Detection Engineering: Defending Networks with Purpose

GIAC (GCIA) Gold Certification

Author: Peter Di Giorgio, peter.di.giorgio00@gmail.com
Advisor: *John Hally*

Accepted: *23 June 2021*

Abstract

Detection engineering is becoming a common term in the information security industry, but it is still a maturing concept. From the perspective of a military philosopher, this paper will explore the tactics, techniques, and procedures behind detection engineering. The goal is to give analysts, researchers, and decision-makers tools to apply in their organizations today. This research explores a method to template threats to an organization, analyze a capability against the threat template for detection gaps, and engineer detections to close observed gaps. With a handful of open-source tools, it will be possible to achieve a military-grade defensive posture. Network defenders will be able to use detection engineering to defend networks with knowledge and purpose.

1. Introduction

Sun Tzu has a saying: "If you know the enemy and know yourself, your victory will not stand in doubt: if you know Heaven and know Earth, you may make your victory complete" (Tzu, 2014). The wisdom of this famous military philosopher applies to any conflict, including those in cyberspace.

In more concise modern language, Sun Tzu's words can be distilled into three simple statements: Know Yourself; Know the Terrain; Know the Enemy (or Adversary). Knowing is a process, not an outcome. Information security analysts may know the situation one minute and run into the unknown the next minute. For this reason, having some structure behind an organizational quest for knowledge is essential. This structure can come from many sources. Policies, standards, and guidance will play a role, especially when informed by threat research and observations in the environment. Another source is security analysts and engineers studying their network and the threat to design the best conditions to highlight adversary activity. That process is called detection engineering.

2. Detection Engineering

Understanding detection engineering begins with a look at intrusion detection. Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of adversary behavior or security risks (Bace, 2000). Intrusion detection can be overwhelming work, and many times, detection gaps are discovered well after an intrusion or penetration test.

Companies were estimated to have spent over 123 billion dollars in cybersecurity in 2020 (Columbus, 2020), driving the median global dwell time for attacks down to 24 days (FireEye Mandiant Services, 2021). This is a staggering 32-day decrease from the previous year. Unfortunately, the global median dwell time for non-ransomware intrusions in 2020 was 45 days (FireEye Mandiant Services, 2021). The industry will not see significant results in finding intrusions before they are destructive until it reduces the time it takes for an analyst to move from event discovery to incident identification. Information environments that are designed, or engineered, to detect events or behaviors

indicative of adversary behavior will focus analysts on the activity of real interest. For many medium to small organizations, systematic detection gap analysis coupled with cheap and readily available tools and data sources leads to effective detection engineering.

Detection Engineering is not only for analysts. It will help organizations understand their information environment and their capability investments. Like so many organizational processes, detection engineering is not a standalone activity. Josh Day, a Gigamon detection engineer, shows in figure 1 how the threat intelligence process feeds detection engineering and leads to a continual cycle of detection maintenance (Day, 2020). Understanding those relationships is essential as detection engineering steps are analyzed/considered.

Figure 1: Detection Engineering Relationships (Day, 2020)

Detection Engineering begins with an understanding of an organization's capabilities and the data the organization generates. Inventorying capabilities (tools, appliances, operating systems) and data sources for event generations is the first step in the process. The inventory of data types will drive the detection possibilities. Native endpoint logs, packet metadata, and network intrusion detection (NIDS) signature alerts have value when detecting adversary behavior in an information environment. The next step is to assess those data sources against adversary behaviors we would like to see.

Detection gap analysis is not a new concept. The shift is in defining the precision of the gap analysis. The data generated by most capabilities is rich with events of interest. The challenge lies in where an organization can land on the "detection spectrum." Jared Atkinson, a detection engineer from Specter Ops, highlights this organizational variable below in Figure 2. A small security team may not have the capacity to handle the potentially large volume of false positives, while precise detections may miss indications of technique variants (Atkinson, 2020). Because detection engineering is a cyclical process, analysts may choose to work from broad to precise

detections when conducting capability analysis against templated threats to understand the investigative load of alerts.

Figure 2: Detection Spectrum (Atkinson, 2020)

Once a capability has been tested against a templated threat, detection gaps are identified based on an organization's risk tolerance, and analysts can create new detections to close gaps. New detections may come in many shapes and sizes. It may be as simple as a configuration change to include an alert, or as complex as generating events from several log sources in conjunction with alerting logic for those specific events. Ultimately, data and analysis drive detection engineering.

3. Practical Application of Detection Gap Analysis

The environment maintains a simple design to remain focused on the capability we are attempting to analyze. For this practical application, an open-source host intrusion detection system, Wazuh, running on a Windows 10 operating system, has been selected. Security Onion will be used as an enterprise security monitoring platform to capture network and host-generated data. Security Onion has a user interface well suited to observe and document adversary behavior tests and capabilities to close detection gaps through various rulesets and alert generation techniques.

3.1. Define data required to observe a threat tactic, technique, or procedure.

The practical application of gap analysis requires threat behavior to analyze. The MITRE ATT&CK windows enterprise matrix (MITRE, 2021) templates the threat for this exercise. The MITER ATT&CK framework is a widespread knowledge base of adversary behavior and taxonomy (MITRE, 2021) used in a variety of applications. It's

beneficial in this case because the Atomic Red Team project maps its adversary emulations to the MITRE ATT&CK matrix.

The Atomic Red Team is a handy tool for gap analysis. It is "a library of simple tests that every security team can execute to test their defenses" (Red Canary, 2021). The project contains many tests run from a command line in Windows or Linux systems. This practical application uses the Invoke-AtomicRedTeam PowerShell module as an execution framework used to quickly launch tests from the command line (Red Canary, 2021). This practical application will utilize 68 tests that cover 33 unique techniques across nine tactics of the MITRE ATT&CK matrix.

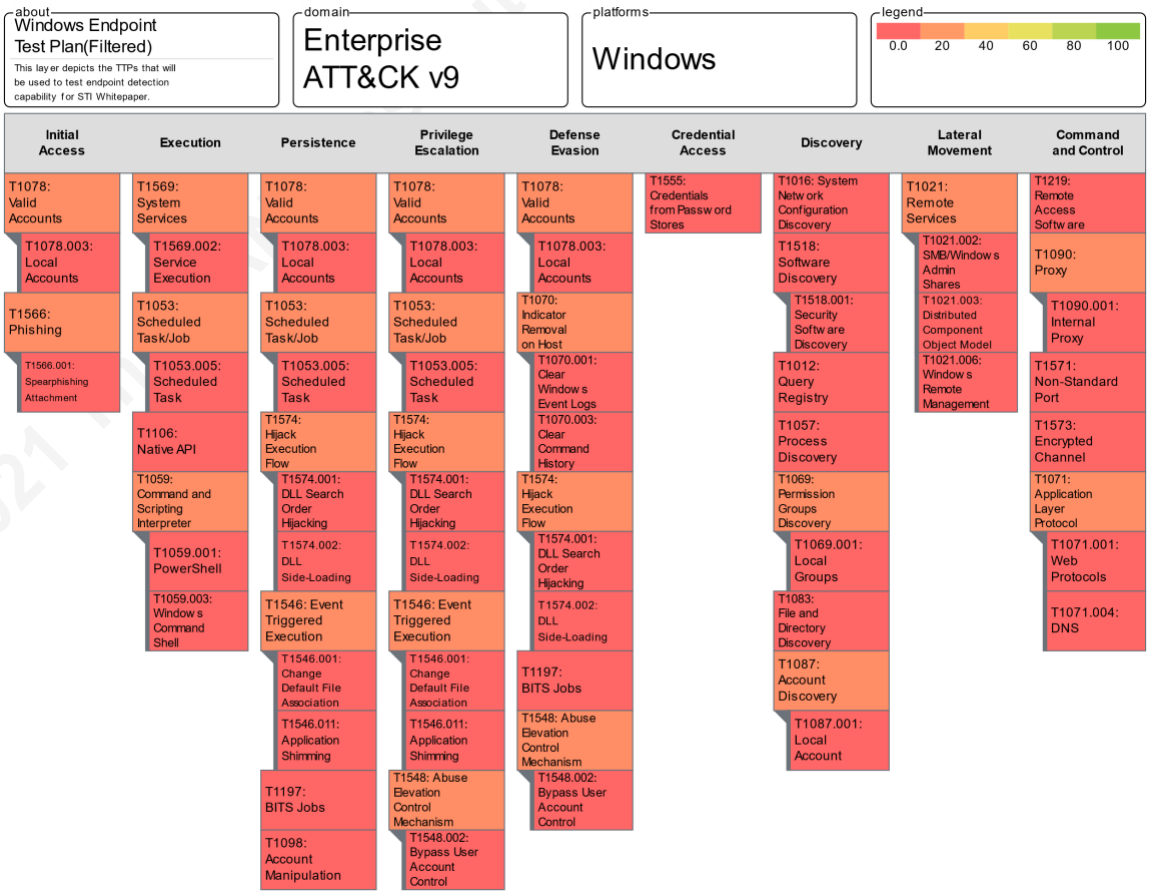


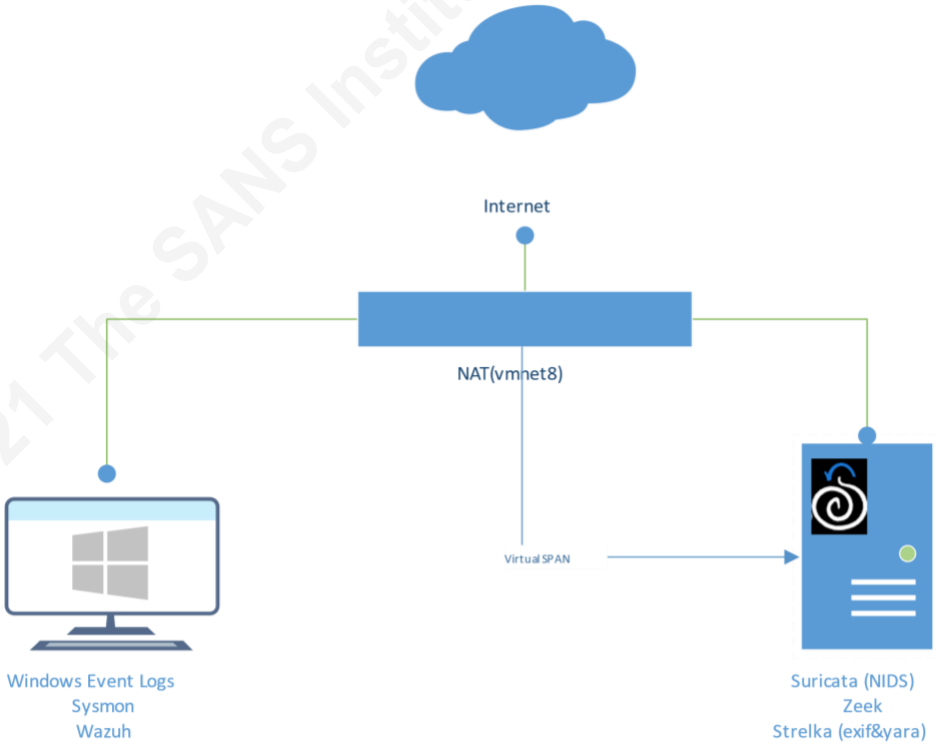
Figure 3: Windows Endpoint Test Plan (Filtered)

Figure 3 displays the techniques selected. Another great tool from the MITRE ATT&CK project is the MITRE ATT&CK Navigator (MITRE, 2021). This webpage can

be accessed from the internet or hosted in an organization's environment. Security Onion 2 integrates Navigator for analysts to track campaigns or detection engineering (Security Onion, 2021). Analysts can export their work in images, like the one in Figure 3, or JSON files like the one hosted in this paper's repository (Appendix A).

3.2. Design data collection method through endpoint or network sources

Figure 5 shows the environment used for this paper. The data sources listed below the Windows 10 endpoint and Security Onion server are most important to the process of gap analysis. Those data sources will generate the events or alerts needed to observe the templated threat activity. Those data sources will be explored in the following sections.



3.2.1. Windows Event Logs

The Windows 10 operating system generates many log sources that create a large number of events stored in a unique Windows Event log format. Many services generate logs, but there are specific logs that have more value to cybersecurity operations. An obvious log source for security monitoring is the Security log, but it isn't the only way to

receive accurate indications of malicious activity. Here are some additional Windows event logs to collect (Lee & Pilkington, 2018):

- Application
- Security
- System
- Windows PowerShell
- Microsoft-Windows-PowerShell/Operational
- Microsoft-Windows-RemoteDesktopServices-RdpCoreTS/Operational
- Microsoft-Windows-SmbClient/Security
- Microsoft-Windows-SMBServer/Security
- Microsoft-Windows-TaskScheduler/Operational
- Microsoft-Windows-TerminalServices-RemoteConnectionManager/Operational
- Microsoft-Windows-Windows Defender/Operational
- Microsoft-Windows-Windows Firewall with Advanced Security/Firewall
- Microsoft-Windows-Winlogon/Operational
- Microsoft-Windows-WinRM/Operational
- Microsoft-Windows-WMI-Activity/Operational

3.2.2. System Monitor (Sysmon)

Another popular application analysts can use to generate events in the Windows operating system is System Monitor (Sysmon). Sysmon is a system service developed by Microsoft as part of the Windows Sysinternals suite of tools. Sysmon monitors and logs system activity such as process creation, network connections, registry changes, and changes to file creation time (Russinovich & Garnier, 2021). Sysmon logs its events to a Windows event log named Microsoft-Windows-Sysmon/Operational. With a configuration file like the one at https://github.com/SwiftOnSecurity/sysmon-config, Sysmon has the following capabilities (Russinovich & Garnier, 2021):

- Logs process creation with full command line for both current and parent processes.
- Records the hash of process image files using SHA1 (the default), MD5, SHA256, or IMPHASH.
- Multiple hashes can be used at the same time.
- Includes a process GUID in process create events to allow for correlation of events even when Windows reuses process IDs.
- Includes a session GUID in each event to allow correlation of events on the same logon session.
- Logs loading of drivers or DLLs with their signatures and hashes.
- Logs open for raw read access of disks and volumes.
- Optionally logs network connections, including each connection's source process, IP addresses, port numbers, hostnames, and port names.
- Detects changes in file creation time to understand when a file was created. Modification of file create timestamps is a technique commonly used by malware to cover its tracks.
- Automatically reload configuration if changed in the registry.
- Rule filtering to include or exclude certain events dynamically.
- Generates events from early in the boot process to capture activity made by sophisticated kernel-mode malware.

3.2.3. Wazuh

Wazuh is a security detection, visibility, and compliance open-source project. It was born as a fork of OSSEC HIDS and was later integrated with Elastic Stack and OpenSCAP, evolving into a more comprehensive solution (Wazuh, 2021). Of the many components in the project, the Wazuh agents can monitor various operating systems, produce alerts/events, and transport logs. Figure 4 outlines the agent's potential in endpoints:

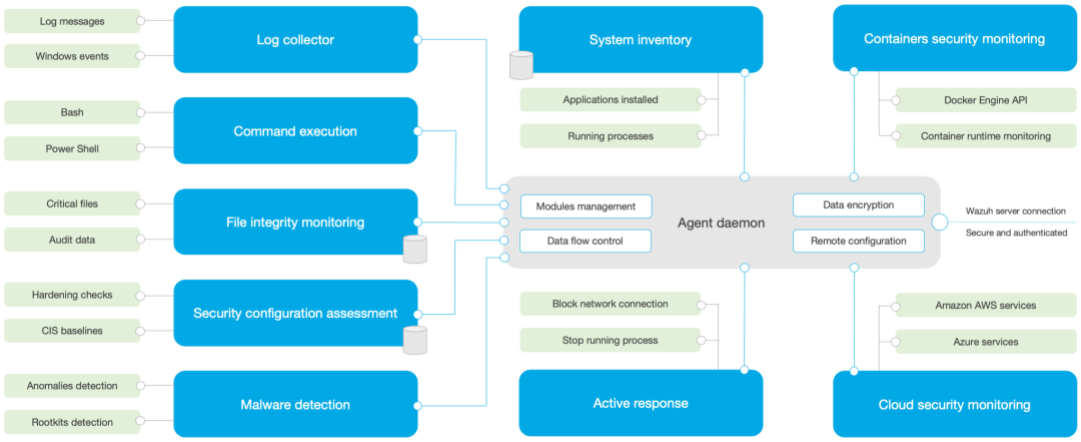


Figure 4: Wazuh Agent Capabilities (Wazuh, 2021)

3.2.4. Security Onion

Security Onion is a free and open Linux distribution for enterprise security monitoring, log management, and threat hunting (Security Onion, 2021). The project integrates many free and open tools into one platform. These tools can be divided/categorized into four areas: Infrastructure Management, Network Monitoring, Host Monitoring, and Analyst Interface. This paper will focus on capabilities in the last three areas. The previous section highlighted the key host monitoring capabilities, and this section will focus on network monitoring and analyst interface.

Zeek, Suricata, and Strelka are specifically useful network monitoring tools. Zeek is an open-source network analyzer used for network security monitoring (Zeek, 2021). Zeek analyzes packets and generates logs based on the packet's headers and protocol characteristics. Zeek's protocol analyzers are extremely powerful and will generate a significant amount of useful information about every packet it processes. Suricata is an open-source network intrusion detection/prevention system and network security monitoring engine (Open Information Security Foundation, 2021). In Security Onion 2, the default configuration uses Suricata as an IDS. Suricata generates alerts based on NIDS signatures or will create events based on protocol parsers for packet metadata. Finally, Strelka is a container-based file scanning system used by Security Onion 2 to

scan files extracted from network traffic by Zeek or Suricata and pass event messages from those scans into Security Onion's log storage for analysis (Target, 2021).

The host and network monitoring capabilities are critical in collecting and generating events from a given environment. Those events are only helpful if an organization cannot make sense of the events to drive decisions. Analysts can leverage several capabilities for this use case. The Security Onion Console is the primary interface for analysts to interact with alerts and events. In addition to data interaction capabilities, tools like the ATT&CK Navigator provide basic navigation and annotation of MITRE ATT&CK matrices (MITRE, 2021). Finally, Playbook is a flexible capability for building detection strategies across logs stored in Security Onion (Security Onion, 2021). Analysts can use these three tools together to measure the performance of detections in a network, document threat-related detections or projections, and close detection gaps.

3.3. Implement alert or event generation

With the environment set, the capability in question, Wazuh, must be tested. The Atomic Red Team is a helpful tool to start generating benign and identifiable threat behavior in a lab or production environment. As each Atomic test executes, the operating system will create a variety of network and host artifacts. The Wazuh agent for the test will have default configurations and will collect the Windows logs identified above. The agent will transmit events and alerts to a Wazuh server integrated into the Security Onion system. The Security Onion system passively monitors network traffic and will generate network events and network intrusion detection alerts. This testing infrastructure accurately measures detections by isolating a couple of capabilities and minimizing variables introduced by complex enterprise networks.

Each test is discrete, which makes measurement possible within given time windows. Like production networks, time synchronization is essential to ensure that endpoint and network event generation accuracy. Marking the start time and stop time of each test is extremely important. Those time bounds can be leveraged in Security Onion to focus a large dataset on the test window and make analysis much more manageable.

The Atomic Red Team execution framework for the Windows operating system leverages a PowerShell module run from a command line. Analysts can integrate Atomic Red Team tests with other commands to document time, record output, and save information for analysis. With enough testing and error handling, it may be possible to script an entire test set. The tests conducted for this paper followed the same repeatable steps to ensure measurement and observation of results could be quickly and accurately. Each test used the following steps in the command line:

```
Start-Transcript -File <ex C:\Users\student\Desktop\T1234.md> -Append
Write-Host "# Test T1234"
Write-Host "## Test 1"
Get-Date
Invoke-AtomicTest <Technique ID> -TestNumbers <Test Number>
Write-Host "## Test 2"
Get-Date
Invoke-AtomicTest <Technique ID> -TestNumbers <Test Number>
Stop-Transcript
```

Figure 5: PowerShell command sequence for tests

4. Detection Gap Analysis

An essential part of this process is documenting the results of capability tests. For this paper, the MITRE ATT&CK Navigator is the reference to record and visualize the test results. The ATT&CK Navigator is an open-source web application designed to provide basic navigation and annotation of ATT&CK matrices (MITRE, 2021). Test results will be scored according to the chart in Figure 6. For techniques with multiple tests, the final technique score will be the sum of the test scores divided by the number of tests. Once scoring is complete, the sum will be taken of all the technique scores and divided by the total possible score (33 techniques * 100 = 3300). As depicted in Figure 7, Comments are included in the results

Score	Measurement	Color
0	No Alert or Log	Red
50	Log and No Alert	Amber
100	Log and Alert	Green

Figure 6: Atomic Test Scoring Chart

and are viewed by loading the appropriate JSON files from the paper's Github page listed in Appendix A.

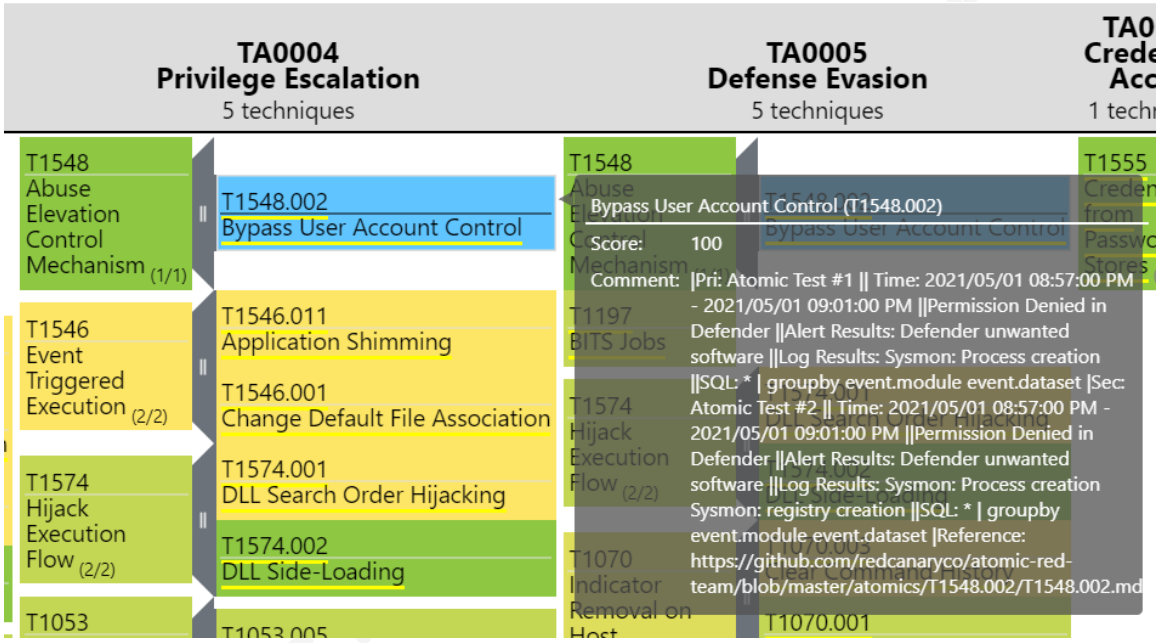


Figure 7: Add Test Results to Technique Comments

4.1. Results without additional detections

Figure 8 shows the results of the first test. After all 33 tests were completed, the final detection score was 61.36%. The score is meaningless without a point of comparison. For now, the score will stand as the base benchmark for the capability on its own. When compared to other capabilities under the same conditions and tests, the score will have more meaning. The overall score will be revisited after designing some detection gaps signatures with the Sigma framework.

Figure 8 displays the individual scoring of all the techniques. 72% of the tests scored 75 points or below points, which means that the combination of Wazuh, Sysmon, and Windows events logs recorded events relevant to the simulated attacks but did not always generate an actionable alert. The gap in meaningful alerts reduced the focus an analyst gains on attack techniques to drive rapid and accurate incident identification. In order to build effective detections, analysts must understand the recorded events from each test.

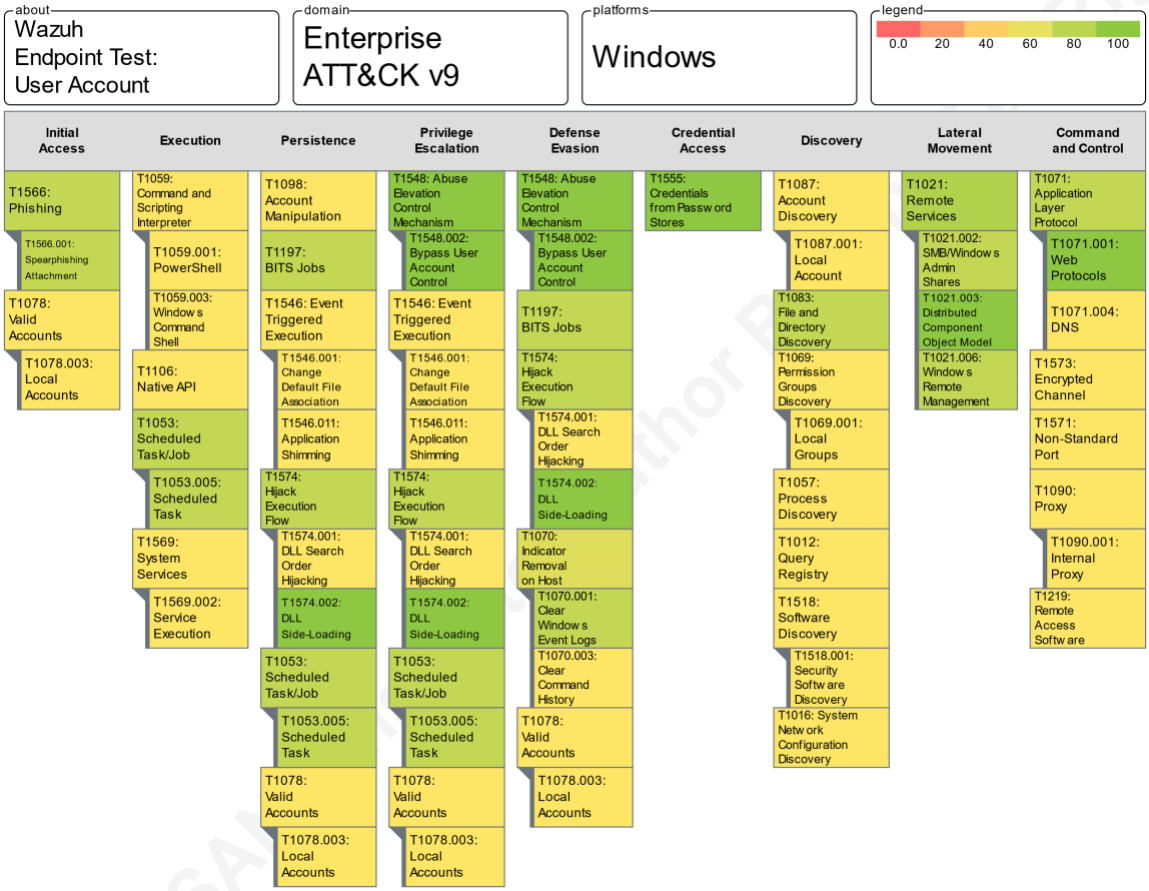


Figure 8: Wazuh Detection Analysis

In Figure 9, technique T1059.003 under the Execution section scored a 50. A score of 50 means that Wazuh recorded events associated with that technique but did not generate alerts. Since the tests were carefully timestamped, we can quickly review the results from that test. The following figure displays the recorded output from the test with the timestamps in bold text.

```
# T1059.003

**********************
Windows PowerShell transcript start
Start time: 20210501161356
Username: DESKTOP-5BHJDMU\student
RunAs User: DESKTOP-5BHJDMU\student
Configuration Name:
Machine: DESKTOP-5BHJDMU (Microsoft Windows NT 10.0.19042.0)
Host Application: powershell.exe
Process ID: 4340
PSVersion: 5.1.19041.906
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.19041.906
BuildVersion: 10.0.19041.906
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
Transcript started, output file is C:\Users\student\Desktop\T1059_003.md

## Test 1

PS C:\Users\student> Get-Date

Saturday, May 1, 2021 4:14:21 PM

PS C:\Users\student> Invoke-AtomicTest T1059.003 -TestNumbers 1 -InputArgs
@{"command_to_execute" = "dir c:\"; "script_path" = "$env:TEMP\T1059_003_script.bat"}
PathToAtomicsFolder = C:\Tools\atomics
Executing test:
T1059.003-1 Create and Execute Batch Script

Done executing test:
T1059.003-1 Create and Execute Batch Script

## Test 2

PS C:\Users\student> Get-Date

Saturday, May 1, 2021 4:14:50 PM

PS C:\Users\student> Invoke-AtomicTest T1059.003 -TestNumbers 2
PathToAtomicsFolder = C:\Tools\atomics
Executing test:
T1059.003-2 Writes text to a file and displays it.

Done executing test:
T1059.003-2 Writes text to a file and displays it.

"Hello from the Windows Command Prompt!"
PS C:\Users\student> stop-transcript
**********************
Windows PowerShell transcript end
End time: 20210501161528
**********************
```

Figure 9: T1059.003 Test Output

Using the Hunt interface in the Security Onion Console, filtering by time focuses analysts on the test events. Figure 10 displays the events create within the time window of the first test. There are 281 events. The Sysmon process creation event is a valuable piece of information that will be used as a pivot point in the data to get to a discrete observable to create a detection around.

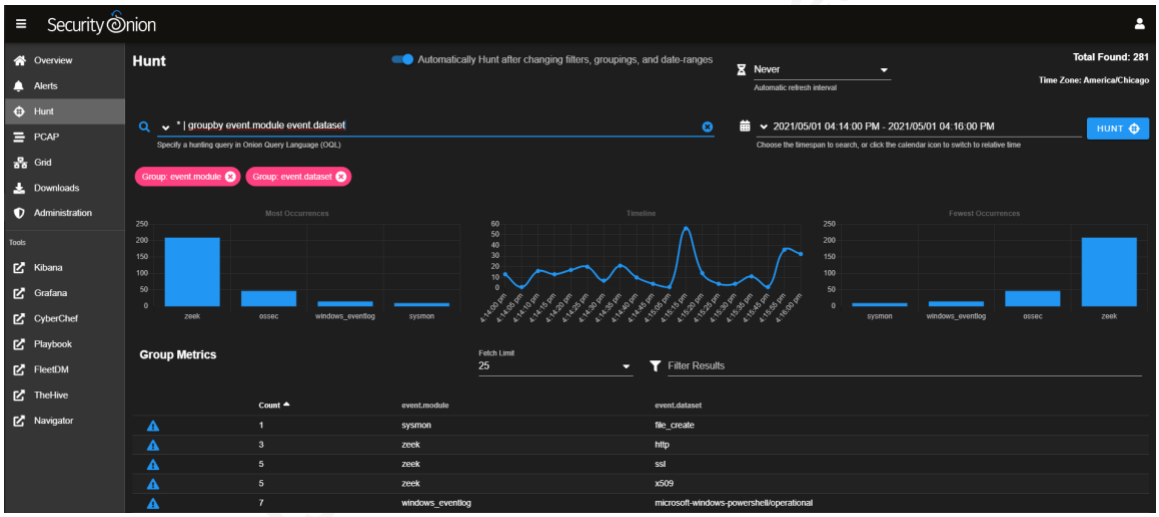


Figure 10: Hunt Interface T1059.003 Test 1

Count ▲	event.module	event.dataset
1	sysmon	file_create
3	zeek	http
5	zeek	ssl
5	zeek	x509
7	windows_eventlog	microsoft-windows-powershell/operational
8	windows_eventlog	windows powershell
9	sysmon	process_creation
19	zeek	file
47	ossec	ossec
88	zeek	dns
89	zeek	conn

Figure 11: Hunt Interface Event Module and Dataset Count

The first test for Technique 1059.003 executes a batch script from a temporary directory on the target endpoint with PowerShell. An analyst would look for PowerShell as the process executable with ".bat" in the command line argument in the process creation events. Figure 12 displays an excerpt of that event.

	\\\\powershell.exe","parentCommandLine":"powershell.exe"}}},"location":"EventChannel"}
process.command_line	\"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe\" & {Start-Process C:\\Users\\student\\AppData\\Local\\Temp\\T1059_003_script.bat}
process.entity_id	{969c0824-c4bc-608d-3c04-000000001500}
process.executable	C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe
process.parent.command_line	powershell.exe
process.parent.entity_id	{969c0824-92a4-608d-4e01-000000001500}
process.parent.executable	C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe
process.pe.company	Microsoft Corporation
process.pe.description	Windows PowerShell
process.pe.file_version	10.0.19041.546 (WinBuild.160101.0800)
process.pe.original_file_name	PowerShell.EXE
process.pe.product	Microsoft® Windows® Operating System
process.ppid	4340
process.working_directory	C:\\Users\\student\\AppData\\Local\\Temp\\
tags	["beats_input_codec_plain_applied"]
user.name	DESKTOP-5BHJDMU\\student
winlog.channel	Microsoft-Windows-Sysmon/Operational
winlog.computer	DESKTOP-5BHJDMU

Figure 12: Sysmon Process Creation Event Excerpt

A detection with the Sigma rule framework can be created from this event, which is covered in the next section.

5. Detection Engineering to Close gaps

Using the knowledge of how Wazuh performs, defenders can leverage other capabilities to close detection gaps and increase the speed of incident identification. It is essential to understand the this is not a "one and done" process. This process is cyclical and should be reviewed regularly by all parties involved in creating and analyzing these detections.

For this environment, we will leverage the Sigma rule framework from Playbook in Security Onion version 2.3. Sigma is a generic signature format used to describe and share SIEM detections (SigmaHQ, 2021). In Security Onion, analysts manage Sigma rules in Playbook, a web application for managing a detection playbook. Each detection playbook consists of plays built on sigma rules and converted to ElastAlert queries through automation (Security Onion, 2021).

An analyst can build a Sigma to detect a batch script run from PowerShell through the technique above. The Sysmon process creation event gives all the critical information needed to target the specific behavior and minimize false positives. The example sigma rule follows in Figure 13 along with the Play generated in Playbook in Figure 14.

```yaml
title: Batch Script Execution
id: atomic-red-team-t1059-003
status: experimental
description: Detects the execution of Batch Script from powershell, which is often used by attackers for command execution
references:
    - https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1059.003/T1059.003.md
author: Peter Di Giorgio
date: 2021/05/30
tags:
    - attack.execution
    - attack.t1059
logsource:
    category: process_creation
detection:
    selection:
        Image: '*\powershell.exe'
    selection2:
        OriginalFileName: 'powershell.exe'
    selection3:
        CommandLine|contains: ".bat"
    condition: selection and selection2 and selection3
falsepositives:
    - Admin activity
    - Scripts and administrative tools used in the monitored environment
level: high
```

Figure 13: Batch Script Sigma Rule

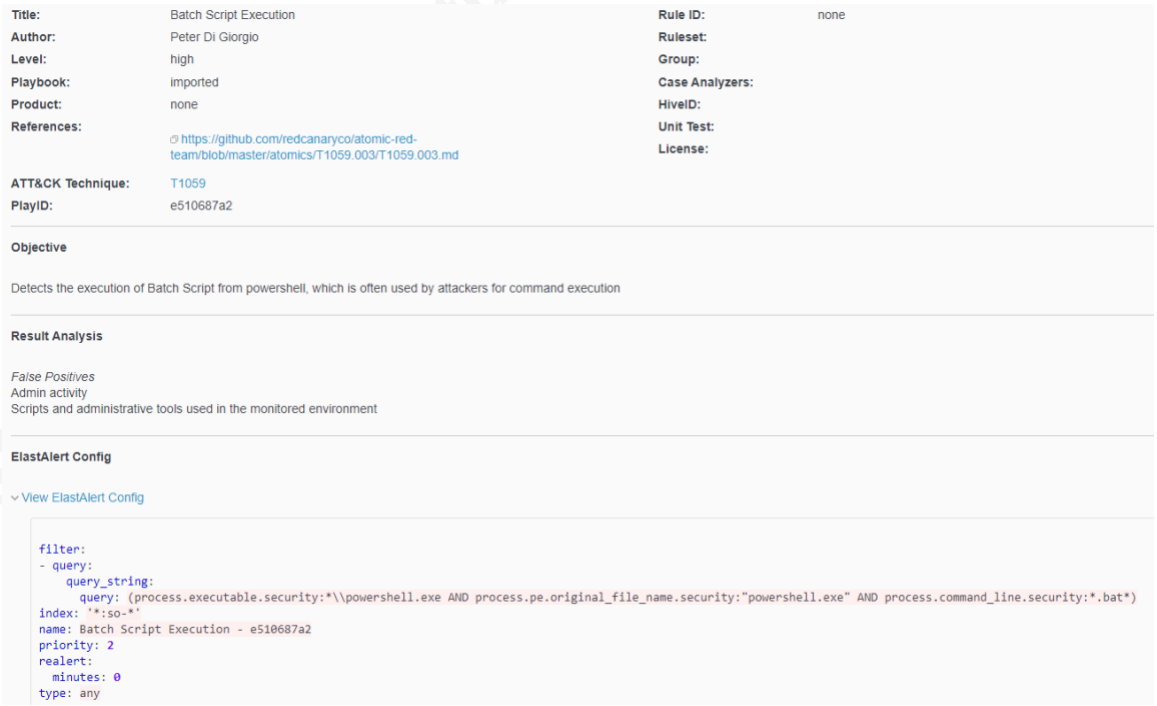


Figure 14: Batch Script Play

Luckily, this work does not have to be done on an individual basis. A growing community of analysts is building and sharing Sigma rules. Security Onion 2.3 already integrates the open-source rule repository hosted on the Sigma Github page. The following section will look at the added coverage gained by the rule written above and the open-source Windows Sigma rules.

5.1. Results with Gap observation in place

Figure 15 shows the detection results with additional detections in place. By integrating Sigma rules through Playbook, Security Onion generated alerts via ElastAlert for potentially malicious activity recorded by log collectors. The result is a 37.12% increase in detection coverage which is significant.

Figure 15: Wazuh Detection Analysis with Sigma Detections

The scoring for the second test used the chart in Figure 6. During this test, the additional opensource Sigma rules and the Batch Script Rule written above successfully generated relevant alerts from endpoint logs collected by Wazuh. The figure below shows the alerts generated for the T1059.003 Atomic Red Team attack. The third alert from the top is the Sigma rule written above. The open-source rules from the Sigma repository generated the remaining rules.

	Count	rule.name	event.module
	2	Local Accounts Discovery	playbook
	2	Whoami Execution	playbook
	1	Batch Script Execution	playbook
	1	System File Execution Location Anomaly	playbook

Figure 16: T1059.003 Playbook Alerts

6. Recommendations and Implications

The tests conducted for this research are simple examples to explore detection gap analysis and detection engineering processes. Depending on the threat, the Atomic Red Team tests won't capture all the tactics for the techniques on the ATT&CK Matrix, which should trigger a risk assessment process or a re-evaluation of threat. Additionally, some of the detections available through open-source rules or written by an organization may result in a high false-positive rate depending on the network. Many of the Atomic Red Team test tactics are very similar to enterprise administrative tasks. The challenge for organizations will be finding the balance between detection coverage and accuracy in relation to its resources.

It may not be feasible, technically or fiscally, for an information security team to achieve 100% detection of a templated threat across an organizational network. Threats regularly evolve, which drives organizations to make decisions about their visibility and resources. Achieving a high detection level may not scale well in robust enterprise networks without a large team of analysts investigating them. By working through the cyclical nature of the detection engineering process, researchers and analysts can explore data types or techniques to increase the accuracy or identification time of detections and share those lessons with the community.

7. Conclusion

The detection gap analysis process was utilized to expose visibility or alerting gaps in a capability. Through a systematic approach, analysts can close detection gaps and confirm the results. Most importantly, defenders can use open-source cybersecurity

tools during the process to test, observe, and document. Analysts can use this process and tools to help organizational leads make informed decisions.

The cyber threat landscape is constantly changing. With new threats manifesting at an alarming rate, threat analysts and cybersecurity engineers struggle to stay ahead of threats and communicate their capabilities and gaps to business leaders. As cybersecurity spending grows, companies are under immense pressure to spend wisely and effectively manage capacity. Detection engineering gives organizations repeatable evaluation processes to deliberately select, on-ramp, and off-ramp capabilities or organizational processes with the backing of objective information.

With the detection engineering process, the cybersecurity industry can defend an information environment throughout its lifecycle with the confidence to communicate what the environment can see and not see. The power of that knowledge enables organizations to defend their networks with purpose. Let the wisdom of a great military philosopher guide the processes to gain an understanding of an environment. Remember three simple phrases: Know Yourself, Know your Terrain, and Know your Adversary.

References

Atkinson, J. (2020, February 21). Detection Spectrum. Retrieved from SpecterOps: https://posts.specterops.io/detection-spectrum-198a0bfb9302

Bace, R. G. (2000). Intrusion Detection. United Kingdom: Macmillan Technical Publishing.

Columbus, L. (2020, August 9). Cybersecurity Spending To Reach $123B In 2020. Retrieved 05 09, 2021, from Forbes: https://www.forbes.com/sites/louiscolumbus/2020/08/09/cybersecurity-spending-to-reach-123b-in-2020/?sh=3022f1b8705f

Day, J. (2020, February 24). So, You Want to Be a Detection Engineer? Retrieved May 9, 2021, from Gigamon Blog: https://blog.gigamon.com/2020/02/24/so-you-want-to-be-a-detection-engineer/

FireEye Mandiant Services. (2021). M-Trends 2021: Special Report. Milpitas: FireEye Mandiant Services.

Lee, R., & Pilkington, M. (2018, December). Hunt Evil Poster. Poster_FOR508_V4.2_12-18. SANS DFIR.

MITRE. (2021, April 27). Enterprise Windows Matrix. Retrieved from MITRE ATT&CK: https://attack.mitre.org/versions/v9/matrices/enterprise/windows/

MITRE. (2021, May 9). Frequently Asked Questions. Retrieved from MITRE ATT&CK: https://attack.mitre.org/resources/faq/

MITRE. (2021, April 27). Navigator 4.3. Retrieved from MITRE ATT&CK: https://mitre-attack.github.io/attack-navigator/

Peter Di Giorgio, peter.digiorgio@sans.student.edu

Red Canary. (2021, May 9). Home Page. Retrieved from Atomic Red Team: https://atomicredteam.io/

Red Canary. (2021, April 12). Invoke-AtomicRedTeam. Retrieved from Github: https://github.com/redcanaryco/invoke-atomicredteam

Russinovich, M., & Garnier, T. (2021, April 21). Sysmon v13.10. Retrieved from Sysinternals: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

Security Onion. (2021). About. Retrieved from Security Onion Documentation: https://docs.securityonion.net/en/2.3/about.html

Security Onion. (2021). ATT&CK Navigator. Retrieved from Security Onion Docs: https://docs.securityonion.net/en/2.3/attack-navigator.html

Tzu, S. (2014). The Art of War. (L. Giles, Trans.) San Diego, CA: Canterbury Classics.

Wazuh. (2021, May 9). Overview. Retrieved from Wazuh Documentation: https://documentation.wazuh.com/current/user-manual/overview.html

Zeek. (2021, May 7). About Zeek. Retrieved from Zeek Documentation: https://docs.zeek.org/en/master/about.html

Appendix A

The data, commands, and tests used for this whitepaper can be found at https://github.com/lock-wire/STI-DetectionEngineering.

Peter Di Giorgio, peter.digiorgio@sans.student.edu