# APT Attack Emulation and Data Labeling

*Generating labeled APT host datasets using CALDERA and Sysmon*

Fikret Kadiric


Thesis submitted for the degree of
Master in Programming and System architecture
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2022

# APT Attack Emulation and Data Labeling

*Generating labeled APT host datasets using CALDERA and Sysmon*

Fikret Kadiric

APT Attack Emulation and Data Labeling

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

# Abstract

Cyber criminals and others seeking to use the cyber domain for malicious purposes are actively pursuing and attempting to exploit new vulnerabilities. Cyber attacks are constantly changing and new attacks are developed. While cyber criminals are trying to attack and breach systems, defenders on the other hand are attempting to fend off the attackers. The introduction of Intrusion Detection Systems (IDS) has been a great contribution to the defense of the cyber domain. These systems can be trained on datasets containing attack data in order to automatically detect these attacks in the future.

However, due to challenges regarding labeling and production of attack data, publicly available labeled host datasets are rare. The situation becomes even worse when it comes to APT datasets. APT attacks tend to utilize new vulnerabilities and tools, these attacks are constantly adapting and evolving over time. As such, APT datasets may quickly become outdated, leaving defenders to rely on outdated datasets. In order to keep pace with the evolving attacks and detect these, we need datasets containing these attacks.

This thesis examines a new approach to generating labeled datasets through an automated labeling tool developed by the author of this thesis. Attack data is generated by an adversary emulation tool (CALDERA) while recording the occurring system changes through System Monitor (Sysmon). It is found that the labeling tool is capable of generating fine-grain labeled datasets, applying labels on the attack technique level directly tied to MITRE ATT&CK. This new approach enables the creation of new datasets in a convenient and efficient manner, allowing researchers to create specific datasets if desired. This thesis is a contribution to the research within the field of cybersecurity.

# Acknowledgements

First and foremost, I would like to thank and express my gratitude to my supervisors Gudmund Grov, Espen Hammer Kjellstadli, and Audun Jøsang for valuable guidance, discussions, providing research ideas, and giving valuable help in writing this thesis.

Secondly, I wish to thank my classmate Julie L. Gjerstad for scientific discussions and company during challenging times.

Finally, I would like to thank my friends and loved ones for all their support, love and motivation. This thesis would never have see the light of day if it was not for you.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Context

Cybercriminals are actively pursuing and attempting to exploit vulnerabilities residing within e.g., computer networks, or software. These vulnerabilities, also called weaknesses, can occur as a result of misconfigurations, flaws, or user errors to name a few. If successfully exploited, a vulnerability may give an attacker entry to the network, allowing them to conduct further malicious operations. Discovering and mitigating vulnerabilities may stop most novice hackers, but it is seldom enough to keep a determined and skilled hacker at bay. A mitigated vulnerability does not necessarily mean that it cannot be exploited. Advanced hackers may gain access by exploiting unknown vulnerabilities or flaws that are yet to be discovered, or by chaining together multiple low priority vulnerabilities. No matter how secure a system is, it is difficult to guarantee perfect security.

Dealing with advanced persistent threat actors, also called APT, requires a deeper level of defense and detection. What separates an APT from most other hackers is their ability to access confidential information using advanced techniques, with a slower attack process in order to reach their goals [39]. The attack phase of an APT often consists of five main phases: reconnaissance, foothold-establishment, lateral movement/discovery, data exfiltration, and post-exfiltration [1]. Once inside, APTs can blend in with the background "noise" by utilizing legitimate processes, often combining the use of multiple processes to perform malicious activities. APTs are usually highly capable and well funded threat actors, with advanced tools and close to unlimited resources and time. Ideally, we want to detect these attacks as early as possible, but as APT attacks are usually more sophisticated and complicated they often manage to bypass the security mechanisms in place. The goal of an advanced attacker is not merely to compromise the network and achieve unauthorized access, but usually includes more specific goals such as: stealing data and information, or disrupting critical services. These goals require comprehensive post-compromise work, where the attacker desires to stay in the network undetected for as long as possible.

Intrusion detection systems, also called IDS, are tools that automatically monitor and analyze the behavior of a computer or a network environment and report any suspicious activities detected. IDS grants an additional layer of system security by recognizing patterns that are known to relate to malicious traffic or directly tied to adversaries. These systems are generally classified in two distinct ways, either based on their operational contexts e.g., network or host-based detection, or on how they aim to fulfill their task e.g., supervised detection or unsupervised detection. Supervised detection utilizes a training dataset in order to train the IDS to differentiate between malicious and benign traffic, while unsupervised detection is trained on the normally functioning system as a baseline for comparison between expected and unexpected traffic [9]. Research within the field of intrusion detection systems has flourished over the years. However, many researchers struggle to find suitable datasets to evaluate and test the efficiency of their proposed models [27]. Issues regarding privacy concerns, data labeling, different research objectives, and availability of datasets are some of the issues described by Nehinbe [35]. Machine learning, which is widely used in anomaly-based IDS, learns from the provided datasets. The datasets may either be labeled with e.g, malicious and non-malicious data, or without labels. From these observations the algorithms used are able to generalize new observations, thus allowing future observations to be automatically classified. The importance of correctly labeled datasets for IDS efficiency evaluations has been highlighted by several researchers e.g., Catania et al. [7], and Davis et al. [14].

## 1.2 Problem Description

While there are multiple publicly available datasets related to network based traffic, only a few host based datasets exist. One reason for the lack of endpoint base datasets may be that most malware or cyber attacks produce network activities, and thus can be detected based on network traffic. When it comes to APT datasets, the situation becomes even worse. To the best of our knowledge, there is only one publicly available dataset [33]. Traditional intrusion detection systems that utilize signature based detection, may struggle to detect the presence of APTs [33]. These systems are designed to detect individual and known attack patterns while an APT attack involves several connected malicious activities. In order to train intrusion detection systems to detect these attacks, we need datasets containing APT attack data. As the detection signatures in IDS stem from previous known attacks, being able to quickly adapt and train the IDS is crucial to keep pace with the development of new attacks. However, in order to train the IDS we first need a dataset containing these new attacks.

Generating such APT datasets presents multiple challenges [40]:

- Storage and processing problems related to big amounts of data. Due to the nature of APT attacks occurring over a long period of time, we need to collect enough data to properly represent this, ideally having data covering at least several months.
- There is no set path that all APT attacks follow, APT attacks may vary in terms of e.g., tools, techniques, tactics, and procedures.
- APT attacks are constantly adapting and evolving over time, they also tend to utilize new vulnerabilities and tools.
- Labeling of APT dataset should include expert knowledge.
- Simulating an APT attack in a realistic manner.

Aside from the challenges directly tied to APT attack data, there are also several other challenges particularly related to the process of labeling the datasets [28]:

- The data is generally generated in large volumes, usually making manual labeling of all lines infeasible.
- A single action may manifest itself in multiple log sources.
- Log lines corresponding to malicious actions may be interrupted by normal log messages as processes are frequently interleaving.
- Execution of a malicious command may cause manifestations in logs at a much later time due to delays or dependencies on other events.

Using a training dataset with correctly labeled logs is a crucial aspect related to the efficiency of the IDS, ideally with labels in greater details than e.g., "benign" and "malicious".

## 1.3 Research Questions

Developing and label a dataset containing APT attack data presents multiple challenges. The main focus of this thesis will be on generating and applying labels to the logs. This focus can be reflected in the following research question:

> **Research Question**: What are the possibilities to develop a labeling tool around CALDERA and Sysmon, to create fine-grain labeled APT datasets?

This research question is further guided by the following sub questions:

1. How can CALDERA and Sysmon be combined to generate APT logs??
2. How can labels be applied at the attack technique level?
3. How can benign traffic be integrated in the APT dataset?

## 1.4 Research Methods

This thesis is a study in cyber security, a subfield within the computer sciences field. Fundamentally the thesis is following a scientific research method through experimental research design [16]. It is necessary to design an experiment that generates data in order to evaluate the truth of the hypothesis. The system around the experiment is controlled to ensure that only specific inputs of the system are altered to determine the effect. The following characteristics are exhibited in a good experiment [16]:

- **Clear Design** - Includes clear explanations of assumptions, procedures to be executed, method used to detect an effect, and the logic behind choices made.
- **Precise** - Experiments conducted need to be precise in their definitions and procedures.
- **Repeatable** - Experiments need to be documented so that they are repeatable, repeating an experiment number of times should yield similar results. They must also be documented from the design through the execution so the experiment may be repeated.
- **Reproducible** - The documentation of the experiment should allow for others to follow and reproduce the experiment. This often means including more information in the documentation, such as personal beliefs and intentions, and software developed and used.

The experimental design relies on testing the relationship between two variables. An independent variable is used in order to measure its effect on the other dependent variable. The dependent variable is an observable and measurable variable, while the independent variable is the input to a system that potentially causes an effect on the dependent variable. The dependent variable is defined and related back to the following hypothesis:

> **Hypothesis** *(H)*: CALDERA and Sysmon can be combined with a developed automated labeling tool in order to generate fine-grained labeled APT dataset.

This hypothesis claims that fine-grained labeled APT datasets can be created by an developed automated labeling tool utilizing CALDERA and Sysmon. The variable measured will be how specific the labels applied are, and how successful the tool is at finding the malicious events produced by CALDERA in Sysmon. Labels of low granularity often only include a statement if the data is malicious or benign in nature, while specific and detailed labels (fine-grained) result in a higher label granularity. Higher label granularity can be achieved by applying labels which can relate the data back to a specific attack type e.g., tactics and techniques. Labels will be measured based on their level of granularity.

## 1.5 Thesis outline

This section provides an overview of the thesis chapters and their content:

**Chapter 2:** This chapter describes the background of the thesis, introducing the essential topics and tools used in this thesis. It starts with introducing publicly available datasets related to the research community of Intrusion detection and prevention systems, before gradually introducing other subjects relevant for this thesis.

**Chapter 3: Architecture and Data Generation** This chapter describes the components and configurations within the developed environment, in which the APT emulations was conducted. Secondly, the process of generating attack data and the executed emulation is described. Lastly, the process and tool used in order to generate benign data is described.

**Chapter 4: Data Collection and Labeling** This chapter describes the process of collecting and processing the logs. It gives an insight into how the developed labeling tool applies fine-grain labels to the logs, and the various iterations of the tool leading up to the final version.

**Chapter 5: Results** This chapter presents the results of the emulation and the final dataset created with the labels applied.

**Chapter 6: Discussion and Related Work** This chapter will discuss and compare the final result of the thesis with related research. Limitations related to the presented dataset, labeling approach, and developed tool is also discussed.

**Chapter 7: Conclusion and Future Work** This chapter will conclude the thesis based on the discussion, answering the research questions of the thesis. Suggestions for future work is also presented in this chapter.

## 1.6 Contribution

With the developed labeling tool and approach for applying fine-grain labels, the research has addressed the following hypothesis:

> **Hypothesis** *(H)*: CALDERA and Sysmon can be combined with a developed automated labeling tool in order to generate fine-grained labeled APT dataset.

The research conducted in this thesis has demonstrated the possibilities of combining CALDERA and Sysmon to generate raw APT attack data. A developed automated labeling tool capable of applying labels based on tactics and techniques from MITRE ATT&CK is presented as the main contribution. Additionally, a labeled APT29 host dataset developed by combining CALDERA, Sysmon, and the labeling tool is presented. The developed labeling tool and APT29 host dataset, have have been published to GitHub Enterprise (UIO) [24], and regular GitHub [25].

# Chapter 2

# Background

## 2.1 Existing datasets

Most of the current datasets openly available focuses on network connections and transmitted packages, paying little to no attention to what changes are occurring on the endpoint. However, advanced attackers are known to exploit zero-day vulnerabilities, thus avoiding detection from traditional signature-based intrusion detection systems. Once a machine is infected, numerous operations are being executed on that specific machine, which cannot be detected in the network data. Endpoint based anomaly detection could help defenders overcome the limitations of traditional signature-based detection, by detecting operations and changes occurring on the endpoint. The most notable and widely used datasets are the DARPA [13] and KDD Cup 99 datasets. The DARPA evaluation dataset was created by Lincoln Lab on behalf of DARPA, containing various information collected through simulating network attacks. The KDD 99 dataset was created by processing the tcpdump portions of the DARPA dataset. Both of these datasets were considered valuable datasets by experts, despite being created through a simulation and not from real attack data. However, these datasets lack endpoint-related information and thus are only valid for network intrusion research.

There have been some attempts at generating a public dataset for the Windows operating system, such as the ADFA dataset. This dataset was generated by recording the system calls performed during a simulated zero-day attack [12]. This introduced two new datasets to the research community, namely the ADFA-WD [12] and ADFA-WD:SAA [12] datasets. The major drawbacks of these datasets are that only basic information was collected and an insufficient amount of vulnerabilities were used to generate the malicious activity. The activity was generated using twelve vulnerabilities and the information gathered consist of DLL file names and the called function name. The vulnerabilities were all known vulnerabilities that have been exploited in automated hacking tools, such as Metasploit [38]. While this dataset is a fine addition to the available datasets, it is unfortunately incomplete [21].

These datasets contain what are considered regular malware attack vectors where the goal is to successfully infect a network or a client. While the detection of APT involves the identification of long-term attack behavior. Myneni et al. [33] has attempted to generate a dataset suitable for APT behavior which considers further post-compromise phases, such as data exfiltration and lateral movement. Attack data was generated by having a red team simulate the attack behavior of an APT. Vulnerabilities were discovered by using various scanning tools, and later exploited through scripts and known attacking tools e.g., metasploit [38], in order to establish a foothold within the network. From this foothold, the red team moved laterally within the network by employing known lateral movement techniques before collecting and exfiltrating data.
The comparison of multiple datasets done by Myneni et al. [33] revealed a massive lack of data related to the later stages of an advanced attack. As seen in **figure 2.1**, most of the datasets only covered the early attack phases, such as reconnaissance and foothold establishment. Only the DAPT2020 dataset covers data exfiltration, which often is the goal of most advanced attacks.

| Dataset / APT Phase | UNB-15 | CICIDS | NSL-KDD | Mawi | ISCX | DARPA | HERITRIX | DAPT 2020 |
|---|---|---|---|---|---|---|---|---|
| Normal Traffic | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Reconnaissance | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ |
| Foothold Establishment | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Lateral Movement | | | | | | | | ✓ |
| Data Exfiltration | | | | | | | | ✓ |

Figure 2.1: Attack phases covered by known datasets Myneni et. al [33]

Ideally, we want to detect the attack as soon as possible, but training for e.g., data exfiltration detection, could allow defenders to interfere with the exfiltration and potentially stop it, effectively stopping the attacker at the finishing line. By utilizing virtualization to create the enterprise network, the DAPT2020 dataset bypassed one of the key challenges related to the sharing of data, namely anonymization, the privacy concerns related to an organization and their customers. The need for anonymization is completely removed as there is no real sensitive information in the data, which would not be the case if the data stemmed from a real-world attack. On the other side, virtualization has some drawbacks as well. It is hard to successfully match the complexity and size of any larger corporation. The traffic considered normal or benign traffic from this network may also deviate from the benign traffic found in real-world networks. A potential lack of traffic diversity could cause biases in any systems trained on the data. What is considered normal or malicious behavior is subjected to changes with the continuous development of new systems and cyber-attacks.

Datasets that may be considered representative of realistic data can quickly become outdated. To constantly generate new datasets, a framework for generating both benign and attack data is desirable. Despite these potential drawbacks of virtualization, the DAPT2020 [33] dataset contains unique and valuable attack data.

## 2.2 Traffic generation using virtualization

One reason for the lack of publicly available data from real-world captures arises from the sensitive nature of data. The inspection of data can reveal sensitive information and even confidential or personal communication. Due to the strict regulations of sensitive data applied by the General Data Protection Regulation (GDPR) and governmental law enforcement, a breach of such information could be catastrophic for both organizations and third parties involved. Given the sensitive nature of real-world data, two alternative approaches are being pursued: simulation and anonymization. Anonymization through sanitation is done by removing or anonymizing any data that might be considered sensitive information. Despite the effort, sensitive information can still be leaked by recovering the scrubbed sensitive information [11]. Even if one could successfully remove all the sensitive data, training data-driven detection systems on the data can be problematic. By definition, anomaly based detection looks for the kind of artifacts that tend to be removed during the anonymization process [26]. Due to this, researchers are forced to generate their own datasets, usually through simulated networks and endpoints.

The use of virtualization technology has gained major traction as the cost of assembling a full-scale physical network with all the components and endpoints necessary, is not a feasible task for most researchers. However, generating realistic synthetic datasets is not an easy task, with multiple problems that have to be addressed. The lack of variation is a problem commonly found in these synthetic datasets. When training a system to differentiate between malicious and legitimate action, we need both benign and attack data. Attack data can be generated through various methods eg., manually performing the attack, executing malware, or through scripts and tools. A common strategy used to generate benign traffic is to execute a series of scripted actions [10] e.g., internal file transfers, web browsing, file download. These scripts often cover the most regularly performed actions or protocol activities. But they are seldom enough to cover the vast range of legitimate actions occurring within the complex network of larger corporations. By moving the environment onto a virtualized network, any external influences become eliminated, and the environment becomes isolated from fluctuations and faults. These fluctuations and faults commonly occur as a result of the complexity of modern networks. This removes the variations in the response behavior of particular services, potentially creating homogeneity and predictable behavior.

The lack of variations results in a clearer separation of benign and malicious activities, which in turn leads to overoptimistic results from any detection systems trained on the data [10]. To address this issue, and several other issues related to data generation through virtualization, Clausen et al. [10] proposed a new framework for traffic generation by clustering a series of interacting containers with one another, where each container was responsible for either hosting a specific service or performing a specific task. Executing a scenario e.g., a step of an attack, would trigger the launch of several containers, dynamically launching new containers depending on the results of an operation. In order to introduce some variation, a series of sub-scenarios were created. These sub-scenarios consisted of simple variations of a scenario, such as mixing up the use cases for different protocols and determining if the specific operations are successful or not.

Most notably was the use of this framework to achieve ground truth, being able to separate and label the data from different origins with stronger certainty. The noise of background processing was minimal since each container was only running a specific piece of software or application related to a scenario. Data from each container was collected separately so that the distinction between the traffic's origin was clear, this could allow for higher label granularity. Label granularity refers to how specific a label is. Higher label granularity could be achieved by labeling the data by the distinct types of attack it represents e.g., host discovery, privilege escalation, or credential dumping. While benign data could be labeled by e.g., service, software, or protocol related to the traffic. The impact label granularity has on machine learning classification has been studied within the field of image recognition. Studies within this field have shown that better label granularity can lead to higher accuracy and higher training data efficiency, which may reduce the amount of training data needed [8]. To what extent label granularity may affect the training of intrusion detection systems has not been studied, but it stands to reason that more granular labels are desirable to distinguish between the different types of activities.

Clausen et al. [10] demonstrated the possibilities of network traffic generation through containerization, with a particular focus on achieving ground truth. This was successful largely through the isolation of traffic scenarios into separate container arrangements. With some modification, the proposed method for generating labeled datasets by Clausen et al. [10] could potentially be implemented to generate endpoint-based datasets. The primary data collection would have to occur on the endpoint as the primary goal is to detect changes occurring on the endpoint. However, we can not completely ignore the network activities since the network traffic is necessary for correlation purposes. An example of this could be that an adversary is executing network connection discovery commands on an endpoint, these could be detected through process command-line parameters and process monitoring.

In order to trace this attempt further, we would need network data to determine if the adversary attempted to move laterally to another endpoint. If both these behaviors are observed in correlation with one another, it would indicate a potential attack with a stronger probability.

## 2.3 Data Labeling

The process of applying labels to the logs can be approached in various ways. Perhaps the most superficial approach to labeling logs is to label all logs generated during a timed interval of an attack as malicious. While perhaps easily accomplished, this approach is prone to mislabeling the logs, as benign actions occurring within the timed interval will be wrongly labeled as malicious. Another aspect is that this approach greatly hinders the reproducibility of the log data, as benign processes are prone to occur at random intervals (e.g. benign processes, updates/patches or normal operation), executing the same attack scenario would result in different datasets with various random mislabeled benign activities. A training dataset labeled with this approach would result in an high false positive (FP) rate, as random benign actions could be detected as malicious actions [14].

Another possible approach is to use two dataset, one containing only benign activities while the other contains both benign and malicious activities. If the same benign actions were performed in both datasets, any differences found in the dataset containing malicious actions could be considered malicious. This would reduce the amount of benign actions being mislabeled compared to the timed interval approach. However, a single action may manifest itself in different ways in the logs, either due to delays, dependencies, missing resources or other errors. As benign background processes are prone to occur at random intervals, deviations amongst benign actions in the datasets could occur. Therefore, there is still a possibility of benign actions being mislabeled with this approach.

Some Intrusion Detection Systems (IDS) report individual events as a deviation of what is considered "normal" activity and flag these as anomalies. There are also those IDS that focus on multi-step attacks or support classification of the attacks (e.g., denial of service, reconnaissance, exploitation attempt etc). In order to differentiate among the anomaly types and enable attack-type classifications, the training dataset must include labels that differentiate between types of attacks or attack steps.

## 2.4 Windows Event Logs

Logs are records of events that occur within a system, these events can be the results of an operation conducted by a person or by a running process. In Windows, these events are stored as Windows Event logs in the .EVTX file format. The information collected, and available in the logs is determined by the Windows system's audit policy.

Windows uses nine audit policy categories, and 50 subcategories which allow for a more granular control over what information is logged. These policies defines the type of system objects that will be monitored (e.g., files, registry keys, network events), what type of access should be recorded (e.g., read, write, permission changes), and who's accesses should be monitored (e.g., users, system, all). Depending on what information is to be collected, the configuration can be found in either the System Access Control List (SACL) or in the Local Security Policy. The SACL enables the logging of attempts to access a security object (e.g., files and registry objects), which can generate audit records when an access attempt fails, succeeds or both. The Local security policy controls what sorts of events and objects generate events in the logs, it contains an Audit Policy section and an advanced Audit Policy section. The advanced audit policy section allows for more granular audit controls. While enabling an Audit Policy, the policy may be enabled to log Success events, Failure events, or both, depending on the policy as some policies only generate success events. Only enabling logging of Failure events for each category is generally not recommended, as many of the most important events (e.g., changes to critical user accounts/groups, account lockouts, security setting changes) are Success events.

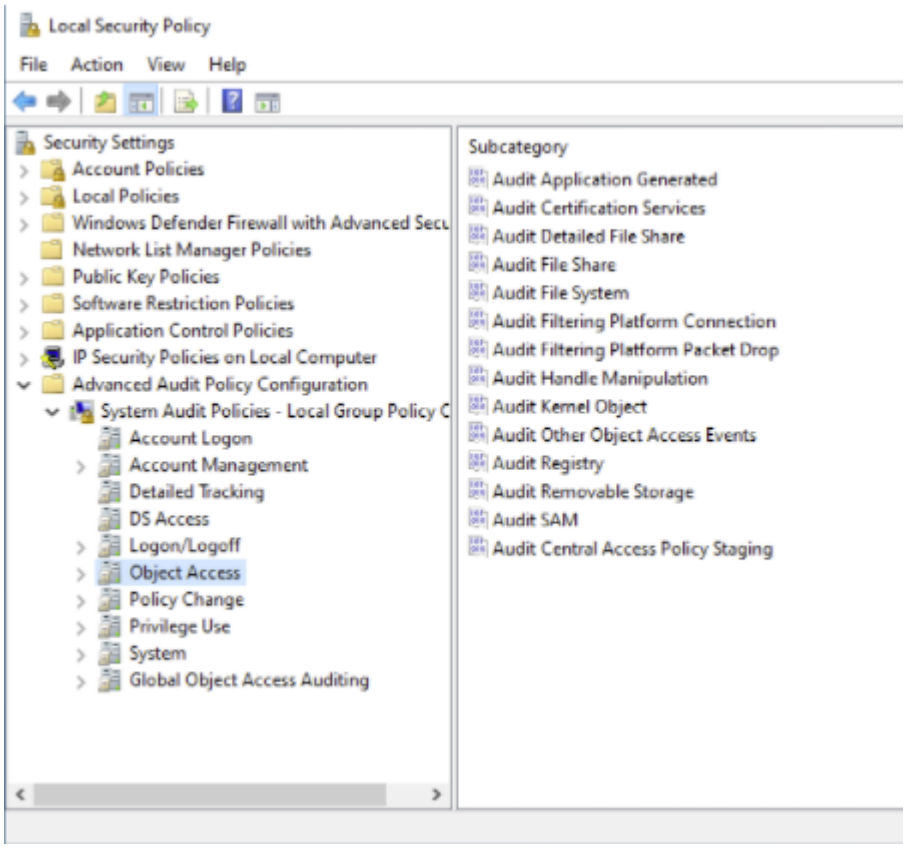



Figure 2.2: System Audit Policies - Local Security Policy

Logs can be collected once the logging mechanisms have been enabled and configured. Turning on all the logging mechanisms would provide the most coverage, the drawback of this is the volume of log events. The generated volume would quickly overwhelm commodity storage systems and impact the performance of the monitored systems. If the logs are sent to a centralized log management solution, the volume of logging generated may impact the network's performance, particularly if the information is transmitted across low bandwidth connections. Limiting the volume generated is thus crucial in order to avoid any unintended side effects on normal business processes. This may be achieved by configuring the policies to filter out the majority of irrelevant events, while still maintaining the ability to detect malicious behavior and collect relevant information regarding the operations conducted prior to the event. However, deciding what configuration yields the most coverage while limiting the volume is no easy task. Ideally we want coverage over events related to the occurring incident, while filtering out "background noise". What is considered relevant events varies depending on the ongoing operation and what activities are being conducted as a result. If the goal is to detect an data exfiltration event, monitoring for file creations and modifications as well as network connections is essential, but enriching this data with account logon/creation or system events aids greatly when attempting to trace down the root of the issue. In order to gain the most out of the logs the correct audit policy configurations must be applied. Start with the Audit Policy Recommendations from Microsoft [32] is a good starting point, but further configuration of the policies and perhaps other logging sources may be necessary to gain the desired coverage.

## 2.5 Sysmon

While the recommended audit policies grants some visibility of system activities, tools such as System Monitor (Sysmon) [48] provide even greater visibility. Sysmon is a highly configurable tool from the Windows Sysinternals Suite that monitors and logs system activities. It provides detailed information regarding process creation/termination, driver and library loads, network connections, file creations, registry changes, process injections and more. In comparison to the built-in Windows audit logs, Sysmon is capable of monitoring a wider range of system activities, with each corresponding event containing significantly more information and details. The increased information and details is of great help when detecting and mapping corresponding indicators of compromise. **Figure 2.3** contains a complete list of events that Sysmon can record [48].

| Description | ID | Description | ID |
|---|---|---|---|
| Process Creation | 1 | Registry Object Renamed | 14 |
| File Creation Time Modification | 2 | File Stream Created | 15 |
| Network Connection | 3 | Sysmon Configuration Changed | 16 |
| Sysmon service state changed | 4 | Named Pipe Created | 17 |
| Process Terminated | 5 | Named Pipe Connected | 18 |
| Driver Loaded | 6 | WMI Filter | 19 |
| Image Loaded | 7 | WMI Consumer | 20 |
| Remote Thread Creation Detected | 8 | WMI Consumer Filter | 21 |
| Raw Access Read Detected | 9 | DNS Query | 22 |
| Process Accessed | 10 | File Delete | 23 |
| File Created | 11 | Clipboard Changed | 24 |
| Registry Object Added or Deleted | 12 | Process Tampering | 25 |
| Registry Value Set | 13 | Error | 255 |

Figure 2.3: Sysmon Event Table

In order to reduce the amount of Sysmon logs generated, administrators can easily specify what events are to be logged or filtered through the use of matching rules, or completely disable the logging of specific ID's. This is done through the configuration file of Sysmon, allowing administrators to tailor the logging based on their needs. Since the configuration is done from one configuration file, users can easily import and share existing configurations as a starting point.

Threat hunting using Sysmon has been proposed and tested. Mavroeidis et al. [30] utilized Sysmon in an automated end-point threat hunting system. Threats were identified through automatically analyzing Sysmon logs in order to classify system processes with different threat levels based on their identified characteristics. Once identified, the threat was presented through a RDF graph, showing the related system changes and actions. Threat hunting and malware detection using Sysmon has also been proposed in numerous other systems. A comprehensive list of related work can be found on GitHub [31]. This shows the extent of possible use-cases for Sysmon as a log source. Given the amount of use-cases and the use of Sysmon as a stand alone log source within academic studies [29, 30], it stands to reason that Sysmon possesses the necessary capabilities to generate logs with enough coverage to conduct host-based threat and malware detection.

## 2.6 MITRE ATT&CK

The MITRE ATT&CK framework [43] is a knowledge base and model for adversary behavior. This framework is designed to reflect the various attack phases of an adversary and the platforms they are known to target. The framework consist of the following core components:

- Tactics, denoting short-term, tactical adversary goals during an attack e.g., Privilege Escalation, or Exfiltration
- Techniques, describing the means by which adversaries achieve tactical goals e.g, Privilege Escalation through Process Injection, or Exfiltration Over Web Service
- Sub-techniques, describing more specific means by which adversaries achieve tactical goals at a lower level than techniques e.g., Dynamic-link Library Injection, or Exfiltration to Code Repository
- Documented groups/APTs usage of tactics, techniques, and procedures.
- Mapping APTs to specific techniques.

Initially, the framework only covered post-compromise adversary tactics against Windows systems, but has grown to cover both Linux and macOS, as well as the tactics leading up to the compromise of an environment. A visualized representation of the relationship between tactics, techniques, and sub-techniques can be found in the ATT&CK Matrix [43]. A key component to accurately emulating an adversary is knowing how the adversary behaves. This can be achieved through information on what tactics and techniques the adversary utilizes. Many threat emulators have their attack models based on the techniques listed in MITRE ATT&CK [43]. The framework can also be used as a tool to create adversary emulation scenarios [42] or construct specific adversary profiles.

## 2.7 Adversary Emulation

The easiest form of adversary emulation comes in the form of executing small, specific scripts for testing individual techniques and procedures. Atomic Red Team [5] is an open-source project from Red Canary that provides this through its collections of scripted cyber attacks. Defenders can manually execute these scripts, simply by copying them over into the command-line, as a way to generate indicators of compromise and test if they are able to detect certain techniques. These are singular tests that allow defenders to focus on individual ATTACK techniques, giving an easy way to check their coverage towards the ATTACK-based techniques.

With more than 250 techniques within ATT&CK [43], testing for coverage on all of these techniques individually is clearly a time-consuming and labor-intensive task. Some of the techniques are also completely legitimate commands that may be executed during normal operations. Pivot off only one indication solely, and determining that it is a malicious instance can be challenging. To confirm that this is indeed a malicious instance, defenders can look for other techniques that might have been used in conjunction with the previous detection. A domain admin login by itself does not mean that the account has been compromised, but if seen in conjunction with group discovery or credential dumping it might raise a concern. Atomic red focuses strictly on technique execution which can be valuable for signature detection development. A more complex tool is necessary for those seeking to fully emulate an adversary.

### 2.7.1 Automated Adversary Emulation

In order to generate realistic attack data, the simulated environment, endpoint, and attack scenarios should be representative of those found in the real-world. Attacks should be conducted by using the same tools, techniques, tactics, and procedures as an attacker would use. Datasets of a post-compromise APT attack differentiates from regular attack data. These adversaries often utilize a conjunction of multiple techniques and legitimate processes for malicious purposes. There is a thin line between normal and abnormal behavior on a system, the key component for APT detection is the correlations between multiple anomalies detected [33]. Most organizations choose to conduct red team exercises to simulate and test their defenses in the face of an adversarial attack. During these exercises, highly-trained security consultants enact attack scenarios by simulating a realistic cyber-attack, using the same methods and techniques that adversaries leverage. Unfortunately, red teams can be difficult to employ, particularly on a regular basis. Conducting red team assessment is a manual and time-consuming task, the cost of hiring personnel who possess the right knowledge and skillset is also fairly significant. These issues of cost, time, and personnel make it difficult for organizations to incorporate red teaming into their security procedures.

In order to make such assessments more feasible, tools such as Atomic Red [5], which automates some of the tasks at a technical level, have risen in popularity. These tools allow the red team operators to focus more on the tactical level of the attack. Automated adversary emulation aims to extend this automation to the tactical level as well, effectively leaving the operators only to construct the attack scenarios. By implementing an automated red teaming system, the problems of cost, time, and personnel are immediately addressed. However, this process is not without challenges. The effectiveness of such a system depends heavily on the automation techniques, how decisions are made and how the system deals with uncertainties that occur during execution.

Applebaum [2] gives an insight into the complexity of decision making and emulation techniques, as well as comparing the performance of the different techniques. Six automation techniques were chosen to show how an attacker can choose to execute an operation in a given scenario. Three of the techniques only consider the immediately possible actions, while the other three operate using a planning paradigm, where they attempt to plan for actions to execute in the future. These planning-based strategies chain together multiple actions to achieve a future goal e.g., in order to achieve exfiltration, the plan could be to escalate, enumerate the host, and collect information to exfiltrate. The best possible plan to execute may not always be clear, as adversaries are often operating with uncertainty in the environment e.g., what services or endpoints exist within the network, or if the endpoints are vulnerable to exploits. To combat this the planning-based agents evaluate the situation by constructing a simulation of the environment based on knowledge currently available and evaluate possible actions. Once an action is executed the response is observed and added to the internal knowledge base of the agent.

A total of six agents, one for each automation technique, were included in Applebaum's experiment as attackers. These simulated attackers leveraged techniques from six different tactics in the MITRE ATT&CK framework [43]. The core of the MITRE ATT&CK framework, is a set of high-level tactics that describe the goal of an adversary by classifying the tactics and techniques commonly used by adversaries. Tactics represent certain objectives an adversary might have, such as privilege escalation or exfiltration. Techniques represent various ways to achieve a given objective, such as exfiltration over alternative protocols for instance Net/SMB or FTP. The six tactics simulated covered the following post-compromise behavior: lateral movement, privilege escalation, exploit execution, credential dumping, host discovery, and account discovery. The results of each agent were measured based on a percentage of: endpoints that the adversary established a foothold on, account credential obtained, endpoints that the adversary was able to execute exfiltration from.

The work done by Applebaum et al. [2] revealed the possibilities of automated decision techniques, despite the attacker having no prior knowledge or insight of the network. The planning-based strategies outperformed the immediate execution ones across all scoring methods. However, the time spent in order to make a decision was much longer on the planning-based strategies compared to the immediate-execution ones. While the planning-based strategies outperformed the immediate execution ones, there may be time-critical scenarios that necessitate quicker decision making.

### 2.7.2 CALDERA

One of the open-source tools available for automated adversary emulation is the tool CALDERA, developed by MITRE [6]. CALDERA offers an intelligent and automated adversary emulation, intended to test endpoint security of a Windows system against common post-compromise adversarial techniques. It comes with a remote access agent, a database, and server components. The database contains predefined attacks taken from the ATT&CK framework [43] which can be freely combined to create custom attacks. Once configured, CALDERA offers a graphical web interface that allows operators to control and monitor the agents, and give visual feedback on ongoing operations. Even without further configuration CALDERA offers several adversaries profiles, which are collections of several techniques and tactics, designed to create a specific scenario on an endpoint or network. These predefined scenarios are limited to persistence, privilege escalation, discovery, command and control communication using a remote access trojan (RAT), and lateral movement. However, CALDERA is not limited by these predefined scenarios, the database that comes with CALDERA allows operations to create customized attack scenarios in a user friendly manner. Through a dropdown menu operators can choose the desired tactics and technique, and directly add them to the desired scenario. These tactics and techniques can be further customized by editing the commands that are being executed. On the other hand, as CALDERA is developed for emulating post-compromise techniques, the attack tactics related to the initial compromise is limited to spear phishing techniques. CALDERA is thus best suited when the goal is to emulate post-compromise behavior.

The infrastructure of CALDERA consists of two main components: a master server, and a remote access tool. The remote access tool is initially placed on one or multiple endpoints within the network and communicates with the master server. The master server is responsible for the decision making process, constantly updating its internal knowledge database with the information received from the remote access tool. CALDERA uses a planning-based approach when deciding what actions to take, a plan is developed based on the current information in its knowledge database. From the initially compromised endpoint, CALDERA then expands its foothold onto other endpoints by placing new remote access tools once the endpoint has been compromised. Once the remote access tool is in place it starts to communicate back to the master server, by compromising new endpoints and collecting data from these the master server is able to gradually expand its knowledge database and map the compromised environment. This decision making process was tested and explained further by Applebaum et al. [3]. CALDERA was also tested on several small internal networks within the authors organization, where it was able to detect network misconfigurations and vulnerabilities.

Result from experiments conducted by Applebaum et al. [3] validated the intelligence component behind CALDERA, showing that the decision engine was able to string multiple actions together to create complex attack patterns, successfully generating attack artifacts and compromising new endpoints as it moves through the network.

# Chapter 3

# Architecture and Data Generation

As an attempt to address the shortcomings of existing datasets, Gharib et al. [18] presented eleven features necessary for a comprehensive and wholesome framework for generating IDS/IPS benchmarking datasets. Although the datasets and studies discussed are mainly regarding network related attacks, we belive the presented criteria can be applied for host based datasets. The following characteristics are presented: Complete Network Configuration, Complete Traffic, labeled Dataset, Complete Interaction, Complete Capture, Available Protocols, Attack Diversity, Anonymity, Heterogeneity, Feature Set and Metadata. The following characteristics can be derived from the work presented by Gharib to suite a host based data generation framework:

1. **Realistic Configuration:** In order to capture the real effects of the attacks in a realistic scenario, the testbed has to be realistically configured. Ideally, the attacks and systems should be as realistic as possible to the attacks seen in the wild.

2. **Complete Capture:** While including all of the occurring operations on an OS may result in larger log files, having a complete capture of this is beneficial when calculating the false-positive percentage of systems trained on the datasets. This involves not removing part of the traffic which is non-functional or deemed as noise. All interactions occurring on the system as the result of an attack must be captured and represented in the dataset. The dataset should represent complete scenarios (e.g., each step in a kill chain or specific scenario), and the resulting outcome of these attacks on the system. This also involves capturing data from other sources, such as network traffic.

3. **labeled Dataset:** Dataset can be unlabeled, partially-labeled or fully-labeled. While generating unlabeled datasets may be an easier task, the value of fully-labeled datasets are greater. Label datasets are useful beyond just as validation sets, a fully-labeled dataset may greatly reduce the false positive rate.

Higher label granularity also allows for more specific training e.g., detecting various stages of an attack or what tactic and technique are being detected. Partially-labeled involves having a small fully-labeled dataset and a larger unlabeled dataset, utilizing a semi-supervised approach when training the system. The desired dataset is a fully labeled datasets with preferably high label granularity, including both normal and malicious traffic.

4. **Attack Diversity:** One common shortcoming of existing datasets is their lack of attack diversity. This lack of diversity could cause biases on systems trained on the dataset. Cyber criminals are constantly changing and developing new attacks, covering all possible attack scenarios is not a feasible task. Instead, the goal could be to cover specific attack scenarios or tactics and the techniques within. One possible solution is a modular approach with multiple smaller datasets for each specific tactic or scenario, which later can be merged to a complete dataset. A modular approach allows researchers and defenders to specifically select the attack data relevant.

5. **Anonymity:** Another common shortcoming is the privacy issue which is halting the exchange of datasets stemming from real world attack data. In order to avoid these issues the datasets are scrubbed for personal data, this anonymization often leads to valuable artifacts being removed from the datasets. The privacy issues are removed from synthetic datasets as these are created in a controlled lab environment.

6. **Heterogeneity:** A dataset can be considered homogeneous if the samples have similar or identical traits or heterogeneous if the opposite is true. This could be in the form of including multiple log sources (e.g., network traffic, operating system logs, or network equipment logs). In the case a of pure host based dataset, this can arguably be achieved through the inclusion of logs regarding file systems, AV solution, certificates, processes and call traces. While homogeneous datasets are useful for analyzing specific types of detection, a heterogeneous dataset provides a deeper insight into all aspects of the detection.

The following chapter describes the design and configuration of the developed environment and the systems within it. Further, it presents the actions and operations conducted in order to generate both malicious and benign data.**Section 3.1** gives a overview of the environment, while **Section 3.2** presents the Windows and network domain within the environment. **Section 3.3** and **Section 3.4** describes the configuration of Sysmon and the hosts present in the environment. **Section 3.5** describes how the attack data was generated, actions taken in order to generate the data, and a detailed description of the chosen attack simulation. **Section 3.6** covers how the benign data was generated.

## 3.1 Environment Overview

This section will provide an overview of the developed environment and its components used during this research. The environment consists of multiple virtual machines (VMs) hosted on a personal machine. Vagrant [47] is utilized in order to build the computer domain which will act as the enterprise network, while the CALDERA server is hosted on a separate Ubuntu VM. Hosting CALDERA on a separate server removes the potential of having fragments related to the operation of CALDERA within the developed dataset.

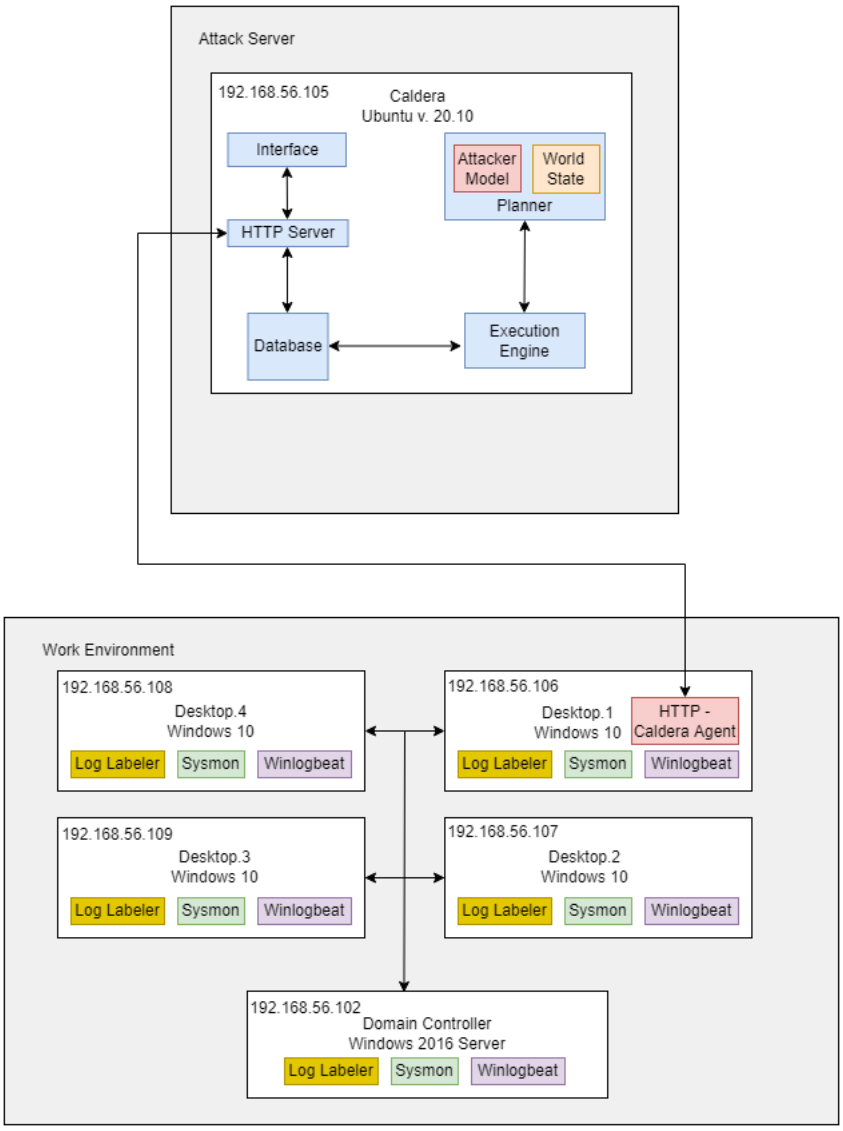



Figure 3.1: Environment overview

Initially, the framework only consisted of a single virtual machine acting as the host, and a Ubuntu server hosting CALDERA.

Vagrant [47] and DetectionLab [15] was later introduced in order to develop a more realistic configuration of an enterprise network , with computer domains, multiple hosts, and a domain controller. The addition of multiple hosts presents CALDERA with possible targets for lateral movement. This environment was developed for the purpose of generating raw logs, the labeling tool is independent of the work environment, but depended on CALDERA performing the attack. The labeling tool is explained in greater details in **subsection 4.2.1**.

## 3.2 Windows Domain

The Windows domain is mainly built upon the work conducted by Chris Long and his DetectionLab project [15]. Detectionlab comes pre-configured with 4 total hosts:

1. **Domain Controller**: Domain controller on a Windows 2016 server
2. **Windows Event Forwarder**: WEF, A Windows 2016 server that manages Windows Event Collection
3. **Endpoint Host**: A Windows 10 host to simulate a non-server endpoint
4. **Logger**: An Ubuntu 16.04 host running Splunk for collecting logs.

This domain was then customized by replacing the logger and event forwarder (WEF) server with three additional endpoint hosts to serve as possible targets for lateral movement attacks, the final domain and environment is illustrated in **Figure 3.1**. By removing these servers we are eliminating the possibility of unwanted artifacts in the logs as a result of the endpoints connecting and transmitting data to the WEF and Logger. For this experiment, the domain is limited to four endpoints, one domain controller and one Ubuntu server hosting CALDERA. Given the nature of this experiment, WEF and Logger servers were deemed redundant.

### 3.2.1 Domain Network

All Windows-related hosts are hosted on the same private virtual network with the domain controller to simulate the work environment. Each of the configured Windows hosts has a preferred predefined IP within the range of 192.168.56.[106-109], where the domain controller was given the IP of 192.168.56.102. Some of the payloads used in the experiment are pre-compiled and configured to to connect back to the static IP address of 192.168.56.105, CALDERA was hosted on this IP address to avoid any potential callback issues.

## 3.3 Sysmon configuration

While not strictly necessary, having a Sysmon configuration based on the use case is recommended as it helps to tune and filter the logs generated before processing them. Sysmon will quickly fill up the log files with the default configuration. The built-in filtering abilities of Sysmon are a part of what makes Sysmon a more sophisticated tool than the built-in windows audit log feature, it would be unwise not to take advantage of this. By restricting the log volume and removing irrelevant data, we minimize possible constraints on the system and network which may occur as a result of large log files being transmitted and processed.

Applying filters to Sysmon can be done by applying filter conditions such as: is, is not, contains, contains any, contains all, excludes, excludes any, excludes all, begin with, end with, less than, more than, image. Used in conjunctions with the onmatch rule in Sysmon, which states whether or not the events are to be logged, administrators can create very specific configurations files, customizable down to any information regarding the process e.g., hash, filename, IP, filepath, port number, etc. Two recommended configurations of Sysmon are that of SwiftOnSecurity [41], or the configuration from Olaf Hartong [36]. Both of these configurations were studied and found quite similar, effective at filtering out the most noisy background process while maintaining coverage of known attack vectors. The major difference in Hartong's configuration file was the inclusion of detection based on image loaded (e.g., DLL files), file deletion/overwrite events, processes accessing other processes and process tampering. These were disabled by default from SwiftOnSecurity, with the argument that they could potentially cause high system load. However, it was found that the inclusion of these, through Hartong's configuration file, did not cause high system load on our test-bed. This is mostly likely due to the detailed filtering in place.

#### Examining Configuration Files

An experiment was conducted in order to test the system load and coverage of both the configuration file of SwiftOnSecurity and Olaf Harton. Multiple different attack scenarios were executed, including operations related to discovery, command and control, credential dumping, execution, privilege escalation through process injection and defense evasion. Olaf Hartong's configuration file did detect more benign background activities in comparison to the configuration of SwiftOnSecurity, however this was quickly outweighed by the additional insight granted from monitoring DLL loading, process access and tampering. Variations in system load was not noticeable during the execution, this may be due to the limited timespan of which the experiment was conducted. Olaf Hartong's configuration file is chosen as the desired configuration as this configuration was successful in detecting several of the malicious operations not detected by SwiftOnSecurity and the system load was not impacted to an notable extent. The configuration file from Olaf Hartong is therefor applied.

## 3.4 Host Configuration

All Windows hosts are running Windows 10, Domain Controller is running on a Windows Server 2016 and the CALDERA server is running on Ubuntu 20.10. Within DetectionLab, Active Directory (AD) is pre-configured and the hosts have their Windows Audition configurations set through the group policy (GPO) in AD, including command-line process auditing.

The logging configuration is based on the best practices by Microsoft [32], which can be expected to be used in realistic configurations. Once new hosts are added to the domain they are also added to the group which in return sets their Audition configurations. This process is done automatically when adding new hosts to the domain through the vagrant configuration. The following steps were applied to each host in order to prepare the domain:

- Installing Sysmon and applying the configuration file of Olaf Hartong [36]
- Installing and configuring Winlogbeat to collect and convert EVTX log files into JSON files. Due to how the tool is developed, the logs need to be in JSON format in order for the labeling tool to process the logs.
- Deactivating Windows Defender, optional configuring it to notify only.
- Installing and configuring the GHOSTS Client agent for simulating benign user activities.

In addition to this, the CALDERA agent was placed on one single host to grant CALDERA entry to the work environment. This host will act as the initial attack vector. Only one host is initially infected in order to simulate one compromised host in an entire network. CALDERA will attempt to move laterally from this host in order to infect the rest of the environment.

## 3.5 Attack Data Generation

Generation of attack data was conducted explicitly through the use of the adversary emulation tool CALDERA. The generated data is related to typical post compromise behavior, as one host in the environment is already infected at the start of the emulation.

### 3.5.1 CALDERA Agent

A CALDERA agent that communicates back to the server through a specified HTTP contact was dropped onto the first endpoint. The agent contacts the server based on the hosting IP of the server and on port 8888.

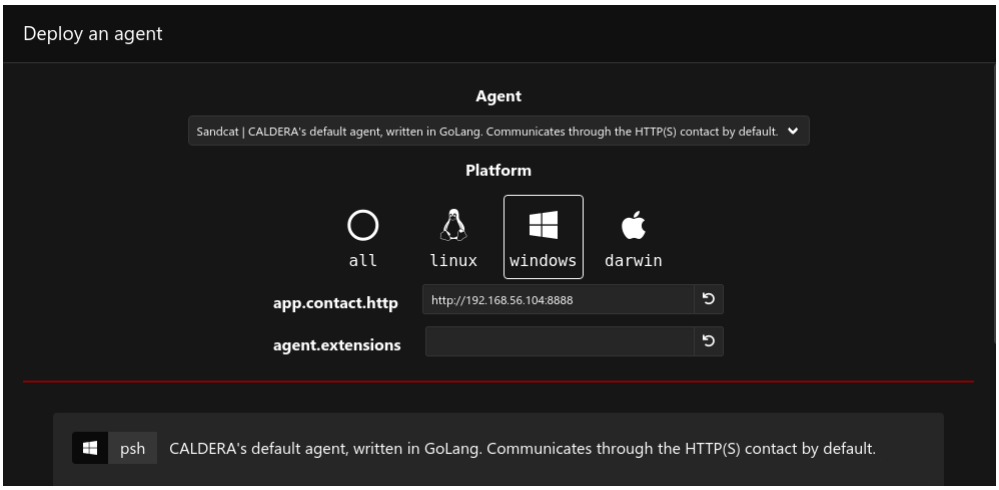



Figure 3.2: Deploying agent through CALDERA interface

The agent is created through the interface of CALDERA, where the user can specify what operating system the attack target is running. CALDERA will dynamically build and compile the agent based on the information supplied. In order to drop the agent on the target host the following code from CALDERA is manually executed through PowerShell on the target machine:

```
1 $server="http://192.168.56.105:8888";
2 $url="$server/file/download";
3 $wc=New-Object System.Net.WebClient;
4 $wc.Headers.add("platform","windows");
5 $wc.Headers.add("file","sandcat.go");$data=$wc.DownloadData($url);
6 $name=$wc.ResponseHeaders["Content-Disposition"].Substring(
7 $wc.ResponseHeaders["Content-Disposition"].IndexOf("filename=")+9)
      .Replace("`"","");
8 get-process | ? {$_.modules.filename -like "C:\Users\Public\$name.
      exe"} | stop-process -f; rm -force "C:\Users\Public\$name.exe"
       -ea ignore;[io.file]::WriteAllBytes("C:\Users\Public\$name.
      exe",$data) | Out-Null;Start-Process -FilePath C:\Users\Public
      \$name.exe -ArgumentList "-server $server -group red" -
      WindowStyle hidden;
```

Listing 3.1: Deploy CALDERA Agent

Once the code has been executed the CALDERA agent is fetched from the CALDERA server and displayed in the agents tab of the interface.

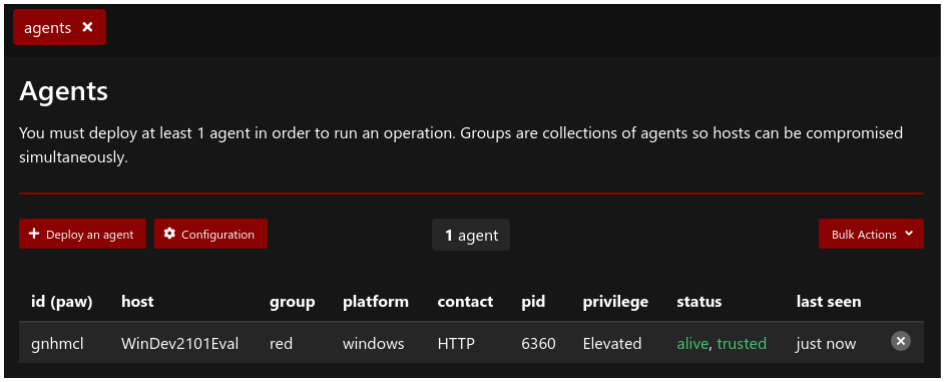



Figure 3.3: Agent Tab

### 3.5.2 Executing Attack Scenario

Attack scenarios may be executed once a agent has been placed on a targeted host machine. Through the interface provided by CALDERA, the attacker may choose to either execute manual commands, choose amongst a total of 1228 abilities (924 for the windows platform) relating back to specific tactics and techniques from MITRE ATT&CK, or execute entire attack scenarios which can be manually created or directly imported from the adversaries profile of CALDERA.

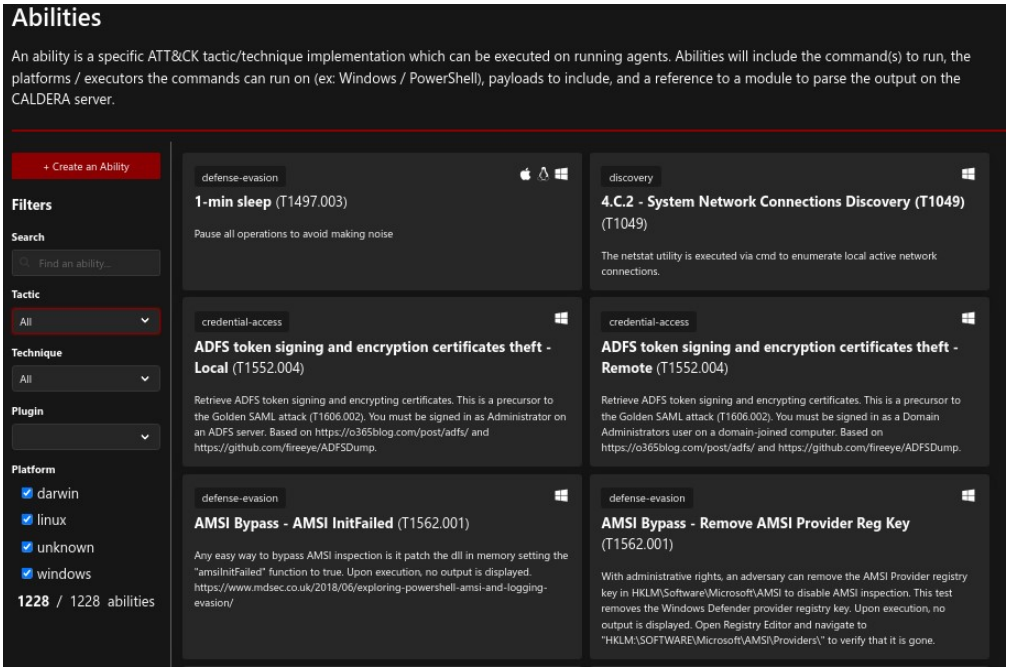



Figure 3.4: CALDERA Abilities search

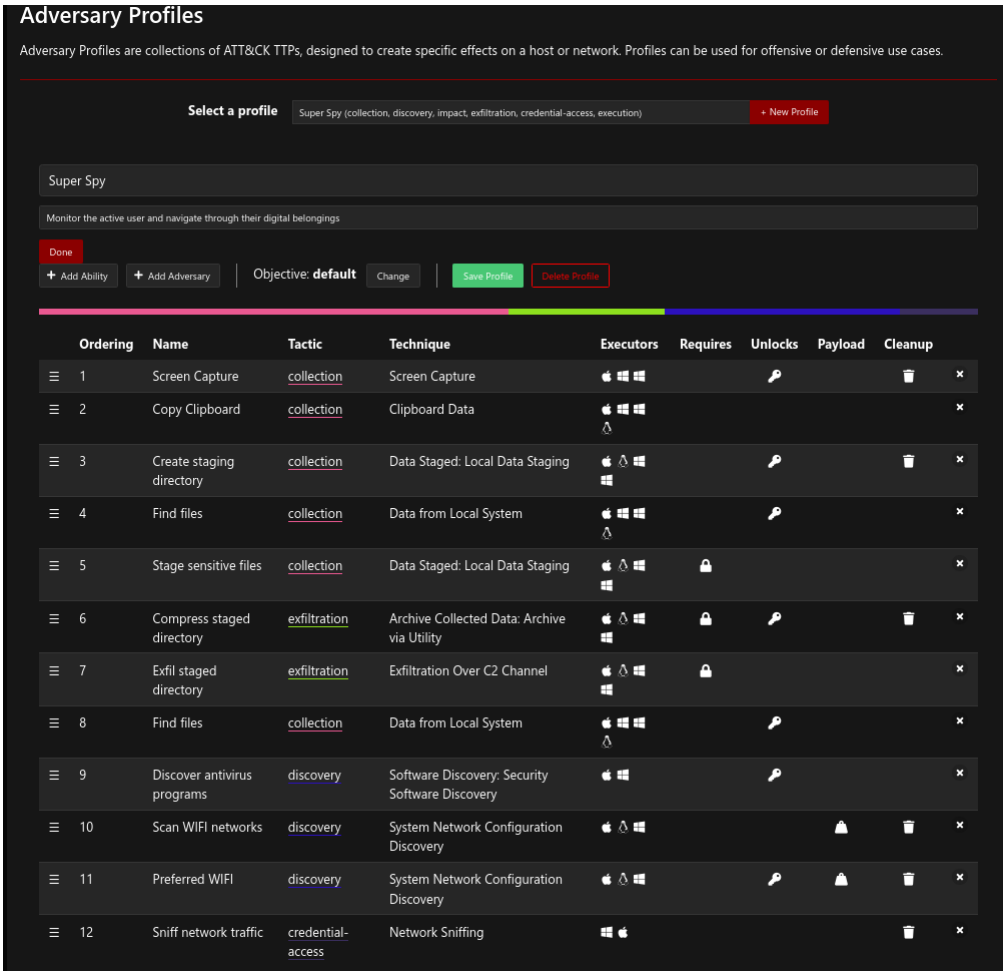



Figure 3.5: CALDERA Adversary Profile

Some of these smaller scenarios e.g., **Figure 3.5**, were executed during the testing and development phase of the project, initially to test the detection capabilities of Sysmon, and later during the development and improvements of the labeling tool. Having smaller logs also made manually inspecting the resulting logs more feasible. With CALDERA version 4.0.0, MITRE implemented a new plugin which integrates MITRE's adversary emulation library [4] with CALDERA. The emulation library contains fully developed emulations plans for multiple threat groups, including APT29 [22].

### 3.5.3 APT29 Emulation

The emulation plan of APT29 is developed from publicly available sources, describing the motivations, objectives and attributed tactics, techniques, and procedures mapped to MITRE ATT&CK. APT29 is a threat group that has been attributed to the Russian Foreign Intelligence Service (SVR), who have been in operation since at least 2008. This group reportedly compromised the Democratic National Committee starting in the summer of 2015, and was named as one of the perpetrators of the cyber espionage campaign that exploited the SolarWinds Orion platform [46]. The emulation plan chains together techniques into a logical order that has been observed across previous APT29 operations. The operations are divided into two distinct scenarios, with a total of 20 steps and 79 operations conducted through CALDERA.

#### Scenario 1

Scenario one consists of a rapid espionage mission that focuses on gathering and exfiltrating data, before transitioning into stealthier techniques in order to achieve persistence, further data collection, credential access, and finally lateral movement.

#### Step 1 - Initial Breach

It starts with an initial breach, where a user clicks an executable payload which creates a command and control connection over port 1234 and spawns a interactive shell for further steps.

| | Ordering | Name | Tactic | Technique | Executors |
|---|---|---|---|---|---|
| ☰ | 1 | RTLO Start Sandcat | execution | Masquerading: Right-to-Left Override | |
| ☰ | 2 | PowerShell | execution | Command and Scripting Interpreter: PowerShell | |

Figure 3.6: Step 1 CALDERA Operations

#### Step 2 - Rapid Collection And Exfiltration:

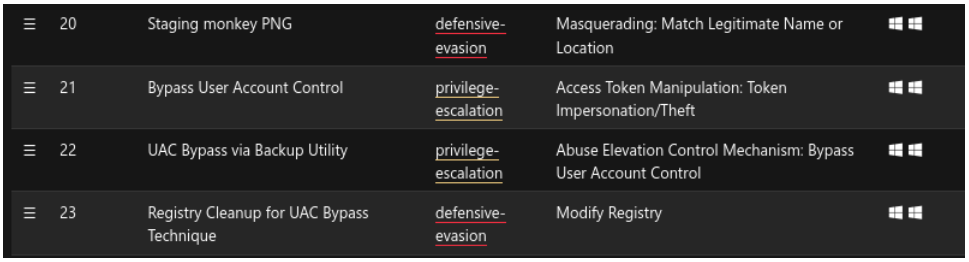
A one-liner command is executed on the targeted machine, searching the filesystem for documents and media files. These are then collected and the content is compressed into a single file. Once the file is ready for transfer it is exfiltrated over the existing command and control connection. This step consist of the collection technique T1119, T1005, T1002, and the exfiltration technique T1041.

| | | | | |
|---|---|---|---|---|
| ☰ | 3 | Automated Collection | collection | Automated Collection |
| ☰ | 4 | System Network Configuration Discovery (2) | discovery | System Network Configuration Discovery |
| ☰ | 5 | System Network Configuration Discovery (2) (2) | discovery | System Network Configuration Discovery |
| ☰ | 6 | System Owner / User Discovery | discovery | System Owner/User Discovery |
| ☰ | 7 | Data from staged fileand Exfiltration over C2 Channel | exfiltration | Exfiltration Over Command and Control Channel |

Figure 3.7: Step 2 CALDERA Operations

### Step 3 - Deploy Stealth Toolkit

During this step T1105 - Ingress Tool Transfer is used to deploy a new payload containing a PowerShell script concealed in a legitimately formed image file to the targeted machine. T1122 and T1088 are then used in order to escalate the privileges through user account control (UAC) bypass, followed by the execution of the new payload. Artifacts of the privilege escalation are then removed from the Registry utilizing T1112 before initiating the next step.

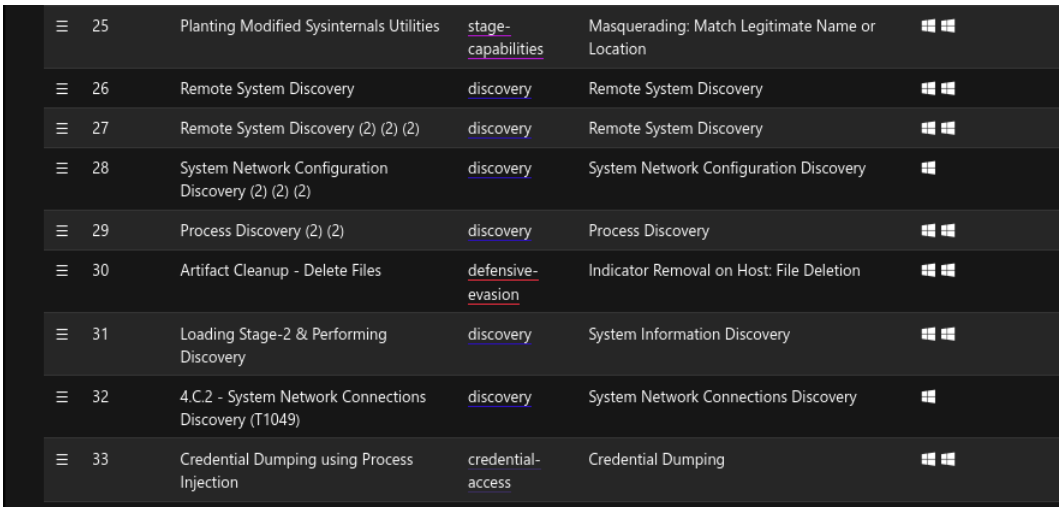


| | | | | |
|---|---|---|---|---|
| ☰ | 20 | Staging monkey PNG | defensive-evasion | Masquerading: Match Legitimate Name or Location |
| ☰ | 21 | Bypass User Account Control | privilege-escalation | Access Token Manipulation: Token Impersonation/Theft |
| ☰ | 22 | UAC Bypass via Backup Utility | privilege-escalation | Abuse Elevation Control Mechanism: Bypass User Account Control |
| ☰ | 23 | Registry Cleanup for UAC Bypass Technique | defensive-evasion | Modify Registry |

Figure 3.8: Step 3 CALDERA Operations

### Step 4 - Defense Evasion and Further Discovery

T1105 is then used again to upload additional tools through the newly elevated access, before a new interactive PowerShell shell is spawned. The tools are then decompressed and placed onto the target for further usage through the newly spawned shell. Using the Tasklist utility through PowerShell (T1057), running processes are enumerated in order to discover and terminate the initial access from step 1, various files associated with the initial access are also deleted. A PowerShell script is then executed, which in return performs a wide variety of reconnaissance commands (T1016, T1033, T1063, T1069, T1083).

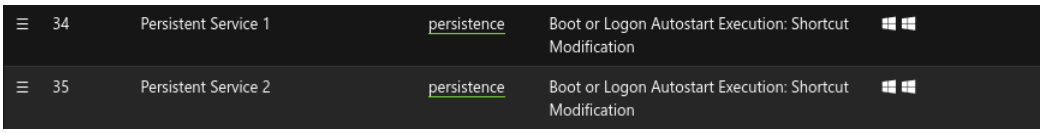


| | | | | |
|---|---|---|---|---|
| ☰ 25 | Planting Modified Sysinternals Utilities | stage-capabilities | Masquerading: Match Legitimate Name or Location | |
| ☰ 26 | Remote System Discovery | discovery | Remote System Discovery | |
| ☰ 27 | Remote System Discovery (2) (2) (2) | discovery | Remote System Discovery | |
| ☰ 28 | System Network Configuration Discovery (2) (2) (2) | discovery | System Network Configuration Discovery | |
| ☰ 29 | Process Discovery (2) (2) | discovery | Process Discovery | |
| ☰ 30 | Artifact Cleanup - Delete Files | defensive-evasion | Indicator Removal on Host: File Deletion | |
| ☰ 31 | Loading Stage-2 & Performing Discovery | discovery | System Information Discovery | |
| ☰ 32 | 4.C.2 - System Network Connections Discovery (T1049) | discovery | System Network Connections Discovery | |
| ☰ 33 | Credential Dumping using Process Injection | credential-access | Credential Dumping | |

Figure 3.9: Step 4 CALDERA Operations

**Step 5 - Persistence Establishment**

Two distinct means of persistent access is established. One through creating a new windows service (T1031), and the second one by creating a malicious payload in the Windows Startup Folder (T1060) which executes when the machine starts up.



| | | | | |
|---|---|---|---|---|
| ☰ 34 | Persistent Service 1 | persistence | Boot or Logon Autostart Execution: Shortcut Modification | |
| ☰ 35 | Persistent Service 2 | persistence | Boot or Logon Autostart Execution: Shortcut Modification | |

Figure 3.10: Step 5 CALDERA Operations

**Step 6 - Credential Access**

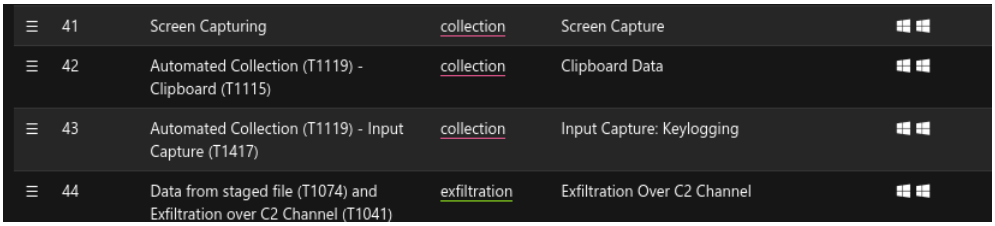
A tool delivered in step 4 is used to access credentials stored in local web browsers (T1081, T1003). This tool is masqueraded as accesschk.exe (T1036), which is a legitimate utility tool. Private keys are harvested leveraging T1552.004, searching for file with common key and certificate file extensions e.g., .key, .pgp, .gph, .ppk. Password hashes are extracted and harvested from the Security Account Manager (SAM) database through the Registry (T1003).

| | | | | |
|---|---|---|---|---|
| 36 | Access Token Manipulation (2) | defensive-evasion | Access Token Manipulation | |
| 37 | Credentials In Files- Chrome | credential-access | Credential Dumping | |
| 38 | Query Registry (2) (2) | defensive-evasion | Access Token Manipulation | |
| 39 | Credentials In Files (T1081) - Private Keys Extraction | credential-access | Unsecured Credentials: Private Keys | |

Figure 3.11: Step 6 CALDERA Operations

### Step 7 - Collection and Exfiltration

This step involves the collection and exfiltration of screenshots (T1113), data from the victim's clipboard (T1115), and keystrokes (T1417). These files are then exfiltrated over the command and control channel (T1041).

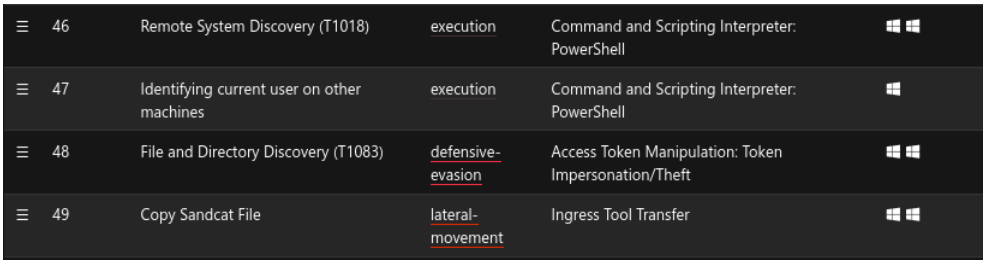


| | | | | |
|---|---|---|---|---|
| 41 | Screen Capturing | collection | Screen Capture | |
| 42 | Automated Collection (T1119) - Clipboard (T1115) | collection | Clipboard Data | |
| 43 | Automated Collection (T1119) - Input Capture (T1417) | collection | Input Capture: Keylogging | |
| 44 | Data from staged file (T1074) and Exfiltration over C2 Channel (T1041) | exfiltration | Exfiltration Over C2 Channel | |

Figure 3.12: Step 7 CALDERA Operations

### Step 8 - Lateral Movement

Other hosts in the domain are enumerated through the use of LDAP queries (T1018), before creating a remote PowerShell session onto a secondary victim (T1021), potential lateral movement targets can be seen in **Figure 3.1**. A payload is then loaded onto the new victim (T1027) and executed utilizing the PSExec utility and the credentials stolen (T1078) in Step 6.



| | | | | |
|---|---|---|---|---|
| 46 | Remote System Discovery (T1018) | execution | Command and Scripting Interpreter: PowerShell | |
| 47 | Identifying current user on other machines | execution | Command and Scripting Interpreter: PowerShell | |
| 48 | File and Directory Discovery (T1083) | defensive-evasion | Access Token Manipulation: Token Impersonation/Theft | |
| 49 | Copy Sandcat File | lateral-movement | Ingress Tool Transfer | |

Figure 3.13: Step 8 CALDERA Operations

### Step 9 - Collection

Additional tools are uploaded to the secondary victim (T1105) before initiating a search for documents and media files on the file system (T1083, T1119). The files are then collected (T1005), encrypted, and compressed (T1002, T1022) into a single file (T1074) before exfiltration through the existing C2 connection (T1041). Files associated with this access are deleted (T1107) before moving on to the next step.

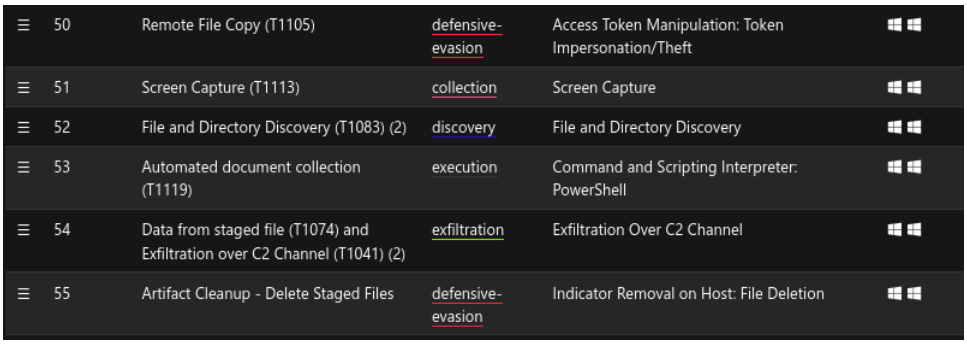



Figure 3.14: Step 9 CALDERA Operations

### Step 10 - Execution of Persistence

In this step, the original victims machine is rebooted, triggering the established persistence mechanism in step 5. This includes the execution of the new service (T1035), and the payload in the Windows Startup folder (T1060).

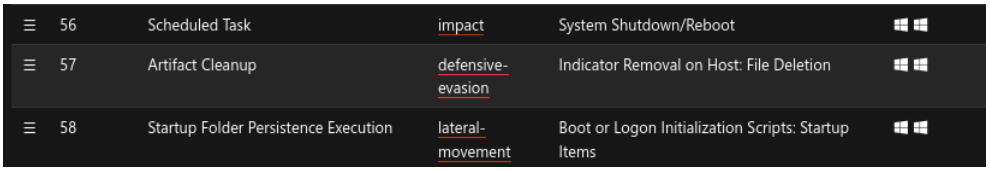



Figure 3.15: Step 10 CALDERA Operations

### Scenario 2

Once step 10 in scenario 1 has been completed CALDERA moves onto scenario 2 of the emulation. This scenario consist of a stealthier and slower approach to compromise the initial target, establish persistence, gather credentials, enumerating and compromising the entire domain.

### Step 11 - Initial Breach

Similar to Step 1 in scenario 1, the initial breach of scenario 2 occurs once a legitimate users clicks a link file payload, which in return executes an alternate data stream (ADS) hidden on another file (T1096).

Unlike the payload in step 1, this payload verifies that it is not executing in a virtualized analysis environment through a series of enumeration commands (T1497, T1082, T1033, T1016, T1057, T1083) before establishing persistence through a Windows Registry Run key entry (T1060). Finally, the ADS executes a PowerShell stager to create a command and control connection over port 443.

**Step 12 - Fortify Access**

In this step the access is fortified by modifying the time attributes (T1099) of the command and control payload, matching it with the time attribute of a random file found in the System32 directory. Software discovery is then performed in order to discover registered AV products (T1063) before querying the Windows Registry (T1012) to find software installed by the user.

**Step 13 - System Enumeration**

Local enumeration is performed using various Windows API calls in order to collect local system information (T1082), network configuration (T1016), current user context (T1033), and running processes (T1057).


Figure 3.16: Step 11-13 CALDERA Operations

**Step 14 - Privilege Escalation**

Privileges are elevated by a user account control (UAC) bypass (T1122, T1088). The elevated access is then used to create and execute code within a custom Windows Management Instrumentation (WMI) class (T1047). This code downloads and executes Mimikatz in order to dump credentials (T1003.001) from the Local Security Authority Subsystem Service (LSASS).

### Step 15 - Persistence Establishment

Additional persistence is established by creating a WMI event subscription (T1084) which executes a PowerShell payload when the user signs in.

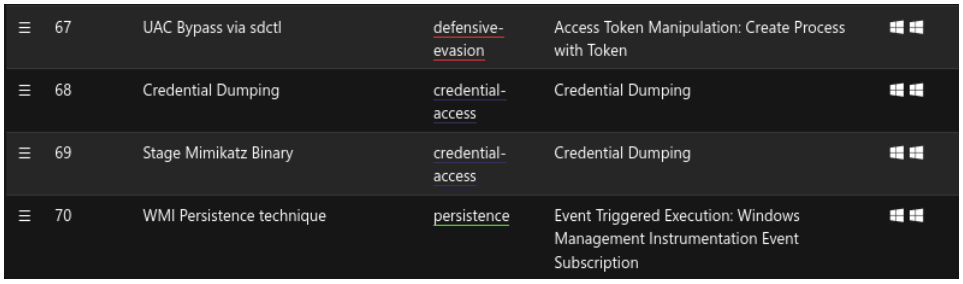



Figure 3.17: Step 14 and 15 CALDERA Operations

### Step 16 - Lateral Movement

Here the native Windows application programming interface (Windows API) is used to enumerate the domain controller of the environment (T1018) and the domains security identifier (SID) (T1033). The domain controller can be seen in **Figure 3.1**. Once these are discovered, a new remote PowerShell session to the domain controller (T1028) is created by using the credentials dumped from step 14 (T1078.002). Mimikatz is then copied to the domain controller using this new connection in order to dump the hash of the KRBTGT account.

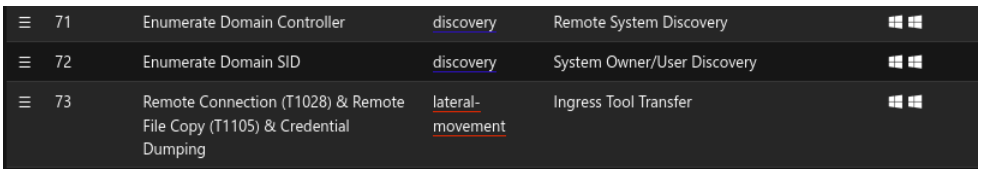



Figure 3.18: Step 16 CALDERA Operations

### Step 17/18 - Collection and Exfiltration

In this step, emails stored in the local email client are harvested (T1114) and collected (T1005). The collected data is then staged, compressed, and obfuscated as a GIF file (T1072) by changing the first few bytes used to recognize filetypes. The file is then exfiltrated through OneDrive (T1048).

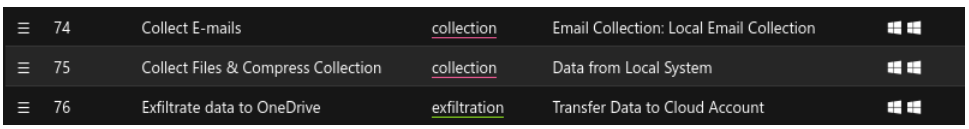



Figure 3.19: Step 17 and 18 CALDERA Operations

### Step 19/20 - Clean up and Execution of Persistence

Clean up is performed by loading and executing the Sdelete binary (T1055) within PowerShell, making the deleted files associated with the access nearly unrecoverable. The emulation ends with a reboot of the infected client, triggering the previously established persistence mechanisms.

| | | | | |
|---|---|---|---|---|
| ☰ | 77 | Data Wiping of staged files | impact | Disk Wipe: Disk Content Wipe |
| ☰ | 78 | Execute Invoke-Mimikatz | credential-access | Credential Dumping |
| ☰ | 79 | Triggering Persistent | persistence | Signed Binary Proxy Execution: Rundll32 |

Figure 3.20: Step 19 and 20 CALDERA Operations

### Emulated Techniques and Tactics

**Figure 3.21** illustrates the expected distribution of techniques and tactics based on the sequence of events from the steps above and information extracted from MITRE ATT&CK Emulation Library [4].

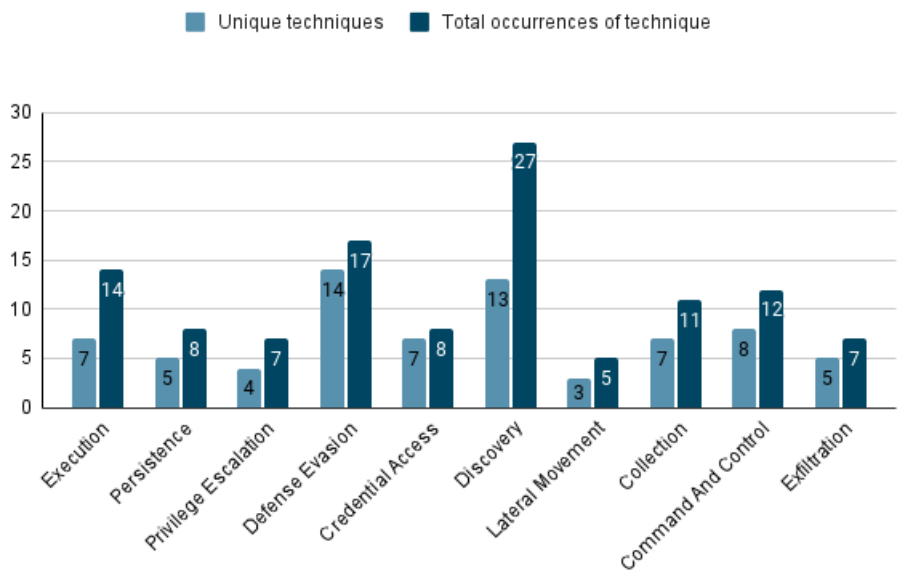



Figure 3.21: Distribution of expected techniques in APT29 emulation plan

The majority of techniques in this emulation are related to the Discovery tactic, with 13 unique techniques leveraged across 27 individual operations followed by defense evasion techniques. Lateral movement techniques is the least represented tactic in this emulation, with only 5 operations involving lateral movement techniques. However, the distribution of tactics in the logs may differ from the distribution seen in **figure 3.21** as some tactics will generated multiple log lines related to the same operation while other will generated fewer log lines.

## 3.6 Benign Data Generation

The inclusion of benign data in a dataset is crucial when training an Intrusion Detection/Prevention system. In order to train the system to differentiate between malicious and benign actions, both must be present in the training set. Benign data generated, and expect behavior, is tied to what end user is desired to simulate e.g., average user, administrator, or software developer. It is expected that administrators will perform more advanced tasks in comparison to an average end user. Typically an administrator will perform tasks such as network or host maintenance/configuration, or executing command line operations. This will manifest itself completely different in the logs in comparison to the logs generated from an average user or a software developer. Simulating a benign user can be accomplished in various ways. The user may be simulated by manually performing various tasks on the client, running scripts which execute the desired tasks, or by utilizing automation software. In order to support the reproducibility of the datasets generated in this thesis an automation software was chosen to generated the benign data.

### 3.6.1 GHOSTS NPC Simulator

In order to preserve the reproducibility of the dataset, benign user data was generated through the GHOSTS (General HOSTS) framework [44] developed by the Software Engineering Institute (SEI) at Carnegian Mellon University[1]. The framework is used to build autonomous non-player characters (NPC) that can represent an array of possible encounters through simulation of users, contexts, and situations across computer networks. With the goal of creating behavior patterns that would realistically mimic typical human behavior. GHOSTS operates in a similar manner to CALDERA, utilizing an agent placed onto the targeted machine in order to execute predetermined commands. GHOSTS consist of two main parts: a server handling the distribution of tasks and collection of reports from the clients; and a client responsible for the automation and execution of tasks. The server is hosted in a Docker container and is further divided into three parts with individual containers:

- Grafana - Vizualisation tool giving an overview of the clients and their activity

[1] https://www.sei.cmu.edu/

- API Server - Responsible for managing the clients
- PostgreSQL Database - Responsible for storing the information and data generated by the clients.

GHOSTS offers the possibility to configure specific timelines which are used to specify what user actions should be taken at which point in time. User actions are predefined within the framework, it is however possible to modify these or develop customized actions if necessary. **Figure 3.22** outlines the operational capabilities in GHOSTS. Highlighted in blue color are the capabilities most likely to generate relevant host based logs. However, web browsing and e-mail activities will still be present in the logs as Sysmon detects the outgoing traffic.

| Capability | User Action | Methods |
|---|---|---|
| Web Browsing<br>(Network) | Browse<br>Enter text<br>Click link or button | Random, Specific, Looping |
| Terminal Commands<br>(Host) | Execute cmd commands<br>Execute Powershell commands | Random, Specific, Looping |
| Inter-NPC communication<br>(Network) | Email creation and management | Specific, Looping |
| Office Document Management<br>(Host) | Creation and saving of common file formats for word processor, spreadsheet, and presentation documents | Random, Specific, Looping |

Figure 3.22: GHOSTS Capabilities [45]

### 3.6.2 GHOSTS Timeline

The timeline for generating benign traffic is not tailored to simulate one specific type of user e.g., administrator, or average user. It will instead focus on taking advantage of the possible capabilities and user actions available in GHOSTS and the timeline repository [19]. Since Microsoft Office user actions requires an valid Microsoft Office license key which is not present in the Windows VM's, the capabilities utilized will be web browsing, terminal commands and e-mail creation/management.

# Chapter 4

# Data Collection and Labeling

## 4.1 Data Collection

Data collection is done once the emulation has been executed and completed. The data collected is limited to Sysmon logs, some of the possible traits covered by Sysmon are:

- Process creating with full command line for both current and parent processes.
- Process ID and GUID that allow for correlation of events
- Driver and DLL loading.
- Read access of disk and volumes.
- Network connections, including source process, IP address, port numbers, hostnames, and port names.
- File changes, creations, and modifications.
- Changes in the registry.

### 4.1.1 Collecting Windows Logs

Windows logs are by default in EVTX format, with an underlying XML structure. These logs can be extracted either directly through the folder `C:\Windows\System32\winevt\Logs` in EVTX format, or through Windows Event Viewer. Windows Event Viewer allows for the log files to be exported in either EVTX, XML, Comma Separated (CSV) or plain text (txt) format. However, the EVTX format is only used by Microsoft and not supported by other operating systems. Ideally, the logs collected should use a common representation format to account for information sharing amongst different components. This is done to ensure that individual components (e.g., logging, alert-generation, analysis), from different vendors, can work together. Raw log files are also typically unstructured, making it difficult for humans to read, and to query the logs for useful information.

An alternative format is JavaScript Object Notation (JSON), which is a highly readable data-interchange format, it is easy for humans to read and for machines to parse [23]. JSON format has grown quite popular as a standard format for structured logging, it is both readable and reasonably compact. JSON format provides a standardized format for structuring data and most programming languages can parse it. A standardized format is desired in the case of other operating systems being added as further work on this thesis.

### 4.1.2 Winlogbeat

Winlogbeat [**Winlogbeat**] was introduced to developed environment as a tool for both log collection and file conversion. Winlogbeat is most commonly used as a data shipper which sends the collected logs from clients to a centralized log management solution. Event logs are read using WindowsAPIs, filtered based on user-configured criteria and then sent to the configured outputs. Winlogbeat can capture event data from any event logs running on the system e.g, application, hardware, security or system events. It is part of the "Beats family" from the Elasticsearch Stack [17]. Beats is a free and open platform for data shippers. The following are the types of Beats from Elasticsearch:

- *Filebeat*: Comes with internal modules that simplify the collection, parsing, and visualization of common log formats.
- *Metricbeat*: Shipper for Metrics, collects metric information from the client system e.g, CPU usage, memory, file system, disk and network IO statistics.
- *Packetbeat*: Collection and shipper for Network Data.
- *Winlogbeat*: Collection and shipper for Windows Event Logs from windows systems.
- *Auditbeat*: Collection and shipper for Linux audit data, and monitors the integrity of files.

With the scope of this thesis being Windows Logs and endpoint detection, Winlogbeat is the only Beat being implemented. The addition of new Beats is possible for further work. Packetbeat could be utilized for the collection of detailed network related data, the Sysmon configuration in place monitors the relevant network connection. In this case, this is only limited to C2 traffic and data exfiltration. These events will not go undetected as they can be detected by Sysmon as outgoing network connections.

A drawback with Winlogbeat is the increase of the log sizes in comparison to CSV log files. JSON files are generally larger than CSV files as more characters and elements are used to represent the same data. In addition to this, multiple new fields are added to each event by Winlogbeat.

As Winlogbeat essentially is a log shipper, designed to ship logs to a centralized log management solution, theses fields are added in order to differentiate between the different clients sending the logs and for query purposes. A file comparison conducted on log files processed by Winlogbeat and raw CSV log files revealed that Winlogbeat files are on average 340% larger. However, compared to an equivalent EVTX file, the JSON files are on average 53% smaller. The trade-off for a standardized format and ease of processing is that of file size. The difference in file size can be reduced by dropping irrelevant fields added by Winlogbeat. Fields can be dropped through the configuration of Winlogbeat, the drop_field processor in the configuration file of Winlogbeat specifies which fields to drop.

## 4.2 Data Processing

Processing of the data begins once the desired emulation has been completed. Winlogbeat is responsible for extracting the logs generated by Sysmon and converting these files from .EVTX to .JSON format before storing these on a specified location. The developed labeling tool queries for new files in this specific location, if new files are found the tool will start processing the files. Each line of the file is then read by the tool and labels are applied. The label process and tool is described in greater detail in the following subsections.

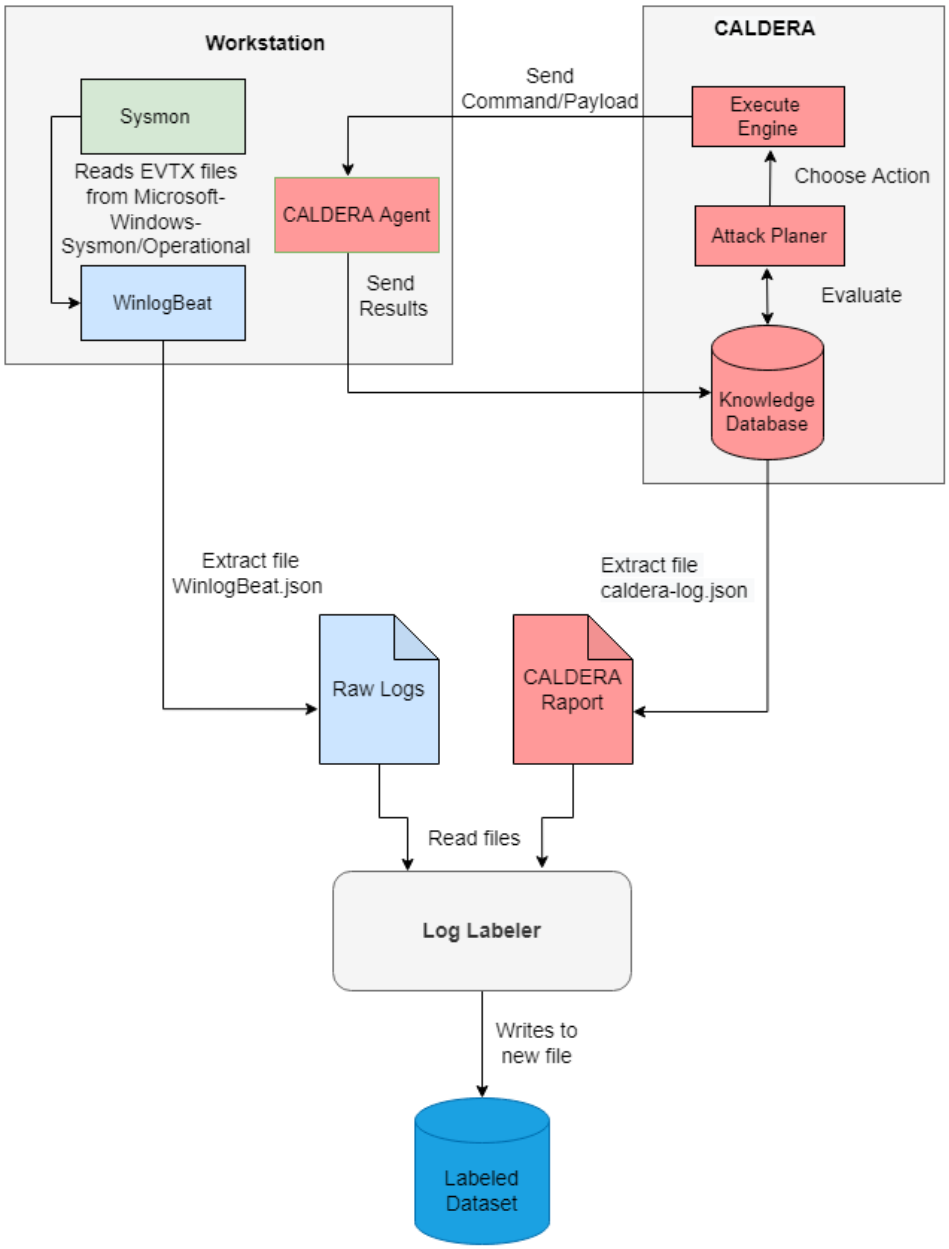



Figure 4.1: Data processing overview

### 4.2.1 Applying labels to logs

As part of the contribution from the research conducted in this thesis, a tool was developed that can convert raw log files into labeled datasets which can be used to train machine learning based intrusion detection systems. This involved labeling each line in the log file stating whether the action was part of a malicious or benign operation, as well as what step in the attack chain, in terms of MITRE ATT&CK technique or tactic, the malicious operation is related to. The efficiency of the IDS is highly dependent on correctly labeled training data as highlighted by Catania et al. [7] and Davis et al. [14], as a part of quality assurance the processed log files were manually reviewed and analyzed. The labeling tool searches for JSON files created by Winlogbeat from the Sysmon logs in a specified folder where unprocessed logs are stored. These files are then processed and the output is saved in a standardized JSON format in a separate folder containing labeled logs. The tool leverages process relations by matching the process identifiers (PIDs) of malicious events with PIDs found in subsequent events, utilizing recursion when processing the log files. A unique identifier is given to a process once it is created, thus only one process can be directly tied to a PID. The tool was developed through three main iterations, the final and complete version is centered around a report that CALDERA can generate after an emulation. Data from this report is extracted and leveraged to find the initial process of each operation, and what ATT&CK tactic and technique the process is related to. Any process or operation seen in relation to the initial process are considered part of the ongoing operation and labeled accordingly.

#### First Iteration

In the first iteration the labeling tool started off with defining two lists within the program, one containing the Winlogbeat file to be processed, and one empty list which is gradually filled with labeled logs. This was done so that the tool could iterate through the Winlogbeat file and move events found to malicious or benign to the processed list once labeled. The tool would search for an occurrence of the CALDERA agent by matching the process name in the events with the name of the agent. This signals the start of the malicious attack as each attack starts with a command being executed by the agent. The naming of the agent is specified in CALDERA and was coded into the tool. Once the executable has been detected, the process ID (PID) is extracted from the list. This event is then moved from the unprocessed list onto the processed list and labeled as malicious. When the first known malicious PID was found, the tool would start searching for any occurrences of this PID in the following events. If the PID was observed in any relation to an event, it would be considered malicious and any new PID's found in this event would be extracted and considered malicious. Once the tool had found all events in relation to the initial PID it would start a new iteration searching for the next known malicious PID. The list containing unprocessed events shrinks as malicious events are removed

from it, causing each subsequent occurrence of recursion to have less events to search through. The remaining events are labeled as benign and moved onto the processed list once all malicious events have been analysed.

This worked to an extent, the tool was able to find the malicious events but it was not able to apply labels based on what technique or tactic they were related to. Wrongly labeling of benign events also occurred as the tool did not take into consideration what timestamp the event had. An PID used for a malicious operation thirty minutes earlier is not necessary conducting malicious operations at a later time. Additionally, changing the name of the CALDERA agent would require the tool to be modified in order to search for the new name. The tool was completely reworked due to these issues.

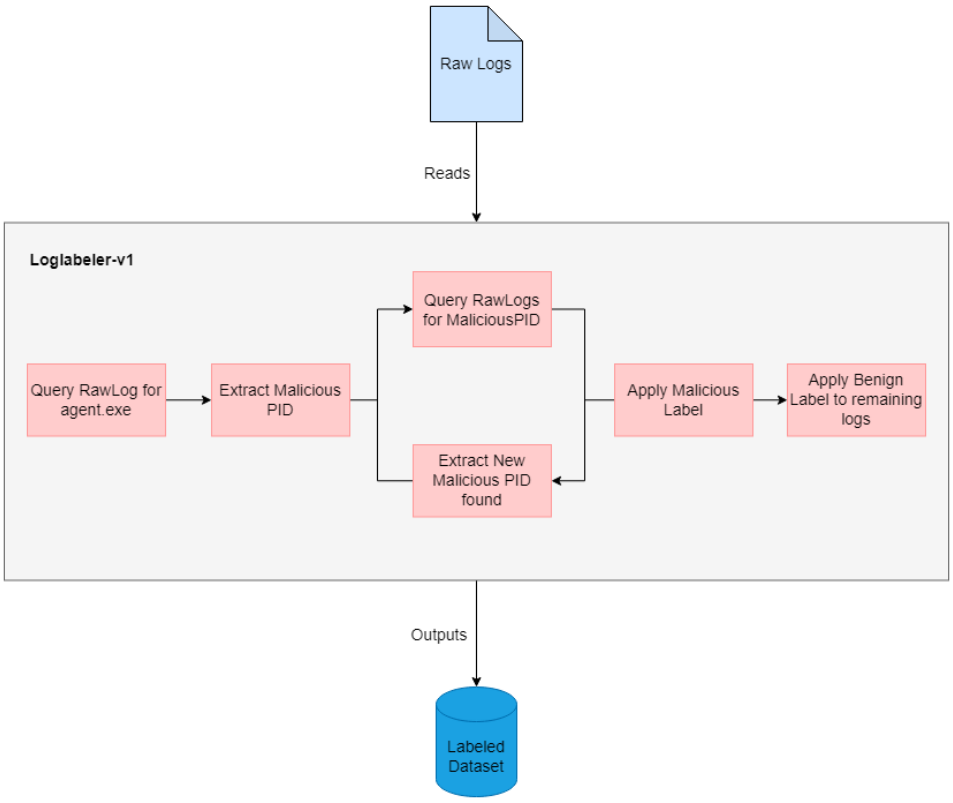



Figure 4.2: Operation Flow Label Tool version 1

### Second Iteration

During this thesis, MITRE released version 4.0.0 of CALDERA, this version had the capability to produce an report with information regarding each operation conducted during the emulation.

The second iteration of the tool leveraged this report to extract the agent PID, the first child process of the agent for each operation, and the ATT&CK tactic and technique related to the specific operation. With this information available, the tool was able to apply fine-grain labels including what ATT&CK tactic and technique each event was related to.

The logic and idea behind the first version was kept as the tool still utilized iteration and the PID's to find the malicious events. The tool reads the CALDERA rapport, extracting information from the AgentMetaData and AttackMetaData fields. AgentMetaData contains information regarding the CALDERA agent, such as the PID of the process spawned by the agent and the PID of the agent itself. The MetaData field contains information regarding the tactic and technique related to the specific PID, which is used to apply the labels. Each operation in the CALDERA report is read as an individual object. Once the information from the object has been extracted the tool start to process the Winlogbeat file, reading each line from the file and matching the PID observed with the known malicious PIDs. If the event is found to be malicious and the PID related to the event is not currently in the list of malicious PIDs, the tool will add this PID to the list, which is related back to the specific operation. This event is then labeled as malicious and the tactic and technique related to the PID is applied. In the next line of Winlogbeat the tool will now search for both of the PIDs within the list, repeating the process for any malicious events. Once the tool reaches the end of the Winlogbeat file, it proceeds to read the next object from the CALDERA report, replace the list of malicious PIDs with the new PID and relate the new list to a specific tactic and technique. This process is repeated until all objects in the report has been searched for in the Winlogbeat file.


```
"command": "MGNiNzEwX1QxMDU1LmV4ZQ==",
"delegated_timestamp": "2022-04-19T12:13:14Z",
"collected_timestamp": "2022-04-19T12:13:58Z",
"finished_timestamp": "2022-04-19T12:13:59Z",
"pid": 4576,
"agent_metadata": {
  "username": "WIN10\\vagrant",
  "location": "C:\\Users\\Public\\sandcat.go-windows.exe",
  "pid": 4656,
  "ppid": 508,
  "privilege": "Elevated",
  "host": "win10",
  "contact": "HTTP",
  "created": "2022-04-19T11:48:26Z"
},
"ability_metadata": {
  "ability_name": "Process Injection via C#",
  "ability_description": "Process Injection using C#\nreference: https://github.com/pwndizzle/c-sharp-memory-injection\n
      Excercises Five Techniques\n1. Process injection\n2. ApcInjectionAnyProcess\n3. ApcInjectionNewProcess\n4. IatInjection\n5
      Upon successful execution, cmd.exe will execute T1055.exe, which exercises 5 techniques. Output will be via stdout.\n"
},
"attack_metadata": {
  "tactic": "multiple",
  "technique_name": "Process Injection: Asynchronous Procedure Call",
  "technique_id": "T1055.004"
},
"agent_reported_time": "2022-04-19T12:13:57Z"
```


Figure 4.3: One operation in the CALDERA report

CALDERA generates the data presented in **figure 4.3** after an attack emulation has been completed. However, the CALDERA log file only contains the PID of the agent and the initially spawned process. By leveraging process relations and recursion the tool is able to find all processes linked to a malicious operation through process relations. The argument for this approach is that there are no benign actions performed by the CALDERA agent as the agent is the initial access point of the attack.

Therefore it is assumed that any interactions on the environment performed by the agent (e.g., process injection, privilege escalation, file creation/modification etc), are malicious. If the PID of the agent is observed in any relation to another process, be it a benign or malicious process, that specific process is considered part of a malicious action. An example of this could be the agent performing process injection by executing code in the address space of a separate process. This can be detected through Sysmon [48] with the event ID 8 (Remote Thread Creation Detected).

**EventData1 – Event ID 1 – Process Create**

| | |
|---|---|
| **RuleName** | technique_id=T1059.003,technique_name=Windows Command Shell |
| **UtcTime** | 2022-04-19 12:13:57.921 |
| **ProcessId** | 4576 |
| **Image** | C:\Windows\System32\cmd.exe |
| **OriginalFileName** | Cmd.Exe |
| **CommandLine** | cmd.exe /C 0cb710_T1055.exe |
| **CurrentDirectory** | C:\Users\vagrant\ |
| **User** | WIN10\vagrant |
| **ParentProcessId** | 4656 |
| **ParentImage** | C:\Users\Public\sandcat.go-windows.exe |
| **ParentCommandLine** | "C:\Users\Public\sandcat.go-windows.exe" -server http://192.168.56.104:8888 -group red |
| **ParentUser** | WIN10\vagrant |

**EventData2 – Event ID 1 – Process Create**

| | |
|---|---|
| **RuleName** | technique_id=T1059.003,technique_name=Windows Command Shell |
| **UtcTime** | 2022-04-19 12:13:57.935 |
| **ProcessGuid** | {04787c88-a785-625e-8f02-000000001400} |
| **ProcessId** | 9076 |
| **Image** | C:\Users\vagrant\0cb710_T1055.exe |
| **OriginalFileName** | T1055.exe |
| **CommandLine** | 0cb710_T1055.exe |
| **CurrentDirectory** | C:\Users\vagrant\ |
| **User** | WIN10\vagrant |
| **ParentProcessId** | 4576 |
| **ParentImage** | C:\Windows\System32\cmd.exe |
| **ParentCommandLine** | cmd.exe /C 0cb710_T1055.exe |
| **ParentUser** | WIN10\vagrant |

Figure 4.4: Truncated Sysmon logs related to a process injection executed by CALDERA

The labeling tool starts off with extracting the PID, technique_name, and techinque_id found in **Figure 4.3**. This is placed onto a temporary "maliciousPID" list within the labeling tool, which is later used to match the PIDs in the list with PIDs observed in events within the Winlogbeat file.

In **Figure 4.4** we observe the resulting Sysmon event from executing the MITRE ATT&CK technique T1055.004 - Process Injection: Asynchronous Procedure Call [37], CALDERA utilizes the payload 0cb710_T1055.exe in order to execute this technique. The information found in this event is also found in the Winlogbeat file which is being processed. The labeling tool compares the ProcessId (4576) and ParentProcessId (4656) with the PID in the "maliciousPID" list, and detects that the ParentProcessId (4656) is in the "maliciousPID" list. This event is then deemed malicious, and labels are applied as shown in **Figure 4.5**.


Figure 4.5: Truncated labeled log related EventData1 in **Figure 4.4**

The PID 4576 has now been seen in a relation to a malicious PID, as such this PID is considered malicious and extracted onto the "maliciousPID" list. This PID is then compared with all PID's found in subsequent events in order to find all malicious operations conducted and the next malicious PID. In EventData2 **Figure 4.4**, we can observe that PID 4576 spawns a new process, ProcessId 9076, when executing the process injection payload. Since PID 4576 is now in the "maliciousPID" list, this event is found to be malicious and is labeled accordingly. All PIDs found in relation to the first malicious PID will receive the same labels as the first PID.

**EventData3 – Event 8 – CreateRemoteThread**

| | |
|---|---|
| **RuleName** | - |
| **UtcTime** | 2022-04-19 12:13:58.169 |
| **SourceProcessGuid** | {04787c88-a785-625e-8f02-000000001400} |
| **SourceProcessId** | 9076 |
| **SourceImage** | C:\Users\vagrant\0cb710_T1055.exe |
| **TargetProcessGuid** | {04787c88-a786-625e-9002-000000001400} |
| **TargetProcessId** | 5880 |
| **TargetImage** | C:\Windows\System32\notepad.exe |
| **NewThreadId** | 2652 |
| **StartAddress** | 0x00007FFB2659EB60 |
| **StartModule** | - |
| **StartFunction** | - |
| **SourceUser** | WIN10\vagrant |
| **TargetUser** | WIN10\vagrant |

**EventData4 – Event 10 – Process Accessed**

| | |
|---|---|
| **RuleName** | technique_id=T1003,technique_name=Credential Dumping |
| **UtcTime** | 2022-04-19 12:13:58.712 |
| **SourceProcessId** | 4644 |
| **SourceThreadId** | 6076 |
| **SourceImage** | C:\Windows\system32\WerFault.exe |
| **TargetProcessId** | 5880 |
| **TargetImage** | C:\Windows\System32\notepad.exe |
| **CallTrace:** | |

C:\Windows\SYSTEM32\ntdll.dll+9c584|C:\Windows\SYSTEM32\ntdll.dll+74c27|C:\Windows\system32\dbghelp.dll+10ea0|C:\Windows\system32\dbghelp.dll+16931|C:\Windows\system32\dbghelp.dll+1134e|C:\Windows\system32\faultrep.dll+1ec27|C:\Windows\system32\faultrep.dll+1f00c|C:\Windows\system32\faultrep.dll+e349|C:\Windows\system32\faultrep.dll+f9af|C:\Windows\system32\faultrep.dll+d383|C:\Windows\system32\faultrep.dll+d2da|C:\Windows\system32\faultrep.dll+ba78|C:\Windows\system32\WerFault.exe+2c7ea|C:\Windows\system32\WerFault.exe+4345|C:\Windows\system32\WerFault.exe+480dd|C:\Windows\System32\KERNEL32.DLL+17bd4|C:\Windows\SYSTEM32\ntdll.dll+6ced1

| | |
|---|---|
| **SourceUser** | WIN10\vagrant |
| **TargetUser** | WIN10\vagrant |

Figure 4.6: Truncated Sysmon logs related to a process injection executed by CALDERA

Further, it can be observed in **Figure 4.6** that SourceProcessId 9076, which was found to be malicious by relation to PID 4576 in **Figure 4.4**, interacts with the TargetProcessId 5880 (notepad.exe). Since notepad.exe is the target of a malicious process it is assumed that the following actions from notepad.exe will be malicious in nature. In this scenario, all of the event from **Figure 4.4** and **4.6** will receive a "isMalicious: True" label, in addition to a "verdict: Malicious Process Injection: Asynchronous Procedure Call - T1055.004" label stating what tactic and technique the event is related to. This is repeated for all processes found in relation to any of these malicious PIDs, gradually developing a process tree of all processes spawned or affected in any way by the operation. **Figure 4.7** illustrates the process tree developed from the process relations in **Figure 4.4** and **4.6**.

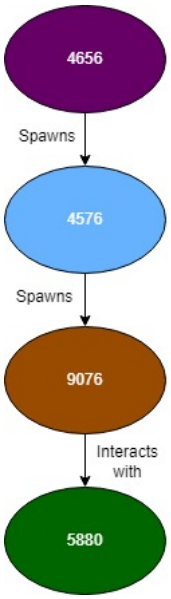



Figure 4.7: Process Tree developed during the execution of T1055.044

Once the labeling tool has found and labeled all operations related to **Figure 4.3**, the tool moves on to the next operation found in the CALDERA log, extracting the new information and repeating the labeling process.

### Final Iteration

With the second iteration of the labeling tool it was now able to find all processes with an PID relation and apply labels with a high granularity. There was still an issue with some benign events being wrongly labeled as a result of the process executing operations at a later time. This issue was addressed by implementing a function which checks if the malicious event in Winlogbeat is within the time frame of the malicious operation. Each operation in the CALDERA log has a agent delegated and finished time stamp. This was leveraged in order to check if the detected malicious event is within these two timestamps. In order for this to function as intended the internal clock of the CALDERA server and the targeted host has to be synchronized. However, the final event in each malicious operation had the same timestamp as the agent finished time stamp, resulting in these events not being correctly labeled. This was addressed by developing a method for truncating the millisecond part of the timestamps.

Argument for this time based approach is that each operation occurs within the delegated and finished time stamp. Finished time stamp is applied once the agent reports back the outcome of the operation. If an event is found to be malicious, but is outside of the expected operation time, the label "uncertain +" technique id and tactic is applied in addition to the "isMalicious: False" verdict. The false verdict is applied since there is a higher probability that the event is benign than malicious. However, the additional "uncertain" label indicates that the event may need to be manually reviewed in order to determine the true nature of it.

The tool was also adjusted in order to correctly label Command and Control traffic as Command and Control. These events received the label from the PID related to the traffic, scenarios occurred where Command and Control traffic was labeled as eg., process injection, this was addressed by checking if the event ID from Sysmon was three (3). Event ID 3 in Sysmon is Network Connection, if an event is malicious with Event ID 3 and within the expect time frame it is considered as Command and Control traffic. Manual inspection after this change showed that all Command and Control labels were correctly applied during the emulation.

**Final Iteration Code Review**

The program is developed in C#, it starts by reading the Winlogbeat and CALDERA json files and storing the data in the Winlogbeats and maliciousOperations variables. A new instance of the LogLabler class is created and the variables from Winlogbeats and maliciousOperations are passed in as arguments. Finally the method for finding and marking malicious events is called on the logLabeler object and the output is saved to a file through the SaveJsonToFile() method.

```
1
2 var Winlogbeats = GetWinlogbeats(@"C:\Users...\Winlogbeat.json");
3 var maliciousOperations = GetMaliciousOperations(@"C:\Users...\
      caldera-log.json");
4 var logLabeler = new LogLabeler(Winlogbeats, maliciousOperations);
5 logLabeler.FindAndMarkAllDescendantMaliciousOperations()
6     .SaveJsonToFile(@"C:\User...\output.json")
```

Listing 4.1: Code snippet from Program.cs

Each line of the Winlogbeat file is read and deserialized into a Winlogbeat object, and each object is added to a list for further processing. A catch clause is implemented to catch invalid Winlogbeat input which would have prevented the program from running. This is seen in **listing 4.2**

```
1
2  List<Winlogbeat> GetWinlogbeats(string path)
3  {
4      IEnumerable<string> WinlogbeatReadLines = File.ReadLines(path)
           ;
5      var Winlogbeats = new List<Winlogbeat>();
6      var i = 0;
7      foreach (var Winlogbeat in WinlogbeatReadLines)
8      {
9          try
10         {
11             i += 1;
12             var deserializedWinlogbeat = JsonConvert.
                   DeserializeObject<Winlogbeat>(Winlogbeat);
13             if (deserializedWinlogbeat != null) Winlogbeats.Add(
                   deserializedWinlogbeat);
14         }
15         catch (Exception e) {
16             Console.WriteLine(e); }
17     }
18     return Winlogbeats;
19 }
```

Listing 4.2: Code snippet from Program.cs

Malicious operations are placed onto a list containing the process ID, tactic and technique related to the malicious operation.

```
1
2  IEnumerable<MaliciousOperation>? GetMaliciousOperations(string
       path)
3  {
4      IEnumerable<MaliciousOperation>? maliciousOperations =
5          JsonConvert.DeserializeObject<List<MaliciousOperation>>(
               File.ReadAllText(path));
6      return maliciousOperations;
7  }
```

Listing 4.3: Code snippet from Winlogbeat.cs

The constructor of LogLabler takes inn the list of Winlogbeat and maliciousOperations as arguments, as shown in **listing 4.5**. A new list called MaliciousPid is created based on the list of maliciousOperations. Information regarding the tactic, technique and timestamps of the operations are added to this list. In line 6, the PID of the process spawned by the agent is added, line 9 adds the PID of the agent while line 13 adds the parent PID of the agent. This was done so that all PIDs initially related to a malicious operation is added to the list of malicious PIDs. This list is later used in order to search for new malicious PIDs and relate these back to specific tactics and techniques.

```
1
2  public LogLabeler(List<Winlogbeat> Winlogbeats, IEnumerable<
       MaliciousOperation> maliciousOperations)
3          {
4              this.Winlogbeats = Winlogbeats;
5              var operations = maliciousOperations.Select(x => new
                   MaliciousPid(x.pid, x.attack_metadata.technique_id
                   ,
6                   x.attack_metadata.technique_name, x.
                       delegated_timestamp, x.finished_timestamp, x.
                       agent_metadata)).ToList();
7
8              operations.AddRange(maliciousOperations.Select(x =>
                   new MaliciousPid(x.agent_metadata.pid,
9                   x.attack_metadata.technique_id, x.attack_metadata.
                       technique_name, x.delegated_timestamp, x.
                       finished_timestamp,
10                  x.agent_metadata)).ToList());
11
12             operations.AddRange(maliciousOperations.Select(x =>
                   new MaliciousPid(x.agent_metadata.ppid,
13                  x.attack_metadata.technique_id, x.attack_metadata.
                       technique_name, x.delegated_timestamp, x.
                       finished_timestamp,
14                  x.agent_metadata)).ToList());
15
16             this.maliciousPids = operations.Distinct().ToList();
17         }
```

Listing 4.4: Code snippet from LogLabler.cs

The FindAndMarkAllDescendantMaliciousOperations method is responsible for identifying and labeling all malicious events and contains the recursion function of the tool. For each malicious PID, find all Winlogbeat events that match the malicious PID, and mark them as malicious. Then, find all PIDs in the event and add this to a new list. Once all events related to the initial list have been found and marked, the initial list is replaced with the new list and the process is repeated. If malicious events are found outside of the expected timestamp, a "uncertain" verdict is applied and the PIDs in the event are not extracted.

```
1
2  public LogLabeler FindAndMarkAllDescendantMaliciousOperations()
3 {
4     var newMaliciousPids = new List<MaliciousPid>();
5     foreach (var pid in this.maliciousPids)
6     {
7         if (pid.pid == null) continue;
8         foreach (var Winlogbeat in Winlogbeats)
9         {
10            if (!Winlogbeat.MatchesMaliciousPid(pid)) continue;
11            if (Winlogbeat.IsWithinMaliciousOperationTimePeriod(
                  pid))
12            {
13                Winlogbeat.isMalicious = true;
14                Winlogbeat.verdict = "Malicious " + pid.
                      technique_name + " - " + pid.technique_id;
15            }
16            else
17            {
18                if (Winlogbeat.winlog.event_id == "3")
19                {
20                    Winlogbeat.isMalicious = true;
21                    Winlogbeat.verdict = "Malicious, command and
                          control traffic";
22                }
23                else if (Winlogbeat.Timestamp != null &&
                      Winlogbeat.Timestamp.Value.TrimMilliseconds()
                      >=
24                      maliciousPids.Select(x => x.agentMetadata.
                          created).Min().TrimMilliseconds())
25                {
26                    Winlogbeat.verdict = "Uncertain "  + pid.
                          technique_name + " - " + pid.technique_id;
27                }
28            }
29            newMaliciousPids.Add(new MaliciousPid(Winlogbeat?.
                  process?.pid, pid.technique_id, pid.technique_name
                  ,
30                  pid.delegated_timestamp, pid.finished_timestamp,
                      pid.agentMetadata));
31            newMaliciousPids.Add(new MaliciousPid(Winlogbeat?.
                  process?.parent?.pid, pid.technique_id, pid.
                  technique_name,
32                  pid.delegated_timestamp, pid.finished_timestamp,
                      pid.agentMetadata));
33        }
34    }
35    this.maliciousPids = newMaliciousPids.Distinct().ToList();
36    if (maliciousPids.Count > 0)
          FindAndMarkAllDescendantMaliciousOperations();
37    return this;
38 }
```

Listing 4.5: Code snippet from LogLabler.cs

# Chapter 5

# Results

## 5.1 Emulation of APT29

CALDERA was successful in emulating most of the attack steps present in the emulation plan. However, both the environment and the emulation plan had to be adjusted in order to successfully execute a portion of the emulation plan. The infrastructure was configured as instructed in the emulation library [4], but further adjustments were necessary in order to to produce an successful emulation. Some steps in the emulation requires knowledge learned from previous steps e.g., host/network discovery, admin user names etc, while others are dependent on successfully execution of previous steps e.g., establishing persistence before reboot. The majority of issues during execution was related to the dependencies of previous steps. The most critical issue occurred during step 4 shown in 3.5.3, where CALDERA executes a PowerShell command to download Sysinternals Suit. Once downloaded, the content is extracted and merged with an modified Sysnternals Suite containing various executables used in later phases of the emulation. CALDERA was able to initiate the download however, it did not allow the download to complete before attempting to merge these files. Due to this, CALDERA was not able to find multiple of the executables required in subsequent steps e.g., Step 4 - Defense Evasion and Further Discovery (3.5.3), Step 5 - Persistence(3.5.3), and Step 6 - Collection and Exfiltration(3.5.3), to name a few. This resulted in a cascading effect of failing operations. This was addressed by placing the Sysinternal Suits file at the designated location, CALDERA initiates the download and attempts to move the file to the designated location. If the file is already present at the designated location, CALDERA will instead proceed with merging the malicious payload with the file.

CALDERA was also unable to successfully trigger the persistence mechanisms in place. These were triggered by rebooting the victim machine. Once the victim machine was rebooted, the CALDERA agent would fail to beacon home to the server. The server would assume that the agent is dead and end the emulation. This problem occurred even when manually placing the persistence mechanisms and attempting to trigger these. As of such, the command to reboot the victim machine was

completely removed from the emulation plan. Once these modifications were made, the emulation was able to fully execute with an acceptable ratio of successful operations, according to the CALDERA report.

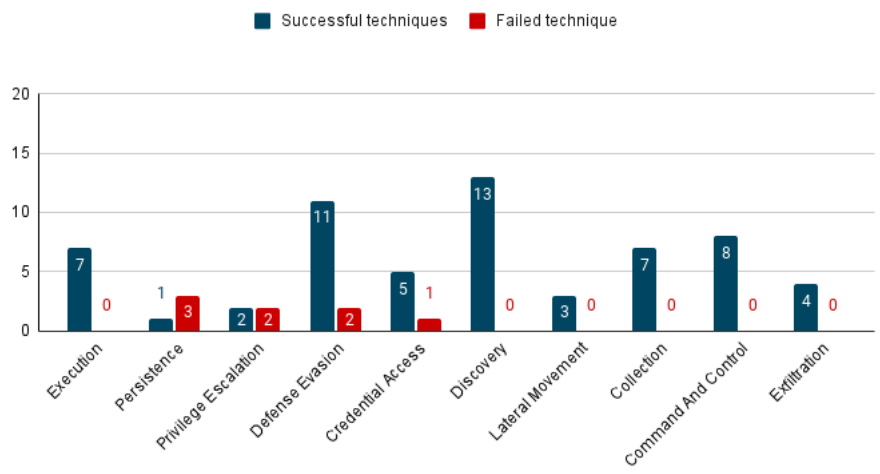



Figure 5.1: Successful and failed techniques from the CALDERA report

**Figure 5.1** presents the amount of successful and failed techniques within each tactic found in the emulation plan. The figure is only based on the report provided by CALDERA. If the agent is able to execute a command and transmit the resulting data back to the server, it is considered successful. This is also true in the scenario of an operation being partially successful. Operation 35 in step 5 (3.5.3) is an example of where the operation is only partially successful but is reported as successful. The operation executes the following command in an attempt to establish persistence:

```
Set-Location -path "C:\Program Files\SysinternalsSuite";
if (Test-Path -path "readme.ps1") {
  . .\readme.ps1;
  Invoke-Persistence -PersistStep 2;
  write-host "[+] Persistence 2 invoked.";

} else {
  write-host "[!] readme.ps1 not found.";
  return 1;
}
```

Readme.ps1 is found and executed, the script calls for multiple executables within the modified Sysinternals Suits folder. Two of these are not found in the folder and the execution returns an ItemNotFoundExeption. Since the agent was able to execute the initial script and some data was returned, it was considered successful in the report while the operation was in fact not successful. This report also does not take into consideration which

operations in the emulation plan were dropped or skipped. An accurate number of failed and successful techniques and tactics can be found by comparing the emulation plan with the executed operations and manually analyzing the results of each operation.

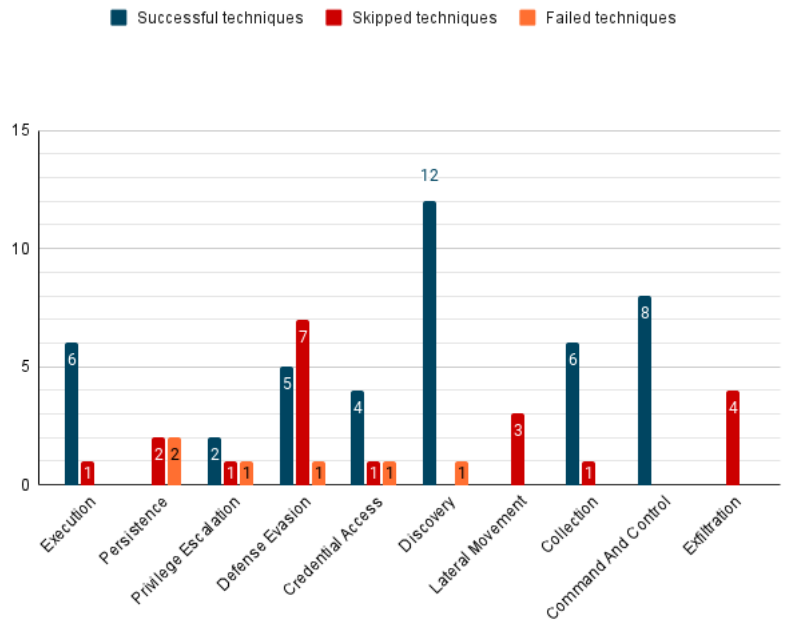



Figure 5.2: Successful, Failed, and Skipped tactics

**Figure 5.2** gives an accurate description of successful, failed, and skipped techniques. The majority of the techniques not executed are related to CALDERA skipping these operations. CALDERA may decide to skip an operation if conditional data is missing from CALDERA's knowledge database e.g., failed to update or receive expected output from previous steps, or as a result of facts learned during previous executions. The latter option is ruled out as the simulation was executed without using prior facts. All persistence techniques present in the emulation either failed or were completely skipped. Persistent establishment in operation 35, step 5, was reported as successful by CALDERA while in fact failing. This could possibly have impacted CALDERA's decision to skip the other persistence establishment operations. Lateral movement techniques also performed poorly during the simulation, all operations were skipped despite successfully detecting the other clients and domain controllers through network discovery techniques. Exfiltration techniques were also entirely skipped without an clear indication of why this failed. Collection of files and data prior the exfiltration was completed, staging directories got created and prepared for exfiltration, but the exfiltration of these files never occurred.

## 5.2 Final Dataset

Since the dataset is converted into a standardized format it may be uploaded to a SIEM solution such as Splunk for further analysis. Splunk was used to manually review the labels produced by the labeling tool. It is expected that the labeled logs would share similar traits with **figure 5.2** regarding the distribution of tactics. The created APT29 datasets consists of 2,276 events, 868 malicious and 1 406 benign.

### Benign and Malicious data distribution

Assuming that process ID's observed outside the related operation's time frame are considered malicious, 49.3% of the datasets will be related to malicious activities. The label " uncertain + technique " is applied to logs where the process ID was found to be malicious, but the timestamp did not match the time frame of CALDERA during that operation. As shown in **figure 5.3** there is a clear bias towards malicious logs, realistically there would only be a fraction of malicious event in comparison to benign events. This bias occurred as the framework utilized for generating benign traffic performed poorly and the data collection ended shortly after the adversary emulation. As a result, benign activities should in this case be performed manually over a longer time period, this would however hinder the reproducibility of the dataset as the manual activities would have to be replicated in a timely manner. Most of the traffic generated through the GHOSTS framework was related to network activities, while the host based activities did not execute properly. There was also issues regarding the loop function of the framework, where it only successfully looped through the browsing activities while Powershell commands and other operations which would generated more host based logs only executed once. Collecting the data at a later time would result in benign activities of similar nature e.g., network browsing.

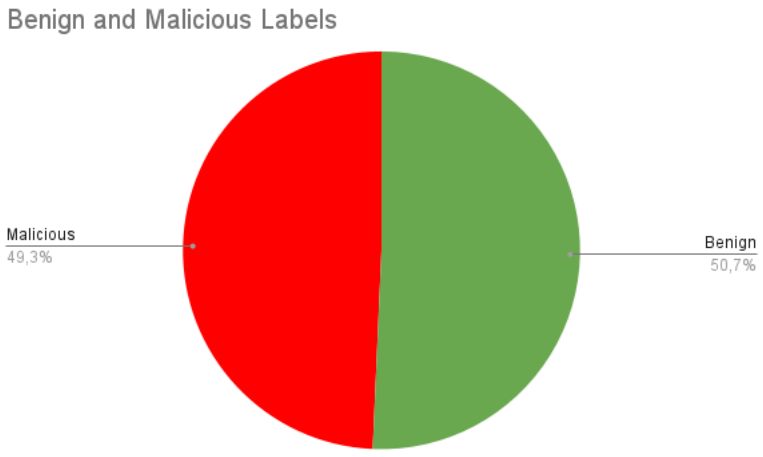



Figure 5.3: Distribution of benign and malicious labels

Taking into consideration that process ID's observed outside the operation time are not determined to be malicious, the label distribution changes to 38,2% malicious, which is still considered to be bias towards malicious logs. This is a more accurate description of the distribution as manually inspecting the log lines labeled as "uncertain" revealed that the majority of these are in fact benign activities. Out of the 278 events that received this label, roughly 30 (10.8%) was manually deemed malicious. Most of these were related back to the execution of scripts, which occurred shortly after the time frame of the operation. **Figure 5.4** is strictly based on the labels applied from the labeling tool.

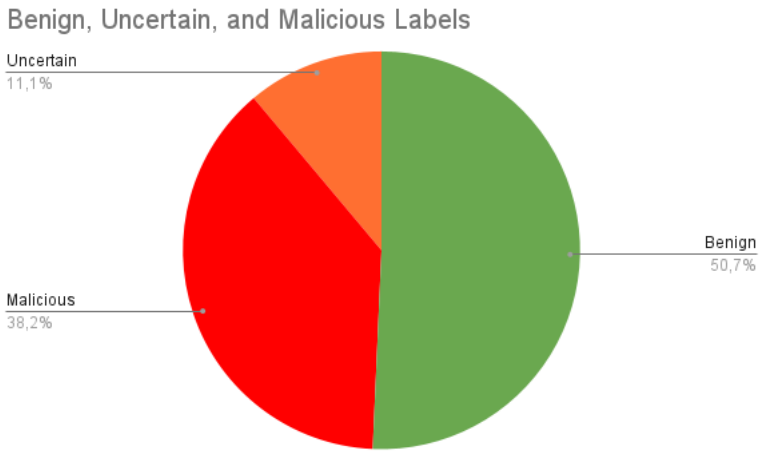



Figure 5.4: Distribution of benign, uncertain, and malicious labels

### Malicious Logs

**Figure 5.5** shows the distribution of techniques based on all malicious labeled log lines. 32,719% (284/868 malicious events) of malicious labels applied were related to discovery techniques. This is to be expected as the emulation successfully executed twelve discovery techniques and the emulation plan contained a clear majority of discovery related techniques. While 21,198% (184/868 malicious events) of labels applied were related to execution techniques, the emulation plan contained one unique execution tactics. However, this technique was leveraged in multiple steps, execution of this techniques also resulted in a significantly larger amount of events in comparison to other techniques. T1059.001 - Command and Script Interpreter usually involves the execution of scripts or payloads, which resulted in a significant longer chain of related process ID's compared to some discovery techniques which only involved executing a single PowerShell command. T1059.001 was also used in order to prepare the payload for other tactics.

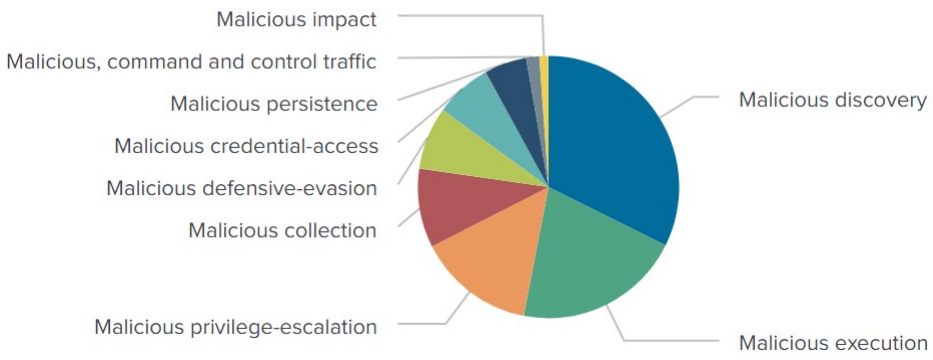



Figure 5.5: Distribution of tactics across malicious events

Lateral Movement and Exfiltration techniques are absent in the dataset as CALDERA was not able to successfully execute these operations. However, events related to Persistence (4,954%, 43/868) are found. As previously discussed, this tactic was executed and reported as successful even if the outcome was in fact a failure. The events found related to this tactic are artifacts from the attempt to establish persistence, which can still be useful. Privilege-Escalation, Collection, Defensive-Evasion, and Credential-Access constituted respectively 14,17%, 9,216%, 8,41%, and 6,682% of the malicious events. All of these tactics did not have an process ID's observed outside of the agents operational timestamp, therefore no events related to these tactics received the "uncertain" label. Privilege-Escalation operations generated a fair amount of events (14,17%, 123/868) despite only successfully executing two operations. Privilege-Escalation occurred through a process injection script, which can be observed loading various DLL's, interacting with User Account Control (UAC), and certificates. Command and control traffic label was added in addition to the techniques and tactics extracted from the CALDERA report, these events were detected through Sysmon event ID 3 - Network Connect and applied once a malicious PID contacted the CALDERA server.

**Techniques observed in the dataset**

A total of 41 variations of techniques labeled is observed when analyzing the dataset in Splunk. However, there was duplicates as some log lines are labeled as "uncertain" followed by the technique and technique ID. A total of nine techniques had additional log lines with this label, a total of 32 unique techniques were observed during the APT29 emulation. Techniques related to the Discovery tactic is observed as the third and fourth most commonly labled techniques. However, the three rarest techniques are also related to the Discovery tactic. These being Network Connections Discovery (T1049), Query Registry (T1012), and Software Discovery (T1518).

| 1 | Verdict | Technique | Count | Percent |
|---|---|---|---|---|
| 2 | Command and Scripting Interpreter: PowerShell | T1059.001 | 184 | 21.198157 |
| 3 | Process Injection | T1055 | 98 | 11.290323 |
| 4 | Process Discovery | T1057 | 53 | 6.105991 |
| 5 | System Information Discovery | T1082 | 52 | 5.990783 |
| 6 | Credential Dumping | T1003 | 45 | 5.184332 |
| 7 | Permission Groups Discovery | T1069 | 40 | 4.608295 |
| 8 | Remote System Discovery | T1018 | 34 | 3.917051 |
| 9 | System Owner/User Discovery | T1033 | 28 | 3.225806 |
| 10 | Screen Capture | T1113 | 27 | 3.110599 |
| 11 | Access Token Manipulation: Token Impersonation/Theft | T1134.001 | 25 | 2.880184 |
| 12 | Indicator Removal on Host: File Deletion | T1070.004 | 21 | 2.419355 |
| 13 | Boot or Logon Autostart Execution: Shortcut Modificatior | T1547.009 | 21 | 2.419355 |
| 14 | Modify Registry | T1112 | 18 | 2.073733 |
| 15 | Automated Collection | T1119 | 18 | 2.073733 |
| 16 | Input Capture: Keylogging | T1056.001 | 16 | 1.843318 |
| 17 | Account Discovery | T1087 | 16 | 1.843318 |
| 18 | Access Token Manipulation: Create Process with Token | T1134.002 | 16 | 1.843318 |
| 19 | System Network Configuration Discovery | T1016 | 15 | 1.728111 |
| 20 | System Service Discovery | T1007 | 14 | 1.612903 |
| 21 | Command and control traffic | | 13 | 1.497696 |
| 22 | Unsecured Credentials: Private Keys | T1552.004 | 13 | 1.497696 |
| 23 | Event Triggered Execution: Windows Management Instru | T1546.003 | 12 | 1.382488 |
| 24 | Software Discovery: Security Software Discovery | T1518.001 | 11 | 1.267281 |
| 25 | Software Discovery | T1518 | 10 | 1.152074 |
| 26 | Signed Binary Proxy Execution: Rundll32 | T1218.011 | 10 | 1.152074 |
| 27 | Email Collection: Local Email Collection | T1114.001 | 10 | 1.152074 |
| 28 | Disk Wipe: Disk Content Wipe | T1561.001 | 10 | 1.152074 |
| 29 | Masquerading: Match Legitimate Name or Location | T1036.005 | 9 | 1.036866 |
| 30 | Indicator Removal on Host: Timestomp | T1070.006 | 9 | 1.036866 |
| 31 | Clipboard Data | T1115 | 9 | 1.036866 |
| 32 | Query Registry | T1012 | 8 | 0.921659 |
| 33 | System Network Connections Discovery | T1049 | 3 | 0.345622 |

Figure 5.6: Technique label distribution in dataset

Network Connections Discovery was conducted by executing the netstat command in the command shell (cmd.exe), which manifested itself in the logs as a "new process created" without any further traces. This is expected as the command only returns network connections, routing tables, and various network statistics without performing any further operations. However, if the operation had saved the result of the netstat command in a file or leveraged this information in any other way, it would be expected to see further traces of the operation in the logs. Further Network Discovery was conducted utilizing T1016, which generated similar traces utilizing commands such as ipconfig, arp and netsh. Distribution of tactic and technique labels is in pair with the expected distribution based on what operations were successful.

## 5.3 Labeling Tool

The developed labeling tool is able to successfully apply labels with a high level of granularity. Labels are applied in two distinct ways:

- Adding a data field to each line named “isMalicious:”, which can either be true or false.
- Adding an additional data field for further specification of malicious labels named “verdict:”. This field is enriched with the MITRE tactic and technique ID of the malicious activities related with the event e.g., “Verdict: Malicious System Information Discovery - T1082”.

Labels applied are directly linked to tactics and techniques found within MITRE ATT&CK, which can be utilized to develop a labeled dataset containing all the steps within a kill chain. This could prove valuable when analyzing and detecting a kill chain, individual phases of the kill chain can be simulated, labeled and later merged with the other phases in order to develop a complete dataset. The labeling tool successfully found all process identifiers (PID’s) related in any manner to the malicious PID’s extracted from the CALDERA logs. This is true as long as there is a relation between the PID’s, such as interactions, child/parent/target relations, or ProcessAccess to name a few. In the scenario where a PID does not have a relation to a malicious PID or the PID was not captured, it will be labeled as benign. In order to verify the labeling process multiple other scenarios were executed with minimal background and benign activities. The time windows of log collection was also adjusted to only collect logs during the execution of these scenarios. Verifying the labels on a smaller dataset did prove to be a more feasible task. Sysmon was also closely monitored during these trail emulations. Within a time span of roughly thirty minutes, Sysmon had generated 172 events, while the labeling tool detected 159 malicious events. No suspicious events were labeled as benign, the benign events found were related to the host sending DNS queries to the domain controller, which occurred even when the host was idle ahead of time.

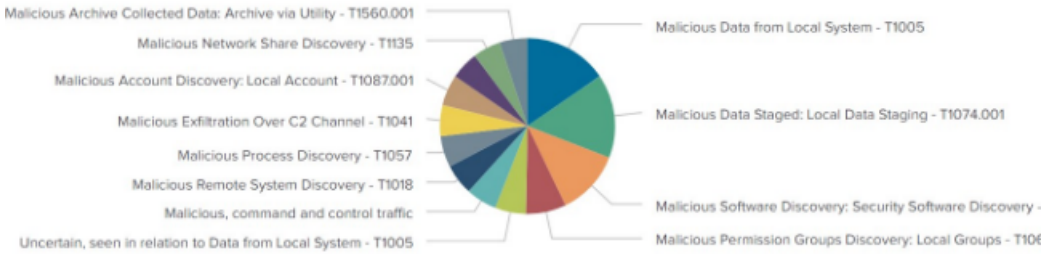



Figure 5.7: Distribution of tactics and techniques from trail emulations

In **figure 5.7** we can observe labels related to Exfiltration tactics, as previously stated these were skipped in the APT29 emulation plan, but CALDERA is clearly capable of executing this tactic as well. Evaluating the tool against other scenarios also verifies that the tool is not directly tied to the APT29 emulation plan.

### Drawbacks

The labeling tool is developed around CALDERA and will not function properly without the CALDERA report. This is due to the labeling tool using data from the CALDERA report in order to find the initial malicious PID's and relate these back to a specific tactic and technique. In order for the tool to function without the CALDERA report it would have to be reverted back to the previous version of it which searched for the process name of the CALDERA agent in order to extract the malicious PIDs. A time based approach for applying higher label granularity could substitute the tactics and techniques extracted from the report, by knowing exactly what time frames each technique was executed during and labeling malicious events accordingly.

The tool was not able to perfectly label the logs as it utilizes process relation, and some events could not be traced back to a malicious PID or had a blank parent PID. In these scenarios the tool will wrongly label these events as benign. However, this is a rare occurrence and only a fraction of events were found to be potentially mislabeled benign after the last development iteration of the tool. Benign events mislabeled as "uncertain" since the process performed benign operations at a later time is present in the APT29 dataset. With a shorter emulation, such as the trail emulation, this drawback did not have a significant impact. Majority of the benign events labeled as "uncertain" were related back to T1059.001 - Command and Scripting Interpreter, with the PID 2388. During a malicious operation a malicious PID was observed targeting 2388, which resulted in this PID being added to the malicious PID list. 2388 was later used by the GHOSTS framework to produce benign events. This PID was observed outside the expected time frame and is given the "isMalicious: false" data field however, since the PID had previously been considered malicious the "verdict" data field receives the "uncertain" label.

Despite these drawbacks the tool was able to successfully label the majority of malicious events with accuracy and high label granularity. All events were not found as roughly 30/278 events that received the "uncertain" label was later found to be malicious. These events did occur close to the excepted time frame and were mostly related to the execution of scripts. The tool is considered a viable source for labeling logs, but refinement is necessary to achieve a perfect score.

# Chapter 6

# Discussion and Related work

This chapter will discuss and compare the final result of the research, and compare the result with the framework presented by Gharib et al. [18] in **Chapter 3**.

## 6.1 Labeling Technique and Approach

The developed labeling tool automatically applies fine-grain labels to the logs on two levels. First, it utilizes process relations through Parent, Child, and Targeted process to develop a process tree containing all the process ID's related to specific malicious operations. This malicious operation is linked to a MITRE tactic/technique through the report generated by CALDERA which is later used to apply fine-grain labels. The script then makes use of the time stamp within the logs to verify that the event occurred within the time period of an operation before applying labels. The time period of each operation is defined by the CALDERA agent attack delegated and finished time stamp, which is extracted from the CALDERA report. A similar time based approach was utilized by Landauer et al [28] when applying labels to logs. Landauer used an attack log containing the time frame of the various attacks, and an attack dictionary containing the expected logs related to each attack, in order to create and apply labels. The attack dictionary was created by carrying out the attacks in an idle system, without normal user activities, and collecting the generated logs. Logs would then be labeled if they matched with the time stamp, and achieved a sufficiently high similarity with the expected logs. This approach did however suffer from misclassification when malicious and normal log lines are similar enough [28]. It is also challenging to guarantee that no background traffic has been collected during the development of the attack dictionary. This may potential create further misclassification of background traffic.

The label approach of Landauer has been further improved upon in the labeling tool developed in this thesis. Time frames of each operation are automatically calculated, removing the requirement of creating a separate file to trace the time frames of each attack. It is also not necessary to conduct the attacks in an idle system in order to later find the

malicious events during the actual simulation. Effectively removing the possibility of misclassification from benign events related to background processes and other operations that may have occurred on the host during the development of the attack dictionary. Malicious events are instead found by leveraging the process relation of events to effectively map all processes spawned or affected in any way by the malicious operation. The granularity of labels applied has also been improved. Labels applied are fine-grain labels that are directly tied to specific tactics and techniques within MITRE ATT&CK. Both of the labeling approaches in the developed tool are conducted automatically by the labeling tool. The only manual operations necessary when applying labels are the extraction of the CALDERA report, and the unprocessed logs. The process of generating attack data is also automated through CALDERA, the attacker only has to select the desired attack scenario or emulation plan. The labeling tool is not limited to the chosen emulation plan, this allows the tool to be utilized in combination with a modular approach, where multiple smaller attack scenarios can be executed and labeled. This allows researchers to specifically select the attack data relevant to the subject, or merge these smaller datasets into a complete dataset. It is also possible to quickly create new datasets once CALDERA releases new APT emulation plans. Being able to quickly develop labeled datasets containing new attacks as these are discovered and implemented into the ATT&CK framework is a valuable asset.

## 6.2 Emulation, Dataset, and Framework

The dataset presented in this thesis is a result of the APT29 emulation plan described in **subsection 3.5.3** and the system activities detected through Sysmon. Artifacts of attack technique that were executed but not successful is present in the dataset. However, this is not the case if the attack was skipped by CALDERA, skipped tactics are presented in **Section 5.1**. Completely skipping specific operations leads to an incomplete dataset. For the dataset to be complete it is expected that it will contain all of the operations within the emulation plan. Exfiltration of data and compromising other hosts with the network is considered a goal of the APT29 emulation plan, this was not achieved as operations related to both these tactics were skipped by CALDERA during the emulation. This is likely due to an issue with the environment used in this thesis, or the version of CALDERA, as Applebaum et al. [2] demonstrated that CALDERA can successfully execute these tactics. CALDERA did not give feedback or error messages indicating why these tactics were skipped. However, the main contribution of this thesis is the developed labeling tool and technique, which was able to successfully apply fine-grain labels as shown in **Section 5.2**. Failed or skipped techniques did not impact the effectiveness of the labeling tool, as it still was able to effectively find malicious events and apply the correct labels to the events.

The decision to skip certain operations affects the attack diversity of the dataset, which could cause biases in systems trained on the dataset. However, as the emulation successfully executed several other tactics, which can be seen by the labels applied in **figure 5.6**, the dataset created in this thesis is considered to have an adequate attack diversity. In addition, since the emulation plan chosen in this thesis involves the execution of several steps, the dataset is beneficial for the research around multistep attacks, where the currently available datasets are rare [34]. It is also acknowledged that the sample size of the dataset is low, 2,276 events where 868 are malicious and 1 406 are benign. It was chosen to end the data collection shortly after CALDERA had finished the emulation. Collecting the logs at a later time would result in a larger sample of benign events due to the approach of only labeling events as malicious if they occurred within the time frame of an operation. All events occurring after the emulation would be found benign as all operations had ended.

Myneni et al. [33] presented a APT focused dataset, DAPT 2020, covering different attack vectors related to the later stages of an advanced attack e.g. Privilege Escalation, Collection, and Exfiltration. The dataset includes system event logs, MySQL Access logs, Audit host IDS logs, Apache access logs, Authentication logs, logs from various services, and DNS logs. The inclusion of multiple log sources makes DAPT 2020 heterogeneous as the samples have different traits. Mynein et al generated benign traffic by having regular users perform what is considered routine business operations throughout a week. This hinders the reproducibility of DAPT 2020 as the benign traffic within the dataset is almost impossible to reproduce without a detailed insight to exactly what and when operations were conducted. Additionally, data in DAPT 2020 is not labeled as the dataset was tested on a semi-supervised IDS, trained on benign data.

The dataset presented in this thesis may be considered homogeneous in comparison to the DAPT 2020 dataset, as the dataset only contains host based logs. However, as the dataset is a pure host based data, it is made heterogeneous through the inclusion of logs regarding file systems, certificates, processes, and call traces. Further, the dataset presented is reproducible as both benign and malicious traffic is generated through automation software, which can be easily replicated. The dataset can also be used and tested on supervised intrusion detection systems as it includes fine-grain labels. However, as shown in **Section 5.1**, the presented dataset is missing data related to Persistence, Lateral Movement, and Exfiltration tactics, all of which are an important aspect of an APT attack.

Gharib et al. [18] presented eleven features necessary for a comprehensive and wholesome framework for generating IDS/IPS dataset. Although the datasets discussed are network based, some of the presented criteria can be applied to host based datasets. The characteristics derived from the work presented by Gharib et al are explained in detail under **Chapter 3**.

1. **Realistic Configuration:** In order to capture the real effects of the attacks, the environment in which the attacks are conducted has to be realistically configured. In regards to a pure host based datasets, this may include using an up-to-date operating system, enabling AV solutions/Firewalls/Windows Auditing, configuring Active Directory, and joining a Domain with other hosts. This was achieved by adding an Active Directory (AD) environment with multiple hosts and one domain controller to the framework. However, in order for CALDERA to work as intended the AV solution and Firewall had to be disabled. Script execution in PowerShell was also enabled as multiple operations involved the execution of scripts through PowerShell. This is not expected from a realistic environment but was necessary in order to successfully perform the emulation.

2. **Complete Capture:** The presented dataset fulfills this in the sense that no artifacts have been removed from the logs, background activities and the outcome of attacks are present within the dataset. However, the dataset does not contain the full kill chain of the APT29 emulation since CALDERA skipped some of the steps within the APT29 emulation plan.

3. **labeled Dataset:** The presented dataset has fine-grain labels directly tied to tactics and techniques within the ATT&CK Matrix, which was achieved through the contributed labeling tool. However, the fine-grain labels are only tied to malicious events, the tool is not able to differentiate between normal user activity (benign) and background/noise. All events found not to be malicious are labeled as benign. This is considered acceptable as the main focus of this thesis is generating APT attack patterns and applying fine-grain labels to the logs.

4. **Attack Diversity:** The dataset is considered to fulfill this criterion as it contains attack data from multiple tactics and techniques within each tactic. The developed labeling tool in combination with CALDERA, can apply fine-grain labels to attacks conducted by CALDERA under the assumption that it has access to the report generated by CALDERA and that the relation between processes is present in the logs. Allowing for the creation of a new dataset containing the desired attack patterns.

5. **Anonymity:** Issues regarding anonymity of the dataset are completely removed through virtualization. The data cannot be related back to any sensitive information.

## 6.3 Limitations

As with any study exploring new fields, this study is subjected to limitations. The presented automated labeling tool is currently limited to CALDERA, leveraging the report generated by CALDERA after an emulation in order to both find and label malicious operations in the raw dataset. The metadata within this report provides the labeling tool with the initial malicious process ID, time frame, and tactic/technique of each operation. The tool is not limited to the testbed designed in this thesis and should work on Sysmon logs in any environment where CALDERA is used to generate attack data. The implemented algorithm for finding malicious events is prone to misclassification due to the nature of interleaving processes if labels were applied solely based on process relations. This was addressed by implementing a time-based label approach, which only labels events as malicious if they are within the expected time frame of an operation. If timed correctly, benign events may still be misclassified as certain operations have a longer time frame, between two to three minutes. This was not found during the manual analysis of the presented dataset, but it is a possibility. If the process chain would be broken by e.g., Process Spoofing, malformed data, or parent PID not found, the subsequent malicious events will go undetected. Process Spoofing was not tested during any of the conducted experiments and is not a part of the APT29 emulation plan. Fine-grain labels are limited to the tactics and techniques within the ATT&CK Matrix, and are only applied to events found malicious. Regarding the labeling of benign events, the tool is not able to differentiate between background and normal user generated events. This was not prioritized as the focus of this thesis was to research the possibilities of generating labeled APT attack data with the help of CALDERA. The developed labeling tool is also limited to host logs, as the implemented algorithm does not work on transmitted network data.

The presented APT29 dataset contains 32 fine-grain attack technique labels, distributed across 8 known tactics. It is acknowledged that the dataset is incomplete as it does not contain all of the expected operations within the APT29 emulation plan. Further, it is acknowledged that the dataset is imbalanced, with a bias toward malicious events. The distribution of 65,8% benign and 35.2% malicious events is shown in **Figure 5.4**. Ideally, only a small portion of the dataset should be malicious as this is representative of realistic datasets. The time period of the emulation is considered unrealistic in comparison with real APT attacks. CALDERA finished the emulation within two hours, while a realistic APT attack is executed over a longer time period. In an ideal case, the emulation would be divided and distributed across several weeks with a slower approach, this was not performed due to time constraints. The presented dataset can be considered a proof-of-concept for the developed labeling tool and technique.

# Chapter 7

# Conclusion and Future Work

This thesis has demonstrated a new approach to generating labeled APT datasets, leveraging CALDERA to generate the attack data and labels, Sysmon to detect system activities and generate logs, and a presented labeling tool for creating datasets containing fine-grain attack labels. Once initially configured, new attack scenarios can be easily executed through CALDERA, raw logs can then be manually extracted and automatically labeled by the presented labeling tool. Allowing for new fine-grain labeled datasets to be created in an efficient and convenient manner. APT29 was emulated in this thesis, an APT29 dataset containing 31 various technique labels is presented as a proof-of-concept for the suggested labeling approach.

**Research Question**: What are the possibilities to develop a labeling tool around CALDERA and Sysmon, to create fine-grain labeled APT datasets?

With regard to the research question, this thesis has demonstrated one possible approach to finding and labeling malicious logs related to a CALDERA emulated APT attack. Logs were produced by Sysmon, which was able to detect malicious operations related to events regarding Process Access/Creation/Termination, Registry activities, Remote Thread Creations, File Creation/Deletion, Windows Management Instrumentation (WMI), Network connection, and changes made to the Registry. Call traces, Command Line Arguments, and process relations are also recorded by Sysmon, granting a detailed insight into the activities occurring on the host.

The developed labeling tool leverages CALDERA's capability to generate an attack report containing meta data about the executed scenario to automatically find and label the corresponding malicious events. The implemented algorithm is based on process relation, utilizing recursion when processing the raw log files for process identifiers (PIDs) observed in relation to a previously known malicious process ID—effectively creating a process tree with PIDs related to the initial operation. These processes and their actions can then be labeled with the tactic and technique observed in the operation. Labels applied are directly tied to specific tactics and tech-

niques within MITRE ATT&CK, resulting in attack labels of high granularity. The resulting dataset can be used for kill chain, or multistep attack detection as it can apply labels related to the various attack phases.

The implemented environment presents a use case for combining CALDERA and Sysmon in order to generate raw APT logs. Benign traffic is introduced through the General HOSTS (GHOSTS) framework in an attempt to diversify the resulting dataset while still maintaining the reproducibility of the generated dataset. The resulting benign data was however not satisfactory as the automated actions were mainly web browsing activities.

In conclusion, the research conducted in this thesis demonstrates the possibilities of using CALDERA and Sysmon to generate raw attack logs. It has introduced a new labeling tool and approach for applying fine-grain attack labels to the raw logs, converting them into a fully labeled dataset. An APT29 dataset is presented, while incomplete and imbalanced, the dataset provides a proof-of-concept for the presented labeling tool and technique. The environment created in this thesis is capable of producing fully labeled attack dataset once new attacks are discovered and implemented into CALDERA and the ATT&CK framework.

## 7.1 Future Work

Based on the experiments conducted in this thesis and their results, several avenues for future work have been found. The developed labeling tool can be further developed in order to differentiate between background traffic and benign user actions. GHOSTS allows for a TrackableId to be implemented into the conducted actions, once the action has been executed a new entry will be created in the server-side PostgresSQL DB of GHOSTS, saving information related to this specific event. This can be leveraged in a similar manner to how malicious processes are found, through process relations, to find all the related benign processes. Benign labels of higher granularity can thus be applied as the benign operations executed can be tied back to the initial PID. However, this would also require improvements to the GHOSTS timeline to include more variations of benign user operations.

In order to reduce the occurrences of the "uncertain" label, which is applied to malicious processes observed outside the operation time frame, the function can be further developed by calculating the difference between the time stamps e.g., malicious PID observed ten seconds after the operation time frame is more likely to be malicious than if observed ten minutes later. A more advanced and accurate approach is to analyze the event ID of the last process seen in relation to the PID outside the operation time frame. If the last operation was a Sysmon Event Id 1 - Process Created, the following operation conducted by this newly created process can with higher accuracy be considered malicious if observed outside the operation time frame. This would require an evaluation score to be applied to the various Sysmon events ahead of time.

Another avenue for future work is to improve the emulation from CALDERA in order to fully emulate all of the steps within the emulation plan. The labeling tool should also be tested on other APT plans once these are implemented into CALDERA. It is also recognized that in order to fully evaluate the presented APT29 dataset, a machine learning model should be trained on the dataset and tested. An interesting topic to further improve this research area could be to replicate the study while capturing both host and network logs. A study similar to the one presented in this thesis, but regarding network dataset, was conducted by Julie L. Gjerstad [20]. The network dataset presented by Julie could be merged with the APT29 host dataset in order to create a complete dataset containing both network and host data.

# Appendices

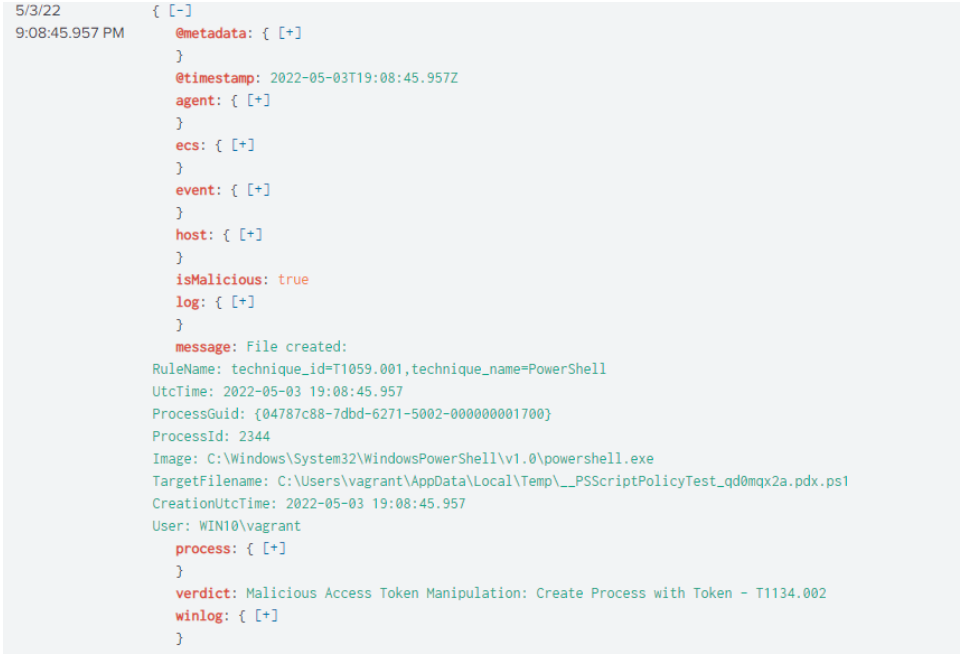
# .1 Labeled logs related to T1134.002


```
   event: { [+]
   }
   host: { [+]
   }
   isMalicious: true
   log: { [+]
   }
   message: Process Create:
RuleName: technique_id=T1059.001,technique_name=PowerShell
UtcTime: 2022-05-03 19:08:45.897
ProcessGuid: {04787c88-7dbd-6271-5002-000000001700}
ProcessId: 2344
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
FileVersion: 10.0.18362.1 (WinBuild.160101.0800)
Description: Windows PowerShell
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: PowerShell.EXE
CommandLine: powershell.exe -ExecutionPolicy Bypass -C ". .\stepFourteen_bypassUAC.ps1;bypass;"
CurrentDirectory: C:\Users\vagrant\
User: WIN10\vagrant
LogonGuid: {04787c88-70e4-6271-70d9-030000000000}
LogonId: 0x3D970
TerminalSessionId: 1
IntegrityLevel: High
Hashes: SHA1=36C5D12033B2EAF251BAE61C00690FFB17FDDC87,MD5=CDA48FC75952AD12D99E526D0B6BF70A,SHA256=908B64B1971A979C7E3E8CE
ParentProcessGuid: {04787c88-71f8-6271-ec00-000000001700}
ParentProcessId: 7816
ParentImage: C:\Users\Public\sandcat.go-windows.exe
ParentCommandLine: "C:\Users\Public\sandcat.go-windows.exe" -server http://192.168.56.104:8888 -group red
ParentUser: WIN10\vagrant
   process: { [+]
   }
   verdict: Malicious Access Token Manipulation: Create Process with Token - T1134.002
   winlog: { [+]
   }
```

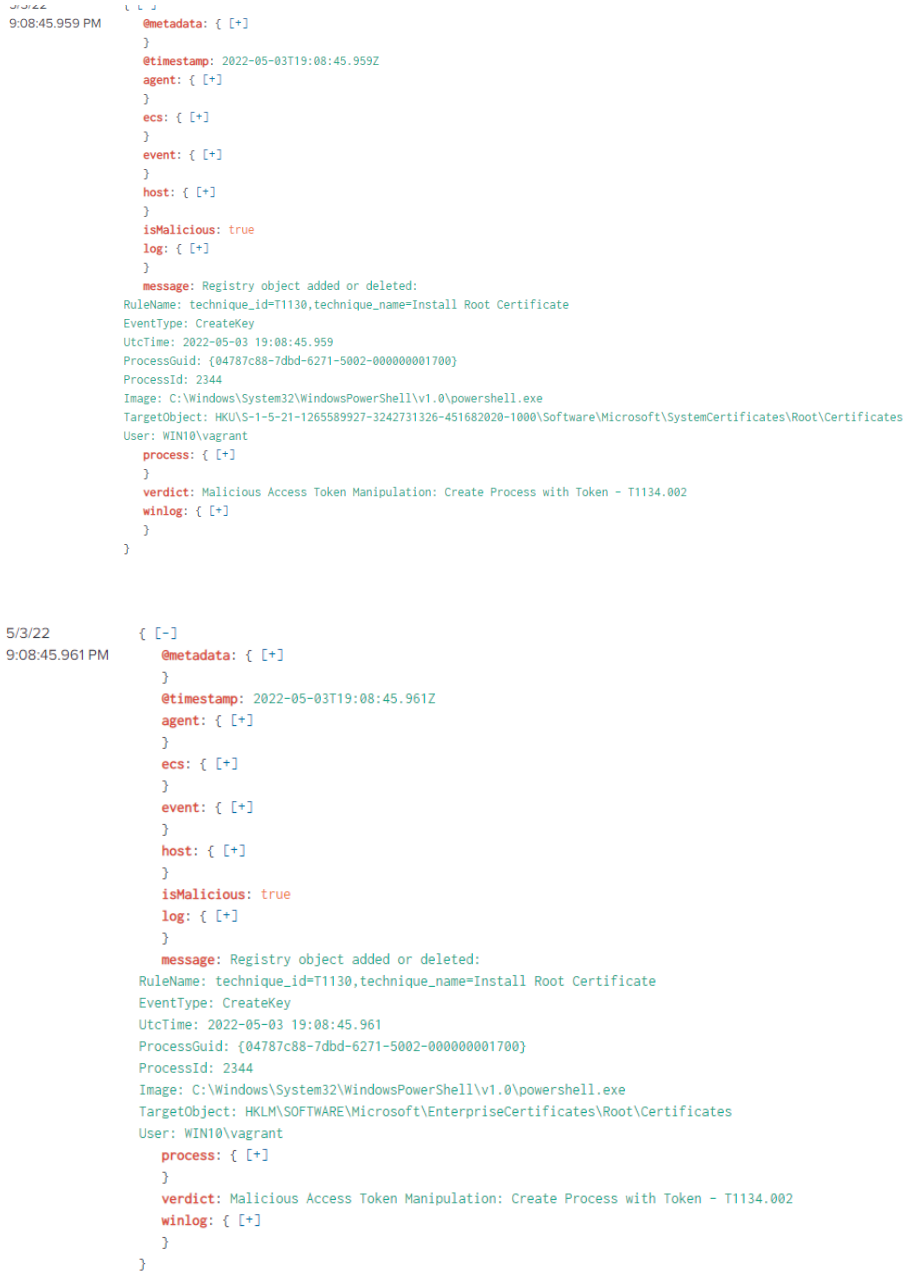


```
5/3/22 9:08:45.894 PM  { [-]
   @metadata: { [+]
   }
   @timestamp: 2022-05-03T19:08:45.894Z
   agent: { [+]
   }
   ecs: { [+]
   }
   event: { [+]
   }
   host: { [+]
   }
   isMalicious: true
   log: { [+]
   }
   message: File created:
RuleName: technique_id=T1059.001,technique_name=PowerShell
UtcTime: 2022-05-03 19:08:45.894
ProcessGuid: {04787c88-71f8-6271-ec00-000000001700}
ProcessId: 7816
Image: C:\Users\Public\sandcat.go-windows.exe
TargetFilename: C:\Users\vagrant\stepFourteen_bypassUAC.ps1
CreationUtcTime: 2022-05-03 19:08:45.894
User: WIN10\vagrant
   process: { [+]
   }
   verdict: Malicious Access Token Manipulation: Create Process with Token - T1134.002
   winlog: { [+]
   }
}
```

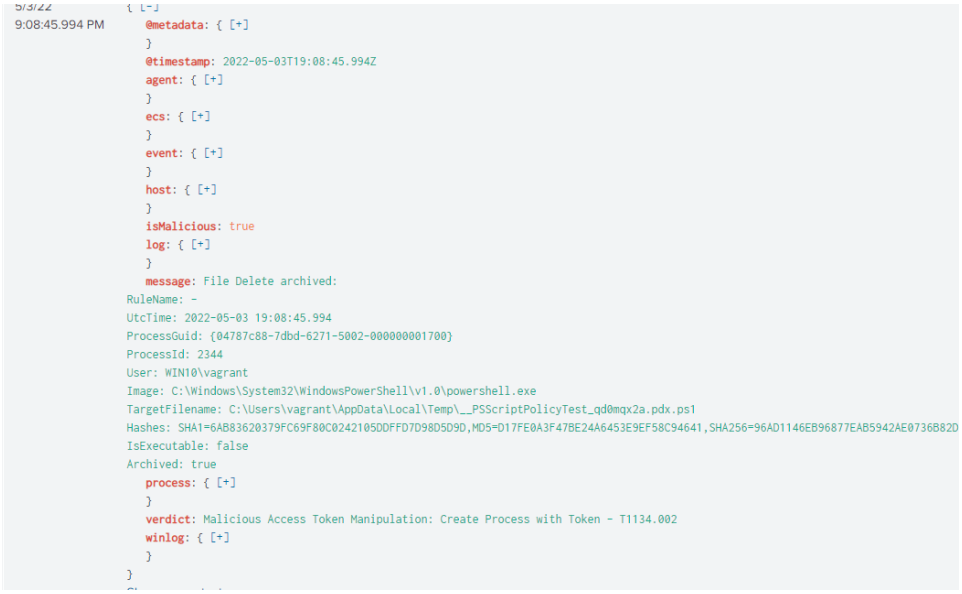

CALDERA Agent PID 7816 - Spawns PID 2344

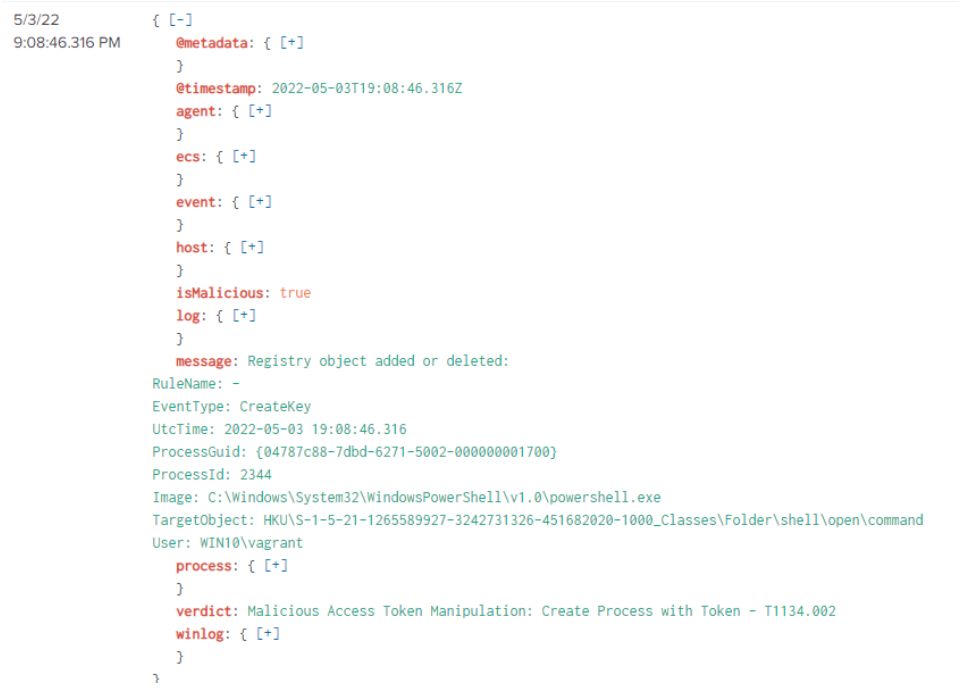




PID 2344 - DLL loading and file created

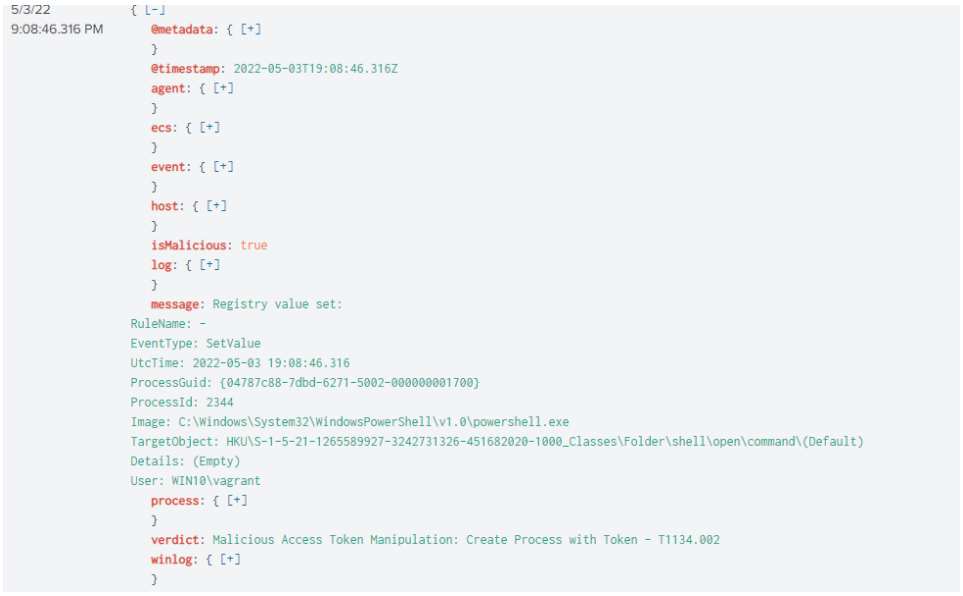


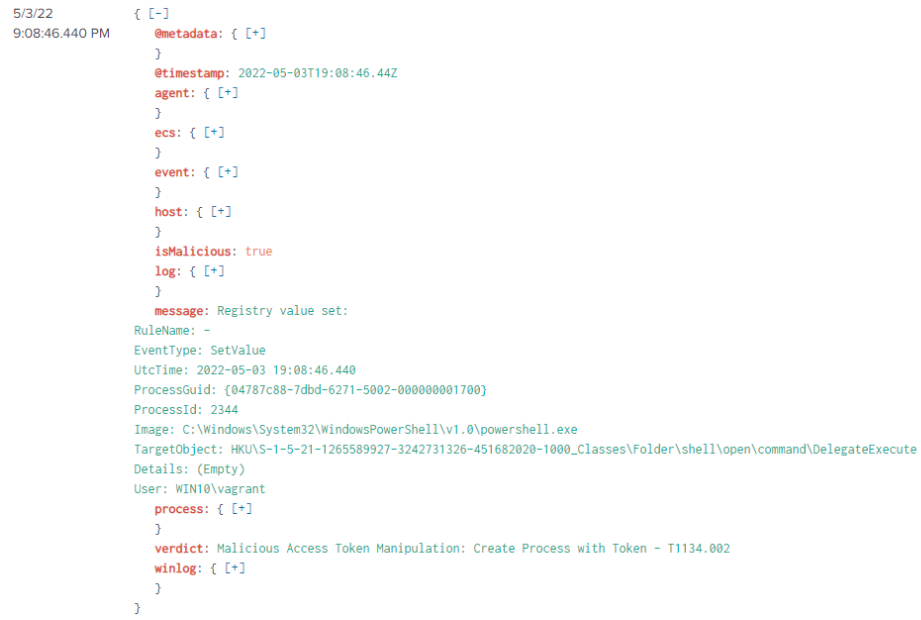

PID 2344 - Changes to Certificates



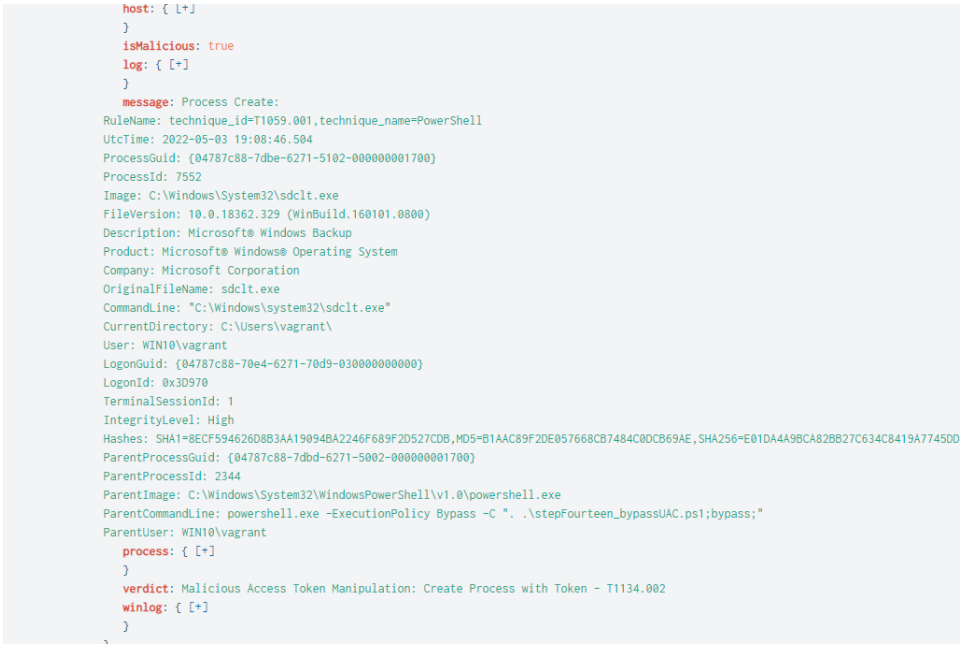



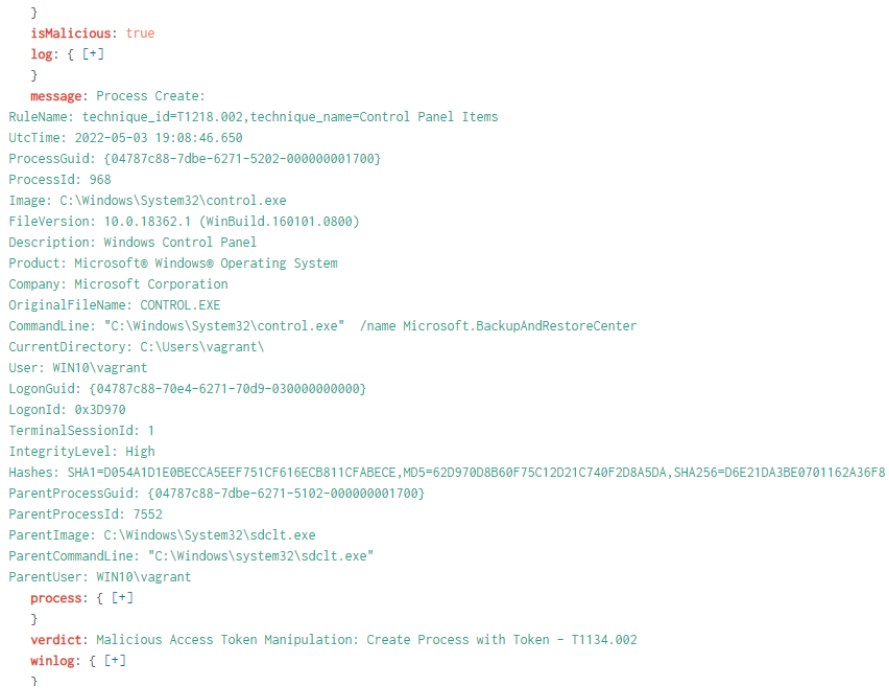

PID 2344 - Delete previous created file, registry changes




PID 2344 - Changes to registry.




PID 2344 - Spawns 7552 which in return spawns 968.

# Bibliography


[1] Adel Alshamrani et al. 'A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities'. In: *IEEE Communications Surveys & Tutorials* 21.2 (2019), pp. 1851–1877.

[2] Andy Applebaum et al. 'Analysis of Automated Adversary Emulation Techniques'. In: *Proceedings of the Summer Simulation Multi-Conference*. SummerSim '17. Bellevue, Washington: Society for Computer Simulation International, 2017.

[3] Andy Applebaum et al. 'Intelligent, automated red team emulation'. In: *Proceedings of the 32nd Annual Conference on Computer Security Applications*. 2016, pp. 363–373.

[4] *APT29 emulation plan, retrieved from https://github.com/center-for-threat-informed-defense/adversary*$_e$*mulation*$_l$*ibrary*.

[5] Atomic-Red-Team. *https://redcanary.com/blog/atomic-red-team-testing/*.

[6] *CALDERA. retrieved from https://www.mitre.org/research/technology-transfer/open-source-software/caldera*.

[7] Carlos A Catania and Carlos Garcıa Garino. 'Automatic network intrusion detection: Current techniques and open issues'. In: *Computers & Electrical Engineering* 38.5 (2012), pp. 1062–1072.

[8] Zhuo Chen et al. 'Understanding the Impact of Label Granularity on CNN-Based Image Classification'. In: *IEEE International Conference on Data Mining Workshops*. 2018.

[9] Clarence Chio and David Freeman. *Machine learning and security: Protecting systems with data and algorithms*. " O'Reilly Media, Inc.", 2018.

[10] Henry Clausen, Robert Flood and David Aspinall. 'Traffic generation using containerization for machine learning'. In: *Workshop Pre-Proceedings in DYnamic and Novel Advances in Machine learning and Intelligent Cyber Security (DYNAMICS)* (2019).

[11] Scott E Coull et al. 'Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces.' In: *Ndss*. Vol. 7. 2007, pp. 35–47.

[12] Gideon Creech. 'Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks.' PhD thesis. University of New South Wales, Canberra, Australia, 2014.

[13] Robert K Cunningham et al. *Evaluating intrusion detection systems without attacking your friends: The 1998 DARPA intrusion detection evaluation*. Tech. rep. MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1999.

[14] Jonathan J Davis and Andrew J Clark. 'Data preprocessing for anomaly based network intrusion detection: A review'. In: *computers & security* 30.6-7 (2011), pp. 353–375.

[15] *DetecionLab, Retrieved from https://detectionlab.network/.*

[16] Thomas Edgar and David Manz. *Research methods for cyber security*. Syngress, 2017.

[17] *ElasticSearch Stack. Beats. Retrieved from https://www.elastic.co/beats/.*

[18] Amirhossein Gharib et al. 'An evaluation framework for intrusion detection dataset'. In: *2016 International Conference on Information Science and Security (ICISS)*. IEEE. 2016, pp. 1–6.

[19] *GHOSTS Timeline Repository, retrieved from https://github.com/cmu-sei/GHOSTS/tree/master/src/Ghosts.Client/Sample%20Timelines.*

[20] Julie Lidahl Gjerstad. *Generating labelled network datasets of APT with the MITRE CALDERA framework*. University of Oslo. 2022.

[21] Waqas Haider et al. 'Windows based data sets for evaluation of robustness of host based intrusion detection systems (IDS) to zero-day and stealth attacks'. In: *Future Internet* 8.3 (2016), p. 29.

[22] *https://attack.mitre.org/groups/G0016/.*

[23] *JavaScript Object Notation (JSON). Retrieved from https://www.json.org/json-en.html.*

[24] Fikret Kadiric. *MasterThesis-LogLabeler*. Version 1.0.0. May 2022. URL: https://github.uio.no/fikretk/MasterThesis-LogLabeler/tree/main.

[25] Fikret Kadiric. *MasterThesis-LogLabeler*. Version 1.0.0. May 2022. URL: https://github.com/Fiik/MasterThesis-LogLabeler.

[26] Kevin S Killourhy and Roy A Maxion. 'Toward realistic and artifact-free insider-threat data'. In: *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*. IEEE. 2007, pp. 87–96.

[27] Robert Koch, Mario Golling and Gabi Dreo Rodosek. 'Towards comparability of intrusion detection systems: New data sets'. In: *TERENA Networking Conference*. Vol. 7. 2014.

[28] Max Landauer et al. 'Have it Your Way: Generating Customized Log Datasets With a Model-Driven Simulation Testbed'. In: *IEEE Transactions on Reliability* 70.1 (2020), pp. 402–415.

[29] Tien-Chih Lin, Cheng-Chung Guo and Chu-Sing Yang. 'Detecting Advanced Persistent Threat Malware Using Machine Learning-Based Threat Hunting'. In: *European Conference on Cyber Warfare and Security*. Academic Conferences International Limited. 2019, pp. 760–XX.

[30] Vasileios Mavroeidis and Audun Jøsang. 'Data-driven threat hunting using sysmon'. In: *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*. 2018, pp. 82–88.

[31] *Michael Haag. Resources for learning about deploying, managing and hunting with Microsoft Sysmon. Retrieved from https://github.com/MHaggis/sysmon-dfir*.

[32] *Microsoft. Audit Policy Recommendations. Retrieved from https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/audit-policy-recommendations*.

[33] Sowmya Myneni et al. 'Dapt 2020-constructing a benchmark dataset for advanced persistent threats'. In: *International Workshop on Deployable Machine Learning for Security Defense*. Springer. 2020, pp. 138–163.

[34] Julio Navarro, Aline Deruyver and Pierre Parrend. 'A systematic survey on multi-step attack detection'. In: *Computers & Security* 76 (2018), pp. 214–249.

[35] Joshua Ojo Nehinbe. 'A critical evaluation of datasets for investigating IDSs and IPSs researches'. In: *2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)*. IEEE. 2011, pp. 92–97.

[36] *Olaf Hartong. Sysmon Configuration File. Retrieved from https://github.com/olafhartong/sysmon-modular*.

[37] *Process Injection: Asynchronous Procedure Call, Retrieved from https://attack.mitre.org/techniques/T1055/004/.*

[38] *Rapid7. Metasploit Penetration Testing Software. Retrieved from http://www.metasploit.com.*

[39] Saurabh Singh et al. 'A comprehensive study on APT attacks and countermeasures for future networks and communications: challenges and solutions'. In: *The Journal of Supercomputing* 75.8 (2019), pp. 4543–4574.

[40] Branka Stojanović, Katharina Hofer-Schmitz and Ulrike Kleb. 'APT datasets and attack modeling for automated detection methods: A review'. In: *Computers & Security* 92 (2020), p. 101734.

[41] *SwiftOnSecurity Sysmon Configuration File, Retrieved from https://github.com/SwiftOnSecurity/sysmon-config*.

[42] *The mitre corporation, "Adversary Emulation Plans," MITRE ATT&CK Retrieved from https://attack.mitre.org/resources/adversary-emulation-plans*.

[43] *The mitre corporation, "MITRE ATT&CK Framework". Retrieved from https://attack.mitre.org/.*

[44] Dustin D Updyke et al. *Ghosts in the machine: A framework for cyber-warfare exercise npc simulation*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA, 2018.

[45] Dustin D Updyke et al. *Ghosts in the machine: A framework for cyber-warfare exercise npc simulation*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA, 2018.

[46] *US Government, APT29 ties to SolarWinds. Retrieved from https://www.whitehouse.gov/briefing-room/statements-releases /2021/ 04/15/fact-sheet-imposing-costs-for-harmful-foreign-activities-by-the-russian-government/.*

[47] *Vagrant, Retrieved from https://www.vagrantup.com/docs.*

[48] *Windows Sysinternals Suite. System Monitor. Retrievend from https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon.*