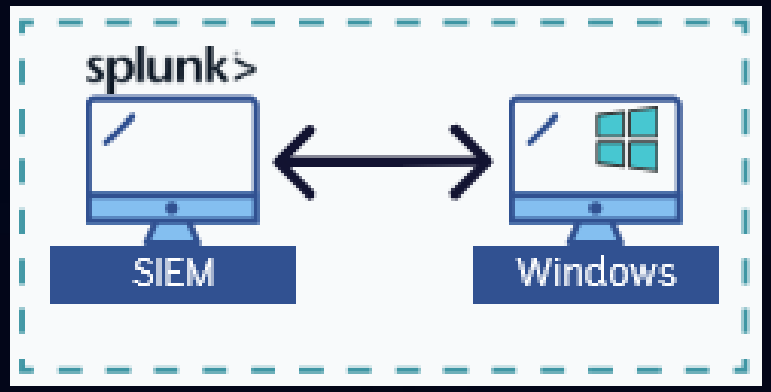
Hands-On:
Detection Engineering

Pyramid of Pain

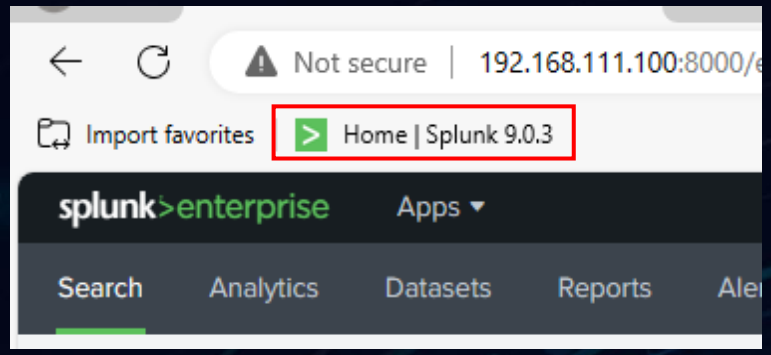


- It is ‘trivial’ for attackers to get around detection mechanisms the lower it is on the pyramid, such as by:
  - Renaming/modifying binaries
  - Using new domain names/IP addresses
- Nearly all indicators have a transitory value that fades over time, except TTPs
- The pyramid provides an ascending priority list of IOCs

Infrastructure Setup

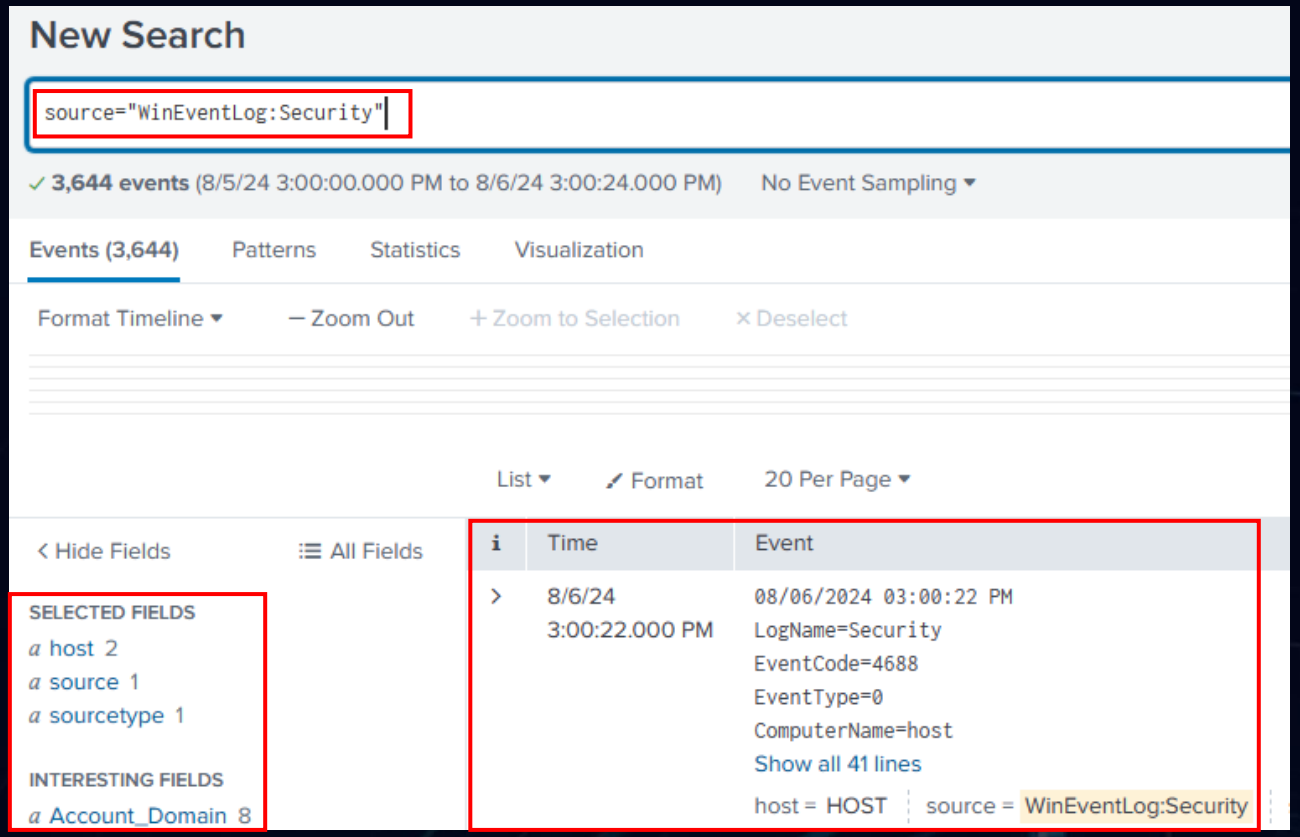


- VirtualBox running VMs
- 2 virtual machines in a network,
  - 1 Windows machine
  - 1 Linux machine running Splunk (SIEM)
- Windows host will be used to run attacks and access Splunk for detection



- Splunk access shortcut on Edge browser
- Splunk enterprise allows searching, analyzing and visualizing data and log files
- Used to create alerts based on search query

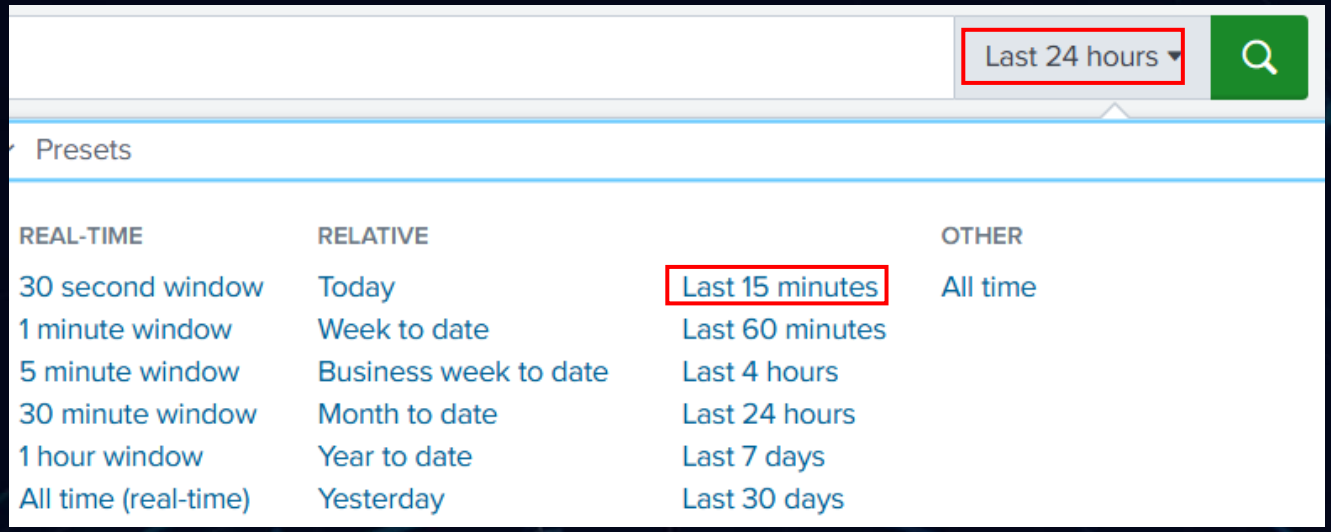
Splunk Interface



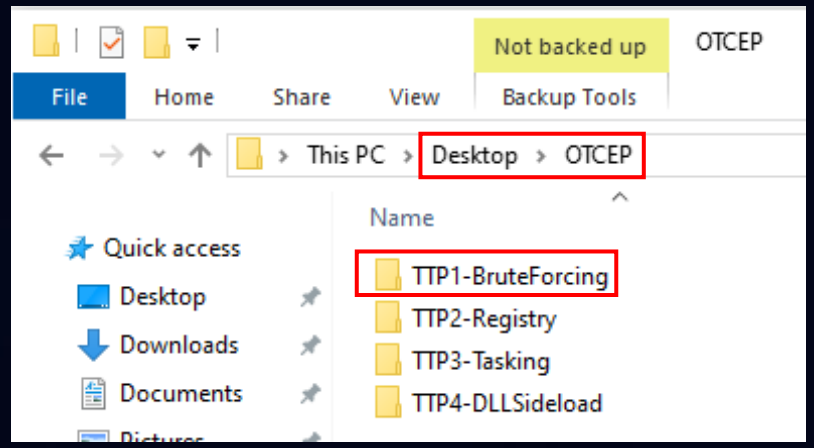
- Search box to enter your query
  - Searches through various log sources
- Results consist of the events returned based on the search query
  - Fields in the left column can be added to the search query to fine-tune it
- Look through the events to confirm if the event is a legitimate attack

Splunk Interface

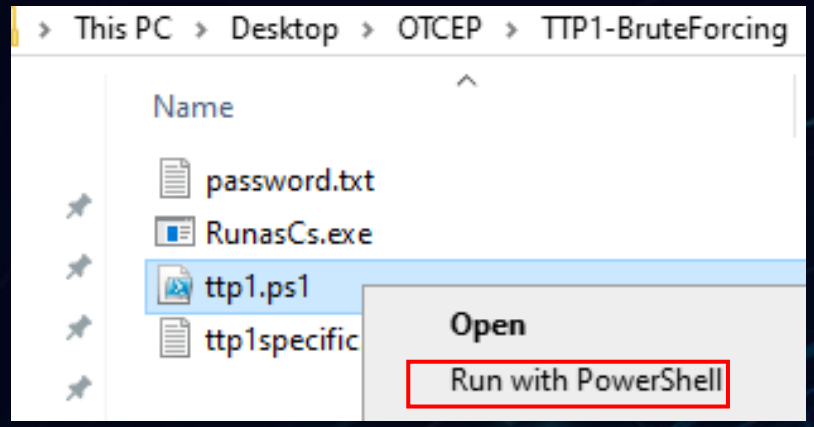
- Drop down list on the right controls how far back the logs are searched through
  - Please set to last 15 minutes to minimize noise from other TTPs



Attack Execution

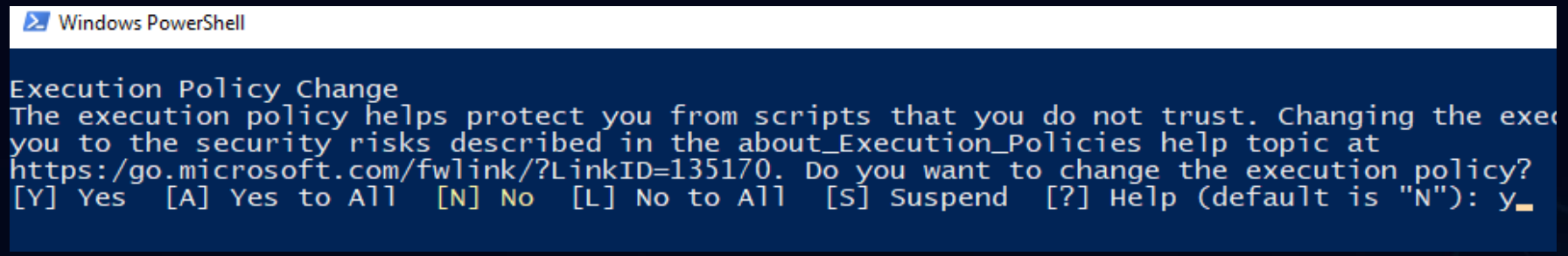


- All attacks are found in the OTCEP folder on the desktop
- Attacks will be run in sequence from 1-4, detection will be run after each attack



- Right-click and run with PowerShell to execute the attacks

Before We Begin



- Enter ‘y’ if the prompt above appears
- Reach out to the facilitators around you if you are unsure or need help

- The hands-on section will serve as a ‘snapshot’ of a simplified PpT engagement
- Your will run the attack as ‘red’ then threat hunt for it as ‘blue’

Splunk Warmup - TTP1: Brute Forcing (Initial Access)

Brute Force

A brute-force attack uses trial-and-error to guess login info, encryption keys or find a hidden web page

Scenario

This scenario demonstrates an adversary performing a brute force attack on a workstation login by iterating through a password list.

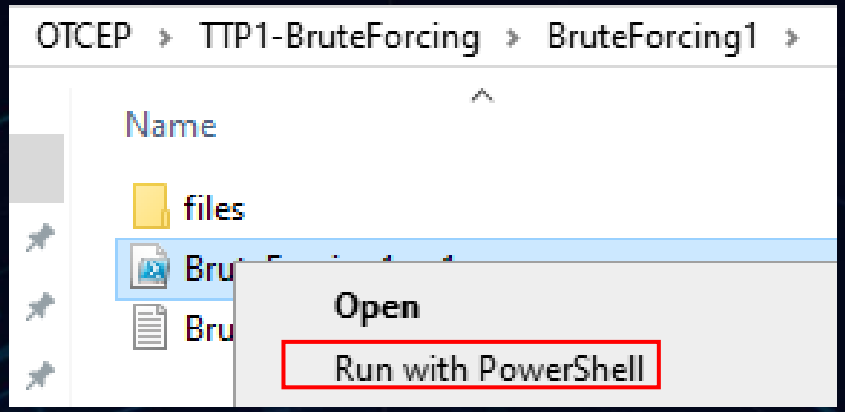
Splunk Warmup - TTP1: Attack

TTP Breakdown

- BruteForcing1.ps1 script specifies the user account to use
- Takes in a list of passwords from password.txt file
- For each password in the list, it attempts a logon to the workstation

Execution

Execute BruteForcing1.ps1 in PowerShell



Splunk Warmup - TTP1: Detection

Detection Strategy

- Detect instances of multiple failed logins for an account
- Log Source: Windows Security Logs Event ID 4625
  - An account failed to log on

Detection Rule

BruteForceDetection.txt

```
source="WinEventLog:Security" EventCode=4625
| stats
  count(eval(EventCode=4625)) AS failed_count
  by Account_Name
| where failed_count>6
```

Splunk Warmup – TTP1: Further Considerations

Scenario

What if an attacker obtains a compromised password but does not know which account to authenticate to? Attacker tried 1 password against all available accounts.

How do you detect this?

- Still use Event ID 4625
- Search for accounts with 1 failed login
- Number of accounts more than 10++ , within short period of time

```
<Rule name="Windows Command Shell" groupRelation="or">
  <OriginalFileName name="technique_id=T1059.003,technique_name=Windows Command Shell" condition="is">cmd.exe</OriginalFileName>
  <Image name="technique_id=T1059.003,technique_name=Windows Command Shell" condition="image">cmd.exe</Image>
</Rule>
<Rule name="PowerShell" groupRelation="or">
  <OriginalFileName name="technique_id=T1059.001,technique_name=PowerShell" condition="image">powershell.exe</OriginalFileName>
  <OriginalFileName name="technique_id=T1059.001,technique_name=PowerShell" condition="image">powershell_ise.exe</OriginalFileName>
  <OriginalFileName name="technique_id=T1059.001,technique_name=PowerShell" condition="contains">Sqlps.exe</OriginalFileName>
  <CommandLine name="technique_id=T1059.001,technique_name=PowerShell" condition="contains">pester</CommandLine>
</Rule>
```

TTP2: Registry Modification (Defence Evasion)

Registry

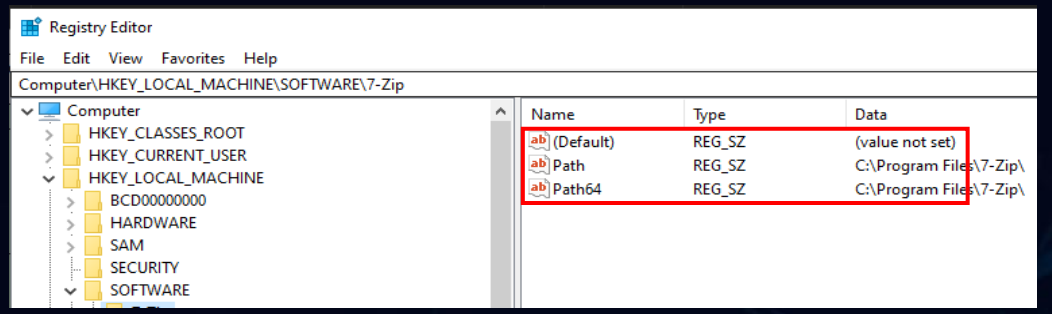
Database that stores low-level settings for Windows OS and applications.

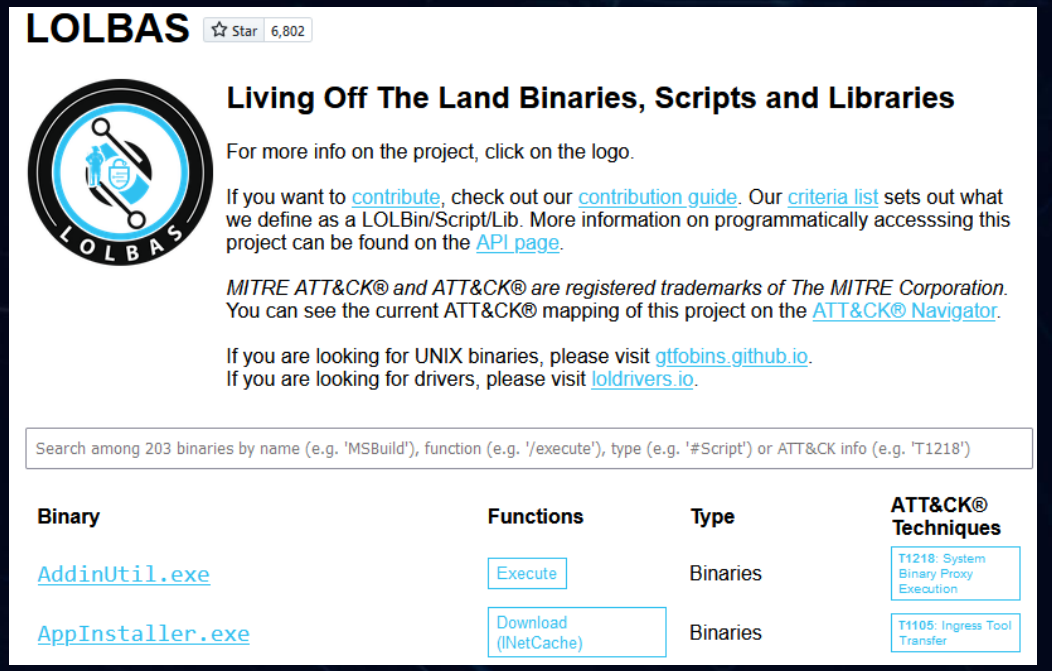
LOLBIN

Living-Off-The-Land-Binaries are non-malicious binaries, existing by default in the OS. Used for legitimate purposes and abused by attackers

Scenario

The attacker abuses a LOLBIN to modify a Registry value to disable a defence mechanism





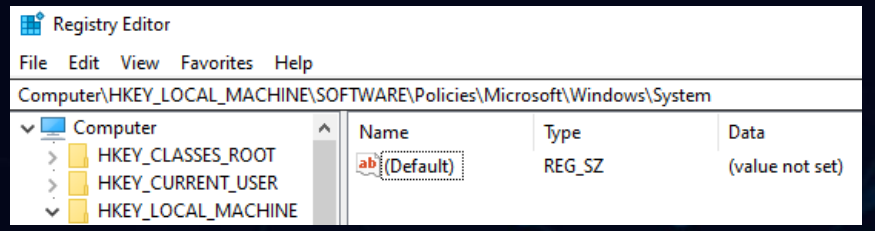
TTP2: Attack 1

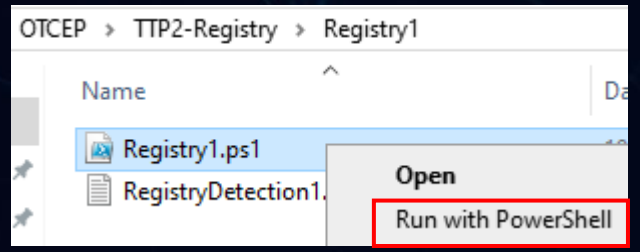
TTP Breakdown

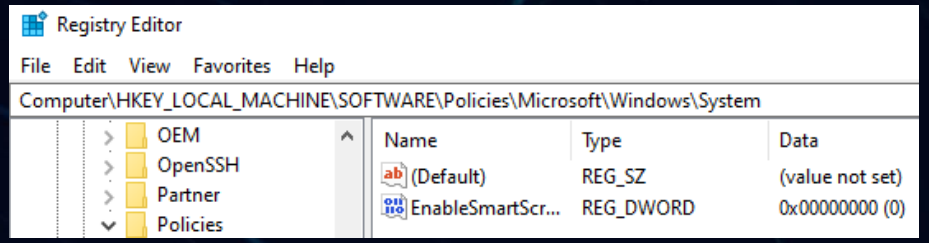
- Registry1.ps1 script uses the **reg.exe** LOLBIN to add a ‘EnableSmartScreen’ parameter in the Registry
- Sets the value of the parameter to 0 to disable it
- Open your Registry Editor file shortcut > Favourites > System

Execution

- Execute Registry1.ps1 in PowerShell







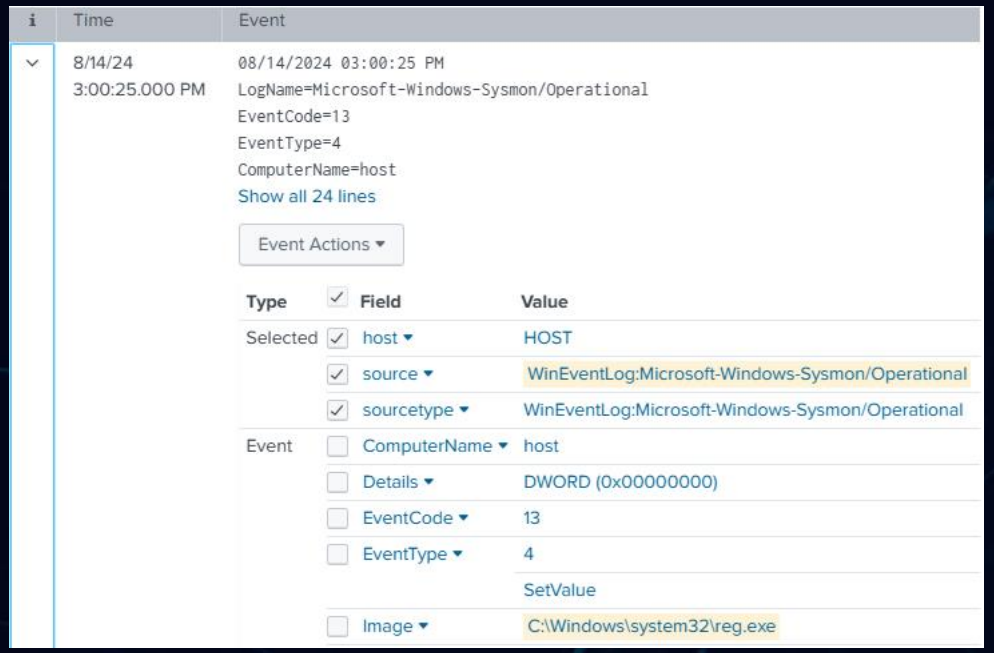
TTP2: Detection 1

Detection Strategy

- Monitor for use of the LOLBIN reg.exe
- Log Source: Sysmon

Detection Rule

RegistryDetection1.txt

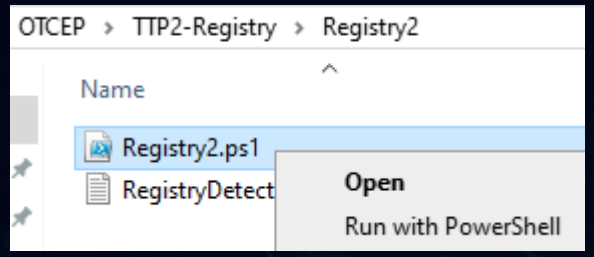



```
source="WinEventLog:Microsoft-Windows-Sysmon/Operational" Image="C:\\Windows\\system32\\reg.exe"
```

TTP2: Attack 2

Execution

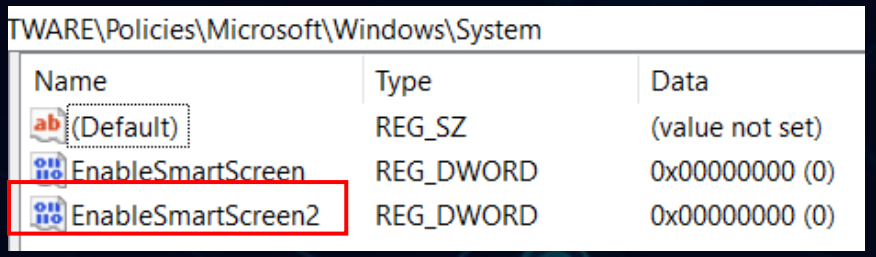
- Execute Registry2.ps1 in PowerShell



Re-run detection rule to see if it was detected

TTP Breakdown

- Registry2.ps1 script uses Windows API calls like
  - RegOpenKeyEx, RegCreateKeyEx
  - RegSetValueEx, RegCloseKey
- Handled by PowerShell cmdlets



Direct calls to Windows API removes the need for LOLBIN usage

TTP2: Detection 2

Detection Strategy

- Rather than LOLBIN usage, we need to monitor for changes to registry
- Many legitimate registry events, only look for events from non-privileged user
- Log Source: Sysmon

Detection Rule

RegistryDetection2.txt

Type	Field	Value
Selected	host	HOST
	source	WinEventLog:Microsoft-Windows-Sysmon/Operational
	sourcetype	WinEventLog:Microsoft-Windows-Sysmon/Operational
Event	ComputerName	host
	Details	DWORD (0x00000000)
	EventCode	13
	EventType	4
		SetValue
	Image	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
	Keywords	None
	LogName	Microsoft-Windows-Sysmon/Operational
	Message	Registry value set: RuleName: technique_id=T1562.001,technique_name=Disa bc-9006-000000000b00} ProcessId: 7936 Image: C:\Windows\System32\Wir nableSmartScreen2 Details: DWORD (0x00000000) User: HOST\vagrant
	OpCode	Info
	ProcessGuid	{078b5ae9-579f-66bc-9006-000000000b00}
	ProcessId	7936
	RecordNumber	5480
	RuleName	technique_id=T1562.001,technique_name=Disable or Modify Tools
	Sid	S-1-5-18
	SidType	0
	SourceName	Microsoft-Windows-Sysmon
	TargetObject	HKLM\SOFTWARE\Policies\Microsoft\Windows\System\EnableSmartScreen2

```
source="WinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=13 User!="NT AUTHORITY\\SYSTEM" RuleName!="-"
```

TTP3: Scheduled Tasks (Persistence)

Task Scheduler

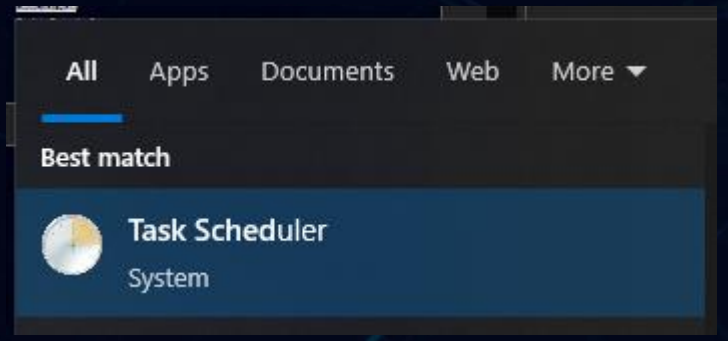
Job scheduler in Windows that launches programs or scripts at pre-defined times or intervals

Persistence

Method to keep access to systems across restarts, changed credentials and other interruptions

Scenario

The attacker schedules a task to execute a binary in order to maintain persistent access to the workstation.



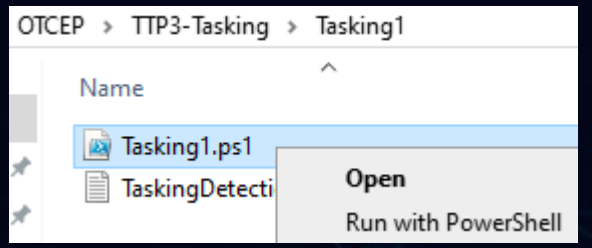
TTP3: Attack 1

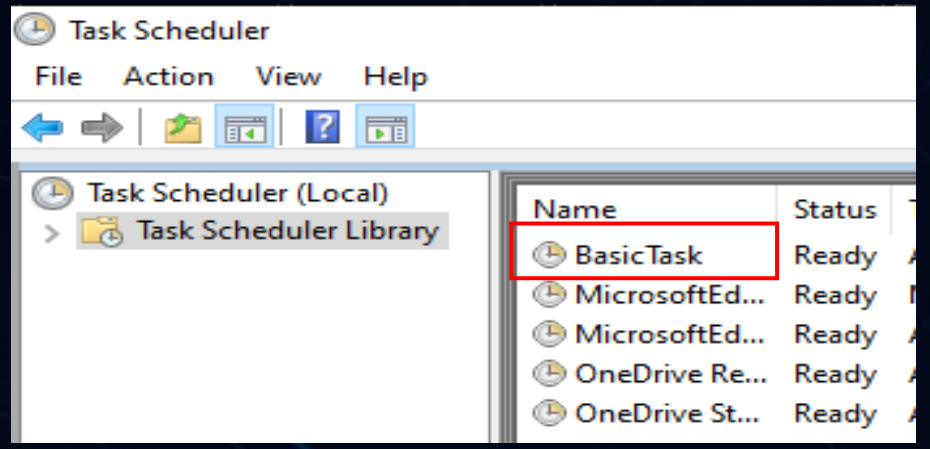
TTP Breakdown

- Tasking1.ps1 uses the LOLBIN schtasks.exe to create a scheduled task
- The task will execute a binary file as part of its actions

Execution

- Execute Tasking1.ps1 in PowerShell





TTP3: Detection 1

Detection Strategy

- Monitor for use of the LOLBIN schtasks.exe
- Log Source: Sysmon

Detection Rule

TaskingDetection1.txt

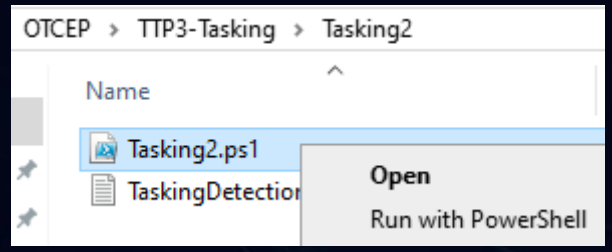
Type		Field	Value
Selected	✓	host ▾	HOST
	✓	source ▾	WinEventLog:Microsoft-Windows-Sysmon/Operational
	✓	sourcetype ▾	WinEventLog:Microsoft-Windows-Sysmon/Operational
Event		CommandLine ▾	"C:\Windows\system32\schtasks.exe" /create /sc once /tn BasicTask /tr calc.exe /st 23:59
		Company ▾	Microsoft Corporation
		ComputerName ▾	host
		CurrentDirectory ▾	C:\Users\vagrant\Desktop\OTCEP\TTP3-Tasking\Tasking1\
		Description ▾	Task Scheduler Configuration Tool
		EventCode ▾	1
		EventType ▾	4
		FileVersion ▾	10.0.19041.3636 (WinBuild.160101.0800)
		Hashes ▾	SHA1=FCE60EBC7EBCC8B09D5821338391D800E7B37591,MD5=D4DA03B7BB20B7E4F6052CC27F168C50F2,IMPHASH=ECCE05491F2E8F279F4790BCB1318C05
		Image ▾	C:\Windows\System32\schtasks.exe

```
source="WinEventLog:Microsoft-Windows-Sysmon/Operational" Image="C:\\Windows\\system32\\schtasks.exe"
```

TTP3: Attack 2

Execution

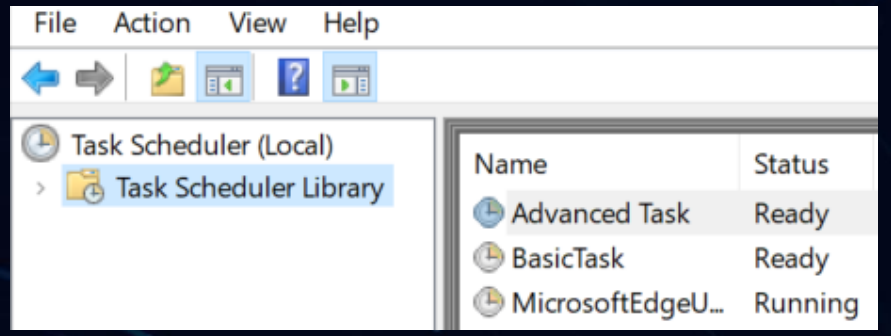
- Execute Tasking2.ps1 in PowerShell



Re-run detection rule to see if it was detected

TTP Breakdown

- Tasking2.ps1 script uses Windows Task Scheduler API like
  - ITaskService::NewTask
  - IRegisteredTask::RegisterTask
- Handled by PowerShell cmdlets



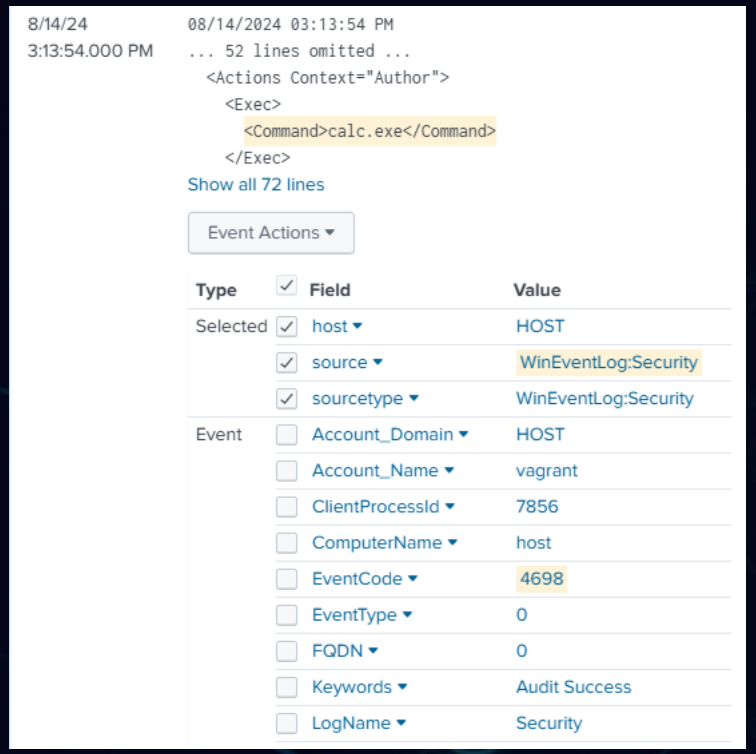
TTP3: Detection 2

Detection Strategy

- Monitor for creation of new scheduled tasks
- Look for binary execution within those scheduled tasks
- Log Source: Windows Security Logs Event ID 4698
  - A scheduled task was created

Detection Rule

TaskingDetection2.txt




```
source="WinEventLog:Security" EventCode="4698" "<Command>*.exe</Command>"
```

TTP4: DLL Sideloading (Reconnaissance)

DLL

Dynamic-link library is a shared library in Windows and can contain executable code, data and resources. There is a search order for where programs look for DLLs.

Scenario

The attacker performs a DLL sideload by hijacking the search order for program and successfully loading their malicious DLL instead.

This scenario simulates when a user opens a program (calc.exe) which results in the execution of the malicious library (WININET.dll).

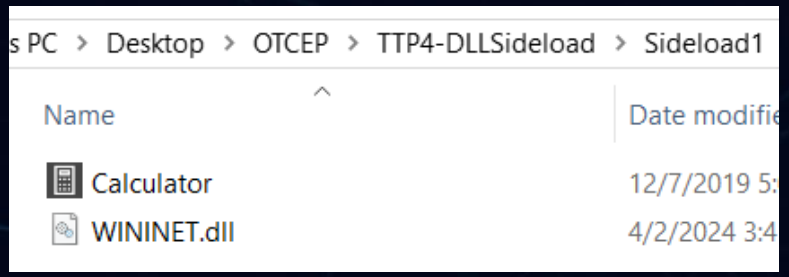
The malicious library executes a series of local discovery commands via cmd:

- whoami
- net share
- dir

TTP4: Attack

TTP Breakdown

- Calc.exe will look for WININET.dll in order to execute properly
- Calculator.exe will first look in the current working directory (Sideload1) for the DLL and use the file if it exists
- Originally it would have loaded the WININET.dll from System 32 folder



Execution

- Double-click Calculator.exe

TTP4: Detection

Detection Strategy

- Monitor for execution of unsigned binaries
- Filter out binaries to those from unexpected places
- Log Source: Sysmon

Detection Rule

SideloadDetection1.txt

Image	C:\Users\vagrant\Desktop\OTCEP\TTP4-DLLSideload\Sideload1\Calculator.exe
ImageLoaded	C:\Users\vagrant\Desktop\OTCEP\TTP4-DLLSideload\Sideload1\WININET.dll
Keywords	None
LogName	Microsoft-Windows-Sysmon/Operational
Message	Image loaded: RuleName: unsinged binary UtcTime: 2024-08-14 07:16:56.795 P op\OTCEP\TTP4-DLLSideload\Sideload1\Calculator.exe ImageLoaded: C:\Users\ File description> Product: TODO: <Product name> Company: TODO: <Company 7,MD5=E6A47D24F6B8992DF6C25D7117649C25,SHA256=4880DA1CD24B2B 2C6AD8BAF Signed: false Signature: - SignatureStatus: Unavailable User: HOS
OpCode	Info
OriginalFileName	WININET.dll
ProcessGuid	{078b5ae9-59e8-66bc-e406-000000000b00}
ProcessId	3768
Product	TODO: <Product name>
RecordNumber	5667
RuleName	unsinged binary
Sid	S-1-5-18
SidType	0
Signature	-
SignatureStatus	Unavailable
Signed	false

```
source="WinEventLog:Microsoft-Windows-Sysmon/Operational" Signed=false ImageLoaded !="C:\\Windows\\*"
```

Conclusion

- Current detection rules result in a large number of false positives or are easily bypassed
  - Make use of multiple log sources in order to filter out noise and make more sense of events
- Detection should increase in granularity over time, allowing for better and more accurate detection
  - Try to create rulesets that are higher up the pyramid of pain avoiding naïve detections such as LOLBINs
- Detection rules should be crafted while considering the system in place: general user behaviors, application configurations etc
- A PpTX can be used to validate your current detection rules
  - If your detection rules are solid, can the ERT execute variations of the attack to see how waterproof your detection rules are

THANK YOU

Scan the QR code to
share your feedback with us