Degree Project in Computer Science and Engineering, specialising in Cybersecurity

Second cycle, 30 credits

Triggering False Alarms in Computer Networks

Evaluation of an Intrusion Detection System’s Performance in Accurately Distinguishing Attacks from Regular Computer Activity

PETER DANIEL

Triggering False Alarms in Computer Networks

Evaluation of an Intrusion Detection System's Performance in Accurately Distinguishing Attacks from Regular Computer Activity

PETER DANIEL

Date: February 13, 2025

Supervisors: Emre Süren, Teodor Sommestad
Examiner: Jan Gulliksen
School of Electrical Engineering and Computer Science
Host company: Swedish Defence Research Agency FOI
Swedish title: Framkallande av Falsklarm i Datornätverk
Swedish subtitle: Bedömning av ett Intrångsdetektionssystems Förmåga att Träffsäkert Särskilja Angrepp från Vanlig Datoraktivitet

© 2025 Peter Daniel

Abstract

A notable trend among malicious actors in the threat landscape is the adoption of `Living-off-the-Land (LotL) binaries` – legitimate system utilities already present in the system – as a defense evasion mechanism. The dual usage of these techniques by administrators and attackers alike have blurred the line between benign and malicious activities, triggering numerous false alarms in `Intrusion Detection Systems (IDS)`, diminishing their efficacy and overwhelming security teams. This thesis investigates the challenges of accurately distinguishing between legitimate and adversarial use of LotL techniques, focusing on how overlapping techniques impact IDS performance. A subset of administrative tasks that resemble adversarial behaviors were automated and simulated in the `Cyber Range And Training Environment (CRATE)`, while the Atomic Red Team framework was used to emulate malicious actions. Testing for detection accuracy was evaluated using signature-based IDS tools, Snort, Sysmon and Wazuh. The findings show that despite using the same binaries and triggering identical signatures, there are subtle behavioral differences between benign and malicious usage of these binaries. Although all the activities generated by the simulations were captured by the security monitoring tools, contextual analysis was required to further distinguish between the two activities. Key recommendations include detection signature refinement and incorporating contextual analysis and anomaly detection to enhance IDS capabilities.

Keywords

Adversary behavior emulation, CRATE, False alarm, IDS, Living-off-the-Land (LotL), Signature-based detection, Snort, Sysmon, System administrator behavior simulation, Wazuh

Sammanfattning

En framträdande trend bland illasinnade aktörer i dagens hotlandskap är användningen av så kallade `Lever-av-landet (LotL)-binärer` – legitima systemkomponenter som redan finns i systemet – som en taktik för att undvika upptäckt. Den dubbla användningen av dessa tekniker, både av systemadministratörer och angripare, har suddat ut gränsen mellan legitima och skadliga aktiviteter. Detta har resulterat i ett stort antal falsklarm i `Intrångsdetektionssystem (IDS)`, vilket minskar deras effektivitet och överbelastar säkerhetsteam. Denna avhandling undersöker de utmaningar som är förknippade med att på ett träffsäkert sätt särskilja mellan legitim och skadlig användning av LotL-tekniker, med särskilt fokus på hur överlappande tekniker påverkar IDS-prestanda. En delmängd av administrativa uppgifter som liknar angripares beteenden automatiserades och simulerades i `cyberanläggningen CRATE`, medan ramverket `Atomic Red Team` användes för att emulera skadliga aktiviteter. Testning av detektionsnoggrannhet utfördes med hjälp av signaturbaserade IDS-verktyg, såsom `Snort`, `Sysmon` och `Wazuh`. Resultaten visar att, trots att samma LotL-tekniker används och identiska signaturer aktiveras, finns det subtila beteendeskillnader mellan legitim och skadlig användning av dessa binärer. Samtliga aktiviteter som genererades i simuleringarna identifierades av säkerhetsövervakningsverktygen, men en kontextuell analys krävdes för att ytterligare särskilja mellan de två typerna av aktiviteter på ett träffsäkert sätt. Väsentliga rekommendationer inkluderar förfining av detektionssignaturer samt implementering av kontextuell analys och anomalidetektion för att förbättra IDS-systemets prestanda och kapacitet.

Nyckelord

Emulering av angriparbeteende, CRATE, Falsklarm, IDS, Lever-av-landet (LotL), Signaturbaserad detektion, Snort, Sysmon, Simulering av systemadministratörsbeteende, Wazuh

Acknowledgments

First and foremost, I would like to express my deepest appreciation and gratitude to my supervisors Teodor Sommestad (FOI) and Emre Süren (KTH) for their valuable guidance throughout the duration of this project. Your insightful feedback and thoughtful suggestions were of utmost importance for the outcome of this study.

I also extend my sincere gratitude to Jonas Hallberg and FOI for the opportunity to conduct my thesis there. I am also grateful to Viktor for providing an overview of CrateBot. Last but not least, I would like to thank all the interview and survey participants for their generosity in sharing their time and insights.

Stockholm, February 2025
Peter Daniel

Contents

1 Introduction 1
1.1 Problem 2
1.1.1 Scientific and engineering issues 3
1.2 Purpose 4
1.3 Sustainability and Ethical issues 5
1.4 Goals 5
1.5 Research Methodology 6
1.6 Delimitations 8
1.7 Structure of the thesis 9

2 Background 11
2.1 Living-off-the-Land Binaries 11
2.2 Threat Detection Methodologies 12
2.2.1 MITRE ATT&CK® 12
2.2.2 LOLBAS Project 15
2.2.3 Sigma Rules 17
2.3 Related work area 20
2.3.1 Major related work 20
2.3.2 Minor related works 23

3 Methodology 25
3.1 Research Process 25
3.2 Qualitative Method 26
3.2.1 Semi-Structured Interview and Online Survey 26
3.2.2 Qualitative Findings 29
3.3 Test Scenarios 37
3.4 Data Collection 41
3.5 Experimental design 41
3.5.1 Test environment 41

4 Implementation **43**
4.1 Testbed Design 43
4.2 Custom Scripts 45
4.3 Configurations 53
4.4 Execution of Test Scenarios 56
4.4.1 Sysadmin Simulation Example 56
4.4.2 Adversary Emulation Example 58

5 Results and Analysis **61**
5.1 Results 61
5.1.1 Detection Efficacy 63
5.1.1.1 Overlap in Signatures 63
5.1.1.2 Sysmon and Wazuh Alerts 66
5.1.1.3 Snort Detection 68
5.2 Analysis 69
5.2.1 Performance Analysis 69
5.2.2 Log Analysis 71
5.2.3 Reliability and Validity Analysis 71

6 Discussion **73**
6.1 Findings 73
6.2 Implications 77
6.3 Methodology Choice 77
6.4 Reflections 77
6.5 Recommendations for Improvement 78
6.6 Atomic Tests and Sigma Contributions 78

7 Conclusions and Future work **81**
7.1 Conclusions 81
7.2 Limitations 82
7.3 Future work 82

References **85**

A Semi-Structured Interview Questions **95**
A.1 Interview: System Administrator 95
A.2 Interview: Log Analysts 102

List of Figures

2.1 A screenshot from the LOLBAS project depicting how attackers can exploit SSH for indirect command execution, along with the detection rule. 15

3.1 Research Process 25
3.2 Living-off-the-Land binaries (LOLBins) used by the sysadmins who participated in the interviews and surveys, with numbers depicting response counts per binary. 32
3.3 Living-off-the-Land binaries (LOLBins) that the log analysts are familiar with or encountered during network monitoring, with numbers representing response counts for each binary. . . 32
3.4 Depicts the process of selecting sysadmin behaviors to simulate in the test cases 37
3.5 ATT&CK techniques selected for testing 38

4.1 Depicts the network mapping in the test environment 44
4.2 Illustrates the system design 45
4.3 Atomic tests added for ssh tunneling and proxying 52
4.4 Shows the summary output of the rule conversion from Sigma to Wazuh 52
4.5 Sysmon configured with Olaf Hartong's template including Event ID 23 - File Delete 53
4.6 Configuring the Wazuh agent to forward sysmon events to Wazuh by adding the following to the ossec.conf file 54
4.7 Configuring the Wazuh agent to forward snort logs and alerts to Wazuh by adding the following to the ossec.conf file 55
4.8 Screenshot of Wireshark installation and network traffic capture 57
4.9 Atomic Test #3 - Msiexec.exe - Execute Local MSI file with an embedded DLL 58

5.1 Sigma-based Wazuh alerts from sysadmin simulation 63
5.2 Sigma-based Wazuh alerts from sysadmin simulation (continued) 64
5.3 Depicts the Wazuh alerts triggered during adversary emulation 65
5.4 Sysmon event for Msiexec quiet install 66
5.5 Wazuh alert visualization for the sysadmin procedure - Port Forwarding Via SSH.exe 67
5.6 Wazuh alert visualization for the atomic test - Port Forwarding Via SSH.exe 67
5.7 Snort detection of PsExec usage 68
5.8 Snort detection of SSH connection 68
5.9 Depicts the Snort rules triggered in Wazuh 69
5.10 Illustrates the encrypted SSH traffic/packets from sysadmin activities as captured in Wireshark 70

List of Tables

2.1 Threat Groups, Windows Binaries, and MITRE ATT&CK Techniques . . . 13
2.2 Threat Groups, Windows Binaries, and MITRE ATT&CK Techniques (continued) . . . 14
2.3 Examples of Sysadmin Command Execution . . . 16
2.4 Examples of Adversary Command Execution . . . 17
2.5 LOLBins and the relevant Sigma Rules they trigger . . . 19

3.1 The Figure illustrates the participants' professional backgrounds, their familiarity with the ten Windows binaries, and the anonymized organizations they belong to . . . 27
3.2 CertUtil-related questions for sysadmins with sample survey responses . . . 29
3.3 CertUtil-Related Questions for Log Analysts with Sample Response . . . 29
3.4 Adversarial usage of living-off-the-land binaries . . . 31
3.5 Administrative usage of living-off-the-land binaries . . . 31
3.6 The overlapping techniques based on practitioners . . . 35
3.7 Sysadmin use case examples based on participants reply . . . 36
3.8 Sysadmin Test Cases and the Sigma Rules they Trigger . . . 39
3.9 Atomic Test Cases. The atomic tests highlighted in blue are newly created . . . 40

4.1 Displays the relevant atomic tests used for adversary emulation along with their original descriptions. . . . 48
4.2 Describes the newly developed atomic tests designed to emulate specific adversarial techniques . . . 49
4.3 Depicts the names of the newly created atomic tests, along with the executors used . . . 49

5.1 Wazuh rules that were triggered by both the sysadmin and adversary activities, along with alarm classification 62

Listings

2.1 Sigma Rule Example for Detecting SSH Proxy Execution . . . 18
4.1 AutoIt script for simulating SSH proxying 46
4.2 T1202.md file 50
4.3 T1202.yaml file 50
4.4 Snort rules for detecting SSH traffic 51
4.5 A sigma based Wazuh rule for detecting SSH Proxying 51
4.6 AutoIt script for quietly installing Wireshark using Msiexec and capturing network traffic with it 56
6.1 Sigma Rule: Suspicious Certutil Decoding 75
6.2 A Wazuh rule for detecting suspicious certutil decoding 76

List of acronyms and abbreviations

ADS	Alternate Data Streams
API	Application Programming Interface
APT	Advanced Persistent Threat
ART	Atomic Red Team
AV	Antivirus
BT	Benign Trigger
C2	Command and Control
CDX	Cyber Defense Exercise
CRATE	Cyber Range And Training Environment
DLL	Dynamic-link library
DMZ	demilitarized zone
EDR	Endpoint Detection and Response
FA	False Alarm
FOI	Swedish Defence Research Agency FOI
FP	False Positive
HIDS	Host-based Intrusion Detection System
IDS	Intrusion Detection System
IOC	Indicators of Compromise
LLM	Large language model
LOLBAS	Living Off The Land Binaries, Scripts and Libraries
LOLBins	Living-off-the-Land binaries
LotL	Living-off-the-Land
LR	Literature Review
MD	Markdown
MDR	Managed Detection and Response
ML	Machine Learning
MSI	Microsoft Software Installer

NIDS	Network-based Intrusion Detection System
NLP	Natural Language Processing
OS	Operating system
RDP	Remote Desktop Protocol
RQ	Research Question
SIEM	Security Information and Event Management
SL	Supervised Learning
SOC	Security Operations Center
sysadmin	System administrator
Sysmon	System Monitor
TA	True Alarm
TTP	Tactics, Techniques and Procedures
UAC	User Account Control
VM	Virtual Machine
WMI	Windows Management Instrumentation
WMIC	WMI command line utility
WmiPrvSe	WMI provider host
Wscript	Windows Script Host

Chapter 1

Introduction

Accurately distinguishing cyberattacks from regular computer activities remains challenging, as attackers often mimic the actions of authorized users and System administrators (sysadmins). A notable trend among threat actors in the current cyberthreat landscape is the adoption of Living-off-the-Land (LotL) tactics, where adversaries leverage native system binaries and utilities to seamlessly blend in with normal system activity and network traffic in order to evade detection [1]. For instance, administration tools such as *PsExec* [2] can be utilized by both system administrators and attackers. Sysadmin use *PsExec* [3] for remote troubleshooting and system configuration, whereas attackers may exploit it for lateral movement within a network [4] [5] [6]. Distinguishing between legitimate and malicious activities becomes arduous, when both sysadmins and attackers are using *PsExec* to jump between machines, but would be easier if only the attacker exhibits this behavior. Given this dual usage, it makes it hard for defenders to discern whether these binaries were used by a sysadmin or an malefactor. False alarms can be as critical as real attacks in such scenarios.

Threat actors nowadays prefer using legitimate, signed binaries already present in the target environment, rather than introducing custom tools to the compromised resource, which poses a significant detection challenge for traditional security solutions [1] [6] [7] [8]. This is particularly problematic for Intrusion Detection System (IDS) and Security Information and Event Management (SIEM) tools, both of which play a crucial role in threat detection and host and network monitoring. IDS focuses on real-time detection of malicious activity by matching known attack patterns (signatures) [9], while SIEM provides a broader analysis by correlating data from multiple sources

[10]. However, they often introduce a major challenge: the overwhelming volume of false positives that they generate [11]. Alahmadi et al. made a distinction between false alarms and benign triggers, defining the former as security alerts generated without an actual threat and the latter as true (legitimate) alarms arising from non-malicious activities that organizations may choose to ignore for business-justified reasons (such as, using vulnerable versions of Java) [12] .

A common way of identifying if there is a perpetrator behind an alarm entails, reaching out to service owners (users) for insights on account activity, reviewing logs for anomalies, evaluating the context of the detected activity using additional data, examining Indicators of Compromise (IOC), and contrasting known legitimate activity with suspicious activity (e.g., by comparing hashes, certificates) to understand the sequence of events [13]. A log analyst uses both context knowledge (knowledge in networking, security and understanding of attacks) and situated knowledge (organizational insight or knowledge of local/operational environment) to be able to distinguish between normal and abnormal activity [9].

1.1 Problem

Living-off-the-Land binaries (LOLBins) are binaries, utilities, processes, or tools native to Operating systems (OSs) - Windows, Linux, or macOS - used for legitimate purposes, but which attackers can exploit on compromised systems. This makes it difficult for network defenders to tell apart their legitimate and malicious use.

LOLBins are often used, on the one hand, by system administrators for legitimate administrative purposes, and on the other hand, by threat actors to perpetrate an attack, making it difficult for even the most experienced log analysts to accurately distinguish between benign executions and a breach, since the binaries leveraged are native to the system, are usually whitelisted, have dual usage and do not require adversaries to deploy scripts on the compromised systems[1].

Seasoned network defenders assess whether a native binary is being used benignly or with nefarious intent by considering the context. This task is made easier when a LotL binary is used for unintended purposes - as that would spell usage by malicious actors, but becomes challenging when used in accordance to its intended standard function - as it might serve either administrative or adversarial objectives. For instance, the Windows utility, *CertUtil*, which is installed as part of Certificate Services [14] [15] [8] and is typically used to manage digital certificates, has other functionalities (such as encoding/decoding and downloading files), and thus can be repurposed to download and install malicious payloads. Detecting LOLBins through pattern matching alone can lead to many false positives due to their dual-use nature. Moreover, IDS – being trigger happy – frequently face the significant issue of producing a large number of False Positives (FPs) [11] [12]. Therefore, the question arises: Which sysadmin tasks can create False Alarm (FA) and is there a difference between legitimate and malicious LotL activity?

1.1.1 Scientific and engineering issues

LotL allow malefactors to camouflage abnormal activity as normal system behavior, making it easier to bypass standard security measures and operate unnoticed. Moreover, the absence of efficient security practices (such as least privilege, baselining, fine-tuning, hardening, and monitoring LOLBins usage) and conventional IOC as well as reliance on untuned Endpoint Detection and Response (EDR) systems makes detecting nefarious LotL activities challenging [6].

This study is based on the hypothesis that “there is a distinction in behavior between a legitimate system administrator and an adversary when using the same LotL binaries (LOLBins), despite generating similar security alerts and log entries.” Given the dual use of LotL binaries, defenders often face the challenge of differentiating between benign and adversarial activities. This leads me to my first Research Question (RQ):

What are some of the most common LotL binaries used by both system administrators and threat actors?

Identifying the LotL binaries that threat groups have in their arsenal which also have administrative usage lays the foundation to further examine how they are leveraged by both administrators and adversaries. The paper further explores the behavioral differences and similarities in their usage. Attackers often mimic legitimate activities to evade detection, but subtle differences in behavior could provide valuable indicators. This brings me to my second research question:

What are the overlaps and differences in the administrative and adversarial usage of LotL binaries? What types of false alarms are triggered due to these overlapping techniques?

Addressing these overlaps is critical, as the false alarms triggered by overlapping usage patterns can overwhelm security teams and make detection systems less effective. Security tools such as IDS and SIEM platforms play a central role in distinguishing legitimate from malicious activities as they are often the last line of defense. Nevertheless, their effectiveness against LotL binaries remains uncertain. Therefore, I raise my third research question:

How effective are signatures in accurately differentiating between legitimate system administrator activities and malicious attacks, given the use of the same LotL binaries?

By investigating these questions, the study seeks to enhance the understanding of how the usage of the overlapping LotL techniques impacts detection strategies and to identify ways to improve existing security measures by pinpointing IOC related to LotL activities.

1.2 Purpose

The purpose of this study is twofold. First, it aims to contribute to academic knowledge by demonstrating that, despite similar security alerts and log entries, there are discernible distinctions between the actions/behaviors of legitimate users (e.g., sysadmins) and digital assailants (adversaries) when utilizing the same LotL binaries. The study will explore where legitimate system administrator activities intersect with malicious adversary behaviors when using LOLBins. By simulating system administrator actions that closely mimic malicious behaviors, the research seeks to identify weaknesses in IDS with default configurations and propose improvements to reduce FPs and enhance detection accuracy. This insight will provide valuable guidance for future researchers and engineers in designing more precise security systems. It will be of particular interest to cybersecurity professionals, including sysadmins, Security Operations Center (SOC) analysts, and IDS developers, who are directly involved in threat detection and network security as it aims to assess the efficacy of IDS/SIEM in distinguishing between benign and malicious activities.

Secondly, the degree project being conducted in collaboration with the Swedish Defence Research Agency FOI (FOI) (i.e., the problem owner), the simulations of sysadmin behavior and test scenarios will be implemented in the cyberrange Cyber Range And Training Environment (CRATE) [16]. Furthermore, the findings will be integrated into FOI's Cyber Defense Exercises (CDXs), contributing to FOI's detection system testing as well as their efforts to strengthen national defense capabilities within the cyber domain and provide a platform for advanced practical training for personnel defending critical systems [16].

On a personal level, this degree project will also demonstrate my ability to conduct independent research that addresses real-world problems while contributing to the advancement of scientific knowledge.

1.3 Sustainability and Ethical issues

The ethical issues that this project could raise are in regards to the collection of information from interview subjects. These concerns can be mitigated by guaranteeing informed consent, which means that the participants are thoroughly briefed on the study's nature and objectives before they agree to participate. Participation must be completely voluntary, with the ability to withdraw at any time without any repercussions. All data gathered from interviews, surveys and tests should be anonymized to ensure the protection of participants' identities, and strict protocols should be in place to maintain privacy and confidentiality. The project must adhere to GDPR and other relevant regulations to protect participants' data.

1.4 Goals

The goal of the project is to evaluate how effectively IDS can differentiate between benign and malicious LotL activities. To satisfy the needs of both the engineering community and academia, as well as the problem owner (FOI), the following sub-goals must be fulfilled:

1. Identifying Windows binaries and LotL techniques common to both system administrators and threat actors
2. Determining the specific sysadmin behaviors to simulate for triggering false alarms
3. Simulating sysadmin behaviors in the test bed CRATE to trigger false alarm
4. Assessing the effectiveness of the IDS by evaluating the results of these test scenarios

1.5 Research Methodology

In order to examine the assumption that despite generating similar security alerts and log entries, there is a distinction in behavior between a legitimate system administrator and a threat actor when using the same LotL binaries, the research adopts a mixed-methods approach, combining Literature Review (LR), qualitative research (semi-structured interviews and surveys) [17], and empirical testing. These methodologies were chosen to provide both theoretical depth and practical relevance, ensuring the research comprehensively addresses the posed research questions.

1. Literature Review (to answer RQ 1 & 2):

   The LR involved examining existing literature and analyzing IDSs and SIEM signatures to identify the intersection between legitimate administrative use, and repurposed adversarial use of common LotL binaries. Some of the key resources used were the LOLBAS Project [18] which offers a curated list of Windows binaries, their unintended use (unexpected functionality), and triggered signature rules; the MITRE ATT&CK Framework [19], providing an extensive taxonomy for adversary behavior and threat modeling; and the Sigma Rule Repository [20], which highlights the false positives triggered by the LotL techniques. This culminated in a gross list of LOLBins and techniques that are prone to triggering false alarms.

2. Qualitative Method (to answer RQ 1 & 2):

   Semi-structured interviews and surveys [17] — where the surveys mirrored the interview questions but offered a more convenient option for participants with busy schedules — were conducted with industry experts. The participants selected were chosen based on their expertise and background in working as either sysadmins in a Windows environment or as log analysts in a SOC. Moreover, to ensure that the sysadmin procedures used were not solely reflective of organizational culture as well as to introduce diverse perspectives, participants were chosen from four different organizations. The goal was to refine the initial list of identified LotL binaries and techniques by establishing a more precise net list of legitimate sysadmin activities. The LOLBins included in the interviews and surveys, which were chosen based on the LR were: PsExec [2], msiexec [21], ssh [22], wmic [23], winget [24], wbadmin [25], certutil [14], ntdsutil [26], cmd [27] and cmdkey [28]. This qualitative approach focused on the nuanced use of LOLBins in legitimate contexts that may overlap with adversarial use. Participants were asked to identify scenarios where repurposed LotL functions serve legitimate administrative purposes, as well as which standard (intended) functionalities they regularly use or have observed being used by other sysadmins. Consulting sysadmins to identify which LotL techniques are commonly used in legitimate administrative tasks helps ensure that the test cases accurately reflect sysadmin behaviors rather than emulating adversarial actions. The findings from these consultations will serve as the basis for selecting the specific sysadmin behaviors to be simulated during the testing phase.

3. Empirical Evaluation (to answer RQ 3):

   Empirical testing will be conducted in a simulated environment — the CRATE cyberrange — to validate the findings from the literature review, interviews and surveys. This approach allows for controlled replication of both sysadmin and adversary behaviors, testing the detection mechanisms under near real-world conditions. Alongside custom scripts and AutoIT [29] to simulate legitimate sysadmin procedures, the adversary emulation framework, Atomic Red Team (ART) [30], will be employed to simulate malicious activities. These simulations aim to trigger FAs based on sigma-signatures, as only the overlapping techniques that can induce FA are part of the test scenarios.

The justification for choosing these research methodologies is as follows: LR was selected to provide an understanding of the administrative and adversarial use of LotL techniques based on existing knowledge. The qualitative research complements the LR by adding real-world perspectives to theoretical findings. This method allows for an exploration of the reasoning and context behind sysadmin behaviors, offering insights into the ”why” behind the actions. This is crucial for distinguishing between similar actions performed by sysadmins and attackers, which may trigger identical security alerts. The flexibility of semi-structured interviews and descriptive qualitative surveys also enables the discovery of unforeseen insights that could be missed in empirical testing alone. A purely quantitative survey would not capture the depth of understanding required for this research, as it would focus on frequency rather than the reasoning behind actions. Ethnographic methods were not considered due to time constraints and the difficulty of observing sysadmins in a live environment. Empirical testing was chosen as it offers a clear method to validate the distinctions between sysadmin and adversarial behavior. This method allows for direct observation of how detection systems respond to both legitimate and malicious actions, offering a way to identify false positives and assess how effectively these systems differentiate between the two. Testing in a simulated environment (like the one available at FOI) offers a controlled setting where administrative and adversarial behaviors can be safely and repeatedly analyzed without disrupting operational systems. A real-world testing approach was rejected due to the risk of operational impact, and a purely theoretical approach was avoided as it would limit the practical applicability of the research.

1.6 Delimitations

This project imposes several limitations to maintain focus and ensure feasibility. In regards to LotL utilities, the paper does not cover binaries that are native to non-Windows operating systems, such as Linux binaries (GTFOBins) [31], macOS binaries (LOOBins) [32], nor does it examines cloud binaries [33] [34] or Windows drivers (LOLDrivers) [35]. In addition, it does not seek to develop an IDS capable of addressing all conceivable false positives, but will instead test IDS against standard configurations. The study will emulate selected sysadmin behaviors resembling adversarial techniques, without providing an exhaustive list of techniques. In addition, the scope of this research prioritizes sysadmin activities over regular user activities. Furthermore, the project is constrained by the licenses and tools available within FOI's cyberrange. For instance, the absence of cloud solutions in CRATE prevents the testing of cloud-based scenarios. Finally, the test cases will be conducted exclusively on Windows machines.

1.7 Structure of the thesis

Chapter 2 presents relevant background information on Living-off-the-Land binaries, threat intelligence and detection frameworks (namely, MITRE ATT&CK [19], the Living Off The Land Binaries, Scripts and Libraries (LOLBAS) Project [18], and Sigma Rule Repository [20].) as well as related work. Chapter 3 presents the methodology and method used to address the problem. Chapter 4 details the implementation phase, while Chapter 5 focuses on presenting and analyzing the results. Last but not least, Chapters 6 and 7 discuss the findings, draw conclusions, and explore potential directions for future work.

Chapter 2

Background

This chapter begins with an overview of LOLBins and essential tools and frameworks for adversarial behavior mapping and detection, including MITRE ATT&CK [19], the LOLBAS Project [18], and Sigma Rule Repository [20]. These tools were used to examine the questions: *"What are some of the most common LotL binaries used by both sysadmins and threat actors?"* and *"What are the overlaps and differences in the administrative and adversarial usage of LotL binaries?"* The chapter then discusses prior research and related work.

2.1 Living-off-the-Land Binaries

Intruders and Intrusion Detection Systems are engaged in a hide-and-seek game, where the former needs to constantly innovate to evade detection, while the latter has to update frequently to uncover these stealthy attacks. As shown in several earlier works [1], one such evasion tactic is the use of LotL, which has become prevalent among threat actors and malware authors.

The term `living off the land` was first introduced to the cybersecurity community in 2013 by Christopher Campbell and Matt Graeber at Derbycon 3.0 [36]. The expression originates from survival by only consuming nature's provisions (i.e. eating food available on the land, obtained by harvesting, hunting, etc.). In the context of cybersecurity, this means the use or/and abuse of native/preexisting (built-in or already deployed) system binaries, processes, and legitimate administrative utilities to execute attacks that disguise malicious activity as normal behavior [6] [37]. It has gained popularity among threat actors, because on the one hand, it allows them to evade detection by seamlessly blending into the target environment's legitimate activities and normal traffic, and on the other hand, it allows them to leverage already deployed trusted tools rather than creating custom utilities [6]. Because these dual-use tools are pervasive and frequently utilized by sysadmins for valid administrative tasks, it makes it challenging for network defenders to fully restrict access to these binaries, enabling adversaries to remain unnoticed and hide in plain sight.

Malicious attackers exploit these binaries across various operating systems (i.e., Windows, Linux, macOS) and infrastructures (i.e., On-Premise, Cloud, Hybrid) to perform post-exploitation activities such as reconnaissance, enumeration, persistence, data exfiltration, and payload execution [1]. Among the earliest observed Advanced Persistent Threat (APT) instances that utilized LotL tactics in the wild was the threat group *APT29* [38] (associated with Russia's Foreign Intelligence Service) that used *POSHSPY* [39], a stealthy backdoor that leveraged two of windows built-in utilities: *Powershell* [40] (for payload execution) and *Windows Management Instrumentation (WMI)* [41] (for storage and persistence) [42].

Various projects such as LOLBAS [18], GTFOBins [31], LOOBins [32], and LOLDrivers [35] have cataloged numerous exploitable LotL across different OSs (including about 207 on Windows, 390 on Linux, 53 on macOS, and 427 Windows drivers).

2.2 Threat Detection Methodologies

2.2.1 MITRE ATT&CK®

MITRE ATT&CK® is an extensive framework used for understanding adversary behavior and threat modeling, grounded in real-world observations [43]. This model enumerates the behavior of threat actors in terms of Tactics, Techniques and Proceduress (TTPs), where a tactic denotes a goal, a technique indicates the means to attain that goal and a procedure refers to the specific implementation used [44]. The framework currently encompasses 14 adversarial tactics and 202 techniques [45], mapping these behaviors to the APT groups that are known to employ them. Moreover, it offers mitigation strategies to prevent the successful execution of said techniques. For example, adversaries such as the APT41 group [46], may employ password cracking in order to gain unauthorized access into a system, a technique the ATT&CK

framework suggests mitigating through Multi-factor Authentication [47].

In this study, the framework was used primarily to investigate the common LotL binaries leveraged by APT groups as shown in Tables 2.1 and 2.2,, as well as to identify sysadmin and adversary LotL techniques and procedures.

Table 2.1: Threat Groups, Windows Binaries, and MITRE ATT&CK Techniques

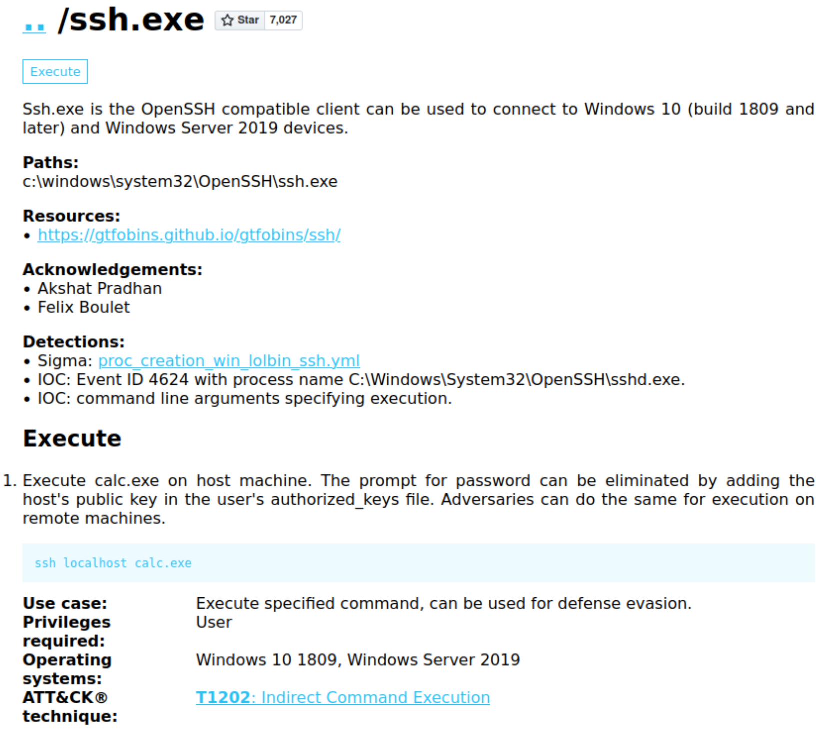
Threat Group	**Windows Binaries**	**Adversarial Goal**	**ATT&CK Techniques**
APT3	cmd.exe, powershell.exe, rundll32.exe, schtasks.exe	Credential Access, Execution, Persistence	T1059.003, T1218.011, T1053.005, T1003
APT29	cmd.exe, powershell.exe, PsExec.exe, schtasks.exe, wmic.exe	Exfiltration, Lateral Movement, Persistence	T1059.003, T1053.005, T1105, T1047, T1071.001
APT33	net.exe, powershell.exe, ProcDump.exe, schtasks.exe, vbscript	Credential Access, Discovery, Exfiltration, Lateral Movement	T1003, T1083, T1059.005, T1071.001, T1053.005
Cleaver	certutil.exe, mshta.exe, net.exe, schtasks.exe, powershell.exe, PsExec.exe	Command and Control (C2), Discovery, Execution, Exfiltration, Persistence	T1588.002, T1218.005, T1071, T1059.001, T1105
FIN5	PsExec.exe, Pwdump, RDP	Credential Access, Lateral Movement, Persistence	T1569.002, T1071.001, T1003, T1021.001, T1588.002

Table 2.2: Threat Groups, Windows Binaries, and MITRE ATT&CK Techniques (continued)

Threat Group	**Windows Binaries**	**Adversarial Goal**	**ATT&CK Techniques**
FIN13	certutil.exe, SSH, wmic.exe	Defense Evasion, Lateral Movement	T1071.001, T1047, T1570, T1105
Gallmaker	powershell.exe, winword.exe	Execution, Persistence	T1059.001, T1204.002
Hexane	bitsadmin.exe, cmdkey.exe, net.exe	Credential Access, Discovery	T1204.003, T1083, T1003
Lazarus Group	mshta.exe, net.exe, regsvr32.exe, SSH	Discovery, Execution, Lateral Movement	T1218.005, T1071.001, T1218.010, T1570
menuPass	certutil.exe, cmd.exe, net.exe, PsExec.exe	Discovery, Exfiltration, Lateral Movement	T1140, T1071.001, T1569.002, T1047, T1105
Sandworm Team	net.exe, ntdsutil.exe, PsExec.exe, rundll32.exe, wbadmin.exe	Credential Access, Discovery, Execution, Impact, Lateral Movement	T1003.003, T1083, T1218.011, T1486, T1105
TA505	cmd.exe, msiexec.exe, net.exe, rundll32.exe, powershell.exe	Command and Control (C2), Discovery, Execution, Exfiltration, Reconnaissance	T1071.001, T1083, T1059.003, T1053.005, T1218.011
Thrip	powershell.exe, PsExec.exe	Execution, Lateral Movement, Reconnaissance	T1059.001, T1570, T1071.001, T1047
Turla	certutil.exe, net.exe, powershell.exe, PsExec.exe, tasklist.exe, wmic.exe, wscript.exe	Command and Control (C2), Discovery, Execution, Exfiltration	T1140, T1071.001, T1059.001, T1569.002, T1057, T1083, T1105, T1218.011

2.2.2 LOLBAS Project

LOLBAS is a curated documentation of LotL binaries, scripts, and libraries that can be leveraged by malefactors for nefarious purposes. The project aims to catalogue these LotL techniques to raise awareness and improve detection strategies. The criteria for inclusion in the project are that the binaries, libraries, and scripts should either be Microsoft-signed files that are pre-installed on the system or installable via trusted vendors. Furthermore, they should provide unintended functionality that attackers might exploit, for instance, hiding data in Alternate Data Streams (ADS), file operations (downloading/uploading files), code executions, User Account Control (UAC) bypass, and more [48]. The project offers ATT&CK® mappings for each LotL binary, script, and library, along with triggered signatures and potential IOCs to monitor. Figure 2.1 illustrates an example using SSH [49]. Tables 3.4 & 3.5 portrays the malicious and legitimate use of LotL techniques for the investigated LOLBins.



Ssh.exe is the OpenSSH compatible client can be used to connect to Windows 10 (build 1809 and later) and Windows Server 2019 devices.

Paths:
c:\windows\system32\OpenSSH\ssh.exe

Resources:
- https://gtfobins.github.io/gtfobins/ssh/

Acknowledgements:
- Akshat Pradhan
- Felix Boulet

Detections:
- Sigma: proc_creation_win_lolbin_ssh.yml
- IOC: Event ID 4624 with process name C:\Windows\System32\OpenSSH\sshd.exe.
- IOC: command line arguments specifying execution.

Execute

1. Execute calc.exe on host machine. The prompt for password can be eliminated by adding the host's public key in the user's authorized_keys file. Adversaries can do the same for execution on remote machines.

```
ssh localhost calc.exe
```

Use case:	Execute specified command, can be used for defense evasion.
Privileges required:	User
Operating systems:	Windows 10 1809, Windows Server 2019
ATT&CK® technique:	T1202: Indirect Command Execution

Figure 2.1: A screenshot from the LOLBAS project depicting how attackers can exploit SSH for indirect command execution, along with the detection rule.

Examples of sysadmin and adversarial command executions for the LOLBins that the interviewees and survey respondents use are shown in Tables 2.3 & 2.4. Some of the malicious command-line examples are based on incidents. For instance, the *PsExec* command execution was based on Volt Typhoon, a state-sponsored actor [50] [51] [6].

Table 2.3: Examples of Sysadmin Command Execution

LOLBin	**Sysadmin Procedure**	**Description**
PsExec	`psexec \\192.168.83.102 -s -i -u Administrator systeminfo`	System information retrieval from a remote host
Msiexec	`msiexec /i "C:\Admin\Downloads\Firefox.msi /qb! /norestart"`	Installs Firefox silently without user interaction
SSH	`ssh administrator@192.168.83.102`	Establishes a secure shell connection to the remote host
Wmic	`wmic product where "name='-Zip'" call uninstall`	Uninstalls a product named ’-Zip’
Winget	`winget install Microsoft.VisualStudioCode`	Installs Visual Studio Code from the package manager
Certutil	`certutil -store root`	Retrieves certificate information from the root store

Table 2.4: Examples of Adversary Command Execution

LOLBin	**Adversary Procedure**	**Description**
PsExec	`psexec \\192.168.83.102 -s cmd /c "cmd.exe /c netsh interface portproxy delete v4tov4 listenaddress=0.0.0.0 listenport=9999"`	Attempts to remove port proxy configurations on a remote host
Msiexec	`msiexec /y "C:\folder\evil.dll"`	Executes malicious DLL from a specified path
SSH	`ssh -o ProxyCommand=regedit.exe .`	Abuses SSH ProxyCommand for indirect command execution and defense evasion
Wmic	`wmic.exe process call create malicious.exe`	Creates a process to launch a malicious executable on the remote machine
Winget	`winget.exe install -manifest manifest.yml`	Installs a potentially malicious application from a manifest file
Certutil	`certutil.exe -urlcache -split -f http://7-zip.org/a/7z1604-x64.exe "C:\Users\Admin\Desktop\7zip.exe"`	Downloads and executes a file from a remote URL

2.2.3 Sigma Rules

Sigma is a generic detection format (language) for SIEMs and a signature database for detecting attacks, with over 3000 threat detection rules in its repository [20]. The repository offers three types of rules: *Generic Detection* for behavior detection, *Threat Hunting* for analyst guidance, and *Emerging Threats* for timely and specific threat coverage (such as Zero-Day exploits and APT campaigns) [20]. Sigma's rule format is adaptable, shareable, simple to compose, suitable for any log file type, as well as vendor agnostic and supports various OSs (such as Windows, Linux, and macOS) and log sources (such as application, cloud, and web), enabling the exchange of known or newly identified adversarial techniques across different platforms. As illustrated in Listing 2.1 with the ssh proxy execution rule, this thesis employed Sigma rules to identify relevant LotL techniques that could be used to simulate false alarms. Table 2.5 lists the relevant sigma rules triggered by some of the LOLBins included in the interviews and surveys.

Listing 2.1: Sigma Rule Example for Detecting SSH Proxy Execution

```

title: Program Executed Using Proxy/Local Command Via SSH.EXE
id: 7d6d30b8-5b91-4b90-a891-46ccaf29598
status: test
description: Detect usage of the ”ssh.exe” binary as a proxy to launch
    ↪ other programs.
references:
  - https://lolbas-project.github.io/lolbas/Binaries/Ssh/
  - https://github.com/LOLBAS-Project/LOLBAS/pull/211/files
  - https://gtfobins.github.io/gtfobins/ssh/
  - https://man.openbsd.org/ssh_config#ProxyCommand
  - https://man.openbsd.org/ssh_config#LocalCommand
author: frack113, Nasreddine Bencherchali
date: 2022-12-29
modified: 2023-01-25
tags:
  - attack.defense-evasion
  - attack.t1218
logsource:
  category: process_creation
  product: windows
detection:
  selection_parent:
    # ParentCommandLine: '”C:\Windows\System32\OpenSSH\sshd.exe” -R'
    ParentImage: 'C:\Windows\System32\OpenSSH\sshd.exe'
  selection_cli_img:
    Image|endswith: '\ssh.exe'
  selection_cli_flags:
    - CommandLine|contains: 'ProxyCommand='
    - CommandLine|contains|all:
        - 'PermitLocalCommand'
        - 'LocalCommand'
condition: selection_parent or all of selection_cli_*
falsepositives:
  - Legitimate usage for administration purposes
level: medium
```

Table 2.5: LOLBins and the relevant Sigma Rules they trigger

TOOLS	SIGMA RULES
PsExec	proc_creation_win_renamed_sysinternals_psexec_service
	proc_creation_win_sysinternals_psexec_execution
	proc_creation_win_sysinternals_psexec_paexec_escalate
	proc_creation_win_sysinternals_psexec_remote_execution
	proc_creation_win_sysinternals_susp_psexec_paexec_flags
	proc_creation_win_renamed_paexec
Msiexec	proc_creation_win_msiexec_web_install
	proc_creation_win_msiexec_masquerading
SSH	proc_creation_win_lolbin_ssh
	proc_creation_win_ssh_port_forward
	proc_creation_win_ssh_rdp_tunneling
Wmic	image_load_wmic_remote_xsl_scripting_dlls
	proc_creation_win_wmic_xsl_script_processing
	proc_creation_win_wmic_squiblytwo_bypass
	proc_creation_win_wmic_eventconsumer_creation
	proc_creation_win_wmic_uninstall_security_products
Winget	proc_creation_win_winget_local_install_via_manifest
Wbadmin	proc_creation_win_wbadmin_delete_all_backups
	proc_creation_win_wbadmin_delete_backups
	proc_creation_win_correlation_multiple_susp_cli
	proc_creation_win_susp_shadow_copies_deletion
Certutil	proc_creation_win_certutil_download
	proc_creation_win_certutil_encode
	proc_creation_win_certutil_decode
Cmd	proc_creation_win_susp_alternate_data_streams
	proc_creation_win_cmd_assoc_execution
	proc_creation_win_cmd_del_greedy_deletion
	proc_creation_win_cmd_curl_download_exec_combo
	proc_creation_win_cmd_del_execution
	proc_creation_win_cmd_dir_execution
	proc_creation_win_cmd_ping_del_combined_execution
	proc_creation_win_cmd_rmdir_execution

2.3 Related work area

2.3.1 Major related work

This paper makes extensive use of MITRE ATT&CK framework [19], the LOLBAS Project [18], and Sigma [20]. A pertinent study that also leverages the ATT&CK framework [19] is provided by Barr-Smith et al. in [1], where the authors investigated the prevalence and purpose of malware using Windows binaries. They analyze these binaries' evasive capabilities, the malware families involved, and the distinctions in behavior between legitimate and malicious uses, as well as the effectiveness of Antivirus (AV) solutions against malicious use in Windows environments. In contrast, this paper aims to examine the overlap in behavior between administrative and adversarial use of LOLBins, focusing on assessing the ability of IDS and SIEM systems (rather than AV engines) to accurately differentiate between legitimate administrative activities (that resemble adversarial behavior) and intrusions.

Barr-Smith et al. [1] observed a 9.41% prevalence of LotL techniques across general commodity malware datasets, with a notably higher prevalence of 26.26% in APT malware datasets. While their measurements did not distinguish between conventional and unconventional usage of LotL techniques, this thesis aims to explore both standard uses (e.g., *Netsh.exe* modifying firewall rules) and repurposed uses (e.g., *Netsh.exe* executing .dll files) of these system components [52] [53] [1], but with the emphasis on identifying the overlapping techniques. Moreover, similar to [1], this study will focus on Windows binaries, because Windows not only dominates the desktop operating system market [54], making it a frequent attack target, but also its binaries can be simulated in CRATE.

Another related work is [12], with an emphasis on the occurrence of false alarms generated by detection tools, such as IDS. Alahmadi et al. highlighted how false positives affect efficiency, the decision-making processes analysts use to filter true threats from noise, and how enhancing tool design, workflows, and alert accuracy (precision) can improve SOC environments. By conducting qualitative interviews and surveys, Alahmadi et al. explore the challenges SOC analysts face with alarm fatigue, caused by the sheer volume of false alerts. This insight is essential to this research, as it involves deliberately triggering false alarms to simulate how personnel training in FOI's CDX might experience such challenges in real-world scenarios. Alahmadi et al. critiqued the term FP for being broad and vague and giving the impression that the security tools themselves are flawed (as they tend to generate an overwhelming number of FPs), suggesting instead using the metrics Benign Trigger (BT), True Alarm (TA), and FA. This study will use these metrics for alarm validation with a slight definition variation: BT will denote instances where conditions are met, but the circumstances represent legitimate administrative use, rather than a TA ignored for business-justified reasons. Furthermore, Luo et al. addressed the complexities of generating realistic traffic that represent authentic user activity and network protocols as a means of evaluating security tools such as IDS and firewalls [11].

Other relevant works draw on industry insights, particularly annual or quarterly reports from incidents investigated by Managed Detection and Response (MDR) services and Endpoint Protection Platforms. These reports [8] [55] reveal some of the LotL tools frequently exploited by cybercriminals to target the monitored enterprises.

When it comes to RQ1, the prevalence of LOLBins varies widely across the cyberthreat landscape, as different APTs seem to favor different system functions based on their campaign tactics and execution purposes (as seen in Tables 2.1 & 2.2). According to a 2023 DFIR report [56], frequently exploited tools in 2022 included: `PowerShell` and `Windows command shell` (for download and execution), `WMI` (for enumeration, lateral movement, persistence, and ransomware propagation), `rundll32` and `regsvr32` (for DLL execution), `ipconfig`, `net`, `systeminfo`, `ping`, and `nslookup` (for discovery), and `PsExec` (for lateral movement and ransomware propagation).

Barr-Smith et al. [1] found significant differences in the purposes of LOLBins execution between legitimate and nefarious scenarios. In malware, the primary execution purposes include reconnaissance, persistence, registry modification, and proxied execution. Their study noted that proxied execution was the most prevalent LotL technique utilized in both benign and malicious activities. According to their findings, `explorer`, `regsvr32`, `sc`, `rundll32`, `taskkill`, `ping`, `net`, `mstsc`, `attrib`, and `regedit` were among the most leveraged binaries in benign datasets. In contrast, `ping`, `rundll32`, `reg`, `wscript`, `xcopy`, `net`, `tasklist`, `ipconfig`, `expand` and `systeminfo` were the most frequently exploited binaries in APT datasets.

CrowdStrike [8] on the other hand, observed in the intrusions that they were engaged in between the period July 2021 and June 2022, that `Rundll32`, `Regsvr32`, `Msiexec`, `Mshta`, `Certutil`, `MSBuild`, `WMI command line utility (WMIC)`, and the `WMI provider host (WmiPrvSe)` were the most frequently leveraged utilities [8]. In addition, Kaspersky's Report [55] noted that, `PowerShell` [40] accounted for 3.3% of all incidents they detected, notably contributing to 20.3% of critical incidents. The next most prevalent executable, `rundll32` (used for loading and running 32-bit Dynamic-link library (DLL)) [57] [58], played a role in 2% of all incidents and 5.1% of critical cases. Further analysis by Kaspersky's MDR team identified other frequently used binaries, such as: `PsExec` (used for running processes in remote services) [2], `CertUtil` (used for handling certificates) [14] [59] [15], `Reg` (used for running operations on the registry) [60][61], `te.exe` (used for executing test binaries) [62][63], and `Windows Script Host (Wscript)` (used for executing scripts) [64][65] — were collectively involved in 1.9% of all incidents and were responsible for 7.2% of critical ones. Moreover, the Kaspersky MDR team came across other binaries, such as `msiexec` [21], mshta [66], `regsvr32` [67], and `netsh` [68], to mention a few.

Thus, in regards to RQ1: the most common LOLBins that were observed to be frequently utilized, along with their ATT&CK mappings could be summarized as follows:

- **Execution**: `PowerShell`, `Windows command shell`, `rundll32`, `regsvr32`, `mshta`, `MSBuild`, `te.exe` and `Wscript`
- **Lateral Movement and Persistence**: `PsExec`, WMI, WMIC
- **Discovery**: `ipconfig`, `tasklist`, `net`, `systeminfo`, `ping` and `nslookup`
- **Defense Evasion**: `certutil`, `msiexec`, and `netsh`

It is worth mentioning that the binaries above have been loosely grouped into a single tactic to avoid redundancy (multiple entries), as many of them are versatile and can serve multiple purposes. For instance, `netsh` can be exploited for defense evasion (to modify firewall settings), discovery (to discover firewall settings), Command and Control (C2) (to setup proxy tunnel), and persistence (to execute a helper) [53].

2.3.2 Minor related works

There have been some recent studies that laid emphasis on LotL attack detection and proposed various strategies to tackle the issue. For instance, techniques involving Natural Language Processing (NLP) and Supervised Learning (SL) have been utilized to examine command strings and identify misuse of LOLBins [69]. Active learning frameworks, such as LOLAL (which uses machine learning algorithms and text embedding methods), have been designed to enhance detection accuracy, achieving an F1 score of 96% at identifying LotL attack samples (command-line datasets) [70]. Monitoring runtime Application Programming Interface (API) signatures has been proposed as a way to detect fileless malware and malicious activities linked to LotL attacks [71]. Moreover, deep learning framework such as LOLWTC have shown potential by employing word embeddings and deep networks to classify command-line text, attaining F1 score of 0.9945 in tests [72]. These diverse approaches focused on malware detection and classification rather than evaluating the efficacy of IDS or accurately distinguishing benign and malicious activities, as is the focus of this study.

Another relevant works are the collaborative guide penned by various Western national security agencies, addressing prevalent LOLBins and cyber defense gaps [6] and The DFIR Report [73] which is an excellent source for incident reports and obtaining adversarial procedures. When examining prevalent LOLBins used in the wild and the APT groups that have them in their arsenal, this study builds on some of the APT listings covered by Bars smith et al. [1] and the LotL executables identified in resources [55] [8] [6] .

In addition, ART [30] will be used to emulate adversarial activity, whereas Olaf Hartong's sysmon-modular [74] will supply a standard configuration for System Monitor (Sysmon) [75].

Chapter 3

Methodology

The purpose of this chapter is to provide an overview of the research methodology used in this thesis. Section 3.1 details the research process. Section 3.2 outlines the qualitative research and the findings of the semi-structured interviews and online surveys. Section 3.3 describes the test scenarios. Section 3.4 focuses on the data collection techniques used for this research. Section 3.5 describes the experimental design.

3.1 Research Process

The steps conducted to carry out this research is depicted below in Figures 3.1.

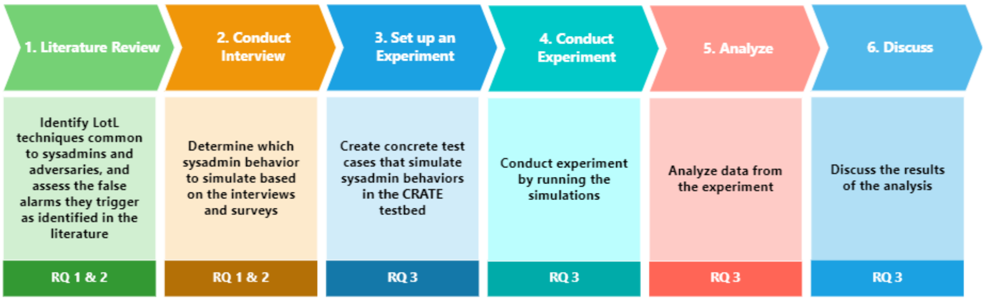

1. Literature Review
Identify LotL techniques common to sysadmins and adversaries, and assess the false alarms they trigger as identified in the literature
RQ 1 & 2
2. Conduct Interview
Determine which sysadmin behavior to simulate based on the interviews and surveys
RQ 1 & 2
3. Set up an Experiment
Create concrete test cases that simulate sysadmin behaviors in the CRATE testbed
RQ 3
4. Conduct Experiment
Conduct experiment by running the simulations
RQ 3
5. Analyze
Analyze data from the experiment
RQ 3
6. Discuss
Discuss the results of the analysis
RQ 3

Figure 3.1: Research Process

3.2 Qualitative Method

3.2.1 Semi-Structured Interview and Online Survey

One of the main focus of this paper is to identify legitimate actions that are seen as threats and normal activities flagged as suspicious. The LR initially explored around twenty LOLBins, of which ten were manually selected for further investigation, based on prevalence, administrative usage and potential to induce FA. Hence, the objective of the qualitative method was to identify the sysadmin and adversarial behaviors that overlap and to ascertain which of the non-standard LotL functions exploited by malefactors have legitimate sysadmin use case in the real-world.

Both semi-structured interviews and online surveys were composed of the same questions, as the aim of the online surveys were to increase reachability and participation. The questions were based on LotL techniques identified from adversarial procedure examples, and sigma-rules. The wording of the questions needed to follow four criterias based on [17]: open-ended (allowing interviewees to answer in their own word), neutral (with the interviewer remaining unbiased to interviewees opinions), singular (focusing on one idea per question) and clear (in terms of words used). Once questions were prepared for each of the selected binaries, two versions were made to tailor them for the two main professions: sysadmin and log analysts. Although the questions were available in both English and Swedish, all willing participants were Swedish speakers and hence was conducted in Swedish.

The recruitment of participants involved sending emails to government agencies, IT-departments at universities, SOC firms; reaching out to sysadmins via LinkedIn and Discord as well as assistance from my supervisors and referrals (where suggested contacts forwarded the information to more suitable individuals). The willing participants included four Windows system administrators, two log analysts, and one network engineer. Four participants belonged to the same organization, while the rest were from three separate organizations. After being briefed on the purpose of the study, participants were given the option to choose between an interview (conducted via Zoom) or a survey, of which three agreed to an interview and the rest opted for the survey. The participants' professional backgrounds, their familiarity with ten Windows binaries, and the anonymized organizations they belong to could be seen in Table 3.1. One of the interviewees consented for the interview to be recorded, while the other two chose to fill in a survey following their interview instead. The audio recording was then transcribed, resulting in a textual data of 4782 words, which was then manually used to complete one survey. Only the techniques that got a validated response from the participants will be included in the test cases.

Table 3.1: The Figure illustrates the participants' professional backgrounds, their familiarity with the ten Windows binaries, and the anonymized organizations they belong to

ORG-ID	BACKGROUND	FAMILIAR LOLBINS	TOTAL LOLBINS FAMILIARITY
A	Sysadmin	PsExec, Msiexec, SSH, Wmic, Winget, Cmd	6
B	Sysadmin	PsExec, Msiexec, SSH, Wmic, Winget, Wbadmim, Certutil, Ntdsutil, Cmd, Cmdkey	10
C	Log Analyst	Msiexec, Wmic, Certutil, Cmd	4
D	Sysadmin	PsExec, SSH, Cmd	3
D	Log Analyst	PsExec, SSH, Cmd	3
D	Network Engineer	SSH, Cmd	2
D	Sysadmin	PsExec, SSH, Wmic, Winget, Certutil, Cmd	6

Below are examples of Certutil-related questions that were directed at sysadmin:

1. How do you use certutil in your daily routine?
2. a) Have you ever used CertUtil to download files from the internet (e.g., certutil -urlcache -split -f; -verifyctl)? b) What are the legitimate scenarios for this?
3. What kind of administrative tasks would require encoding and decoding files to/from Base64 using CertUtil (e.g., certutil -encode | certutil -decode, -decodehex)?

The questions above were slightly tailored for participants with background in log analysis:

1. a) Have you observed system administrators using CertUtil in their routine tasks? b) How do you distinguish between legitimate and potentially malicious usage in your logs?
2. a) Have you seen CertUtil being used by admins to download files from the internet (e.g., certutil -urlcache -split -f, -verifyctl)? b) What indicators help you determine if this is a legitimate scenario or a potential adversary tactic?
3. a) What administrative tasks have you seen that require encoding and decoding files to/from Base64 using CertUtil (e.g., certutil -encode, certutil -decode, -decodehex)? b) How do you verify if these actions are legitimate or suspicious?

The justification for questioning Certutil's standard and non-standard functionalities is due to its ability to download web files as well as obscure files through encoding and decoding as a defense evasion tactic, making it crucial to examine whether sysadmins employ these features routinely [59]. The responses from participants could be seen from Tables 3.2 and 3.3. The complete list of interview (survey questions) is available in Appendix A.

Table 3.2: CertUtil-related questions for sysadmins with sample survey responses

`CertUtil Questions`	`Sysadmin Responses`
1 How do you use CertUtil in your daily routine?	**Response 1:** Signing certificate requests **Response 2:** Importing of certificates
2 a) Have you ever used CertUtil to download files from the internet (e.g., certutil -urlcache -split -f; -verifyctl)?	**Response 1:** No **Response 2:** ”Hell no, 100% harmful”
3 What kind of administrative tasks would require encoding and decoding files to/from Base64 using CertUtil (e.g., certutil -encode | certutil -decode, -decodehex)?	**Response 1:** Conversion between different certificate formats **Response 2:** ”Don’t see the need for it”

Table 3.3: CertUtil-Related Questions for Log Analysts with Sample Response

`Certutil Questions`	`Log analyst Response`
1 a) Have you observed system administrators using CertUtil in their routine tasks?	N/A
2 a) Have you seen CertUtil being used by admins to download files from the internet (e.g., certutil -urlcache -split -f, -verifyctl)?	a) Yes
b) What indicators help you determine if this is a legitimate scenario or a potential adversary tactic?	b) Considerations include who is logged in, where the command is run from, and the command line used
3 a) What administrative tasks have you seen that require encoding and decoding files to/from Base64 using CertUtil (e.g., certutil -encode, certutil -decode, -decodehex)?	When verifying code on web pages, controlling proxy functionality, and similar tasks

3.2.2 Qualitative Findings

The LOLBins included in the interviews and surveys, which were chosen based on the LR were: PsExec [2], msiexec [21], ssh [22], wmic [23], winget [24], wbadmin [25], certutil [14], ntdsutil [26], cmd [27] and cmdkey [28]. The LotL techniques were categorized into various functions such as credentials, download, dump, encode/decode, execute, modify system settings, recon and upload, inspired by LOLBAS [48] and partly based on Crowdstrike's white paper [8] and interview/survey participants response as depicted in Tables 3.4 and 3.5. The primary techniques investigated for each windows binary can be summarized as follows:

- **PsExec**: execution of End-User License Agreement (EULA), remote execution and installation, escalation to local system, interactive vs non-interactive sessions, using PAExec and renaming binary
- **Msiexec**: Microsoft Software Installer (MSI) package installation/uninstallation/modification , installation from online source, dll file and custom msi file execution
- **SSH**: serving as proxy, for port-forwarding and RDP-tunneling
- **Wmic**: binary execution, binary execution on remote systems, script execution (JScript/VBScript), copying file from source to destination, volume shadow copy creation (of NTDS.dit)
- **Winget**: installation from microsoft store, URL-based downloading (specified in manifest.yml) and package installation via local manifest
- **Wbadmin**: deletion of backups and shadow copies, restoring NTDS.dit version and SYSTEM hive into file path
- **Certutil**: signing certificates, importing certificates, editing certificates, encoding/decoding and downloading
- **Ntdsutil**: dumping of Active Directory NTDS.dit database
- **Cmd**: file deletion (using wildcard), force deletion (-f) or deletion in quiet mode (-q), downloading/uploading, combination of downloading and executing using curl, add content to an Alternate Data Stream (ADS), payload execution stored in an Alternate Data Stream (ADS), enumeration and port scanning
- **Cmdkey**: retrieving credential information from host

Table 3.4: Adversarial usage of living-off-the-land binaries

LOLBIN	CREDENTIALS	DOWNLOAD	DUMP	ENCODE /DECODE	EXECUTE	MODIFY SYSTEM SETTINGS	RECON	UPLOAD
PsExec					🕵		🕵	🕵
Msiexec					🕵			
SSH		🕵			🕵			🕵
Wmic					🕵	🕵	🕵	
Winget		🕵			🕵			
Wbadmin			🕵					
Certutil		🕵		🕵		🕵		
Ntdsutil			🕵					
Cmd		🕵		🕵	🕵	🕵	🕵	🕵
Cmdkey	🕵							
Rundll32					🕵			
Powershell	🕵	🕵	🕵	🕵	🕵	🕵	🕵	🕵

Table 3.5: Administrative usage of living-off-the-land binaries

LOLBIN	CREDENTIALS	DOWNLOAD	DUMP	ENCODE /DECODE	EXECUTE	MODIFY SYSTEM SETTINGS	RECON	UPLOAD
PsExec					👤			
Msiexec					👤			
SSH		👤						👤
Wmic					👤	👤	👤	
Winget		👤						
Wbadmin					👤			
Certutil				👤				
Ntdsutil			👤					
Cmd					👤	👤	👤	
Cmdkey	👤							
Rundll32					👤			
Powershell	👤	👤	👤	👤	👤	👤	👤	👤

The LOLBins that the participants frequently used or encountered in their daily routine were PsExec, msiexec, ssh, wmic, winget and cmd as seen in Figures 3.2 and 3.3. According to the participants, some of the techniques that do not have legitimate sysadmin use cases include utilizing certutil for downloads and ssh for RDP tunneling. The whole overlapping techniques and adversarial behavior – based on the interviews/surveys – are illustrated in Table 3.6.

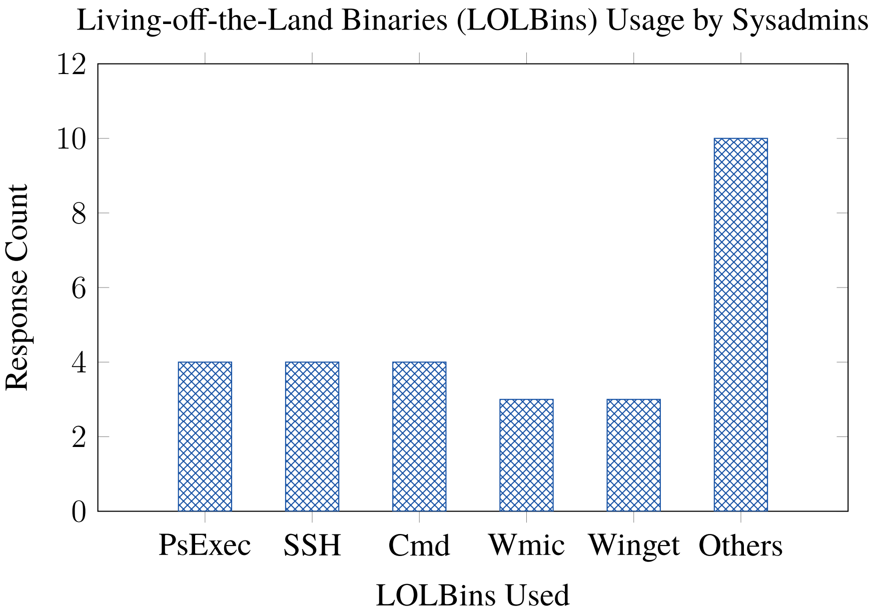


Figure 3.2: Living-off-the-Land binaries (LOLBins) used by the sysadmins who participated in the interviews and surveys, with numbers depicting response counts per binary.

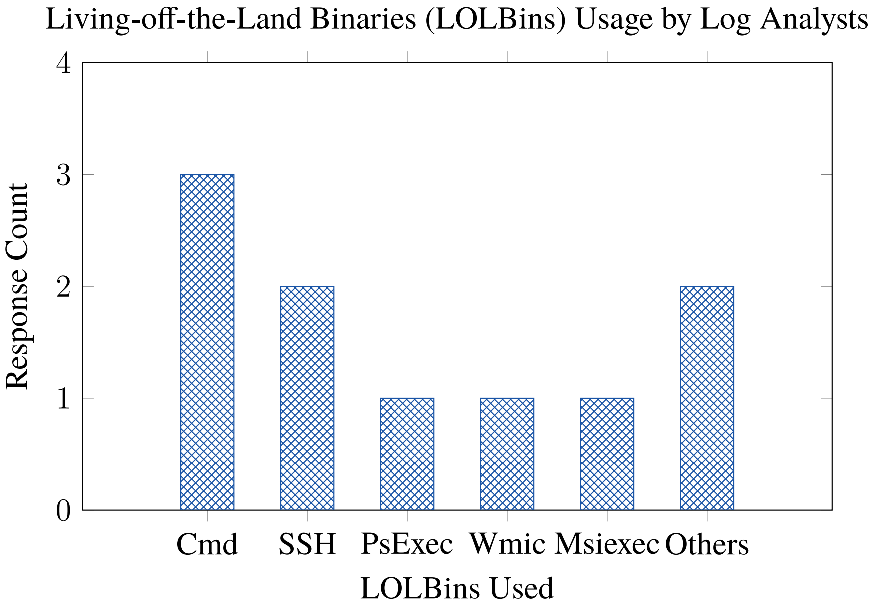


Figure 3.3: Living-off-the-Land binaries (LOLBins) that the log analysts are familiar with or encountered during network monitoring, with numbers representing response counts for each binary.

The findings from the semi-structured interviews and online surveys can be divided into three categories: LotL techniques with confirmed administrative usage, LotL techniques mainly seen as adversarial, and LotL techniques with uncertain sysadmin use case due to respondents' limited familiarity.

1. **LotL techniques with confirmed administrative usage:**
    - **SSH for Proxying and Remote Access:** Both groups confirmed SSH's widespread use among sysadmins, particularly for secure connections and remote management on Unix/Linux systems. Proxy usage was commonly cited, especially in network-segmented environments.

    - **PsExec with Privilege Elevation:** The sysadmins does occasionally use PsExec with elevated privilege, for tasks such as software installations or inspecting security hive.

    - **Msiexec for Offline Software Installation:** sysadmins use Msiexec primarily for software installations. Direct URL downloads are uncommon, but local installations are common. SOC respondents also view Msiexec as mostly legitimate when executed in standard system directories (System32 or SysWOW64)

    - **Using curl for Downloads:** According to respondents, curl is used in automation scripts or to retrieve updates in environments lacking graphical browsers, though sysadmins typically avoid executing files immediately after download (as it is a typical adversarial behavior). The log analysts supported this as legitimate when limited to trusted sources and monitored for unusual IP addresses or ports.

2. **LotL techniques mainly seen as adversarial:**
    - **SSH for RDP Tunneling:** This technique was generally discouraged by the sysadmins and was only to be used as a temporary solution (if someone has trouble getting in through a certain port and needs to switch ports)

  - **Using Certutil for downloading:** CertUtil's downloading feature was marked as adversarial, although one survey respondent mentioned seeing an administrator use it for downloading without providing additional details.

  - **Using WMIC with XSL Script Formatting:** Although WMIC itself has legacy use cases, specific actions such as XSL formatting and remote execution were regarded adversarial by participants, especially when executed from unfamiliar paths or users. Most system administrators avoided these features.

  - **Cmdkey for Credential Listing:** extremely unusual/rare for cmdkey to be used. The suggestion received was to alarm and control its use at each opportunity.

3. **LotL techniques with uncertain sysadmin use case:**

   - **PAExec:** Although it is a secure PsExec implementation and alternative, most participants did not know about it. Hence, its legitimate use case was unclear.

   - **SSH ProxyCommand:** There wasn't enough feedback regarding the legitimacy of the specific usage of ProxyCommand either from administrators or log analysts.

The findings from the interviews and surveys aids in identifying the intersecting techniques and determining which sysadmin behaviors, resembling adversarial actions, to emulate. Only LotL techniques that have confirmed administrative uses are to be emulated, as these accurately reflect legitimate sysadmin behaviors. The overlapping techniques and adversarial behaviors along with their ATT&CK mapping, are shown in Table 3.6. Moreover, the legitimate sysadmin behaviors and the use cases provided by participants are displayed in Table 3.7.

Table 3.6: The overlapping techniques based on practitioners

LOLBIN	Overlapping Behavior	Adversarial Behavior	ATT&CK Technique	IOC
PsExec	Remote execution, remote installation, escalation to local system, open interactive session	Commands run locally or with obfuscated parameters, renaming binary	Remote Execution, Privilege Escalation	Time of day, unprivileged account privilege escalation attempts, number of failed authentication attempts, creation of new accounts/services
Msiexec	Install/Uninstall/Modify MSI package, run custom MSI file, install from online sources (ex. PuTTY)	Call/run DLL files, web installation, unusual execution directory	Download, Execute, Modify System Settings	Uninstallation from unexpected locations or as a subprocess to unusual processes
SSH	Proxy, port-forwarding, valid account login	RDP tunneling	Recon, Execute, Upload	Encrypted SSH stream hiding RDP traffic
Wmic	Execute binary remotely, execute binary from wmic	Copy files, create shadow copies, execute script (JScrpt/VBScript)	Execute, Modify System Settings, Upload	Wmic execution from unauthorized sources or non-designated VLANs, lack of expected MFA or unexpected VLAN access
Winget	Download/install software from Microsoft Store, install packages via local manifest	Install packages from a URL specified in a local manifest	Download, Execute, Modify System Settings	N/A
Wbadmin	Delete backups	Delete shadow copies	Modify System Settings	Unexpected backup deletions outside regular maintenance, absence of technician confirmation
Certutil	Sign/import certificates, encode/decode	Download files	Modify System Settings, Encode/Decode	Usage of Certutil to download file from the internet
Cmd	Greedy file deletion, enumeration via dir command (with §switch), port scanning with for loops, downloading (using curl), deletion in quiet mode (-q)	Execution from ADS, forced deletion, combination of downloading and immediate execution with curl	Execute, Modify Settings, Upload, Download, Recon	Unusual ADS use (particularly with sensitive files), systematic file deletion outside routine maintenance

Table 3.7: Sysadmin use case examples based on participants reply

Tools	Sysadmin-behaviors	Use cases according to Participants
PsExec	Remote execution	For troubleshooting remote machines where PSRemoting has stopped working; for executing scripts
	Local system escalation	When inspecting the security hive; for packaging softwares for SCCM
Msiexec	Install MSI file	For installing softwares
	Install from online sources	Installing PuTTY
SSH	Proxying	We have isolated networks that can be accessed through SSH gateways
	Port forwarding	For accessing admin interfaces not publicly accessible over the network
Winget	Install from Microsoft Store	Installation from the store
	Local manifest installation	I’ve seen an automated packing solution in PowerShell doing this
Certutil	Encoding/decoding	Conversion between different certificate formats
Cmd	File deletion with wildcards	Deleting older backup files, e.g., ‘del *.bak‘

3.3 Test Scenarios

In an attempt to actively trigger detection systems through command sequences that a sysadmin might occasionally execute, it was crucial to first identify sysadmin behavior that resembles adversarial actions. LOLBins commonly used for both administrative and adversarial purposes were manually selected based on the LR [18][19][20][1][6][8][55], overlapping techniques were identified through semi-structured interviews and online surveys, and a subset of sysadmin behavior with confirmed administrative use case (based on the qualitative findings) were chosen to be emulated and used for test cases, as depicted in Figure 3.4 and Table 3.8. The choice of sysadmin behaviors was based on:

1. Actions performed (deemed reasonable) by interviewees and survey respondents
2. Potential to trigger an alarm
3. Simulatability in CRATE

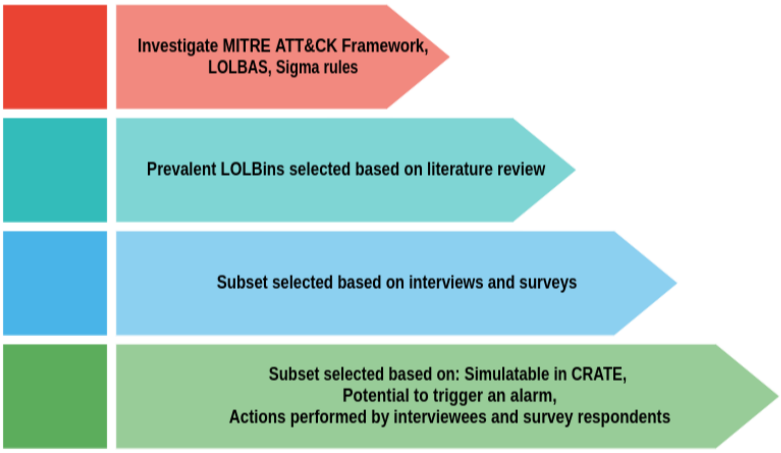
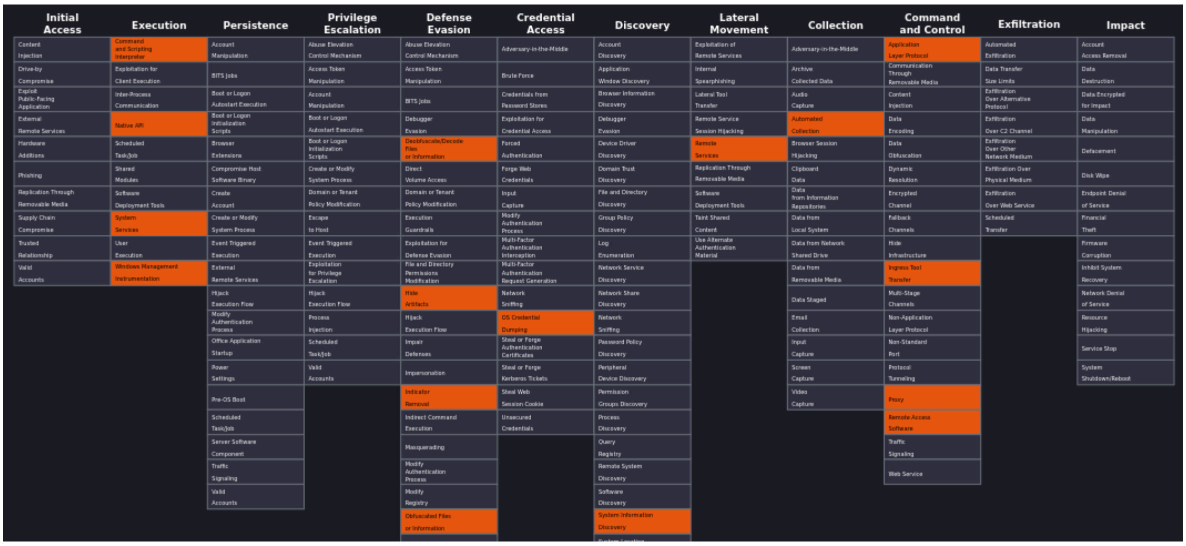

Investigate MITRE ATT&CK Framework, LOLBAS, Sigma rules
Prevalent LOLBins selected based on literature review
Subset selected based on interviews and surveys
Subset selectable based on: Simulatable in CRATE, Potential to trigger an alarm, Actions performed by interviewees and survey respondents

Figure 3.4: Depicts the process of selecting sysadmin behaviors to simulate in the test cases

All the behaviors and techniques that will be used in the test scenarios are defined and mapped in accordance to the MITRE ATT&CK framework [19] (see Figure 3.5 and Table 3.9), which provides an extensive taxonomy for adversary behavior and threat modeling. Furthermore, to evaluate the performance of IDS, traffic generation is needed, where ART [30] will serve for creating malicious activity (as seen in Table 3.9) (with atomic test), while legitimate user activity will be generated by simulating sysadmin behaviors using a custom script built with AutoIT [29] and differences in log traffic will be analyzed. The sysadmin behaviors that will be tested, along with the signatures that will be triggered are provide in Table 3.8.



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Content Injection	**Command and Scripting Interpreter**	Account Manipulation	Abuse Elevation Control Mechanism	Abuse Elevation Control Mechanism	Adversary-in-the-Middle	Account Discovery	Exploitation of Remote Services	Adversary-in-the-Middle	**Application Layer Protocol**	Automated Exfiltration	Account Access Removal
Drive-by Compromise	Exploitation for Client Execution	BITS Jobs	Access Token Manipulation	Access Token Manipulation	Brute Force	Application Window Discovery	Internal Spearphishing	Archive Collected Data	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction
Exploit Public-Facing Application	Inter-Process Communication	Boot or Logon Autostart Execution	Account Manipulation	BITS Jobs	Credentials from Password Stores	Browser Information Discovery	Lateral Tool Transfer	Audio Capture	Content Injection	Exfiltration Over Alternative Protocol	Data Encrypted for Impact
External Remote Services	**Native API**	Boot or Logon Initialization Scripts	Boot or Logon Autostart Execution	Debugger Evasion	Exploitation for Credential Access	Debugger Evasion	Remote Service Session Hijacking	**Automated Collection**	Data Encoding	Exfiltration Over C2 Channel	Data Manipulation
Hardware Additions	Scheduled Task/Job	Browser Extensions	Boot or Logon Initialization Scripts	**Deobfuscate/Decode Files or Information**	Forced Authentication	Device Driver Discovery	**Remote Services**	Browser Session Hijacking	Data Obfuscation	Exfiltration Over Other Network Medium	Defacement
Phishing	Shared Modules	Compromise Host Software Binary	Create or Modify System Process	Direct Volume Access	Forge Web Credentials	Domain Trust Discovery	Replication Through Removable Media	Clipboard Data	Dynamic Resolution	Exfiltration Over Physical Medium	Disk Wipe
Replication Through Removable Media	Software Deployment Tools	Create Account	Domain or Tenant Policy Modification	Domain or Tenant Policy Modification	Input Capture	File and Directory Discovery	Software Deployment Tools	Data from Information Repositories	Encrypted Channel	Exfiltration Over Web Service	Endpoint Denial of Service
Supply Chain Compromise	**System Services**	Create or Modify System Process	Escape to Host	Execution Guardrails	Modify Authentication Process	Group Policy Discovery	Taint Shared Content	Data from Local System	Fallback Channels	Scheduled Transfer	Financial Theft
Trusted Relationship	User Execution	Event Triggered Execution	Event Triggered Execution	Exploitation for Defense Evasion	Multi-Factor Authentication Interception	Log Enumeration	Use Alternate Authentication Material	Data from Network Shared Drive	Hide Infrastructure		Firmware Corruption
Valid Accounts	**Windows Management Instrumentation**	External Remote Services	Exploitation for Privilege Escalation	File and Directory Permissions Modification	Multi-Factor Authentication Request Generation	Network Service Discovery		Data from Removable Media	**Ingress Tool Transfer**		Inhibit System Recovery
		Hijack Execution Flow	Hijack Execution Flow	**Hide Artifacts**	Network Sniffing	Network Share Discovery		Data Staged	Multi-Stage Channels		Network Denial of Service
		Modify Authentication Process	Process Injection	Hijack Execution Flow	**OS Credential Dumping**	Network Sniffing		Email Collection	Non-Application Layer Protocol		Resource Hijacking
		Office Application Startup	Scheduled Task/Job	Impair Defenses	Steal or Forge Authentication Certificates	Password Policy Discovery		Input Capture	Non-Standard Port		Service Stop
		Power Settings	Valid Accounts	Impersonation	Steal or Forge Kerberos Tickets	Peripheral Device Discovery		Screen Capture	Protocol Tunneling		System Shutdown/Reboot
		Pre-OS Boot		**Indicator Removal**	Steal Web Session Cookie	Permission Groups Discovery		Video Capture	**Proxy**		
		Scheduled Task/Job		Indirect Command Execution	Unsecured Credentials	Process Discovery			**Remote Access Software**		
		Server Software Component		Masquerading		Query Registry			Traffic Signaling		
		Traffic Signaling		Modify Authentication Process		Remote System Discovery			Web Service		
		Valid Accounts		Modify Registry		Software Discovery					
				Obfuscated Files or Information		**System Information Discovery**					
						System Location					

Figure 3.5: ATT&CK techniques selected for testing

Table 3.8: Sysadmin Test Cases and the Sigma Rules they Trigger

LOLBins	**Test Cases**	**Sigma Rules**
PsExec	Remote execution	`win_sysinternals_psexec_remote_execution`
	Local system escalation	`win_sysinternals_psexec_paexec_escalate`
Msiexec	Install MSI file	`proc_creation_win_msiexec_install_quiet`
	Install from online sources	`proc_creation_win_msiexec_web_install`
SSH	Proxying	`proc_creation_win_ssh_proxy_execution`
	Port forwarding	`proc_creation_win_ssh_port_forward`
Certutil	Encoding	`proc_creation_win_certutil_encode`
	Decoding	`proc_creation_win_certutil_decode`
	Installing new root certificate	`win_certutil_certificate_installation`
Cmd	File deletion with wildcards	`proc_creation_win_cmd_del_greedy_deletion`
	Directory enumeration	`proc_creation_win_cmd_dir_execution`

Table 3.9: Atomic Test Cases. The atomic tests highlighted in blue are newly created

LOLBins	Test Cases	Atomic Tests
PsExec	Service execution	`Invoke-AtomicTest T1021.002-3`
	Remote execution	`Invoke-AtomicTest T1569.002-2`
	Local system escalation	`Invoke-AtomicTest T1569.002-8`
Msiexec	Quiet install	`Invoke-AtomicTest T1218.007-1`
	Web install	`Invoke-AtomicTest T1218.007-11`
SSH	Proxying	`Invoke-AtomicTest T1202-6`
		`Invoke-AtomicTest T1021.004-3`
	Port forwarding	`Invoke-AtomicTest T1021.004-4`
Certutil	Encoding	`Invoke-AtomicTest T1140-1`
	Decoding	`Invoke-AtomicTest T1140-2`
	Root certificate install	`Invoke-AtomicTest T1553.004-6`
Cmd	File deletion with *	`Invoke-AtomicTest T1070.004-12`
	Directory enumeration	`Invoke-AtomicTest T1217-8`

3.4 Data Collection

The sysadmin simulations and adversary emulations will be carried out on Windows Virtual Machines (VMs) hosted on CRATE. The data collected for correlation will consist of event logs, aggregated from multiple sources, including Snort, Wazuh, and Sysmon during the testing phase. To maintain privacy, these VMs will be solely utilized for this study.

3.5 Experimental design

3.5.1 Test environment

In order to reproduce the experiment setup, one needs to:

- Simulate the sysadmin behaviors using AutoIT scripts, as illustrated in Table 3.8
- Emulate the adversarial behaviors using atomic tests as highlighted in Table 3.9
- Conduct these tests on Windows VMs
- Deploy security monitoring tools, including Snort, Wazuh and Sysmon, to detect the benign and malicous activities generated

Chapter 4

Implementation

The purpose of this chapter is to provide an overview of the implementation phase that included developing custom scripts for simulating sysadmin actions and specific atomic tests (that are missing from the ART framework, but relevant to the test scenarios) to emulate adversarial activities. It also involved setting up the testbed as well as deploying and configuring the security monitoring tools. Section 4.1 highlights the network topology of the test environment implemented and the VMs deployed. Section 4.2 supplies examples of custom script implementations. Section 4.3 shows the necessary configurations used for the security monitoring tools in the testbed CRATE. Section 4.4 describes the execution of the test scenarios.

4.1 Testbed Design

This section briefly descibes Tyrdemo - the test environment used within CRATE and proceeds to highlight the necessary configurations for the security monitoring tools deployed. Tyrdemo's network topology as illustrated in Figure 4.1 was logically segmented into subnetworks that includes server (SRV), demilitarized zone (DMZ), SOC, and client (HQCLIENT) zone segments, mimicing those deployed in production IT Environments. Each segment hosted various roles and services, separated by a central firewall to emulate real-world organizational infrastructures. This enhanced the controlled testing environment with the ability to simulate scenarios found within operationally deployed networks realistically and, therefore, reliably.

All the VMs within Tyrdemo were comprised of Windows 10 IoT

Enterprise LTSC 2021 and Ubuntu Mate 22.04.1 LTS. Both the sysadmin simulations and adversary emulations were conducted on a single Windows machine. To establish PsExec and SSH remote services, an extra Windows VM (used as the remote host for the PsExec operation) and two other Ubuntu VMs (that served as a Proxy and remote servers) were used respectively. Furthermore, A Wazuh manager and Snort3 were deployed on dedicated Ubuntu VMs security monitoring and intrusion detection purposes.

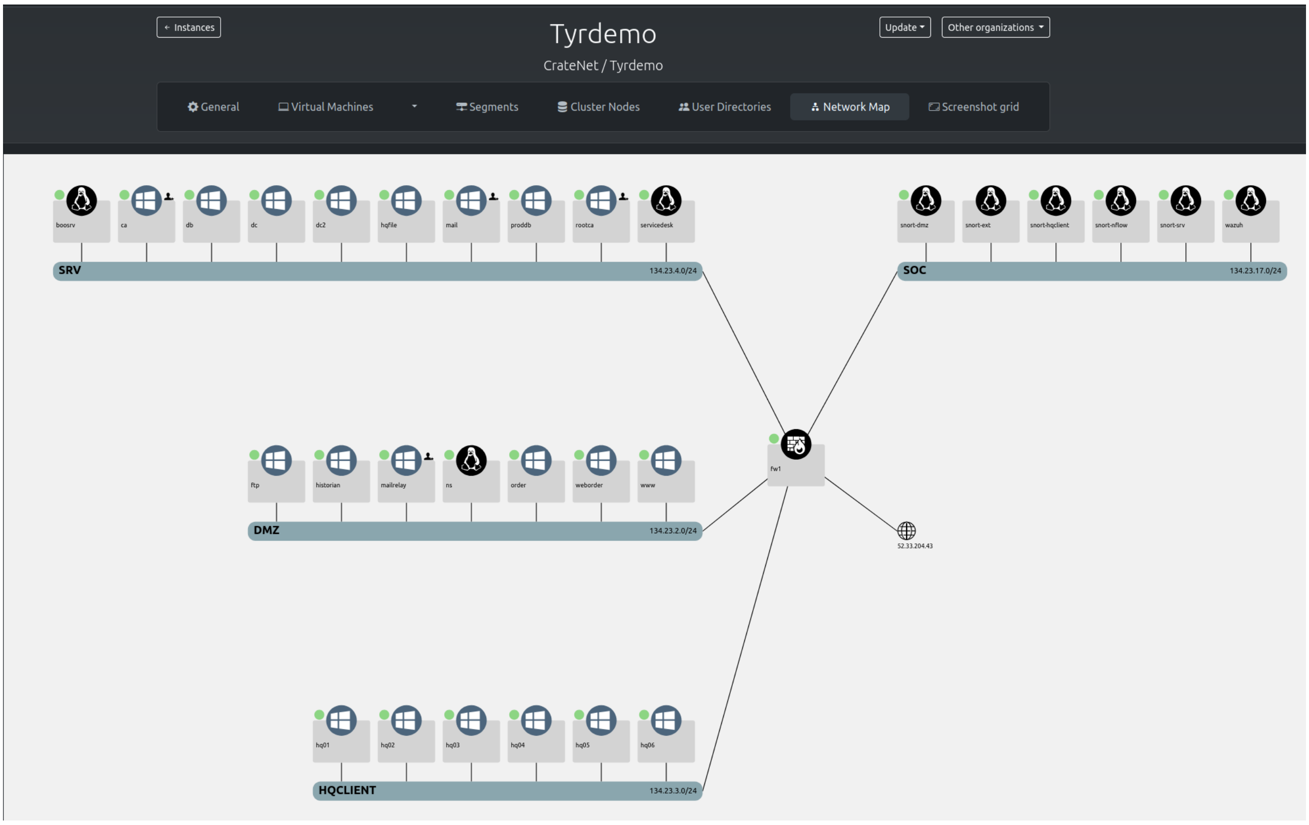

← Instances
Tyrdemo
CrateNet / Tyrdemo
Update
Other organizations
General
Virtual Machines
Segments
Cluster Nodes
User Directories
Network Map
Screenshot grid
boosrv
ca
db
dc
dc2
hqfile
mail
proddb
rootca
servicedesk
SRV
134.23.4.0/24
snort-dmz
snort-ext
snort-hqclient
snort-nflow
snort-srv
wazuh
SOC
134.23.17.0/24
fw1
52.33.204.43
ftp
historian
mailrelay
ns
order
weborder
www
DMZ
134.23.2.0/24
hq01
hq02
hq03
hq04
hq05
hq06
HQCLIENT
134.23.3.0/24

Figure 4.1: Depicts the network mapping in the test environment

4.2 Custom Scripts

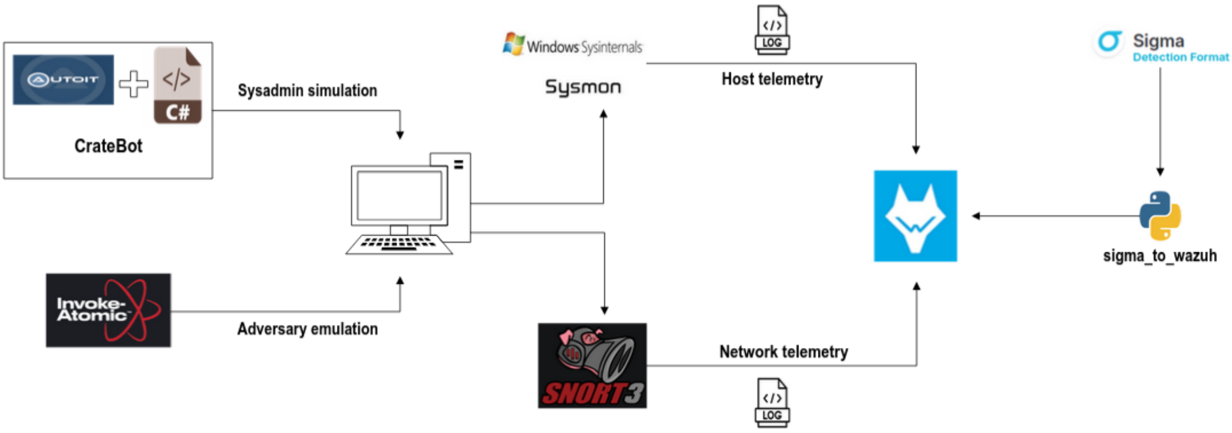

AUTOIT
C#
CrateBot
Sysadmin simulation
Invoke-Atomic
Adversary emulation
Windows Sysinternals
Sysmon
Host telemetry
LOG
SNORT3
Network telemetry
LOG
Sigma
Detection Format
sigma_to_wazuh

Figure 4.2: Illustrates the system design

As portrayed in Figure 4.2, the simulations of sysadmin interactions and activities were implemented using C# and AutoIt automation scripts to automate Windows GUI. These ranged from establishing remote connections to a machine using PsExec and listing processes and directories and deleting files using wildcard patterns, to the quiet installation of Wireshark with Msiexec, opening Wireshark, capturing network traffic and saving of network captures. This also included the download of PuTTY from an online source using Msiexec, which in turn was used to establish an SSH connection to a remote machine. In addition both SSH tunneling and SSH proxying techniques were emulated. Moreover, some tasks on certificate management have been simulated: creating a new self-signed certificate, encoding and decoding, and adding to the Root Certificate Authority store, then listing via certmgr in order to confirm addition of this certificate. Listing 4.1 is a sample of the AutoIt script used for simulating ssh proxying.

Listing 4.1: AutoIt script for simulating SSH proxying

```

#cs --------------------------------------------------------------

 AutoIt Version: 3.3.16.1
 Author: Peter Daniel
 Date: 2024-11-12

 Script Function:
    AutoIt script that simulates a system administrator
    using SSH ProxyCommand to connect to a host via an
    intermediary.

 The syntax for SSH Proxying is:
 ssh -o ProxyCommand=”ssh -W %h:%p
 [username]@[proxy_server]” [username]@[remote_server]

#ce --------------------------------------------------------------

; Define proxy credentials
$proxyUsername = ”<PROXY_USERNAME>”
$proxyHost = ”<PROXY_HOST_IP>”
$proxyPassword = ”<PROXY_PASSWORD>”

; Define target credentials
$targetUsername = ”<TARGET_USERNAME>”
$targetHost = ”<TARGET_HOST_IP>”
$targetPassword = ”<TARGET_PASSWORD>”

; Function to execute a command
Func RunCommand($command, $delay)
    Run($command)
    Sleep($delay)
EndFunc

; Function to send a command
Func SendCommand($command, $delay)
    Send($command)
    Sleep($delay)
EndFunc

; Start a command prompt
RunCommand(”cmd”, 2000)

;
;           LotL Technique: SSH as a Proxy
;      Sigma Rule: proc_creation_win_lolbin_ssh
;         CommandLine | Contains: 'ProxyCommand='
;

; Execute SSH ProxyCommand connection
SendCommand('C:\Windows\Sysnative\OpenSSH\ssh.exe -o
            ProxyCommand=”ssh -W %h:%p ' & $proxyUsername &
            '{Asc 64}' & $proxyHost & '” ' & $targetUsername
            & '{Asc 64}' & $targetHost & '{ENTER}', 2000)

SendCommand($proxyPassword & ”{ENTER}”, 2000)
SendCommand($targetPassword & ”{ENTER}”, 2000)

```

```
; Simulated sysadmin activities
SendCommand(”whoami && uname -r {ENTER}”, 2000)
SendCommand(”netstat -an | grep '5901' {ENTER}”, 2000)
SendCommand(”systemctl list-units --type=service | less {ENTER}”,
            4000)
SendCommand(”q {ENTER}”, 2000)
SendCommand(”sudo apt update {ENTER}”, 2000)
SendCommand(”<REMOTE_PASSWORD> {ENTER}”, 4000)
SendCommand(”sudo apt list --upgradable {ENTER}”, 2000)
SendCommand(”cat /etc/passwd {ENTER}”, 2000)

; Close SSH sessions
SendCommand(”exit {ENTER}”, 2000)
SendCommand(”exit {ENTER}”, 2000)
```

The adversarial behaviors that were emulated during the implementation phase were based on the atomic tests presented in Tables 4.1 and 4.2. Table 4.1 depicts the existing atomic tests from the ART framework that corresponds to the overlapping techniques. In contrast, Tables 4.2 and 4.3 outlines newly developed atomic tests designed to fill the gap in the framework regarding particular techniques such as greedy file deletion and SSH tunneling and proxying.

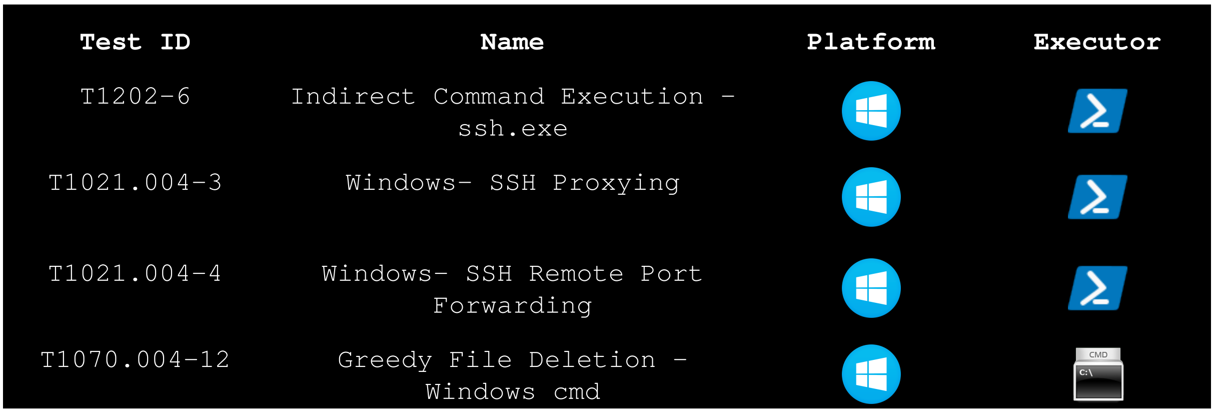
Table 4.1: Displays the relevant atomic tests used for adversary emulation along with their original descriptions.

LOLBin	**Invoke-AtomicTest**	**Description**
PsExec	T1021.002-3	Copies a file to a remote host and executes it using PsExec
	T1569.002-2	Cmd will utilize psexec.exe to spawn calc.exe on a remote endpoint (default:localhost)
	T1569.002-8	Upon successful execution, PsExec will be executed from a suspicious location and create a new pipe to execute CMD
Msiexec	T1218.007-1	Executes an MSI containing embedded JScript code using msiexec.exe
	T1218.007-2	Executes an MSI containing embedded VBScript code using msiexec.exe
	T1218.007-3	Executes an MSI containing an embedded DLL using msiexec.exe
	T1218.007-4	Executes an MSI containing an embedded EXE using msiexec.exe
	T1218.007-11	Execute arbitrary MSI file retrieved remotely. Less commonly seen in application installation, commonly seen in malware execution. The MSI executes a built-in JScript payload that launches powershell.exe.
Certutil	T1140-1	Encode/Decode executable. Upon execution, a file named T1140_calc_decoded.exe will be placed in the temp folder.
	T1140-2	Rename certutil and decode a file
	T1140-6	Creates a root CA with certutil
Cmd	T1217-8	This test will list the bookmarks for Internet Explorer that are found in the Favorites folder

Table 4.2: Describes the newly developed atomic tests designed to emulate specific adversarial techniques

LOLBin	Invoke-AtomicTest	Description
SSH	T1202-6	calc.exe is executed through ssh, which can be used as a defense evasion technique.
	T1021.004-3	An adversary uses SSH Proxying to tunnel an ssh connection through an intermediate proxy host.
	T1021.004-4	An adversary uses the OpenSSH client on a Windows host to set up remote port forwarding, which allows forwarding traffic from a port on the remote machine to a specified port on the local machine.
Cmd	T1070.004-12	Deletes files using greedy/wildcard expression

Table 4.3: Depicts the names of the newly created atomic tests, along with the executors used



Test ID	Name	Platform	Executor
T1202-6	Indirect Command Execution - ssh.exe	⊞	>_
T1021.004-3	Windows- SSH Proxying	⊞	>_
T1021.004-4	Windows- SSH Remote Port Forwarding	⊞	>_
T1070.004-12	Greedy File Deletion - Windows cmd	⊞	CMD C:\

Listing 4.2 displays the Markdown (MD) file for the newly developed Atomic Test T1202-6: Indirect Command Execution via ssh.exe, based on [49], while Listing 4.3 portrays the YAML file.

Listing 4.2: T1202.md file

~~~
# T1202 - Indirect Command Execution
## Atomic Test #6 - Indirect Command Execution - ssh.exe
The ssh.exe utility can be abused to proxy execution through the
    ↪ ProxyCommand, allowing for the execution of programs and commands.
In this case, calc.exe is executed through ssh, which can be used as a
    ↪ defense evasion technique.

Reference: https://lolbas-project.github.io/lolbas/Binaries/Ssh/
**Supported Platforms:** Windows
**auto_generated_guid:** 258dbc5c-df48-4f4e-9fb2-1c5be8abcaaa

#### Inputs:
|     Name     |       Description       |  Type  |  Default Value |
|--------------|-------------------------|--------|----------------|
| payload_path | Path to the executable | String |
    ↪ C:&#92;Windows&#92;System32&#92;calc.exe|

#### Attack Commands: Run with `powershell`!

```powershell
ssh -o ProxyCommand="#{payload_path}" .
~~~
#### Cleanup Commands:
```powershell
Stop-Process -Name CalculatorApp -Force
```
<br/>
```

Listing 4.3: T1202.yaml file

```

attack_technique: T1202
display_name: Indirect Command Execution
atomic_tests:
- name: Indirect Command Execution - ssh.exe
  auto_generated_guid: 258dbc5c-df48-4f4e-9fb2-1c5be8abcaaa
  description: |-
    The ssh.exe utility can be abused to proxy execution through the
        ↪ ProxyCommand, allowing for the execution of programs and
        ↪ commands.
    In this case, calc.exe is executed through ssh, which can be used as a
        ↪ defense evasion technique.
    Reference: https://lolbas-project.github.io/lolbas/Binaries/Ssh/
  supported_platforms:
  - windows
  input_arguments:
    payload_path:
      description: Path to the executable
      type: String
      default: C:\Windows\System32\calc.exe
  executor:
    command: ssh -o ProxyCommand="#{payload_path}" .
    cleanup_command: Stop-Process -Name CalculatorApp -Force
    name: powershell
    elevation_required: false
```
```

Furthermore, Listing 4.4 is a sample of Snort custom rules for detecting SSH traffic.

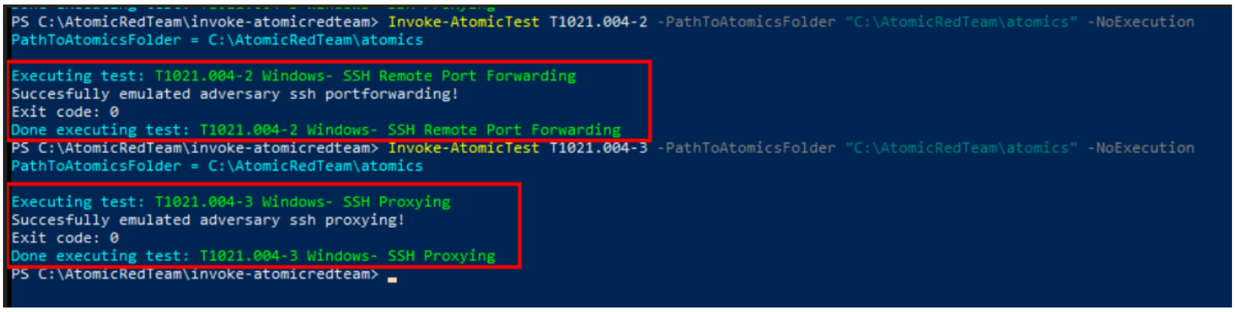
Listing 4.4: Snort rules for detecting SSH traffic

```

Rule: SSH Connection Attempt
alert tcp $HOME_NET any -> $HOME_NET any
(
    msg:"SSH Connection Attempt";
    flow: to_server,established;
    content:"ssh", fast_pattern,nocase;
    service: ssh;
    classtype:misc-activity;
    sid:1000001;
    rev:1;
)

Rule: SSH to Non-standard port
alert tcp $HOME_NET any -> $HOME_NET !22
(
    msg:"SSH to Non-standard port";
    content:"ssh", fast_pattern,nocase;
    service: ssh;
    classtype:misc-activity;
    sid:1000002;
    rev:1;
)
```

Last but not least, Listing 4.5 is a sample of the sigma rule converted for Wazuh for detecting ssh proxying in xml, whereas Figure 4.3 shows the execution of the newly created atomic tests for SSH tunneling and proxying:

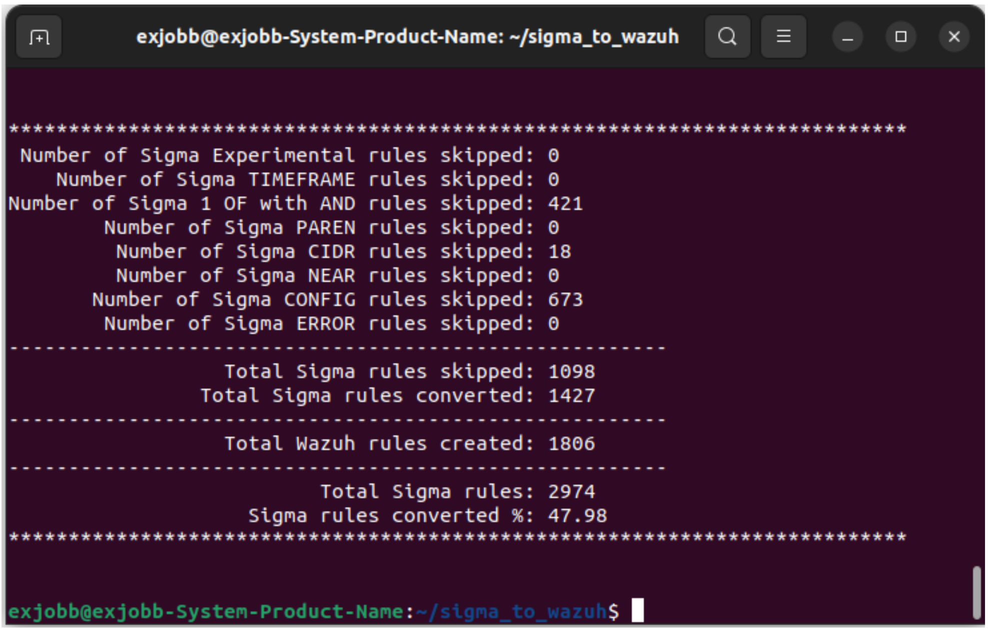
Listing 4.5: A sigma based Wazuh rule for detecting SSH Proxying

```
<!--   SSH: Program executed using proxy/local command via ssh.exe   -->

<rule id="100006" level="10">
    <info type="text">Sigma Rule Author: frack113, Nasreddine Bencherchali</info>
    <info type="text">Detects usage of the "ssh.exe" binary as a proxy to launch other programs.</info>
    <info type="text">Date: 2022-12-29</info>
    <info type="text">Status: test</info>
    <info type="text">ID: 7d6d30b8-5b91-4b90-a891-46cccaf29598</info>
    <info type="link">lolbas-project.github.io/lolbas/Binaries/Ssh/</info>
    <info type="link">https://man.openbsd.org/ssh_config#ProxyCommand</info>
    <info type="text">Falsepositives: Legitimate usage for administration purposes.</info>
    <mitre>
        <id>T1218</id>
    </mitre>
    <description>Program Executed Using Proxy/Local Command Via SSH.EXE</description>
    <options>no_full_log</options>
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.image" negate="no" type="pcre2">(?i)\\+ssh\.exe</field>
    <field name="win.eventdata.parentCommandLine">ProxyCommand=</field>
    <field name="win.eventdata.commandLine" type="pcre2">(?i)PermitLocalCommand</field>
    <field name="win.eventdata.commandLine" negate="no" type="pcre2">(?i)LocalCommand</field>
</rule>
```



```
PS C:\AtomicRedTeam\invoke-atomicredteam> Invoke-AtomicTest T1021.004-2 -PathToAtomicsFolder "C:\AtomicRedTeam\atomics" -NoExecution
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1021.004-2 Windows- SSH Remote Port Forwarding
Succesfully emulated adversary ssh portforwarding!
Exit code: 0
Done executing test: T1021.004-2 Windows- SSH Remote Port Forwarding
PS C:\AtomicRedTeam\invoke-atomicredteam> Invoke-AtomicTest T1021.004-3 -PathToAtomicsFolder "C:\AtomicRedTeam\atomics" -NoExecution
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1021.004-3 Windows- SSH Proxying
Succesfully emulated adversary ssh proxying!
Exit code: 0
Done executing test: T1021.004-3 Windows- SSH Proxying
PS C:\AtomicRedTeam\invoke-atomicredteam>
```

Figure 4.3: Atomic tests added for ssh tunneling and proxying



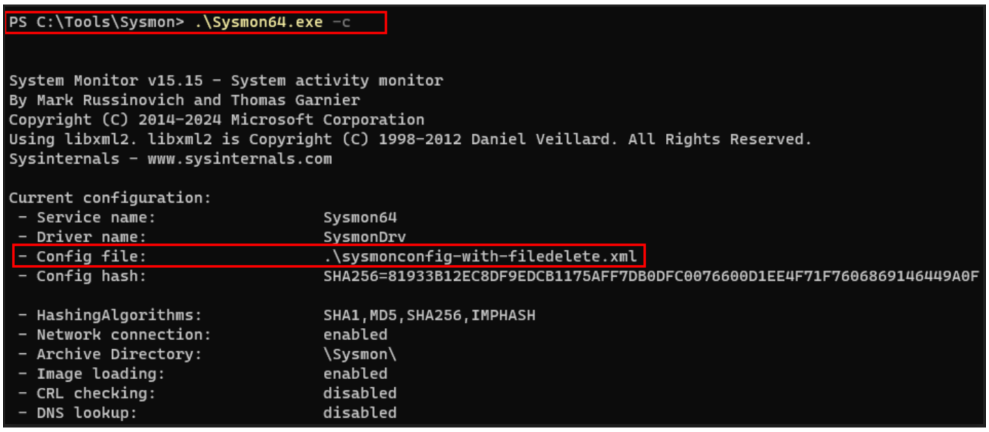
```
***************************************************************************
 Number of Sigma Experimental rules skipped: 0
    Number of Sigma TIMEFRAME rules skipped: 0
Number of Sigma 1 OF with AND rules skipped: 421
        Number of Sigma PAREN rules skipped: 0
         Number of Sigma CIDR rules skipped: 18
         Number of Sigma NEAR rules skipped: 0
       Number of Sigma CONFIG rules skipped: 673
        Number of Sigma ERROR rules skipped: 0
-------------------------------------------------------
                  Total Sigma rules skipped: 1098
                Total Sigma rules converted: 1427
-------------------------------------------------------
                  Total Wazuh rules created: 1806
-------------------------------------------------------
                          Total Sigma rules: 2974
                    Sigma rules converted %: 47.98
***************************************************************************

exjobb@exjobb-System-Product-Name:~/sigma_to_wazuh$
```

Figure 4.4: Shows the summary output of the rule conversion from Sigma to Wazuh

As depicted in Figure 4.4, the sigma to wazuh converter managed to only convert 47.98% of the sigma rules, which only included two of the rules relevant to detect the sysadmin actions. Thus, the remaining ten custom alert rules were written without the use of the converter.

4.3 Configurations

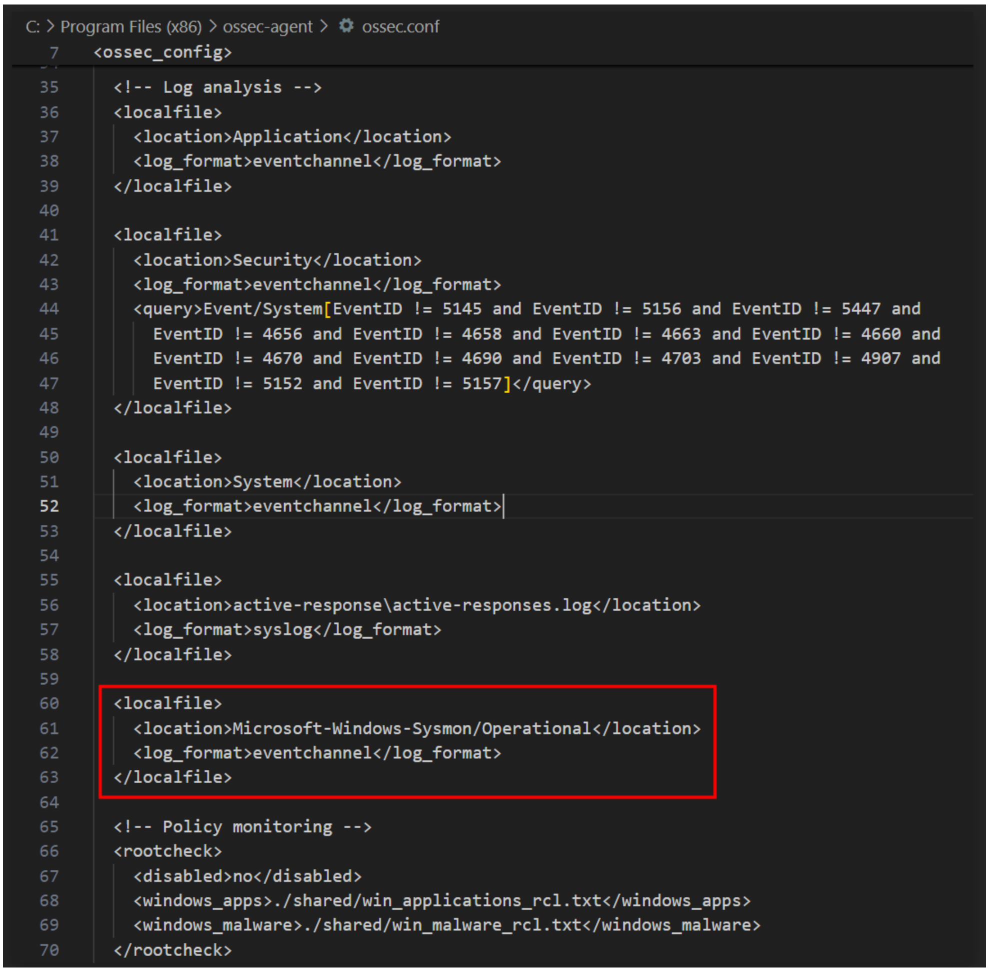
The deployment of security tools involved the installation of Sysmon. The Sysmon configuration template from SwiftOnSecurity [76] was initially used to configure Sysmon on the Windows machines, but was later replaced with Olaf Hartong's modular configuration template with file delete (as seen in Figure 4.5), which also monitors `Event ID 23: FileDelete`. This enabled the detection of greedy file deletion, which is one of the LotL techniques investigated.



```
PS C:\Tools\Sysmon> .\Sysmon64.exe -c


System Monitor v15.15 - System activity monitor
By Mark Russinovich and Thomas Garnier
Copyright (C) 2014-2024 Microsoft Corporation
Using libxml2. libxml2 is Copyright (C) 1998-2012 Daniel Veillard. All Rights Reserved.
Sysinternals - www.sysinternals.com

Current configuration:
 - Service name:                  Sysmon64
 - Driver name:                   SysmonDrv
 - Config file:                   .\sysmonconfig-with-filedelete.xml
 - Config hash:                   SHA256=81933B12EC8DF9EDCB1175AFF7DB0DFC0076600D1EE4F71F7606869146449A0F

 - HashingAlgorithms:             SHA1,MD5,SHA256,IMPHASH
 - Network connection:            enabled
 - Archive Directory:             \Sysmon\
 - Image loading:                 enabled
 - CRL checking:                  disabled
 - DNS lookup:                    disabled
```

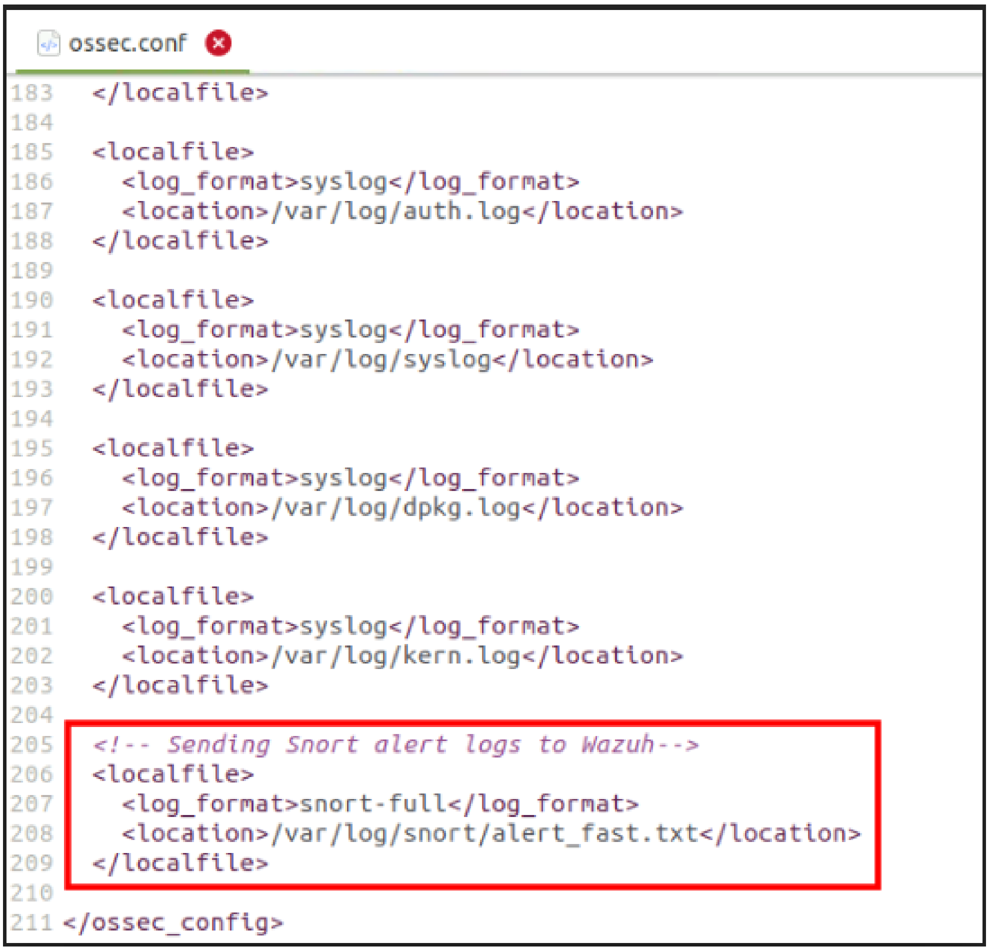
Figure 4.5: Sysmon configured with Olaf Hartong's template including Event ID 23 - File Delete

Wazuh served as a central log aggregation engine for both Sysmon and Snort. Figures 4.6 and 4.7 illustrates the necessary settings needed to be added to the Wazuh agents local configuration file (`ossec.conf`) in order for Wazuh to be able to monitor Sysmon events and ingest Snort logs. The file paths in the Windows and Linux endpoints are as follows: Windows - `C:\Program Files (x86)\ossec-agent\ossec.conf`; Linux - `/var/ossec/etc/ossec.conf`. Once modified, the Wazuh agents needed to be restarted for the changes to take effect.



```
<ossec_config>
  <!-- Log analysis -->
  <localfile>
    <location>Application</location>
    <log_format>eventchannel</log_format>
  </localfile>

  <localfile>
    <location>Security</location>
    <log_format>eventchannel</log_format>
    <query>Event/System[EventID != 5145 and EventID != 5156 and EventID != 5447 and
      EventID != 4656 and EventID != 4658 and EventID != 4663 and EventID != 4660 and
      EventID != 4670 and EventID != 4690 and EventID != 4703 and EventID != 4907 and
      EventID != 5152 and EventID != 5157]</query>
  </localfile>

  <localfile>
    <location>System</location>
    <log_format>eventchannel</log_format>
  </localfile>

  <localfile>
    <location>active-response\active-responses.log</location>
    <log_format>syslog</log_format>
  </localfile>

  <localfile>
    <location>Microsoft-Windows-Sysmon/Operational</location>
    <log_format>eventchannel</log_format>
  </localfile>

  <!-- Policy monitoring -->
  <rootcheck>
    <disabled>no</disabled>
    <windows_apps>./shared/win_applications_rcl.txt</windows_apps>
    <windows_malware>./shared/win_malware_rcl.txt</windows_malware>
  </rootcheck>
```

Figure 4.6: Configuring the Wazuh agent to forward sysmon events to Wazuh by adding the following to the ossec.conf file



```
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/syslog</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/dpkg.log</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/kern.log</location>
  </localfile>

  <!-- Sending Snort alert logs to Wazuh-->
  <localfile>
    <log_format>snort-full</log_format>
    <location>/var/log/snort/alert_fast.txt</location>
  </localfile>

</ossec_config>
```

Figure 4.7: Configuring the Wazuh agent to forward snort logs and alerts to Wazuh by adding the following to the ossec.conf file

4.4 Execution of Test Scenarios

This section presents an example of the sysadmin simulation and adversary emulation based on one of the selected test cases, more specifically, the use of the LotL binary, Msiexec to quietly install MSI packages. The sysadmin activity demonstrated here, invloves the installation of Wireshark, which is a network protocol analyzer tool [77], which in turn was used for capturing network traffic. In contrast, the adversarial behavior was portrayed using the atomic test `T1218.007-3`, which utilizes Msiexec to execute local MSI file with an embedded DLL.

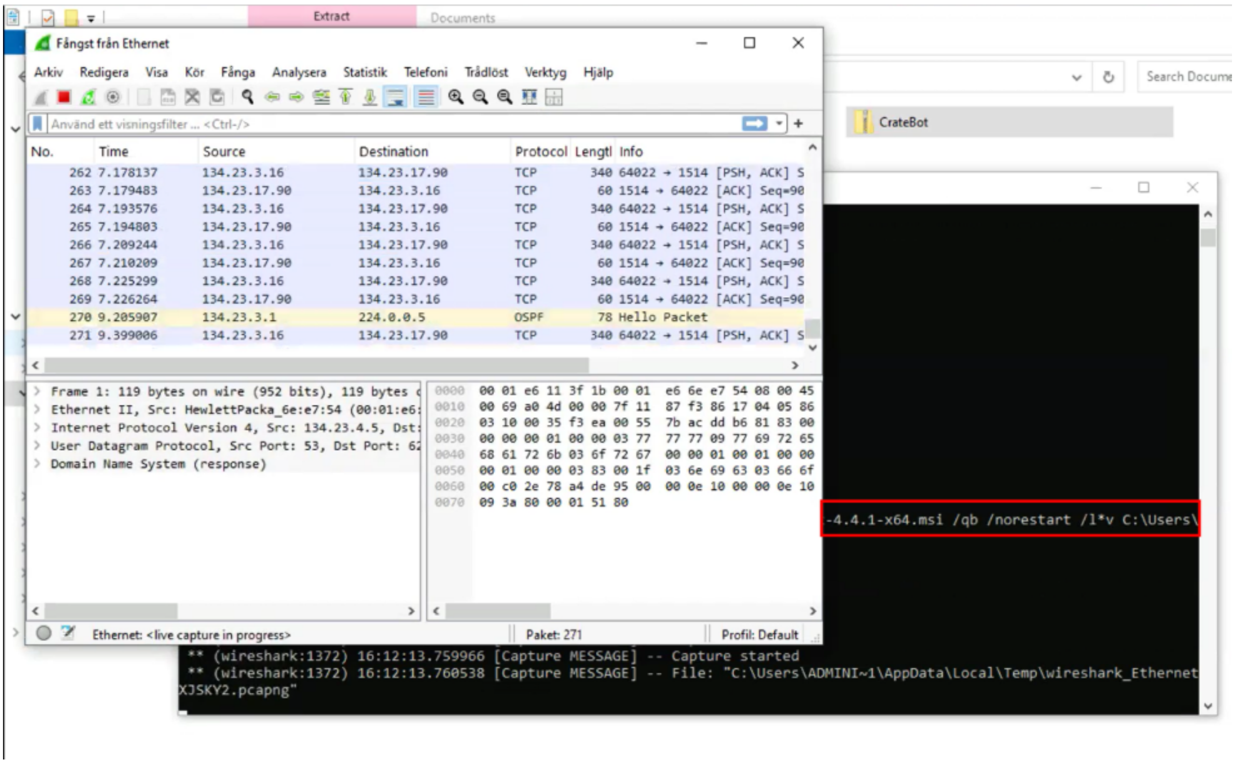
4.4.1 Sysadmin Simulation Example

Msiexec has a legitimate administrative usage, one of which is for installing MSI packages. Thus, Listing 4.6 illustrates a sample script that simulates an administrative task of installing a Wireshark MSI package. And to provide contextual relevance and show that it is part of a sequence of administrative actions rather than an isolated atomic event, the sysadmin is also depicted using the installed tool to analyze network traffic (see Figure 4.8).

Listing 4.6: AutoIt script for quietly installing Wireshark using Msiexec and capturing network traffic with it

```
#cs ----------------------------------------------------------------------

 AutoIt Version: 3.3.16.1
 Author: Peter Daniel 2024-11-15

 Script Function:
    Simulates Sysadmin quietly installing Wireshark, launching
    the application and capturing network traffic with it

#ce ----------------------------------------------------------------------

Local $installerPath = @UserProfileDir &
    "\Downloads\Wireshark-4.4.1-x64.msi"
Local $wiresharkLogPath = @UserProfileDir &
    "\Downloads\wireshark_install.log"

Func RunCommand($command, $delay)
    Run($command)
    Sleep($delay)
EndFunc

Func SendCommand($command, $delay)
    Send($command)
    Sleep($delay)
```

```
EndFunc

RunCommand(”cmd”, 2000)

;
;      LotL Technique: Install the target .MSI file
;    Sigma Rule: proc_creation_win_msiexec_install_quiet
;     CommandLine | Contains: '/i' | '/q' | '/package'
;

SendCommand('msiexec /i ' & $installerPath &  ” /qb /norestart /l*v ” &
    $wiresharkLogPath & ”{ENTER}”, 4000) ; install Wireshark

; Opens Wireshark, captures traffic and saves the capture

RunCommand(@ProgramFilesDir & ”\Wireshark\Wireshark.exe -i 5 -k”, 60000)
SendCommand(”^e”, 2000) ; Ctrl+E --> to stop capturing
SendCommand(”^s”, 2000) ; Ctrl+S --> to save the capture
SendCommand(”capture_log.pcap{ENTER}”, 2000)

SendCommand(”!{F4}”, 2000)
```



```
No.  Time      Source        Destination   Protocol Lengtl Info
 262 7.178137  134.23.3.16   134.23.17.90  TCP      340 64022 → 1514 [PSH, ACK] S
 263 7.179483  134.23.17.90  134.23.3.16   TCP       60 1514 → 64022 [ACK] Seq=90
 264 7.193576  134.23.3.16   134.23.17.90  TCP      340 64022 → 1514 [PSH, ACK] S
 265 7.194803  134.23.17.90  134.23.3.16   TCP       60 1514 → 64022 [ACK] Seq=90
 266 7.209244  134.23.3.16   134.23.17.90  TCP      340 64022 → 1514 [PSH, ACK] S
 267 7.210209  134.23.17.90  134.23.3.16   TCP       60 1514 → 64022 [ACK] Seq=90
 268 7.225299  134.23.3.16   134.23.17.90  TCP      340 64022 → 1514 [PSH, ACK] S
 269 7.226264  134.23.17.90  134.23.3.16   TCP       60 1514 → 64022 [ACK] Seq=90
 270 9.205907  134.23.3.1    224.0.0.5     OSPF      78 Hello Packet
 271 9.399006  134.23.3.16   134.23.17.90  TCP      340 64022 → 1514 [PSH, ACK] S
```

```
> Frame 1: 119 bytes on wire (952 bits), 119 bytes c
> Ethernet II, Src: HewlettPacka_6e:e7:54 (00:01:e6:
> Internet Protocol Version 4, Src: 134.23.4.5, Dst:
> User Datagram Protocol, Src Port: 53, Dst Port: 62
> Domain Name System (response)
```

```
0000  00 01 e6 11 3f 1b 00 01  e6 6e e7 54 08 00 45
0010  00 69 a0 4d 00 00 7f 11  87 f3 86 17 04 05 86
0020  03 10 00 35 f3 ea 00 55  7b ac dd b6 81 83 00
0030  00 00 00 01 00 00 03 77  77 77 09 77 69 72 65
0040  68 61 72 6b 03 6f 72 67  00 00 01 00 01 00 00
0050  00 01 00 00 03 83 00 1f  03 6e 69 63 03 66 6f
0060  00 c0 2e 78 a4 de 95 00  00 0e 10 00 00 0e 10
0070  09 3a 80 00 01 51 80
```

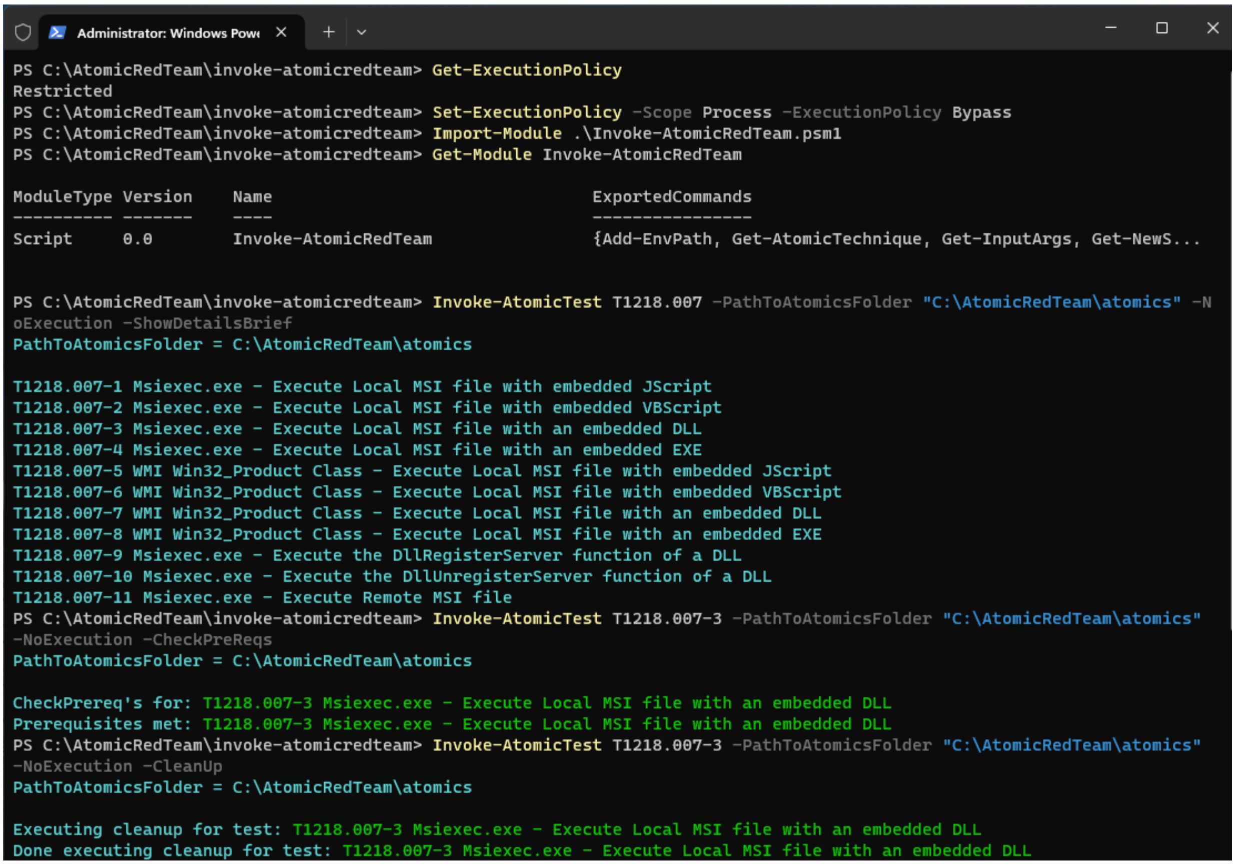
```
-4.4.1-x64.msi /qb /norestart /l*v C:\Users\
```

```
 ** (wireshark:1372) 16:12:13.759966 [Capture MESSAGE] -- Capture started
 ** (wireshark:1372) 16:12:13.760538 [Capture MESSAGE] -- File: "C:\Users\ADMINI~1\AppData\Local\Temp\wireshark_Ethernet
XJSKY2.pcapng"
```

Figure 4.8: Screenshot of Wireshark installation and network traffic capture

4.4.2 Adversary Emulation Example

This subsection provides a highlight of how the atomic tests were executed to emulate adversarial behaviors. As already mentioned in 4.4 and could be seen from Figure 4.9, the atomic test `T1218.007-3` is the one demonstrated here. This test utilizes a defense evasion technique, leveraging the Windows utility msiexec.exe to execute a malicious payload as an embedded DLL within an MSI package, aiming to bypass security controls and masquerade as a normal system process.



```
PS C:\AtomicRedTeam\invoke-atomicredteam> Get-ExecutionPolicy
Restricted
PS C:\AtomicRedTeam\invoke-atomicredteam> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
PS C:\AtomicRedTeam\invoke-atomicredteam> Import-Module .\Invoke-AtomicRedTeam.psm1
PS C:\AtomicRedTeam\invoke-atomicredteam> Get-Module Invoke-AtomicRedTeam

ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Script     0.0        Invoke-AtomicRedTeam                {Add-EnvPath, Get-AtomicTechnique, Get-InputArgs, Get-NewS...


PS C:\AtomicRedTeam\invoke-atomicredteam> Invoke-AtomicTest T1218.007 -PathToAtomicsFolder "C:\AtomicRedTeam\atomics" -N
oExecution -ShowDetailsBrief
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

T1218.007-1 Msiexec.exe - Execute Local MSI file with embedded JScript
T1218.007-2 Msiexec.exe - Execute Local MSI file with embedded VBScript
T1218.007-3 Msiexec.exe - Execute Local MSI file with an embedded DLL
T1218.007-4 Msiexec.exe - Execute Local MSI file with an embedded EXE
T1218.007-5 WMI Win32_Product Class - Execute Local MSI file with embedded JScript
T1218.007-6 WMI Win32_Product Class - Execute Local MSI file with embedded VBScript
T1218.007-7 WMI Win32_Product Class - Execute Local MSI file with an embedded DLL
T1218.007-8 WMI Win32_Product Class - Execute Local MSI file with an embedded EXE
T1218.007-9 Msiexec.exe - Execute the DllRegisterServer function of a DLL
T1218.007-10 Msiexec.exe - Execute the DllUnregisterServer function of a DLL
T1218.007-11 Msiexec.exe - Execute Remote MSI file
PS C:\AtomicRedTeam\invoke-atomicredteam> Invoke-AtomicTest T1218.007-3 -PathToAtomicsFolder "C:\AtomicRedTeam\atomics"
-NoExecution -CheckPreReqs
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

CheckPrereq's for: T1218.007-3 Msiexec.exe - Execute Local MSI file with an embedded DLL
Prerequisites met: T1218.007-3 Msiexec.exe - Execute Local MSI file with an embedded DLL
PS C:\AtomicRedTeam\invoke-atomicredteam> Invoke-AtomicTest T1218.007-3 -PathToAtomicsFolder "C:\AtomicRedTeam\atomics"
-NoExecution -CleanUp
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing cleanup for test: T1218.007-3 Msiexec.exe - Execute Local MSI file with an embedded DLL
Done executing cleanup for test: T1218.007-3 Msiexec.exe - Execute Local MSI file with an embedded DLL
```

Figure 4.9: Atomic Test #3 - Msiexec.exe - Execute Local MSI file with an embedded DLL

Once the execution framework - ”Invoke-AtomicRedTeam” [78] - was installed, the following steps were performed to execute the atomic tests. Here, Atomic Test T1218.007-3 (illustrated in Figure 4.9) is used as an example.

1. **Modify Execution Policy:** This was done to temporarily allow the execution of ART test scripts, bypassing the execution policy for the current session, while maintaining the system's security integrity. This was the command used from a PowerShell prompt:

```
 PS C:\AtomicRedTeam\invoke-atomicredteam>
Set-ExecutionPolicy -Scope Process
-ExecutionPolicy Bypass
```

2. **Import Required Module:** The `Invoke-AtomicRedTeam` module must be imported into the current PowerShell session prior to test execution, using the following command:

```
 Import-Module .\Invoke-AtomicRedTeam.psm1
```

   `Get-Module` was used to confirm that the module was imported and available for use.

3. **Enumerate Available Atomic Tests:** To list the tests available under a specific technique, such as `T1218.007`, the `ShowDetailsBrief` or `ShowDetails` switches can be used:

```
 Invoke-AtomicTest T1218.007 -PathToAtomicsFolder
”C:\AtomicRedTeam\atomics” -NoExecution
-ShowDetailsBrief
```

   `ShowDetailsBrief` provides a concise list of subtests under the designated technique. For instance, `T1218.007` have 11 tests, of which 7 are Msiexec-related.

4. **Check Prerequisites:** Before executing the test, it is necessary to verify that all prerequisites are satisfied using the `-CheckPrereqs` parameter, to ensure proper execution. If the required prerequisites are missing, they can simply be installed with the `-GetPrereqs` flag.

```
 Invoke-AtomicTest T1218.007-3
-PathToAtomicsFolder ”C:\AtomicRedTeam\atomics”
-NoExecution -CheckPrereqs
```

5. **Invoke the Atomic Test:** Once prerequisites are satisfied, execute the atomic test by running the following command:

```
 Invoke-AtomicTest T1218.007-3
-PathToAtomicsFolder "C:\AtomicRedTeam\atomics"
-NoExecution -Cleanup

```

   It is always a good practice to remove any residual artifacts or changes introduced during the test. Using the `-Cleanup` parameter ensures that the system is reverted to its original state, preserving the integrity of the environment.

Chapter 5

Results and Analysis

This chapter presents the results of the conducted test scenarios and provides an analysis of the findings.

5.1 Results

Because only overlapping LotL techniques and behavioral patterns commonly shared by both system administrators and threat actors were tested, it was not surprising that identical sigma signatures were triggered as a result of the benign and malicious activities generated (as shown in Table 5.1). These activities induced by sysadmin and adversary procedures were predominantly host-based telemetry captured by Sysmon. In contrast, the network telemetry produced was minimal, detecting only activities pertaining to SSH and PsExec using Snort.

As highlighted in Table 5.1, the Wazuh alerts resulting from sysadmin activities mistaken as malicious were classified as BT, whereas alerts triggered by the atomic tests were labeled as TA. As stated in 2.3.1 the term FP is rather broad and vague, potentially giving the false impression that the monitoring tools are flawed for producing overwhelming amount of FPs. Consequently, metrics BT and TA were utilized, where BT represents instances where conditions are met, but the triggered events constitute legitimate administrative usage. Since only the LotL techniques that intersected were used to replicate adversarial and administrative behaviors, contextual analysis was necessary to tell apart the benign from the malicious.

Table 5.1: Wazuh rules that were triggered by both the sysadmin and adversary activities, along with alarm classification

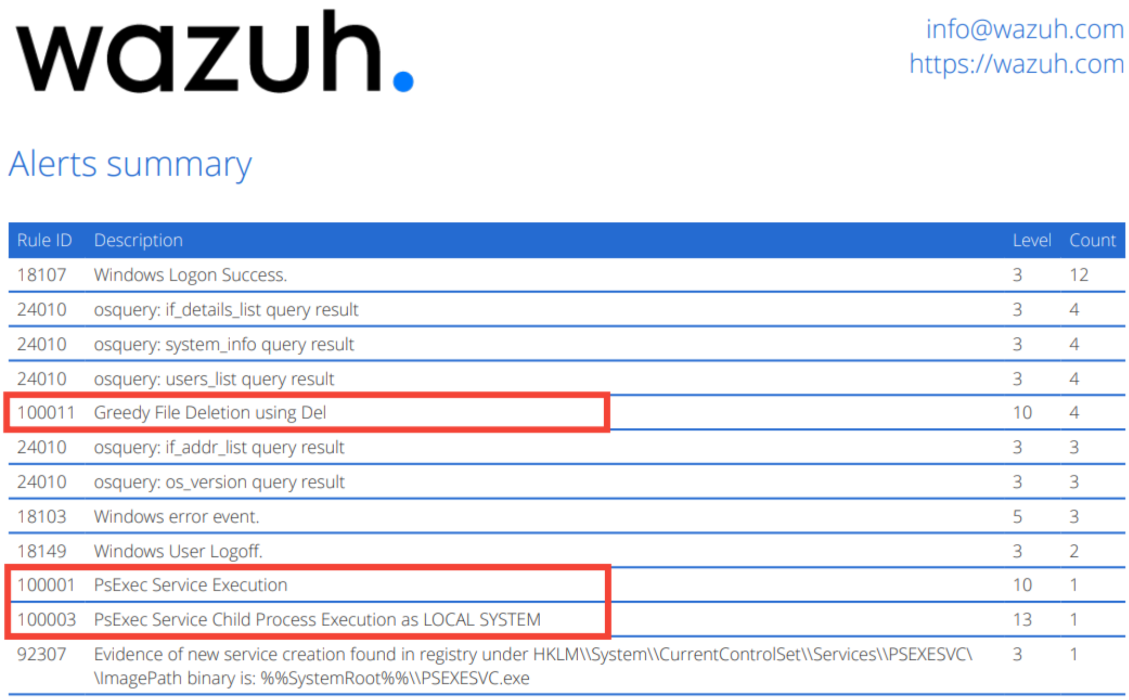
Rule ID	Rule	Sysadmin	Adversary
100001	`PsExec Service Execution`	✓ (BT)	✓ (TA)
100002	`Potential PsExec Remote Execution`	✓ (BT)	✓ (TA)
100003	`PsExec Service Child Process Execution as LOCAL SYSTEM`	✓ (BT)	✓ (TA)
100004	`Msiexec Quiet Installation`	✓ (BT)	✓ (TA)
100005	`Msiexec Web Install`	✓ (BT)	✓ (TA)
100006	`Program Executed Using Proxy/Local Command Via SSH.EXE`	✓ (BT)	✓ (TA)
100007	`Port Forwarding Activity Via SSH.EXE`	✓ (BT)	✓ (TA)
100008	`File Encoded To Base64 Via Certutil.EXE`	✓ (BT)	✓ (TA)
100009	`File Decoded From Base64/Hex Via Certutil.EXE`	✓ (BT)	✓ (TA)
100010	`New Root Certificate Installed Via Certutil.EXE`	✓ (BT)	✓ (TA)
100011	`Greedy File Deletion using Del`	✓ (BT)	✓ (TA)
100012	`File and SubFolder Enumeration via Dir Command`	✓ (BT)	✓ (TA)

5.1.1 Detection Efficacy

As already mentioned, the simulated test scenarios involved running automated sysadmin tasks and atomic tests using overlapping LotL techniques such as Msiexec's quiet installation and SSH tunneling. Results showed a significant overlap in alerts generated by both types of activities. The key findings are summarized below:

5.1.1.1 Overlap in Signatures

A total of 12 sigma signatures – for each of the LotL techniques tested – were triggered in Wazuh due to both legitimate administrative tasks (as depicted in Figures 5.1 and 5.2) and adversarial activities (as seen in Figure 5.3), which underscores the challenge of differentiating between them.



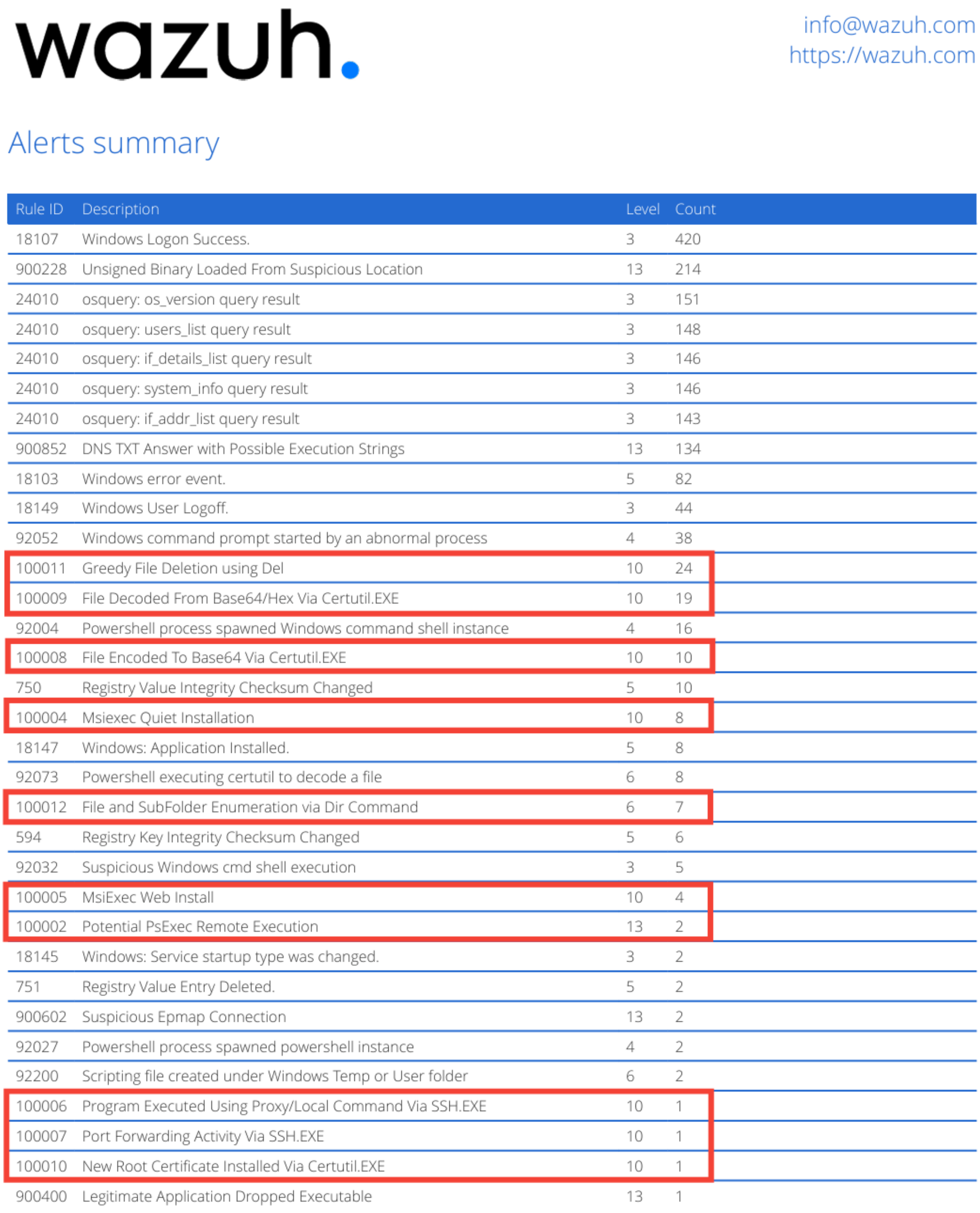
wazuh.

info@wazuh.com
https://wazuh.com

Alerts summary

Rule ID	Description	Level	Count
18107	Windows Logon Success.	3	12
24010	osquery: if_details_list query result	3	4
24010	osquery: system_info query result	3	4
24010	osquery: users_list query result	3	4
100011	Greedy File Deletion using Del	10	4
24010	osquery: if_addr_list query result	3	3
24010	osquery: os_version query result	3	3
18103	Windows error event.	5	3
18149	Windows User Logoff.	3	2
100001	PsExec Service Execution	10	1
100003	PsExec Service Child Process Execution as LOCAL SYSTEM	13	1
92307	Evidence of new service creation found in registry under HKLM\\System\\CurrentControlSet\\Services\\PSEXESVC\ \ImagePath binary is: %%SystemRoot%%\\PSEXESVC.exe	3	1

Figure 5.1: Sigma-based Wazuh alerts from sysadmin simulation

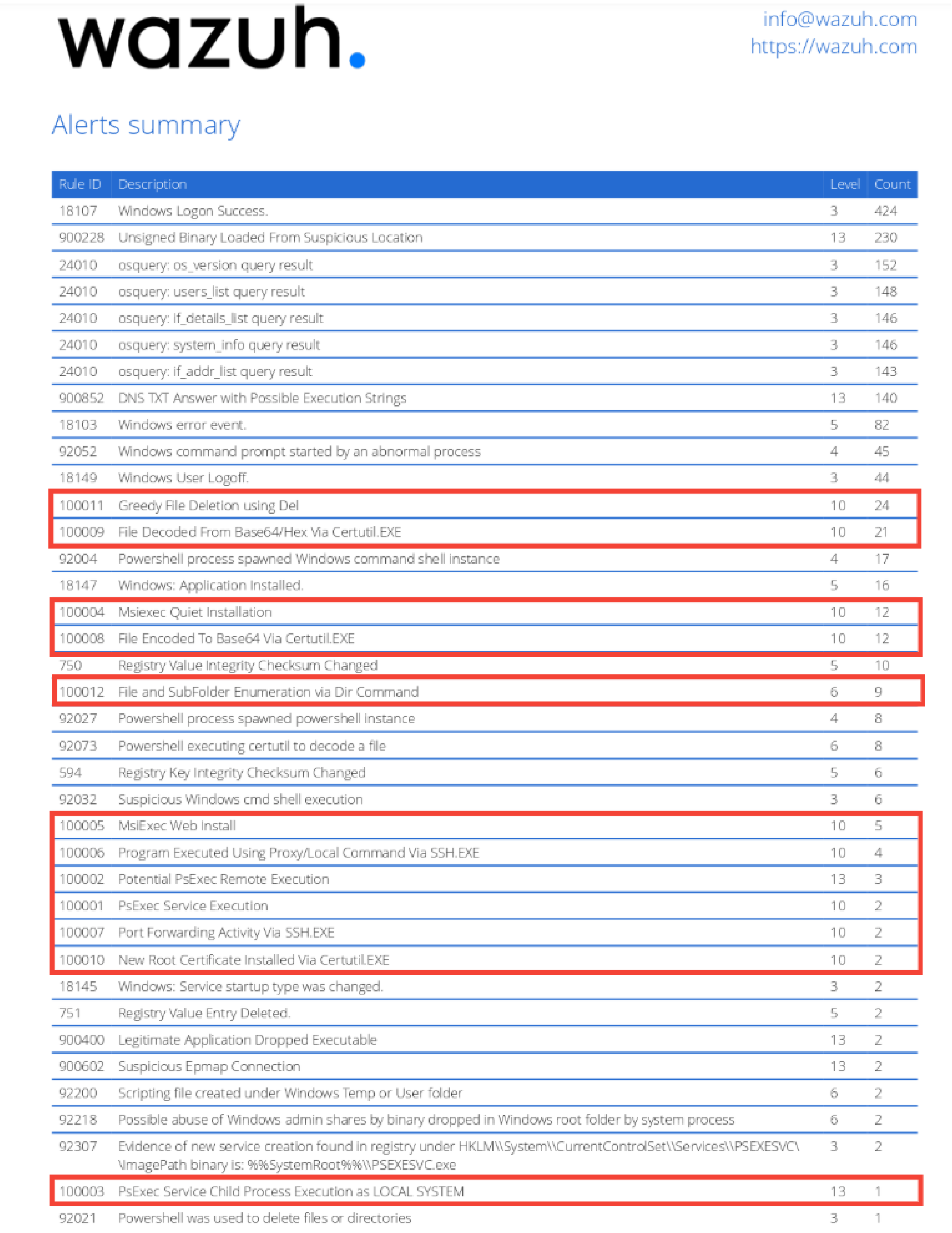


wazuh.
info@wazuh.com
https://wazuh.com

Alerts summary

Rule ID	Description	Level	Count
18107	Windows Logon Success.	3	420
900228	Unsigned Binary Loaded From Suspicious Location	13	214
24010	osquery: os_version query result	3	151
24010	osquery: users_list query result	3	148
24010	osquery: if_details_list query result	3	146
24010	osquery: system_info query result	3	146
24010	osquery: if_addr_list query result	3	143
900852	DNS TXT Answer with Possible Execution Strings	13	134
18103	Windows error event.	5	82
18149	Windows User Logoff.	3	44
92052	Windows command prompt started by an abnormal process	4	38
100011	Greedy File Deletion using Del	10	24
100009	File Decoded From Base64/Hex Via Certutil.EXE	10	19
92004	Powershell process spawned Windows command shell instance	4	16
100008	File Encoded To Base64 Via Certutil.EXE	10	10
750	Registry Value Integrity Checksum Changed	5	10
100004	Msiexec Quiet Installation	10	8
18147	Windows: Application Installed.	5	8
92073	Powershell executing certutil to decode a file	6	8
100012	File and SubFolder Enumeration via Dir Command	6	7
594	Registry Key Integrity Checksum Changed	5	6
92032	Suspicious Windows cmd shell execution	3	5
100005	MsiExec Web Install	10	4
100002	Potential PsExec Remote Execution	13	2
18145	Windows: Service startup type was changed.	3	2
751	Registry Value Entry Deleted.	5	2
900602	Suspicious Epmap Connection	13	2
92027	Powershell process spawned powershell instance	4	2
92200	Scripting file created under Windows Temp or User folder	6	2
100006	Program Executed Using Proxy/Local Command Via SSH.EXE	10	1
100007	Port Forwarding Activity Via SSH.EXE	10	1
100010	New Root Certificate Installed Via Certutil.EXE	10	1
900400	Legitimate Application Dropped Executable	13	1

Figure 5.2: Sigma-based Wazuh alerts from sysadmin simulation (continued)



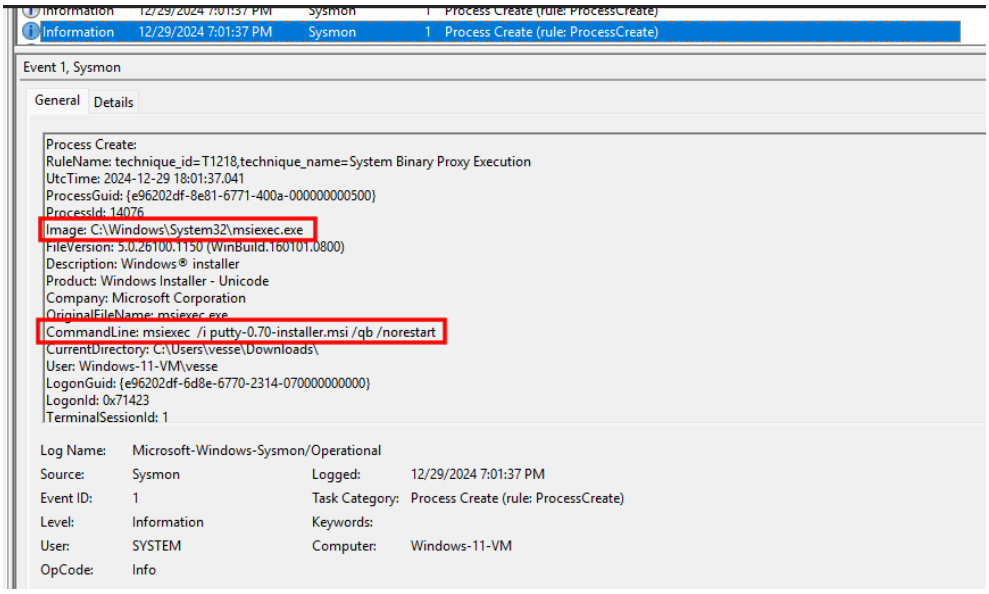
Alerts summary

Rule ID	Description	Level	Count
18107	Windows Logon Success.	3	424
900228	Unsigned Binary Loaded From Suspicious Location	13	230
24010	osquery: os_version query result	3	152
24010	osquery: users_list query result	3	148
24010	osquery: if_details_list query result	3	146
24010	osquery: system_info query result	3	146
24010	osquery: if_addr_list query result	3	143
900852	DNS TXT Answer with Possible Execution Strings	13	140
18103	Windows error event.	5	82
92052	Windows command prompt started by an abnormal process	4	45
18149	Windows User Logoff.	3	44
100011	Greedy File Deletion using Del	10	24
100009	File Decoded From Base64/Hex Via Certutil.EXE	10	21
92004	Powershell process spawned Windows command shell instance	4	17
18147	Windows: Application Installed.	5	16
100004	Msiexec Quiet Installation	10	12
100008	File Encoded To Base64 Via Certutil.EXE	10	12
750	Registry Value Integrity Checksum Changed	5	10
100012	File and SubFolder Enumeration via Dir Command	6	9
92027	Powershell process spawned powershell instance	4	8
92073	Powershell executing certutil to decode a file	6	8
594	Registry Key Integrity Checksum Changed	5	6
92032	Suspicious Windows cmd shell execution	3	6
100005	MsiExec Web Install	10	5
100006	Program Executed Using Proxy/Local Command Via SSH.EXE	10	4
100002	Potential PsExec Remote Execution	13	3
100001	PsExec Service Execution	10	2
100007	Port Forwarding Activity Via SSH.EXE	10	2
100010	New Root Certificate Installed Via Certutil.EXE	10	2
18145	Windows: Service startup type was changed.	3	2
751	Registry Value Entry Deleted.	5	2
900400	Legitimate Application Dropped Executable	13	2
900602	Suspicious Epmap Connection	13	2
92200	Scripting file created under Windows Temp or User folder	6	2
92218	Possible abuse of Windows admin shares by binary dropped in Windows root folder by system process	6	2
92307	Evidence of new service creation found in registry under HKLM\\System\\CurrentControlSet\\Services\\PSEXESVC\\ImagePath binary is: %%SystemRoot%%\\PSEXESVC.exe	3	2
100003	PsExec Service Child Process Execution as LOCAL SYSTEM	13	1
92021	Powershell was used to delete files or directories	3	1

Figure 5.3: Depicts the Wazuh alerts triggered during adversary emulation

5.1.1.2 Sysmon and Wazuh Alerts

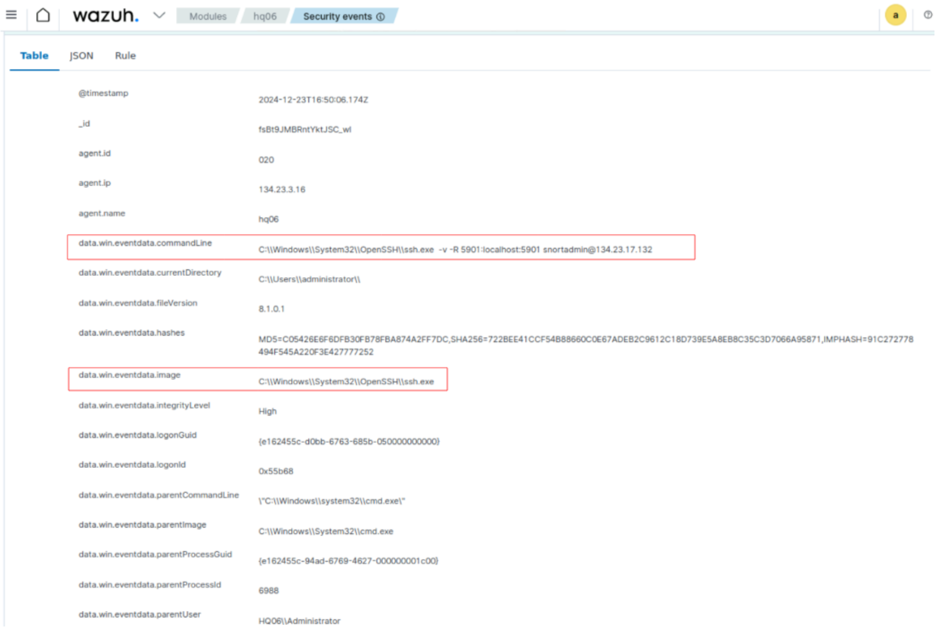
Sysmon is a powerful monitoring tool that logs extensive data, which could be dire to sift through, but Wazuh's customization allowed fine-tuning of detection signatures to streamline the processing of relevant Windows event fields, such as `win.eventdata.CommandLine`. For instance, Figure 5.4 displays event data captured during PuTTY's quiet installation via Msiexec. Fields such as `User`, `Image`, `SourceIp`, `DestinationIp` and `UtcTime` could serve as valuable IOCs, providing crucial context to better understand a given activity and determine whether it is malicious or benign. Sysmon successfully recorded the host-based events generated by the test scenarios and was able to consistently detect all the LotL techniques tested. Figures 5.5 and 5.6 depict the activation of the same signature due to a sysadmin procedure and the atomic test `T1021.004-2` involving SSH tunneling.



```
Information   12/29/2024 7:01:37 PM   Sysmon   1   Process Create (rule: ProcessCreate)
Information   12/29/2024 7:01:37 PM   Sysmon   1   Process Create (rule: ProcessCreate)
```

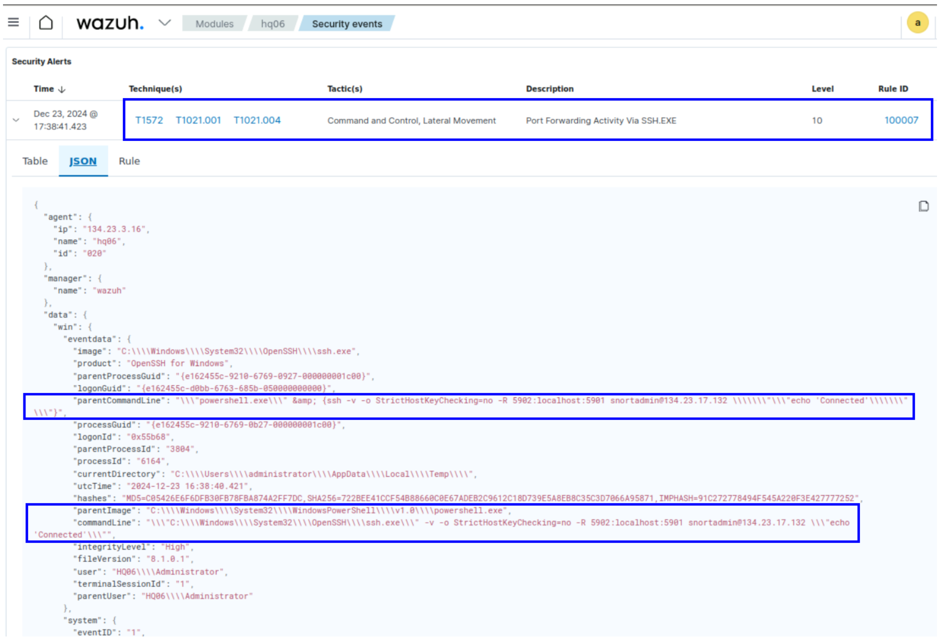
```
Event 1, Sysmon

Process Create:
RuleName: technique_id=T1218,technique_name=System Binary Proxy Execution
UtcTime: 2024-12-29 18:01:37.041
ProcessGuid: {e96202df-8e81-6771-400a-000000000500}
ProcessId: 14076
Image: C:\Windows\System32\msiexec.exe
FileVersion: 5.0.26100.1150 (WinBuild.160101.0800)
Description: Windows® installer
Product: Windows Installer - Unicode
Company: Microsoft Corporation
OriginalFileName: msiexec.exe
CommandLine: msiexec  /i putty-0.70-installer.msi /qb /norestart
CurrentDirectory: C:\Users\vesse\Downloads\
User: Windows-11-VM\vesse
LogonGuid: {e96202df-6d8e-6770-2314-070000000000}
LogonId: 0x71423
TerminalSessionId: 1

Log Name:   Microsoft-Windows-Sysmon/Operational
Source:     Sysmon             Logged:          12/29/2024 7:01:37 PM
Event ID:   1                  Task Category:   Process Create (rule: ProcessCreate)
Level:      Information        Keywords:
User:       SYSTEM             Computer:        Windows-11-VM
OpCode:     Info
```

Figure 5.4: Sysmon event for Msiexec quiet install



@timestamp	2024-12-23T16:50:06.174Z
_id	fsBt9JMBRntYktJSC_wl
agent.id	020
agent.ip	134.23.3.16
agent.name	hq06
data.win.eventdata.commandLine	C:\\Windows\\System32\\OpenSSH\\ssh.exe -v -R 5901:localhost:5901 snortadmin@134.23.17.132
data.win.eventdata.currentDirectory	C:\\Users\\administrator\\
data.win.eventdata.fileVersion	8.1.0.1
data.win.eventdata.hashes	MD5=C05426E6F6DFB30FB78FBA874A2FF7DC,SHA256=722BEE41CCF54B88660C0E67ADEB2C9612C18D739E5A8EB8C35C3D7066A95871,IMPHASH=91C272778494F545A220F3E427777252
data.win.eventdata.image	C:\\Windows\\System32\\OpenSSH\\ssh.exe
data.win.eventdata.integrityLevel	High
data.win.eventdata.logonGuid	{e162455c-d0bb-6763-685b-050000000000}
data.win.eventdata.logonId	0x55b68
data.win.eventdata.parentCommandLine	\"C:\\Windows\\system32\\cmd.exe\"
data.win.eventdata.parentImage	C:\\Windows\\System32\\cmd.exe
data.win.eventdata.parentProcessGuid	{e162455c-94ad-6769-4627-000000001c00}
data.win.eventdata.parentProcessId	6988
data.win.eventdata.parentUser	HQ06\\Administrator

Figure 5.5: Wazuh alert visualization for the sysadmin procedure - Port Forwarding Via SSH.exe



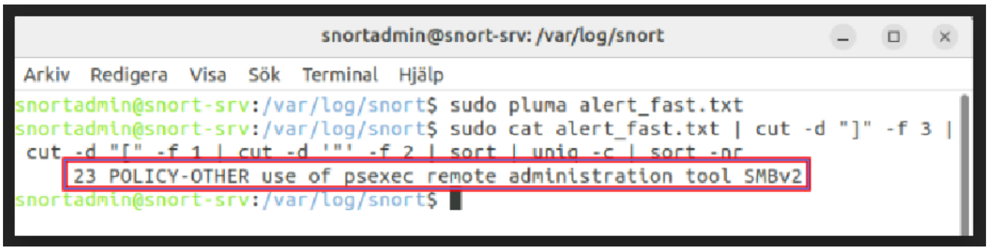
Time ↓	Technique(s)	Tactic(s)	Description	Level	Rule ID
Dec 23, 2024 @ 17:38:41.423	T1572 T1021.001 T1021.004	Command and Control, Lateral Movement	Port Forwarding Activity Via SSH.EXE	10	100007

```
{
  "agent": {
    "ip": "134.23.3.16",
    "name": "hq06",
    "id": "020"
  },
  "manager": {
    "name": "wazuh"
  },
  "data": {
    "win": {
      "eventdata": {
        "image": "C:\\\\Windows\\\\System32\\\\OpenSSH\\\\ssh.exe",
        "product": "OpenSSH for Windows",
        "parentProcessGuid": "{e162455c-9210-6769-0927-000000001c00}",
        "logonGuid": "{e162455c-d0bb-6763-685b-050000000000}",
        "parentCommandLine": "\\\"powershell.exe\\\" &amp; {ssh -v -o StrictHostKeyChecking=no -R 5902:localhost:5901 snortadmin@134.23.17.132 \\\\\\\"\\\"echo 'Connected'\\\\\\\"
\\\"}",
        "processGuid": "{e162455c-9210-6769-0b27-000000001c00}",
        "logonId": "0x55b68",
        "parentProcessId": "3804",
        "processId": "6164",
        "currentDirectory": "C:\\\\Users\\\\administrator\\\\AppData\\\\Local\\\\Temp\\\\",
        "utcTime": "2024-12-23 16:38:40.421",
        "hashes": "MD5=C05426E6F6DFB30FB78FBA874A2FF7DC,SHA256=722BEE41CCF54B88660C0E67ADEB2C9612C18D739E5A8EB8C35C3D7066A95871,IMPHASH=91C272778494F545A220F3E427777252",
        "parentImage": "C:\\\\Windows\\\\System32\\\\WindowsPowerShell\\\\v1.0\\\\powershell.exe",
        "commandLine": "\\\"C:\\\\Windows\\\\System32\\\\OpenSSH\\\\ssh.exe\\\" -v -o StrictHostKeyChecking=no -R 5902:localhost:5901 snortadmin@134.23.17.132 \\\"echo
'Connected'\\\"",
        "integrityLevel": "High",
        "fileVersion": "8.1.0.1",
        "user": "HQ06\\\\Administrator",
        "terminalSessionId": "1",
        "parentUser": "HQ06\\\\Administrator"
      },
      "system": {
        "eventID": "1",
```

Figure 5.6: Wazuh alert visualization for the atomic test - Port Forwarding Via SSH.exe

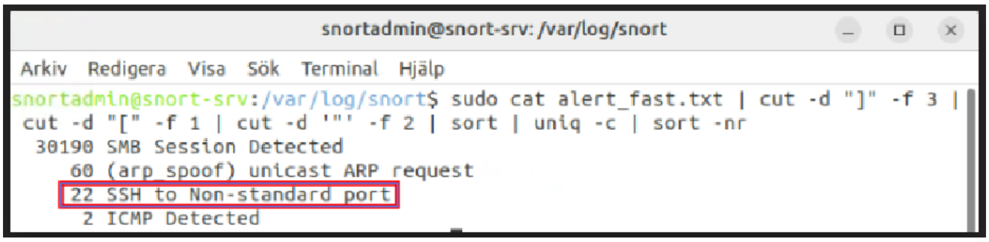
5.1.1.3 Snort Detection

The network traffic generated by the test scenarios was limited to PsExec and SSH related activities, resulting in minimal network telemetry compared to host telemetry. As seen in Figures 5.7 and 5.8, Snort was able to detect and inspect SSH and SMB communications and PsExec usage, although subtle behavioral patterns (such as PsExec escalation to local system) seems to not have been visible.


snortadmin@snort-srv: /var/log/snort
Arkiv
Redigera
Visa
Sök
Terminal
Hjälp

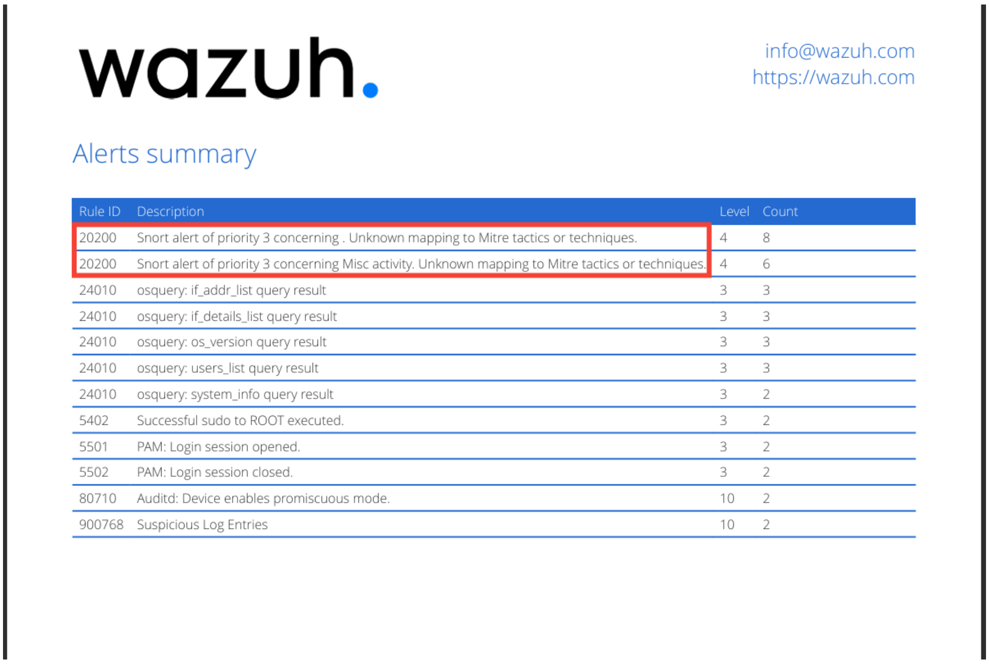
```
snortadmin@snort-srv:/var/log/snort$ sudo pluma alert_fast.txt
snortadmin@snort-srv:/var/log/snort$ sudo cat alert_fast.txt | cut -d "]" -f 3 |
 cut -d "[" -f 1 | cut -d '"' -f 2 | sort | uniq -c | sort -nr
     23 POLICY-OTHER use of psexec remote administration tool SMBv2
snortadmin@snort-srv:/var/log/snort$ 
```

Figure 5.7: Snort detection of PsExec usage


snortadmin@snort-srv: /var/log/snort
Arkiv
Redigera
Visa
Sök
Terminal
Hjälp

```
snortadmin@snort-srv:/var/log/snort$ sudo cat alert_fast.txt | cut -d "]" -f 3 |
 cut -d "[" -f 1 | cut -d '"' -f 2 | sort | uniq -c | sort -nr
  30190 SMB Session Detected
     60 (arp_spoof) unicast ARP request
     22 SSH to Non-standard port
      2 ICMP Detected
```

Figure 5.8: Snort detection of SSH connection



Alerts summary

Rule ID	Description	Level	Count
20200	Snort alert of priority 3 concerning . Unknown mapping to Mitre tactics or techniques.	4	8
20200	Snort alert of priority 3 concerning Misc activity. Unknown mapping to Mitre tactics or techniques.	4	6
24010	osquery: if_addr_list query result	3	3
24010	osquery: if_details_list query result	3	3
24010	osquery: os_version query result	3	3
24010	osquery: users_list query result	3	3
24010	osquery: system_info query result	3	2
5402	Successful sudo to ROOT executed.	3	2
5501	PAM: Login session opened.	3	2
5502	PAM: Login session closed.	3	2
80710	Auditd: Device enables promiscuous mode.	10	2
900768	Suspicious Log Entries	10	2

Figure 5.9: Depicts the Snort rules triggered in Wazuh

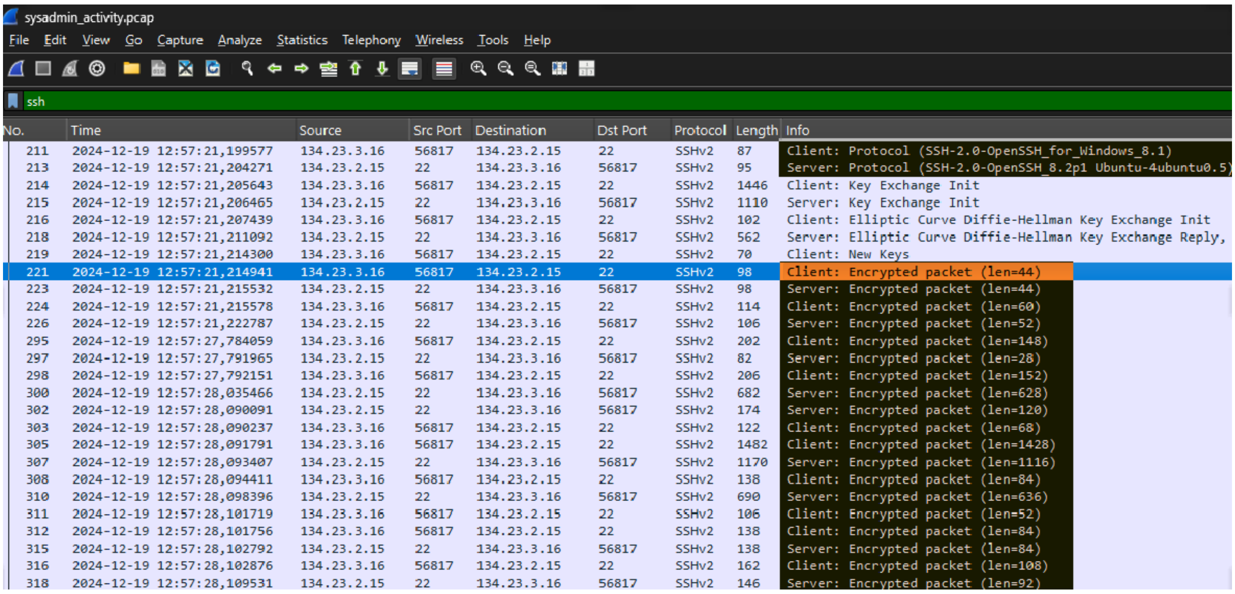
5.2 Analysis

5.2.1 Performance Analysis

Although both Sysmon and Snort are powerful security tools, they both serve different purposes and operate at different levels. Thus, their capabilities in detecting subtle behavioral patterns varied.

Sysmon was able to monitor system-level events, such as process creation, network connections. Using `sysmonconfig-with-filedelete` for configuration also allowed the detection of file deletion. A wealth of information could be gleaned from its logs, including command-line arguments, user and timestamp, as well visibility into behaviors such as privilege escalation (e.g., PsExec Local System Escalation). As already underscored in 5.1, in each test run, Sysmon reliably and consistently captured both administrative and adversarial activities. However, its scope was limited to inspecting only network traffic originating from the host itself (excluding any external network activity).

In contrast, Snort as a Network-based Intrusion Detection System (NIDS), primarily sniffs and inspects network traffic. Hence, only specific patterns in network packets indicative of PsExec and SSH usage were detected on the fly, during the test executions. Snort was able to detect SSH traffic to either the default port 22 or non-standard ports, but not the invocation of specific commands (such as `ProxyCommand`, for instance), as SSH traffic is encrypted (and cannot inspect the payload without decryption). As shown in Figure 5.10, the initial SSH handshake (that is, parts of the connection setup packets) is not encrypted, and from the moment the session is established, all SSH traffic between client and server is encrypted [49]. Thus, Snort relies on non-encrypted metadata, such as `Protocol Header` to be able to detect SSH usage. As could be inferred from Figure 5.10, the protocol headers `SSH-2.0-OpenSSH_for_Windows_8.1` and `SSH-2.0-OpenSSH_8.2p1_Ubuntu-4ubuntu0.5` are sent in plaintext. Subtle behavioral distinctions could not be identified solely using Snort, thus, it would be recommended to also use Host-based Intrusion Detection System (HIDS) like Sysmon as a complement to enhance endpoint visibility.



ssh

No.	Time	Source	Src Port	Destination	Dst Port	Protocol	Length	Info
211	2024-12-19 12:57:21,199577	134.23.3.16	56817	134.23.2.15	22	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_for_Windows_8.1)
213	2024-12-19 12:57:21,204271	134.23.2.15	22	134.23.3.16	56817	SSHv2	95	Server: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5
214	2024-12-19 12:57:21,205643	134.23.3.16	56817	134.23.2.15	22	SSHv2	1446	Client: Key Exchange Init
215	2024-12-19 12:57:21,206465	134.23.2.15	22	134.23.3.16	56817	SSHv2	1110	Server: Key Exchange Init
216	2024-12-19 12:57:21,207439	134.23.3.16	56817	134.23.2.15	22	SSHv2	102	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
218	2024-12-19 12:57:21,211092	134.23.2.15	22	134.23.3.16	56817	SSHv2	562	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply,
219	2024-12-19 12:57:21,214300	134.23.3.16	56817	134.23.2.15	22	SSHv2	70	Client: New Keys
221	2024-12-19 12:57:21,214941	134.23.3.16	56817	134.23.2.15	22	SSHv2	98	Client: Encrypted packet (len=44)
223	2024-12-19 12:57:21,215532	134.23.2.15	22	134.23.3.16	56817	SSHv2	98	Server: Encrypted packet (len=44)
224	2024-12-19 12:57:21,215578	134.23.3.16	56817	134.23.2.15	22	SSHv2	114	Client: Encrypted packet (len=60)
226	2024-12-19 12:57:21,222787	134.23.2.15	22	134.23.3.16	56817	SSHv2	106	Server: Encrypted packet (len=52)
295	2024-12-19 12:57:27,784059	134.23.3.16	56817	134.23.2.15	22	SSHv2	202	Client: Encrypted packet (len=148)
297	2024-12-19 12:57:27,791965	134.23.2.15	22	134.23.3.16	56817	SSHv2	82	Server: Encrypted packet (len=28)
298	2024-12-19 12:57:27,792151	134.23.3.16	56817	134.23.2.15	22	SSHv2	206	Client: Encrypted packet (len=152)
300	2024-12-19 12:57:28,035466	134.23.2.15	22	134.23.3.16	56817	SSHv2	682	Server: Encrypted packet (len=628)
302	2024-12-19 12:57:28,090091	134.23.2.15	22	134.23.3.16	56817	SSHv2	174	Server: Encrypted packet (len=120)
303	2024-12-19 12:57:28,090237	134.23.3.16	56817	134.23.2.15	22	SSHv2	122	Client: Encrypted packet (len=68)
305	2024-12-19 12:57:28,091791	134.23.3.16	56817	134.23.2.15	22	SSHv2	1482	Client: Encrypted packet (len=1428)
307	2024-12-19 12:57:28,093407	134.23.2.15	22	134.23.3.16	56817	SSHv2	1170	Server: Encrypted packet (len=1116)
308	2024-12-19 12:57:28,094411	134.23.3.16	56817	134.23.2.15	22	SSHv2	138	Client: Encrypted packet (len=84)
310	2024-12-19 12:57:28,098396	134.23.2.15	22	134.23.3.16	56817	SSHv2	690	Server: Encrypted packet (len=636)
311	2024-12-19 12:57:28,101719	134.23.3.16	56817	134.23.2.15	22	SSHv2	106	Client: Encrypted packet (len=52)
312	2024-12-19 12:57:28,101756	134.23.3.16	56817	134.23.2.15	22	SSHv2	138	Client: Encrypted packet (len=84)
315	2024-12-19 12:57:28,102792	134.23.2.15	22	134.23.3.16	56817	SSHv2	138	Server: Encrypted packet (len=84)
316	2024-12-19 12:57:28,102876	134.23.3.16	56817	134.23.2.15	22	SSHv2	162	Client: Encrypted packet (len=108)
318	2024-12-19 12:57:28,109531	134.23.2.15	22	134.23.3.16	56817	SSHv2	146	Server: Encrypted packet (len=92)

Figure 5.10: Illustrates the encrypted SSH traffic/packets from sysadmin activities as captured in Wireshark

When it comes to Wazuh and sigma-signatures, Wazuh proved effective at ingesting logs, correlating data, and visualizing alerts. Because the signatures used were modeled based on the LotL techniques, they were able to consistently trigger FAs during each test run. However, signatures require refinement and fine-tuning, as even a slight variation in the command-line arguments or procedures could easily have affected the alert efficiency.

5.2.2 Log Analysis

Despite the LotL techniques used by both parties triggering identical detection signatures, network defenders can still use contextual analysis to determine whether a specific activity is benign or malicious in nature. The log analysis revealed subtle yet significant differences in behavioral patterns. For instance, although Certutil was used for encoding in both scenarios, the context differed: in the legitimate sysadmin case, it was used in the context of certificate management, while in the adversarial case, it was used to encode an executable as a text file for obfuscation purposes. Similarly, the use of Msiexec for quiet installation also varied. In the sysadmin simulation, it was utilized to install legitimate MSI packages. In contrast, in the adversarial emulation, it was leveraged for executing MSI files containing embedded DLLs. Yet another example was the use of SSH as a proxy. In one version of adversarial emulation, this LotL technique was used exploited to execute a binary as part of a defense evasion tactic. Conversely, in the legitimate case, SSH was used within the context of accessing isolated networks through SSH gateway.

In production environments, additional IOCs, such as whether the activity occurred during or outside normal working hours (i.e., unusual timing), or whether it was conducted by authorized user accounts with privileges (i.e., user role), geographic anomalies, etc can further aid in accurately distinguishing between benign and malicious activities.

5.2.3 Reliability and Validity Analysis

To ensure the reliability of the findings, it was vital that the simulations used to depict legitimate administrative activities and malicious behaviors needed to be grounded in real-world scenarios. Thus, the simulated sysadmin actions were based on LotL techniques identified by the interview and survey participants as typical administrative behaviors, as described in the subsection 3.2.2. Moreover, the atomic tests leveraged are aligned with the MITRE

ATT&CK® framework, which models adversarial behaviors observed in the wild.

In addition, it was necessary to repeat the test scenarios several times – for both administrative and adversarial emulations. And as anticipated, both activities were able to consistently trigger the same detection signatures as the default configurations and rules used remained constant. And this consistency can be attributed to the fact that the detection signatures used, were modeled and fine-tuned to match the specific win.eventdata.CommandLine patterns observed in the Sysmon log events generated by the emulations.

Furthermore, efforts were made to ensure content validity, by striving to cover all relevant aspects of LotL techniques – specifically those related to the LOLBins investigated – in the interview/survey questions. Moreover, the ART framework – which maps to MITRE ATT&CK® – was used to replicate threat actor's TTPs seen in the wild. This alignment ensures that the atomic tests accurately reflect genuine adversary behaviors, thereby strengthening their construct validity by ensuring they measure what they are intended to represent.

Chapter 6

Discussion

This chapter presents a brief discussion and reflection regarding the findings, implications, method choice, improvement recommendations, and contributions.

6.1 Findings

This study addressed three main research questions:

What are some of the most common LotL binaries used by both system administrators and threat actors?

The project identified LOLBins such as `PowerShell`, `Windows command shell`, `rundll32`, `MSBuild`, `PsExec`, `SSH`, `WMIC`, `certutil`, `msiexec` and `netsh` as some of the most prevalent binaries encountered in the wild, with `PsExec`, `Msiexec`, `SSH`, `CertUtil`, `PowerShell` and `CMD` being some of the commonly shared utilities by both admins and attackers. Their dual-usage often blurs the line between benign and malicious activities. Specific use cases for these binaries range from software installation (`Msiexec`) to remote services (`SSH`, `PsExec`) to certificate management (`CertUtil`) and command execution (`Cmd`, `PowerShell`).

What are the overlaps and differences in the administrative and adversarial usage of LotL binaries?

Overlaps in LotL behaviors were observed in activities such as `PsExec` privilege escalation, `SSH` tunneling, `Msiexec` quiet installation, `Certutil` encoding and `CMD` greedy deletion. There were also some minor differences in behavioral traits; for instance, sysadmins typically avoid executing files immediately after download, whereas attackers commonly do. RDP tunneling over `SSH` and file downloads via `CertUtil` were some of the techniques tagged as adversarial.

How effective are signatures in accurately differentiating between legitimate administrator activities and malicious attacks, given the use of the same LotL binaries?

The Empirical findings showed that signature-based IDS are prone to false positives (which in actuality where BTs) due to their reliance on static pattern. While `Sysmon`, `Wazuh` – and Snort, in a limited manner – were able to alert on suspicious activity quite well, their output always required contextual analysis to determine intent, thus showing the limitation of signature-based detection in dynamic settings.

One of the main emphasis of this thesis was to trigger FAs in an effort to distinguish benign and malicious activities when identical LotL behaviors are exhibited. In order to reduce FPs and better distinguish threat from noise, the sigma rules can be further enhanced and fine-tuned by including additional filtering conditions. For instance, the sigma rule for Certutil decoding [79] can be modified to include conditions to flag potentially dangerous (unsafe) file types, such as executable files (`.exe`, `.dll`, `.bat`), archive files (`.rar`, `.zip`), and script files (`.ps1`, `.js`) that attackers typically use to deobfuscate files and bypass detection [80] [81] [82] [83]. By specifically monitoring encoding and decoding operations that involve these high-risk file types, network defenders can more effectively identify potentially malicious abuse of system utilities while at the same time reducing FAs from routine administrative usage. Listing 6.1 illustrate the modified sigma rule for detecting suspicious certutil decoding, while Listing 6.2 shows the Wazuh rule version. This refinement could help delineate between the benign sysadmin usage of certutil for certificate management and the attacker's use for deobfuscation of malicious payload as well as minimize FPs.

Listing 6.1: Sigma Rule: Suspicious Certutil Decoding

```
title: Suspicious Certutil Decoding
id: 769c780a-396f-41de-afa9-4dd054942a04
related:
    - id: cc9cbe82-7bc0-4ef5-bc23-bbfb83947be7
      type: derived
status: experimental
description: Detects potential malicious decoding of files using certutil
references:
    - https://attack.mitre.org/techniques/T1140/
    - https://ezprotect.io/what-are-high-risk-file-types-and-how-to-block-\
    ↪ users-from-accessing-them-in-salesforce/?
    - https://cloud.google.com/blog/topics/threat-intelligence/targeted-\
    ↪ attack-in-middle-east-by-apt34/
    - https://lolbas-project.github.io/lolbas/Binaries/Certutil/
    - https://news.sophos.com/en-us/2021/04/13/compromised-exchange-server\
    ↪ -hosting-cryptojacker-targeting-other-exchange-servers/
author: Peter Daniel
date: 2025-02-04
tags:
    - attack.defense-evasion
    - attack.t1140
logsource:
    category: process_creation
    product: windows
detection:
    selection_img:
        - Image|endswith: '\certutil.exe'
        - OriginalFileName: 'CertUtil.exe'
    selection_cli:
        CommandLine|contains:
            - '-decode'
            - '/decode'
            - '-decodehex'
            - '/decodehex'
    selection_extensions:
        CommandLine|contains:
              # Program files
              - '.exe'
              - '.dll'
              - '.scr'
              # Batch files
              - '.bat'
              - '.cmd'
              # Script files
              - '.ps1'
              - '.vbs'
              - '.js'
              - '.jse'
              - '.py'
              - '.rb'
              - '.php'
              - '.asp'
              - '.aspx'
              # Archive files
              - '.zip'
              - '.rar'
              - '.7z'
              - '.cab'
              - '.tar'
              - '.gz'
              # Encoded/encrypted files
              - '.b64'
              - '.dat'
              - '.enc'
              - '.bin'
              # Shortcut/system files
              - '.lnk'
              - '.url'
              - '.inf'
              - '.hta'
              - '.dmp'
              # Disk images/virtual disk files
              - '.iso'
              - '.img'
              - '.vhd'
              - '.vhdx'
              # Macro-enabled document files
              - '.docm'
              - '.xlsm'
              - '.pptm'
              # Windows Registry files
              - '.reg'
    condition: all of selection_*
falsepositives:
    - Unknown
level: high
```

Listing 6.2: A Wazuh rule for detecting suspicious certutil decoding

```
<!-- Suspicious Certutil Decoding -->
<rule id=”100002” level=”12”>
    <info type=”text”>Sigma Rule Author: Peter Daniel</info>
    <info type=”text”>Detects potential malicious encoding of files using
    certutil, targeting specific file extensions</info>
    <info type=”text”>Date: 2025-02-04</info>
    <info type=”text”>Status: experimental</info>
    <info type=”text”>ID: 769c780a-396f-41de-afa9-4dd054942a048</info>
    <info type=”link”>https://attack.mitre.org/techniques/T1140/</info>
    <info type=”text”>Falsepositives: Unknown</info>
    <mitre>
        <id>T1140</id>
    </mitre>
    <description>Suspicious Certutil Encoding Detected</description>
    <options>no_full_log</options>
    <if_group>sysmon</if_group>
    <field name=”win.eventdata.Image” negate=”no”
    type=”pcre2”>(?i).+\\certutil\.exe$</field>
    <field name=”win.eventdata.OriginalFileName” negate=”no”
    type=”pcre2”>(?i)CertUtil\.exe</field>
    <field name=”win.eventdata.CommandLine” negate=”no”
    type=”pcre2”>(?i).*(-decode|-decodehex).*</field>
    <field name=”win.eventdata.CommandLine” negate=”no” type=”pcre2”>
        (?i).*\.(exe|dll|bat|cmd|scr|ps1|vbs|js|zip|rar).*
    </field>
</rule>
```

6.2 Implications

The findings of this study have some implications for the development and deployment of IDS solutions. Embedding behavioral and contextual analysis into detection mechanisms can reduce FPs significantly. While signatures are essential, their efficacy is limited when used in isolation, without supplemental behavior-based detection and context analysis. This underscores the importance of integrating Machine Learning (ML)-based anomaly detection solutions and training analysts.

6.3 Methodology Choice

This research adopted a mixed-method approach by combining theoretical insights with empirical validations. The literature review laid the foundation for identifying overlapping LotL techniques; the qualitative research (i.e., semi-structured interviews and surveys) enriched the findings with real-world perspectives; while empirical tests validated the findings in a controlled environment. This approach allowed for a robust investigation of the RQs.

6.4 Reflections

When converting the Sigma signatures into Wazuh rules, in an effort to fine-tune and adapt them to the simulated sysadmin procedures and atomic tests, a looser variant of the Sigma rules was sometimes used to deliberately trigger FAs. For instance, when testing the LotL technique `SSH Proxy Execution` [84], the signatures specifically monitored and matched the invocation of `ProxyCommand`, since both the emulations did not include the invocation of `LocalCommand` upon a successful SSH connection. That being said, it's worth investigating whether the presence of `LocalCommand` directive in conjunction with `ProxyCommand` can serve as a meaningful IOC. Although the execution of a local command once the SSH connection has been established could be exploited for malicious purposes, such as automation, evasion of detection mechanisms, and maintaining persistence on compromised systems [85], further investigation is needed to determine whether it has legitimate administrative use cases beyond mere convenience.

6.5 Recommendations for Improvement

Sysmon provides a wealth of event data fields (e.g., `win.eventdata.user`, `win.system.computer`, `win.eventdata.UtcTime`), which can be appended to the Wazuh custom rules to further enhance the detection capability and also supply network defenders with better contextual data to accurately distinguish between benign and malicious LotL activities.

In addition, incorporating behavioral analytics alongside anomaly detection techniques could further strengthen detection to spot deviations from baseline behaviors and detect outliers. Signature-based detection is at a disadvantage, when it comes to detecting novel and nuanced attacks and require perpetual fine-tuning and refinement to maintain secure organizational posture.

Some other key recommendations for improvement include:

- Involving log analysts during the testing phase to observe their accuracy in distinguishing between benign and malicious LotL activities
- Conducting interviews and surveys with larger and more diverse participant populations to identify broader patterns
- Comparing Signature vs. Anomaly-based IDS would have provided valuable insights into their respective strengths, weaknesses, and real-world applicability in terms of LotL techniques
- Including non-Windows binaries and cloud environments for broader testing

6.6 Atomic Tests and Sigma Contributions

Four new atomic tests have been developed to emulate specific adversarial techniques, as outlined in Tables 4.2 and 4.3. Notable contributions include the creation of tests for executing commands indirectly via SSH (`T1202-6`) (without invoking cmd) for defense evasion; leveraging SSH Proxying (`T1021.004-3`) and SSH Remote Port Forwarding (`T1021.004-4`) for lateral movement. In addition, a test was developed to simulate file deletion using greedy wildcard expressions (`T1070.004-12`). Each test was implemented with either a `PowerShell` or `CMD` executor. In addition, a Sigma rule titled `Suspicious Certutil Decoding` (see Listing 6.1) has been developed and added to the SigmaHQ repository.

Chapter 7

Conclusions and Future work

7.1 Conclusions

This project set out to examine some of the prevalent Windows binaries observed in the wild, focusing on identifying intersecting LotL behaviors commonly shared by both admins and adversaries. By emulating these behaviors, the study intentionally triggered FAs to test how effectively security monitoring tools can differentiate between legitimate activities and potential threats.

The results demonstrate that while both host- and network-based detection systems can detect LotL behaviors used by administrators and attackers alike, distinguishing between the two relies on network defenders conducting a contextual evaluation. Key insights highlight the necessity of combining behavioral analytics with well-tuned detection signatures to minimize false positives and improve detection accuracy.

This work underscores the overlapping LotL techniques pose a major challenge in separating legitimate from malicious activity. Findings showed that while current signature-based intrusion detection systems effectively flag LotL behaviors, they struggle to reliably tell apart benign and malicious actions, leading to an overwhelming number of alerts, many of which are BTs.

The study emphasizes the need to improve detection strategies by incorporating contextual analysis, refining detection rules, and using behavioral detection methods. Such enhancements would reduce false alerts and significantly boost the accuracy and efficiency of security systems.

7.2 Limitations

This research focused exclusively on Windows binaries, investigating and testing with but a small subset of LOLBins. Only the LotL techniques that were deemed valid administrative behavior by the interview participants were simulated. Despite efforts to include as many participants from various organizations, the limited number of participants restricted the testing of additional LotL techniques. Moreover, cloud scenarios were excluded due to lack of cloud environment in CRATE. Also, the security monitoring tools utilized in this study were limited to signature-based detection methods. Finally, even though the emulations were conducted in a controlled cyber range designed to mimic real-world networks, they may not fully capture the complexity and unpredictability of production environments.

7.3 Future work

Future research can extend this work in multiple ways. One research avenue is further exploring non-Windows LotL binaries, such as Linux binaries (`GTFOBins`) [31], macOS binaries (`LOOBins`) [32], cloud binaries [33] [34] as well as vulnerable Windows drivers (`LOLDrivers`) [35] [86] [87], to better understand LotL techniques across ecosystems, given the shift towards cloud-native architectures and the rapid adoption of Linux-based IoT devices [88].

Another research area that merits further exploration is leveraging Large language models (LLMs) capabilities for contextual understanding, anomaly detection and adaptive learning to improve IDS in distinguishing between benign and malicious LotL usage [89]. LLM-based IDS solutions could play a vital role in the evolving threat landscape.

Future research should also focus on integrating ML, NLP [90] and AI-driven approaches into traditional IDS solutions to enhance the accuracy of LotL attack detection. Recent studies leveraging deep learning frameworks, such as [70] [72], warrant further exploration.

Research Artifacts

Research artifacts can be accessed at: https://github.com/cybercampus-se/triggering-false-alarms.git

References

[1] F. Barr-Smith, X. Ugarte-Pedrero, M. Graziano, R. Spolaor, and I. Martinovic, “Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land,” in *2021 IEEE Symposium on Security and Privacy (SP)*, May 2021. doi: 10.1109/SP40001.2021.00047 pp. 1557–1574, iSSN: 2375-1207. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9519480 [Pages 1, 2, 11, 12, 20, 21, 23, and 37.]

[2] markruss, “PsExec - Sysinternals,” Mar. 2023. [Online]. Available: https://learn.microsoft.com/en-us/sysinternals/downloads/psexec [Pages 1, 7, 22, and 29.]

[3] “PsExec, Software S0029 | MITRE ATT&CK®.” [Online]. Available: https://attack.mitre.org/software/S0029/ [Page 1.]

[4] “Quantum Ransomware,” Apr. 2022. [Online]. Available: https://thedfirreport.com/2022/04/25/quantum-ransomware/ [Page 1.]

[5] “MITRE ATT&CK T1059 Command and Scripting Interpreter.” [Online]. Available: https://www.picussecurity.com/resource/blog/mitre-attck-t1059-command-and-scripting-interpreter [Page 1.]

[6] “https://www.cisa.gov/sites/default/files/2024-02/Joint-Guidance-Identifying-and-Mitigating-LOTL_v3508c.pdf.” [Online]. Available: https://www.cisa.gov/sites/default/files/2024-02/Joint-Guidance-Identifying-and-Mitigating-LOTL_V3508c.pdf [Pages 1, 3, 11, 12, 16, 23, and 37.]

[7] H. Carvey, “Adversary Extends Persistence by Modifying System Binaries,” Dec. 2018. [Online]. Available: https://www.crowdstrike.com/blog/adversary-extends-persistence-by-modifying-system-binaries/ [Page 1.]

[8] “8 LOLBINS EVERY THREAT HUNTER SHOULD KNOW.” [Online]. Available: https://go.crowdstrike.com/rs/281-OBQ-266/images/WhitepaperHuntingForLOLBins2023.pdf?mkt_tok=MjgxLU9CUS0yNjYAAAGWFLoDT1i3idExqs2s7bApbJxnBv6-W5CQkYI01OH84Dtd2efuxOW08MrYhj79Np7cIZfkXJ8DkuLZkByd3qwB5K5c4YAZ4-YeoUwIF1tkBnsLNCfs [Pages 1, 3, 21, 22, 23, 30, and 37.]

[9] J. R. Goodall, W. G. Lutters, and A. Komlodi, “I know my network: collaboration and expertise in intrusion detection,” in *Proceedings of the 2004 ACM conference on Computer supported cooperative work*. Chicago Illinois USA: ACM, Nov. 2004. doi: 10.1145/1031607.1031663. ISBN 978-1-58113-810-8 pp. 342–345. [Online]. Available: https://dl.acm.org/doi/10.1145/1031607.1031663 [Pages 1 and 2.]

[10] M. Nabil, S. Soukainat, A. Lakbabi, and O. Ghizlane, “SIEM selection criteria for an efficient contextual security,” in *2017 International Symposium on Networks, Computers and Communications (ISNCC)*, May 2017. doi: 10.1109/ISNCC.2017.8072035 pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8072035 [Page 2.]

[11] S. Luo and G. Marin, “Generating Realistic Network Traffic for Security Experiments,” in *IEEE SoutheastCon, 2004. Proceedings.*, Mar. 2004. doi: 10.1109/SECON.2004.1287918 pp. 200–207. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1287918 [Pages 2, 3, and 21.]

[12] B. A. Alahmadi, L. Axon, and I. Martinovic, “99% False Positives: A Qualitative Study of SOC Analysts’ Perspectives on Security Alarms.” [Pages 2, 3, and 20.]

[13] K. Knerler, “11 Strategies of a World-Class Cybersecurity Operations Center.” [Page 2.]

[14] Xelu86, “certutil,” Oct. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/certutil [Pages 3, 7, 22, and 30.]

[15] “CertUtil Qualms: They Came to Drop FOMBs - Malware News,” Oct. 2019, section: Malware News. [Online]. Available: https:

//malware.news/t/certutil-qualms-they-came-to-drop-fombs/34283 [Pages 3 and 22.]

[16] "CRATE – Sweden's national cyber training facility." [Online]. Available: https://www.foi.se/en/foi/research/information-security/crate---swedens-national-cyber-training-facility.html [Page 5.]

[17] M. Q. Patton, *Qualitative Research & Evaluation Methods: Integrating Theory and Practice*. SAGE Publications, Oct. 2014. ISBN 978-1-4833-0145-7 Google-Books-ID: ovAkBQAAQBAJ. [Pages 6, 7, and 26.]

[18] "LOLBAS/README.md at master · LOLBAS-Project/LOLBAS." [Online]. Available: https://github.com/LOLBAS-Project/LOLBAS/blob/master/README.md [Pages 6, 9, 11, 12, 20, and 37.]

[19] "ATT&CK® Navigator." [Online]. Available: https://mitre-attack.github.io/attack-navigator/#layerURL=https://lolbas-project.github.io/mitre_attack_navigator_layer.json [Pages 6, 9, 11, 20, and 37.]

[20] "SigmaHQ/sigma," Oct. 2024, original-date: 2016-12-24T09:48:49Z. [Online]. Available: https://github.com/SigmaHQ/sigma [Pages 6, 9, 11, 17, 20, and 37.]

[21] robinharwood, "msiexec," Feb. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/msiexec [Pages 7, 22, and 29.]

[22] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Protocol Architecture," Internet Engineering Task Force, Request for Comments RFC 4251, Jan. 2006, num Pages: 30. [Online]. Available: https://datatracker.ietf.org/doc/rfc4251 [Pages 7 and 29.]

[23] robinharwood, "wmic," Feb. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/wmic [Pages 7 and 29.]

[24] trumanbrown msft, "Using WinGet to Install Apps on Windows IoT Enterprise," May 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/iot/iot-enterprise/deployment/install-winget-windows-iot [Pages 7 and 29.]

[25] robinharwood, “wbadmin,” Feb. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/wbadmin [Pages 7 and 30.]

[26] Deland-Han, “Use Ntdsutil to manage AD files - Windows Server,” Feb. 2024. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-server/active-directory/use-ntdsutil-manage-ad-files [Pages 7 and 30.]

[27] Xelu86, “cmd,” Sep. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cmd [Pages 7 and 30.]

[28] robinharwood, “cmdkey,” Nov. 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cmdkey [Pages 7 and 30.]

[29] “Home.” [Online]. Available: https://www.autoitscript.com/site/ [Pages 7 and 38.]

[30] “redcanaryco/atomic-red-team,” Oct. 2024, original-date: 2017-10-11T17:23:32Z. [Online]. Available: https://github.com/redcanaryco/atomic-red-team [Pages 7, 23, and 38.]

[31] “view | GTFOBins.” [Online]. Available: https://gtfobins.github.io/gtfobins/view/ [Pages 8, 12, and 82.]

[32] “All LOOBins.” [Online]. Available: https://www.loobins.io/binaries/ [Pages 8, 12, and 82.]

[33] R. Pliskin, “Azure LoLBins: Protecting against the dual use of virtual machine extensions,” Mar. 2021. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2021/03/09/azure-lolbins-protecting-against-the-dual-use-of-virtual-machine-extensions/ [Pages 8 and 82.]

[34] GabstaMSFT, “Azure VM extensions and features for Windows - Azure Virtual Machines,” Aug. 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-machines/extensions/features-windows [Pages 8 and 82.]

[35] “LOLDrivers.” [Online]. Available: https://www.loldrivers.io/ [Pages 8, 12, and 82.]

[36] "Living off the Land White Paper." [Page 11.]

[37] "Thrip: Espionage Group Hits Satellite, Telecoms, and Defense Companies." [Online]. Available: https://www.security.com/threat-intelligence/thrip-hits-satellite-telecoms-defense-targets [Page 11.]

[38] "APT29, IRON RITUAL, IRON HEMLOCK, NobleBaron, Dark Halo, StellarParticle, NOBELIUM, UNC2452, YTTRIUM, The Dukes, Cozy Bear, CozyDuke, SolarStorm, Blue Kitsune, UNC3524, Midnight Blizzard, Group G0016 | MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/groups/G0016/ [Page 12.]

[39] "POSHSPY, Software S0150 | MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/software/S0150/ [Page 12.]

[40] sdwheeler, "What is PowerShell? - PowerShell," Mar. 2024. [Online]. Available: https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.4 [Pages 12 and 22.]

[41] stevewhims, "Windows Management Instrumentation - Win32 apps," Mar. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/wmisdk/wmi-start-page [Page 12.]

[42] "Dissecting One of APT29's Fileless WMI and PowerShell Backdoors (POSHSPY) | Mandiant." [Online]. Available: https://cloud.google.com/blog/topics/threat-intelligence/dissecting-one-ofap [Page 12.]

[43] "MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/ [Page 12.]

[44] "FAQ | MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/resources/faq/ [Page 12.]

[45] "Techniques - Enterprise | MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/techniques/enterprise/ [Page 12.]

[46] "APT41, Wicked Panda, Brass Typhoon, BARIUM, Group G0096 | MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/groups/G0096/ [Page 12.]

[47] "Brute Force: Password Cracking, Sub-technique T1110.002 - Enterprise | MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/techniques/T1110/002/ [Page 13.]

[48] "LOLBAS-Project/LOLBAS," Oct. 2024, original-date: 2018-06-08T22:11:05Z. [Online]. Available: https://github.com/LOLBAS-Project/LOLBAS [Pages 15 and 30.]

[49] "SSH - Wireshark Wiki." [Online]. Available: https://wiki.wireshark.org/SSH [Pages 15, 49, and 70.]

[50] "People's Republic of China State-Sponsored Cyber Actor Living off the Land to Evade Detection | CISA," May 2023. [Online]. Available: https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-144a [Page 16.]

[51] M. T. Intelligence, "Volt Typhoon targets US critical infrastructure with living-off-the-land techniques," May 2023. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2023/05/24/volt-typhoon-targets-us-critical-infrastructure-with-living-off-the-land-techniques/ [Page 16.]

[52] Xelu86, "Netsh Command Syntax, Contexts, and Formatting," Oct. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh-contexts [Page 20.]

[53] "netsh, Software S0108 | MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/software/S0108/ [Pages 20 and 22.]

[54] "Desktop Operating System Market Share Worldwide." [Online]. Available: https://gs.statcounter.com/os-market-share/desktop/worldwide [Page 20.]

[55] "https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2021/07/20155845/MDR_analyst_report_q4-2020.pdf." [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2021/07/20155845/MDR_Analyst_Report_Q4-2020.pdf [Pages 21, 22, 23, and 37.]

[56] "2022 Year in Review," Mar. 2023. [Online]. Available: https://thedfirreport.com/2023/03/06/2022-year-in-review/ [Page 21.]

[57] robinharwood, "rundll32," Feb. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/rundll32 [Page 22.]

[58] "Rundll32 on LOLBAS." [Online]. Available: https://lolbas-project.github.io/lolbas/Binaries/Rundll32/ [Page 22.]

[59] "certutil, Software S0160 | MITRE ATT&CK®." [Online]. Available: https://attack.mitre.org/software/S0160/ [Pages 22 and 28.]

[60] robinharwood, "reg commands," Feb. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/reg [Page 22.]

[61] "Reg on LOLBAS." [Online]. Available: https://lolbas-project.github.io/lolbas/Binaries/Reg/ [Page 22.]

[62] DOMARS, "Te.exe Command Options - Windows drivers," Dec. 2021. [Online]. Available: https://learn.microsoft.com/en-us/windows-hardware/drivers/taef/te-exe-command-line-parameters [Page 22.]

[63] "Te on LOLBAS." [Online]. Available: https://lolbas-project.github.io/lolbas/OtherMSBinaries/Te/ [Page 22.]

[64] robinharwood, "wscript," May 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/wscript [Page 22.]

[65] "Wscript on LOLBAS." [Online]. Available: https://lolbas-project.github.io/lolbas/Binaries/Wscript/ [Page 22.]

[66] "Mshta - Red Canary Threat Detection Report." [Online]. Available: https://redcanary.com/threat-detection-report/techniques/mshta/ [Page 22.]

[67] robinharwood, "regsvr32," Feb. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/regsvr32 [Page 22.]

[68] ——, "netsh," Jun. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/netsh [Page 22.]

[69] R. Stamp, *Living-off-the-Land Abuse Detection Using Natural Language Processing and Supervised Learning*, Aug. 2022. [Page 23.]

[70] T. Ongun, J. W. Stokes, J. B. Or, K. Tian, F. Tajaddodianfar, J. Neil, C. Seifert, A. Oprea, and J. C. Platt, "Living-Off-The-Land Command Detection Using Active Learning," in *24th International Symposium on Research in Attacks, Intrusions and Defenses*, Oct. 2021. doi: 10.1145/3471621.3471858 pp. 442–455, arXiv:2111.15039 [cs]. [Online]. Available: http://arxiv.org/abs/2111.15039 [Pages 23 and 82.]

[71] R. Tarek, S. Chaimae, and C. Habiba, "Runtime API Signature for Fileless Malware Detection," in *Advances in Information and Communication*, K. Arai, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-39445-5$_{4}7.ISBN978-3-030-39445-5pp.645--654.[Page$ 23.]

[72] K. Ding, S. Zhang, F. Yu, and G. Liu, "LOLWTC: A Deep Learning Approach for Detecting Living Off the Land Attacks," in *2023 IEEE 9th International Conference on Cloud Computing and Intelligent Systems (CCIS)*, Aug. 2023. doi: 10.1109/CCIS59572.2023.10262997 pp. 176–181, iSSN: 2376-595X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10262997 [Pages 23 and 82.]

[73] "The DFIR Report," Sep. 2024. [Online]. Available: https://thedfirreport.com/ [Page 23.]

[74] O. Hartong, "olafhartong/sysmon-modular," Sep. 2024, original-date: 2018-01-13T21:20:59Z. [Online]. Available: https://github.com/olafhartong/sysmon-modular [Page 23.]

[75] markruss, "Sysmon - Sysinternals," Jul. 2024. [Online]. Available: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon [Page 23.]

[76] SwiftOnSecurity, "SwiftOnSecurity/sysmon-config," Jan. 2025, original-date: 2017-02-01T18:49:36Z. [Online]. Available: https://github.com/SwiftOnSecurity/sysmon-config [Page 53.]

[77] "Wireshark · Go Deep." [Online]. Available: https://www.wireshark.org/ [Page 56.]

[78] "Installing Atomic Red Team." [Online]. Available: https://github.com/redcanaryco/invoke-atomicredteam/wiki/Installing-Atomic-Red-Team [Page 58.]

[79] "sigma/rules/windows/process_creation/proc_creation_win_certutil_decode.yml at master · SigmaHQ/sigma." [Online]. Available: https:

//github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_certutil_decode.yml [Page 74.]

[80] M. Meyers, “What Are High-Risk File Types and How to Block Users from Accessing Them in Salesforce,” Apr. 2024. [Online]. Available: https://ezprotect.io/what-are-high-risk-file-types-and-how-to-block-users-from-accessing-them-in-salesforce/ [Page 74.]

[81] “An overview of unsafe file types in Microsoft products - Microsoft Support.” [Online]. Available: https://support.microsoft.com/en-us/topic/an-overview-of-unsafe-file-types-in-microsoft-products-266a9bd3-50d7-d65a-8fe0-ec4e486c3a14? [Page 74.]

[82] “Obfuscated Files or Information, Technique T1027 - Enterprise | MITRE ATT&CK®.” [Online]. Available: https://attack.mitre.org/techniques/T1027/ [Page 74.]

[83] “OilRig, COBALT GYPSY, IRN2, APT34, Helix Kitten, Evasive Serpens, Hazel Sandstorm, EUROPIUM, ITG13, Group G0049 | MITRE ATT&CK®.” [Online]. Available: https://attack.mitre.org/groups/G0049/? [Page 74.]

[84] “sigma/rules/windows/process_creation/proc_creation_win_ssh_proxy_execution.yml at master · SigmaHQ/sigma.” [Online]. Available: https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_ssh_proxy_execution.yml [Page 77.]

[85] “Stealthy Cyber Attacks: LNK Files & SSH Commands Playbook,” Dec. 2024. [Online]. Available: https://cyble.com/blog/a-stealthy-playbook-for-advanced-cyber-attacks/ [Page 77.]

[86] D. Mellinger, “Hunting Vulnerable Kernel Drivers,” Oct. 2023. [Online]. Available: https://blogs.vmware.com/security/2023/10/hunting-vulnerable-kernel-drivers.html [Page 82.]

[87] “These Are The Drivers You Are Looking For: Detect and Prevent Malicious Drivers | Splunk.” [Online]. Available: https://www.splunk.com/en_us/blog/security/these-are-the-drivers-you-are-looking-for-detect-and-prevent-malicious-drivers.html? [Page 82.]

[88] F. Dang, Z. Li, Y. Liu, E. Zhai, Q. A. Chen, T. Xu, Y. Chen, and J. Yang, “Understanding Fileless Attacks on Linux-based IoT Devices with HoneyCloud,” in *Proceedings of the 17th Annual International Conference*

on Mobile Systems, Applications, and Services, ser. MobiSys '19. New York, NY, USA: Association for Computing Machinery, Jun. 2019. doi: 10.1145/3307334.3326083. ISBN 9781450366618 pp. 482–493. [Online]. Available: https://doi.org/10.1145/3307334.3326083 [Page 82.]

[89] O. G. Lira, A. Marroquin, and M. A. To, "Harnessing the Advanced Capabilities of LLM for Adaptive Intrusion Detection Systems," in *Advanced Information Networking and Applications*, L. Barolli, Ed. Cham: Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-57942-$4_4 4. ISBN 9783031579424 pp. 453 - -464. [Page$ 82.]

[90] R. Ning, W. Bu, J. Yang, and S. Duan, "A Survey of Detection Methods Research on Living-Off-The-Land Techniques," in *2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE)*, Aug. 2023. doi: 10.1109/ICSECE58870.2023.10263445 pp. 159–164. [Online]. Available: https://ieeexplore.ieee.org/document/10263445 [Page 82.]

Appendix A

Semi-Structured Interview Questions

The questions in the semi-structured interviews were the same as those in the online survey and were tailored specifically for system administrators and log analysts. The Windows binaries included were: PsExec, Msiexec, SSH, Wmic, Wbadmin, Winget, Certutil, Ntdsutil, Cmdkey and Cmd. The questions related to each binary are categorized based on specific LotL techniques. The two versions of the questions are presented below.

A.1 Interview: System Administrator

Introductory Questions

1. Could you please describe your background and role?
2. Do you primarily work with Windows, Linux systems, or both?
3. Which living-off-the-land binaries (LOLbins) do you frequently use when managing [Windows/Linux] systems?
4. Are you familiar with any of these binaries: PsExec, MSIExec, SSH, WMIC, Winget, WBAdmin, CertUtil, Ntdsutil, CMD, or Cmdkey? If so, which ones do you use regularly, and for what purposes?

SSH-Related Questions

1. **Using SSH as a Proxy:**

   i. Do you find it common to use SSH as a proxy to launch programs on remote systems?
   ii. Could you share an example of how you've used the Proxy-Command option in a legitimate context?
   iii. When using SSH for proxying, are there commands or switches that might indicate whether an action is legitimate or malicious? How does an administrator's use differ from an attacker's?

2. **SSH for Port Forwarding:**
   i. Although SSH port forwarding is often associated with software developers and testers, do you see administrative use cases for this technique?
   ii. Can you describe a few legitimate scenarios where an admin might use SSH port forwarding?

3. **SSH for RDP Tunneling:**
   i. How might system administrators use SSH to tunnel Remote Desktop Protocol (RDP) connections?
   ii. Could SSH-based RDP tunneling potentially resemble adversarial behavior, like data exfiltration? How might an admin's approach differ from an attacker's?

4. **SSH and Adversary-like Behavior:**
   i. Are there other ways administrators might use SSH that could resemble adversarial behavior, possibly leading to false positives in security monitoring systems?

PsExec-Related Questions

1. **Remote Execution with PsExec:**
   i. How often do you use PsExec for executing commands on remote systems (-u, -p options)?
   ii. Could you provide an example of a legitimate scenario where you used PsExec for remote installation or scripting?

   iii. Have you used PAExec (i.e., a PsExec variant developed by PowerAdmin) instead of PsExec, given its improvements in handling sensitive information by encrypting command-line parameters?
   iv. Are there specific combinations of arguments (e.g., `\\`, `-p`, `-u`) that may resemble adversarial behavior but are commonly used by sysadmins?

2. **Privilege Escalation to Local System:**
   i. Can you share an instance where you escalated privileges to the local system account using the `-s` switch legitimately?
   ii. How might an attacker use this same command for privilege escalation, and what indicators could help distinguish malicious from legitimate use?

3. **Interactive Sessions (-i switch):**
   i. Is it common for system administrators to initiate interactive sessions (`-i` switch) for tasks like process management?
   ii. Could you describe a scenario where an interactive session is more beneficial than a non-interactive one?
   iii. How might an attacker misuse this functionality for persistence or stealth?

4. **Renamed PsExec Service:**
   i. Have you encountered instances where PsExec or PAExec binaries were renamed for legitimate administrative purposes?

Msiexec-Related Questions

1. **Installing and Uninstalling Apps with Msiexec:**
   i. What are the most common administrative uses of Msiexec?
   ii. Do you ever use Msiexec for installing software directly from online sources? If so, could you provide an example?
   iii. How often do you use Msiexec with URLs as parameters, such as installing software directly from vendor sites? What's a typical use case?

   iv. Are there specific flags or parameters you use with Msiexec when uninstalling software?

   v. Have you ever encountered instances where Msiexec was used to remove critical software without prior notice? How did you verify if it was legitimate or potential misuse?

2. **Running Msiexec from Uncommon Directories:**

   i. Have you ever run Msiexec from directories other than the standard locations (e.g., `C:\Windows\System32` or `C:\Windows\SysWOW64`)? Why might this be necessary?

WMIC-Related Questions

1. **Common Administrative Usage:**

   i. What typical tasks do you use WMIC for in your daily work?

2. **Script Library Loading with WMIC:**

   i. Do you ever load script libraries like `jscript.dll` or `vbscript.dll` with WMIC? Can you describe a legitimate use case?

3. **XSL Script Processing with WMIC:**

   i. Have you used WMIC with XSL scripts for formatting or output processing (e.g., `/format` or `−format` options)? Could you provide an example?

   ii. Would using XSL formatting with WMIC likely trigger a security alert? How do you distinguish legitimate from potentially malicious actions?

4. **Remote Command Execution with WMIC:**

   i. Have you executed binaries remotely with WMIC (e.g., `wmic.exe /node:″192.168.0.1″ process call create ″program.exe″`)? What are your typical use cases?

   ii. Remote command execution is often flagged as adversarial behavior. How do you ensure these actions are recognized as legitimate?

5. **Volume Shadow Copy Creation via WMIC:**
    - i. Have you created volume shadow copies (e.g., of `NTDS.dit`) with WMIC? What tasks require this?
    - ii. Since shadow copies can be abused for data exfiltration, how do you monitor for such potential misuse?
6. **Execution of JScript or VBScript via WMIC:**
    - i. Have you ever executed JScript or VBScript through WMIC (e.g., processing a remote XSL stylesheet)? Could you explain a situation where this is necessary?
7. **File Copying with WMIC:**
    - i. Do you use WMIC to copy files from one location to another (e.g., copying `autoit-v3-setup.zip`) ? In what scenarios would this be necessary?

Wbadmin-Related Questions

1. **Backup and Restore (Active Directory NTDS.dit Snapshotting):**
    - i. Deleting backups (using commands with `delete`, `backup`, `-manifest`) can resemble adversarial activity. What's a common reason for doing this?
2. **Deleting Shadow Copies:**
    - i. Have you ever removed shadow copies with WBAdmin (e.g., using `delete catalog`, `-quiet`)? What's a legitimate reason?
    - ii. Since removing shadow copies is often associated with adversarial behavior, in what way does the administrative use differ?

Winget-Related Questions

1. **Installing Packages Using Local Manifest:**
    - i. Have you used Winget with a local manifest (e.g., commands with `install`, `-m`, `-manifest`) instead of Microsoft's trusted sources?

   ii. Under what circumstances would you choose a local manifest over Microsoft's repositories (trusted sources)?
   iii. Have you ever used `winget` to download and install a file from a URL specified in a local manifest, such as `winget install -manifest .\manifest.yml`? What legitimate tasks would this serve?

2. **Bypassing Blocked Microsoft Store Apps Using Winget:**
   i. Have you ever encountered a situation where msstore apps were blocked on a system, and you had to use winget to install apps (e.g., `winget install -accept-package-agreements -s msstore Npcap`)? Did this action trigger any security alerts?

CertUtil-Related Questions

1. **Downloading Files Using CertUtil:**
   i. How do you use CertUtil in your routine?
   ii. Have you used CertUtil for downloading files from the internet (e.g., `certutil -urlcache -split -f;-verifyctl`)? What legitimate scenarios call for this?

2. **Encoding & Decoding Files in Base64:**
   i. What administrative tasks require encoding or decoding files with CertUtil (e.g., `certutil -encode | certutil -decode, -decodehex`)?

Ntdsutil-Related Questions

1. **Usage:**
   i. Could you describe common scenarios where Ntdsutil is used, such as Active Directory maintenance?
   ii. Have you encountered any security alerts while using unusual commands with Ntdsutil?

Cmdkey-Related Questions

1. **Usage:**
   1. How do you typically use `cmdkey` in your day-to-day administrative tasks? For example, do you frequently manage or retrieve stored credentials with this tool?
   2. A common use of `cmdkey` is to list or retrieve credentials using the `/list` option. Are there any other frequent uses of this tool?

Cmd-Related Questions

1. **Usage:**
   1. What are some operations you perform with `cmd` that might appear suspicious and potentially trigger a security alert? For instance, actions like file deletion, system enumeration, or downloading with `curl`. Could you share examples?
2. **File Deletion using `del` or `erase` with Wildcards:**
   1. Could you describe a situation where you needed to delete files using wildcards (e.g., `*.dll` or `*.exe`) with the `del` or `erase` command?
   2. How often do you use options like force delete (`-f`) or quiet mode (`-q`) in these operations? In what scenarios are these flags necessary?
   3. How do you ensure that such deletion actions don't mistakenly resemble malicious behavior, possibly causing security alerts?
3. **Combining Download and Execution using `curl`:**
   1. In what situations would you use the `curl` command to download files or data from remote sources?
   2. Do you sometimes execute downloaded files or scripts immediately after downloading with `curl`? How do you ensure that your activities are recognized as legitimate by security monitoring systems?
4. **Alternative Data Streams (ADS) Usage:**

   i. Have you ever used commands such as `type >`, `makecab .cab`, or `regedit /E` for system administration purposes? If so, in what context?
   ii. Are there situations where you would export registry keys using commands like `reg export`? How often do you perform this, and what are your reasons?

Concluding Question:

1. **Other Binaries:**
   i. Are there other binaries you commonly work with? Could you identify ways these might be used that could unintentionally trigger a security alert?

A.2 Interview: Log Analysts

Introductory Questions

1. Could you please describe your background and role?
2. Do you primarily monitor Windows or Linux systems, or both?
3. Which living-off-the-land binaries (LOLbins) do you commonly encounter when monitoring IT systems?
4. Are you familiar with any of the following binaries: PsExec, MSIExec, SSH, WMIC, Winget, WBAdmin, CertUtil, Ntdsutil, CMD, or Cmdkey? If so, which ones do you encounter regularly, and how do you typically respond to their use?

SSH-Related Questions

1. **SSH as a Proxy**
   i. Do you often encounter an alarm when administrators use SSH as a proxy to launch programs on remote systems?
   ii. Could you provide an example of how the ProxyCommand option could be used in a legitimate administrative context?

    iii. When using SSH for proxying, are there specific commands or switches that could indicate whether the action is legitimate or malicious? What method do you use to differentiate between an admin's usage and an attacker's?

2. **SSH for Port Forwarding**

    i. While port forwarding via SSH is often associated with software developers and testers, do you also observe this technique being used in an administrative context?

    ii. Can you describe legitimate scenarios in which SOC teams might encounter SSH port forwarding being used by system administrators?

3. **SSH for RDP Tunneling**

    i. In your experience, how do system administrators use SSH for tunneling RDP connections?

    ii. Can SSH-based RDP tunneling sometimes resemble adversarial behavior, such as data exfiltration? What differences in approach would you typically see between legitimate admin use and malicious activity?

4. **SSH and Adversary-like Behavior**

    i. Are there other SSH use cases by system administrators that could mimic adversarial behavior and potentially lead to false positives in security monitoring tools?

PsExec-Related Questions

1. **Remote Execution with PsExec**

    i. How often do you encounter PsExec being used by system administrators for executing commands remotely on other machines?

    ii. Can you describe a legitimate incident response case where PsExec was used for remote script execution or installation?

    iii. Have you seen instances of PAExec being used instead of PsExec, given its additional security features, such as encrypting sensitive command line parameters to reduce the risk of interception?

   iv. Are there specific combinations of command-line arguments (e.g., `\\, -p, -u`) that might look like adversarial behavior but are often used by sysadmins?

2. **Privilege Escalation to Local System**

   i. Can you provide an example from your monitoring experience where privilege escalation to the local system account using the `-s` switch was employed legitimately?

   ii. How might an attacker use the same command to escalate privileges, and what indicators could help distinguish malicious from legitimate use?

3. **Interactive Sessions (-i switch)**

   i. Do you usually get alerts when sysadmins open interactive sessions (`-i` switch) with PsExec for tasks like process management?

   ii. Could you describe a scenario where it happened, and then what was the reason why the system administrator did that?

   iii. What method do you use to identify if it is a system administrator or an attacker doing it?

4. **Renamed PsExec Service**

   i. Have you come across cases where PsExec or PAExec binaries were renamed for legitimate administrative purposes during your incident investigations?

Msiexec-Related Questions

1. **Installing and Uninstalling Apps with Msiexec**

   i. What are the most common scenarios where you see Msiexec being used for administrative purposes in your monitoring?

   ii. Do you often encounter Msiexec being used for software installations from online sources? Can you share an example where this was a legitimate use case?

   iii. How frequently do you observe Msiexec being used with URLs as parameters (e.g., installing software directly from vendor websites) in a legitimate case?

    iv. Are there specific flags or parameters you often see used with Msiexec during software uninstallations? Could these same parameters be exploited by an attacker, and how would you differentiate between legitimate and malicious use?
    v. Have you encountered incidents where Msiexec was used to uninstall critical software unexpectedly? How did you investigate to determine if it was a legitimate action or potential misuse?

2. **Running Msiexec from Uncommon Directories**

    i. In your experience, have you ever observed Msiexec being run from unusual directories (e.g., `C:\Windows\System32` or `C:\Windows\SysWOW64`)? Can you explain any legitimate reasons why this might occur, and how you assessed these scenarios?

WMIC-Related Questions

1. **Common Administrative Usage**

    i. Have you observed instances of system administrators using WMIC in their daily work? How do you identify these activities in logs?

2. **Script Library Loading with WMIC**

    i. Have you encountered cases where WMIC was used to load script libraries (e.g., `jscript.dll, vbscript.dll`)? How did you determine whether these activities were legitimate or malicious?

3. **XSL Script Processing with WMIC**

    i. Have you noticed the use of XSL script formatting with WMIC (e.g., `/format or -format options`) by admins? What techniques do you use to determine if these actions are legitimate or potentially malicious?
    ii. How do you assess whether the use of XSL formatting in WMIC logs could be part of an attack? Do you have examples of incidents where this occurred?

4. **Remote Command Execution with WMIC**

   i. Have you detected remote commands being executed with WMIC (e.g., `wmic.exe /node:"192.168.0.1" process call create "program.exe"`)? How do you typically verify whether these commands are legitimate or signs of an intrusion?
   ii. Remote command execution is often seen in adversarial behaviors. How do you distinguish legitimate remote commands from potentially malicious ones in your monitoring?

5. **Volume Shadow Copy Creation via WMIC**
   i. Have you identified volume shadow copies being created via WMIC (e.g., of `NTDS.dit`)? How do you detect whether this indicates an attack or is part of legitimate administrative tasks?
   ii. How do you monitor the use of volume shadow copies to prevent data exfiltration or persistence? What methods do you use to distinguish legitimate use from abuse?

6. **Execution of JScript or VBScript via WMIC**
   i. Have you seen instances of JScript or VBScript being executed through WMIC (e.g., `processing a remote XSL stylesheet`)? How do you determine if this is part of a normal administrative workflow or something suspicious?

7. **File Copying with WMIC**
   i. Have you observed WMIC being used to copy files from one location to another (e.g., `copying autoit-v3-setup.zip`)? How do you assess whether this activity is legitimate or a sign of data theft?
   ii. What are your strategies for identifying file copying as part of malicious activities?

WBAdmin-Related Questions

1. **Backup and Restore (Snapshotting of Active Directory NTDS.dit)**
   i. Have you encountered instances where backups (e.g., `commands containing delete, backup, -manifest`) were deleted? How do you typically assess whether this activity is legitimate or part of adversarial behavior? What IoCs do you look for in logs or

    alerts to differentiate between a legitimate backup deletion and a potential attack?

2. **Deleting Shadow Copies**

    i. Have you seen cases where shadow copies were removed by a sysadmin using wbadmin (e.g., `delete catalog, -quiet`)?

    ii. Since removing shadow copies is often associated with adversaries attempting to cover their tracks, how do you distinguish between legitimate administrative tasks and malicious intent in this context?

Winget-Related Questions

1. **Installing Packages via Winget Using Local Manifest**

    i. Have you noticed system administrators installing packages using a local manifest (e.g., `commands with install, -m, -manifest`) instead of using Microsoft's trusted repository or the Microsoft Store? How do you identify if this is a legitimate action or suspicious behavior?

    ii. Under what circumstances have you seen administrators opting for local manifests over Microsoft's trusted sources, and how do you verify the legitimacy of these actions?

    iii. Have you observed cases where winget was used to download and install a file from a URL specified in a local manifest (e.g., `winget install -manifest .\manifest.yml`)? How do you determine whether this is a legitimate task or a potential security risk?

2. **Bypassing Blocked Microsoft Store Apps Using Winget**

    i. Have you encountered instances where administrators bypassed blocked Microsoft Store apps using winget (e.g., `winget install -accept-package-agreements -s msstore Npcap`)? How do you identify if this is a legitimate action?

Certutil-Related Questions

1. **Downloading Files Using CertUtil**
   1. Have you observed system administrators using CertUtil in their routine tasks? How do you distinguish between legitimate and potentially malicious usage in your logs?
   2. Have you seen CertUtil being used by admins to download files from the internet (e.g., `certutil -urlcache -split -f, -verifyctl`)? What indicators help you determine if this is a legitimate scenario or a potential adversary tactic?
2. **Encoding & Decoding Files to/from Base64**
   1. What administrative tasks have you seen that require encoding and decoding files to/from Base64 using CertUtil (e.g., `certutil -encode, certutil -decode, -decodehex`)? How do you verify if these actions are legitimate or suspicious?

Ntdsutil-Related Questions

1. **Usage**
   1. Have you observed instances of system administrators using ntdsutil for tasks like Active Directory maintenance or restoring snapshots? What logs or alerts typically capture these activities?
   2. What specific commands or procedures involving ntdsutil have you encountered in legitimate administrative scenarios? How do you validate them?
   3. Are there any unusual or less common uses of ntdsutil that you would flag as potentially suspicious? How do you assess these in your monitoring?

Cmdkey-Related Questions

1. **Usage**
   1. How often do you detect the use of cmdkey (e.g. with the `/list` option) for managing or retrieving stored credentials? How do you differentiate legitimate administrative use from potential credential harvesting by an attacker?

   ii. Have you seen instances where cmdkey usage has triggered false positives?

Cmd-Related Questions

1. **Usage**

   i. What kinds of administrative tasks using cmd.exe have you encountered that might resemble malicious activity (e.g., `file deletion, enumeration, or downloading via curl`)? Can you provide examples of false positives you have dealt with in these scenarios?

2. **File Deletion Using Del or Erase with Wildcards**

   i. Have you detected file deletions involving wildcards (e.g., `*.dll, *.exe`) using the del or erase commands? What indicators help you determine if this is a legitimate task or a potential threat?

   ii. How often do you see the `force delete (-f)` or `quiet mode (-q)` flags used in legitimate operations? What context helps you confirm their legitimacy?

3. **Combining Download and Execution Using Curl**

   i. Have you observed instances where curl was used to download files or data from remote sources? How do you verify whether these activities are part of legitimate administrative tasks?

   ii. Do you encounter cases where downloaded files or scripts are executed immediately after being retrieved via curl? What indicators help you distinguish between legitimate execution and potential exploitation?

4. **Alternative Data Streams (ADS) Usage**

   i. Have you noticed the use of commands like `type >, makecab .cab, or regedit /E` in system administration? In what scenarios have these been employed, and how do you assess their legitimacy?

   ii. Do you often see the exporting of registry keys with commands like reg export, in an administrative context? How do you distinguish between routine administrative tasks and potentially malicious activities in such cases?

Concluding Question

1. **Other Binaries:**
   i. Are there any other binaries you've observed that system administrator use that could trigger false alarms? How do you handle cases where legitimate usage closely resembles adversarial techniques?