

Generating labelled network datasets of APT with the MITRE CALDERA framework

Julie Lidahl Gjerstad



Thesis submitted for the degree of
Master in Informatics: Information Security
60 credits

Institute for Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2022

**Generating labelled
network datasets of APT
with the MITRE CALDERA
framework**

Julie Lidahl Gjerstad

© 2022 Julie Lidahl Gjerstad

Generating labelled network datasets of APT with the MITRE CALDERA
framework

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

Printed: Reprosentralen, University of Oslo

Abstract

Threats in the cyber domain are ever-evolving, and threat actors continue to refine their techniques and expand their reach. These increasingly sophisticated attacks are performed by well-funded and well-organised *APT groups*, presenting a challenge for security professionals. To improve the defence mechanisms to detect these complex threats, the cybersecurity community requires sufficient data and *labelled datasets*. However, the number of publicly available APT datasets is limited, leaving defenders to rely on outdated datasets that do not reflect the complicated and dynamic threats we face today.

MITRE ATT&CK is a framework that serves as an industry knowledge base for characterising malware, attacker campaigns, and how adversaries engage with systems during an operation. *MITRE CALDERA* is a tool developed for professionals to test the security of their systems, containing tactics and techniques defined in ATT&CK. CALDERA focus on simulating post-compromise attacks that organisations use to train their defences.

This thesis explores one approach to labelling network datasets. CALDERA was used to emulate one specific APT group in a controlled and targeted experiment. The experiment generated network traffic that included a variety of benign, background and malicious characteristics. The labelling approach is called *LabelGen*, a script to generate network datasets with ground truth labels on attack technique level, mapped to MITRE ATT&CK. Final results and evaluation indicated that the datasets had labels corresponding to relevant attacks. The approach used with LabelGen may be suitable and adaptable to additional CALDERA simulations. This thesis contributes to cybersecurity detection, as LabelGen generates labelled and granular APT network datasets.

Table of contents

List of Figures	iv
List of Listings	v
1 Introduction	1
1.1 Motivation	1
1.1.1 Intrusion Detection System	2
1.2 Research question	4
1.3 Methodology	4
1.4 Contributions	5
1.5 Chapter outline	6
2 MITRE ATT&CK and CALDERA	7
2.1 The MITRE Corporation	7
2.2 MITRE ATT&CK™	8
2.2.1 Background and history	9
2.2.2 Use Cases	10
2.2.3 The ATT&CK Model	10
2.3 Advanced Persistent Threats	12
2.3.1 Adversary emulation plans	15
2.4 MITRE CALDERA™	16
2.4.1 Architecture and example usage	17
2.4.2 CALDERA terminology	19
3 Background	21
3.1 Simulation	21
3.2 Capturing network traffic	21
3.3 Labelling	22
3.4 Challenges in labelling datasets	22
3.5 Current solutions	24
3.5.1 APT detection	27
3.6 Summary and discussion	27
4 Approach and implementation	28
4.1 General plan	28
4.2 Network architecture	29
4.3 APT29	31
4.3.1 APT29 EMU plan	32
4.4 DetectionLab	35

4.5	GHOSTS	37
4.6	Approach	38
4.7	Implementation	39
4.7.1	LabelGen: implementation of labelling	41
4.8	Summary and discussion	45
5	Results and evaluation	47
5.1	The datasets	47
5.2	Manual inspection	50
5.2.1	Packet statistics	52
5.3	Evaluating LabelGen	55
5.3.1	Experimenting with a different attack	55
5.4	Machine Learning	58
5.5	Proof-of-concept	60
5.5.1	Implementation of SVM	61
5.6	Summary and discussion	63
6	Discussion and related work	65
6.1	CALDERA as a framework	65
6.1.1	Alternative tools for emulation	68
6.2	Labelling network traffic	68
6.3	LabelGen	69
6.4	Use of containers/DetGen	70
6.5	GHOSTS and its alternatives	71
6.6	The environment	72
7	Conclusion and future work	73
7.1	Summary of results and findings	73
7.2	Future work	74

List of Figures

1.1	Overview of the whole thesis work.	5
2.1	Part of ATT&CK Matrix for Enterprise, v10.1.	11
2.2	Expanded the tactic "Initial Access", ATT&CK v10.1.	12
2.3	APT attack steps [2].	13
2.4	Adversary Emulation Plan for APT29 in CALDERA.	17
2.5	Visualisation of the CALDERA infrastructure [5].	18
3.1	Injection Timing, a widely deployed labelling strategy [62]. . .	25
3.2	Human-guided labelling [62].	26
3.3	Comparison of current APT anomaly detection methods [2]. . .	27
4.1	Phases of the work in this thesis.	28
4.2	Simplified representation of the experiment environment. . .	29
4.3	Architecture of the experiment setup.	30
4.4	Operation flow of APT29 EMU plan [7].	33
4.5	DetectionLab overview [81].	36
4.6	Overall workflow of the approach in this thesis.	38
4.7	Responsibilities between the VMs during implementation. . .	39
4.8	CALDERA operation, attack steps 42-48.	40
4.9	Order of implementation.	42
5.1	A snippet of the dataset in CSV format.	49
5.2	Packets related to ATT&CK technique T1036.005.	51
5.3	Confirmed beaconing packet prior to attack step T1036.001. .	51
5.4	Number of packets for each technique	54
5.5	Custom adversary profile created in CALDERA.	56
5.6	Custom adversary plan running in CALDERA.	56
5.7	Corresponding timestamps in the new experiment.	57
5.8	The workflow of ML with feature engineering [140]	60
5.9	Confusion matrix with the new dataset.	63
6.1	APT29 Emulation plan: planned and executed steps.	67

List of Listings

1	processDeclareLabels.py	43
2	findCaldera.py	44
3	labelPackets.sh	45
4	One packet shown in JSON format.	49

Acknowledgements

Throughout the writing of this thesis, I received a great lot of support and guidance. To begin, I want to thank Gudmund Grov, Espen Hammer Kjellstadli and Markus Leira Asprusten, my three external supervisors at FFI. Over the last year and a half working on this master project, they have demonstrated commitment through close follow-up, a positive mindset and advice. Furthermore, they have provided me with guidance, motivation and help to find the tools I needed to choose the right direction and complete this master thesis.

I would also like to thank Audun Jøsang, my internal supervisor at University of Oslo, for his constructive and honest feedback and words of encouragement.

Next, I want to express my gratitude to Fikret Kadiric, my study colleague working on a similar thesis, for his help and companionship. I value our collaboration, especially over the last year. Sharing experiences, knowledge, approaches and everyday life has been helpful.

Monika and Fredrik, the people closest to me, deserve special recognition for their love, care and support. I would not have been able to complete this project, let alone this education, without them.

Julie Lidahl Gjerstad,
May 2022

Chapter 1

Introduction

1.1 Motivation

Cyber threats have gradually shifted their focus from primarily targeting nation-states, and their connected entities, more towards also including the private and corporate sectors [2]. These new types of attacks are complex and are performed by well-funded and organised criminal groups, entitled Advanced Persistent Threats (APT) [114]. APT groups operate stealthily and adjust their cyberattacks precisely for each target [114]. Well-funded adversaries have the resources to develop highly advanced tools and deploy techniques to remain undetected within the target system for extended periods [2]. Standard defence mechanisms, such as signature-based anti-virus software and intrusion detection (IDS) and prevention systems (IPS), rarely succeed in detecting these APT attacks [2]. Information security researchers and organisations focus on implementing machine learning (ML) in their efforts to detect advanced threat actors [90]. A successful threat detection system should recognise contextual and anomalous behaviour within large volumes of data. ML algorithms rely heavily on datasets to identify intricate and sophisticated threats like APT [90]. To train models, the algorithms require realistic data, and many of them require *labelled data*. However, there are limited numbers of publicly available, labelled datasets. Consequently, security researchers frequently rely on the few older ones that are accessible [27]. These datasets are often generated in virtualised and isolated environments for specific test cases [111][62]. While this configuration allows for proper control of the gathered data and the network itself, it hides many of the essential aspects that are necessary to correctly differentiate between normal and malicious behaviour [2][62].

Clausen et al. [26] raised questions concerning modern datasets, pointing to the lack of four main characteristics:

- *Lack of variance* in benign traffic, where the range of sub-activities in each protocol is limited, leaving uncertainty about whether it represents real-world usage. Lack of variation in individual protocols leads to homogeneity at the packet exchange and network flow levels.

- *Lack of ground truth* that noticeably defines the malicious behaviour, preferably with more granular labelling than merely 'malicious' and 'benign'. Ground truth labels for network traffic are difficult to come by. This statement is logical, given how difficult - if not impossible - it is to separate traffic from different origins retrospectively, e.g. background processes like software updates, authentication traffic and advertising features.
- *Static design*, in which the dataset only contains data that is representative of the system when it was created, is unavoidable. All currently available NIDS datasets are created with a fixed testbed of host machines that contain specific vulnerabilities for the selected attacks.
- *Limited size*, where the datasets typically contain 5-10 hosts and short capture periods of little more than 5-6 weeks.

Another challenge with labelling datasets originates from the increasing use of encrypted traffic. While the cryptographic protocol *TLS*, or *Transport Layer Security*, protects the majority of internet traffic, it also protects malware authors in hiding malicious network connections [51]. Similarly, attackers are increasingly adopting *IP Flux*, a technology in which IP addresses continuously change the mapping to domain names to circumvent IP blacklisting and monitoring [108]. The examples demonstrate the importance of increasing the capabilities of network security monitoring devices for detecting malicious content hidden in encrypted network traffic and malware [51].

1.1.1 Intrusion Detection System

Intrusions on a computer system or network are often detected using an Intrusion Detection System, abbreviated IDS. This device or software automatically monitors and analyses behaviour and traffic and generates reports if it detects any suspicious activity [79]. IDS is divided into three major categories:

- *Signature-based Detection (SD)* compare patterns or strings to known attacks and reports if detects any patterns that matches entries in its *signature database*. This method relies on knowledge assembled during real and specific attacks, and system vulnerabilities [79].
- *Anomaly-based Detection (AD)* focuses on identifying activities that deviate from known or expected behaviour, e.g. network connections, host or user interactions and behaviour on endpoints. AD relies heavily on high-quality and realistic data to increase its detection capabilities [6]. AD aims to reduce the increasing number of zero-day attacks [27].
- *Stateful Protocol Analysis (SPA)* represents an IDS that can recognise and trace protocol states such as request-response pairing. SPA might appear similar to AD, but it relies on vendor-developed generic profiles

for specific protocols, whereas AD uses pre-loaded network or host-specific profiles [79].

Both SD and AD have advantages and disadvantages. SD recognise intrusions by looking for pre-defined attack patterns and signatures. Accordingly, attackers may encrypt or modify their malware to avoid detection [79]. SD methods are therefore unable to detect new or tailored attacks, leading AD to be a better option [84] [72]. AD builds on the assumption that an attack on a computer system will be noticeably different from regular system activity, and an intruder will exhibit a pattern of behaviour different from that of the typical user [15, 117]. Therefore, AD can identify unknown attacks [79]. However, it is challenging to identify normal user behaviour, in which AD has the consequence of a higher percentage of false positives (FP). How to define normal user behaviour in order to distinguish between benign and malicious activity is a central question within anomaly detection [15].

In addition to the categories mentioned in the above list, *Hybrid IDS* employs multiple methods at the same time, which allows for a more comprehensive and accurate detection [79].

We also differentiate IDS based on whether they detect intrusions at the network or host level. i.e. their scope. A *Network Intrusion Detection System (NIDS)* captures and analyses network traffic, and a *Host-Based Intrusion Detection System (HIDS)* monitor important operating system files [79].

Because of the challenges presented, according to [26], there only exist four publicly available structured datasets for NIDS that incorporate real-world traffic and attacks [26]. Guerra et al. [62] concluded that all current labelling approaches have fundamental flaws in terms of quality, volume and speed of the generated dataset [62]. Clausen et al. [26] mentioned similar difficulties .

This thesis will present my approach to creating a labelled dataset to address the difficulties presented. A controlled experiment was conducted to generate malicious, benign and background network traffic, directing to the issues regarding dataset variation. During the experiment, the network traffic was captured in a logfile. A script, *LabelGen*, was developed to produce a dataset containing ground truth labels from the network logfile. The malicious network traffic is labelled according to specific attacks, providing granularity. *LabelGen* is my solution to the problem of modern datasets having insufficiently detailed labels, and the dataset was created for NIDS using for anomaly detection. This thesis will focus on *LabelGen* and the methods I used to label the network logfile and an evaluation of the labelling to determine its efficiency.

1.2 Research question

By conducting a targeted, controlled experiment, the work in this thesis builds on *experimental research* [54]. The following research question will act as a foundation for the work:

Can MITRE CALDERA be used to generate a dataset of APT behaviour with fine-grained labels that can identify different stages of the attack?

The Hypothesis (H) is that MITRE CALDERA can generate APT network traffic that can be labelled at a fine-grained level to distinguish attack stages. The following objectives will further guide the research, describing what this work expects to accomplish:

1. To be able separate and label APT traffic and benign traffic at an appropriate level of details
2. To distinguish between the respective APT tactics and techniques in the label to capture the different stages of the attack
3. To combine and capture APT network traffic from Caldera with realistic benign traffic in a way that allows for training machine learning-based detection capabilities

1.3 Methodology

Figure 1.1 illustrates the methodology and structure of the work in this thesis. The work can be separated into three different phases. The *experiment phase* refers to the targeted, controlled experiment that was conducted, which builds on theory on *experimental research* [54]. All variables in the experiment were supervised and controlled, i.e. the malware was isolated not to damage real environments, and the hosts on the network were monitored and configured adequately to a confined network with minimal noise. Additionally, following the phases of the attacks and stopping and ending the experiment at chosen times are standard practices in experimental research [54]. The red boxes indicate activities related to the attacks that are simulated, the green refers to the benign activity. A camera in the top of the experiment phase indicate that the activities in this phase was recorded or captured. It shows the main steps of the experiment, with start and stop points for the attacks.

The result of the experiment was a network logfile, which was used in the labelling in LabelGen, in the *processing phase*. LabelGen will be described in detail later. LabelGen produces three files; one labelled logfile and two datasets. The labelled logfile is inspected manually for evaluating LabelGen. The dataset in CSV format, as indicated in Figure 1.1, is used in an evaluation in a proof-of-concept ML. This process is also done to evaluate LabelGen and the usability of the datasets that LabelGen created.

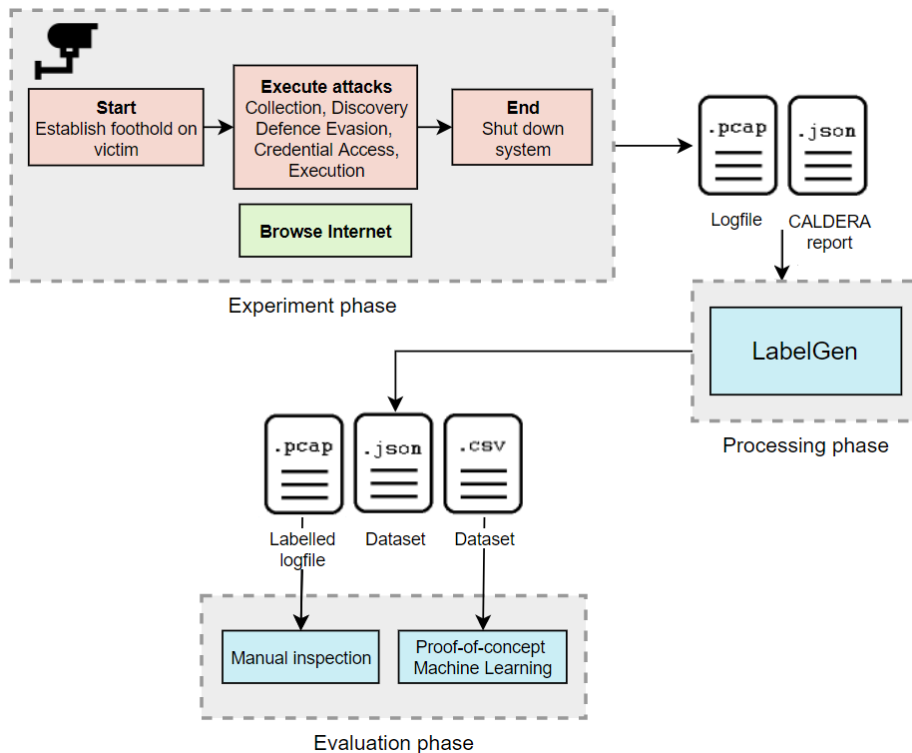


Figure 1.1: Overview of the whole thesis work.

1.4 Contributions

The main contributions of this thesis are:

- LabelGen¹, the script that can label network packets in a network logfile and produce datasets is described,
- an APT dataset is generated,
- the generated dataset has ground truth labels,
- the malicious labels are on technique level, mapping to specific attacks from MITRE ATT&CK,
- the labelled network logfile is described,
- an evaluation of the CALDERA framework is given,
- an ML model is trained for proof-of-concept validation of LabelGen,
- different methodologies for labelling are discussed,
- automatically labels a network logfile with attack techniques from the framework CALDERA.

¹<https://github.com/julielgjerstad/LabelGen>

LabelGen contributes to the field of cybersecurity detection research. Labelling a network logfile (.pcap) opens up many possibilities for future applications. Because the logfile is in such a "raw" format, it may be used for generating flow statistics or be exported to other file types and formats as desired. During initial research, I did not encounter any work that discussed labelled logfiles or how the logfile is labelled in LabelGen.

This thesis addresses one of the challenges described in the section 1.1, in which researchers frequently employ outdated datasets due to the lack of updated, publicly available ones. CALDERA was utilised for this project, and it is regularly updated. With the addition of new functionality to CALDERA, it might be used to generate network traffic for LabelGen. As a result, LabelGen enables more frequent dataset creation.

1.5 Chapter outline

The rest of the thesis is structured as follows:

Chapter 2: Mitre ATT&CK and CALDERA introduces the MITRE Corporation and the CALDERA framework, as well as Advanced Persistent Threats (APT) and Adversary Emulation (EMU) plans.

Chapter 3: Simulation, capturing and labelling network traffic presents simulation, network traffic capture, and labelling as fundamental concepts. Furthermore, it provides a more detailed description of the challenges presented in the introduction, as well as the current solutions researchers have used to address issues.

Chapter 4: Approach and implementation covers the experiment that was conducted and the approach to labelling. The tools, frameworks, and concepts are first introduced to provide details for the rest of the chapter. The subsequent section describes the approach to labelling, followed by the actual labelling implementation, particularly LabelGen.

Chapter 5: Results and evaluation review the results of the implementation. LabelGen is evaluated by inspecting the labelled logfile manually and training a proof-of-concept ML model on one of the datasets. LabelGen will also attempt to label a second, distinct logfile to determine whether it is generalisable or highly customised.

Chapter 6: Discussion and related work reflects on the labelling approach. In the chapter, LabelGen is compared to other methods of labelling. Additionally, the chapter will review and discuss the experiment, alternative tools and experiment environment.

Chapter 7: Conclusion and future work summarises the results and findings, discusses the research question and makes recommendations for future work.

Chapter 2

MITRE ATT&CK and CALDERA

This chapter will present a more detailed context for this thesis, beginning with the MITRE Corporation. MITRE developed the CALDERA framework used in this thesis. The following section covers Advanced Persistent Threats (APT), which provides background information for CALDERA. Finally, the section describing CALDERA will detail its purpose, architecture and usage. This chapter as a whole will address the matters in sufficient depth to lay the groundwork for the subsequent chapters.

2.1 The MITRE Corporation

The MITRE Corporation hereafter referred to as MITRE, is a not-for-profit organisation founded in 1958. Initially, the company provided engineering, and technical guidance to the US federal government [44]. Today, they work in the public interest across federal, state and local governments in the United States and industry and academia worldwide. MITRE focuses on innovative ideas in a variety of fields, including Artificial Intelligence (AI), quantum information science, cyber threat intelligence, and cyber resilience [39].

MITRE works in the public interest, meaning that they have no owners or stakeholders nor compete with the industry. This position serves as the foundation for their objectivity, allowing them to be impartial due to the lack of commercial interests [39].

MITRE's sole focus is to operate *federally funded research and development centers* (FFRDCs), unique organisations that can assist the United States government. They contribute with scientific research and analysis, development and acquisition, along with systems engineering and integration [41]. MITRE operates multiple FFRDCs, which enables them to share knowledge across the corporation [39]. Since 2014, MITRE has operated the National Cybersecurity FFRDC (NFC), which is sponsored by the National Institute of Standards and Technology (NIST) [43]. NFC is Amer-

ica's first and only FFRDC dedicated to cybersecurity and the advancement of secure technologies. MITRE seeks to strengthen and expand collaboration between the public and private sectors by designing and implementing practical cybersecurity solutions by implementing NIST, and other industry standards and frameworks [39]. Through targeted solutions, they work to improve the organisations' ability to identify, protect, detect, respond to, and recover from cyber threats and vulnerabilities [39].

The organisation maintains the *CVE* Program (The Common Vulnerability and Exposures). *CVE* is a knowledge base that provides a reference method for *publicly known* information security vulnerabilities, and exposures [131]. The goal of *CVE* is to identify, define, and catalogue publicly disclosed vulnerabilities by publishing *CVE Records*. This record allows for communication of consistent descriptions of vulnerabilities to assure that cybersecurity and information technology professionals are discussing the same issue and to coordinate efforts to prioritise and address the vulnerabilities [31].

MITRE also operates the *CWE* (Common Weakness Enumeration) List, a community-created list of software weaknesses and vulnerabilities [129]. *CWE* seeks to minimise software and hardware vulnerabilities in products before they are delivered, by educating developers, architects and designers on how to eliminate the most common weaknesses [32]. The system contains a collection of weaknesses that could make systems, networks or hardware vulnerable to attacks, i.e. flaws, faults, bugs or errors in implementation, code, design or architecture. The security community develops *CWE*, and its associated classification taxonomy serves as a common language for describing *CWE Records*.

2.2 MITRE ATT&CK™

It is essential to obtain insight into how attackers innovate their *tactics, techniques and procedures (TTPs)* in their attempts to circumvent established defences. The difficulty of this process has, however, increased [2, 114]. Various obfuscation techniques, such as *polymorphism* and *metamorphism*, are frequently employed in order to avoid detection. Both strategies entail changing the logic of the code, with the former causing it to act differently in certain situations [101] and the latter rewrites and outputs a logically equivalent version of itself during runtime [132]. The purpose is to prevent detection by traditional anti-virus software and not to be identified by manual and automated code inspections, and analysis [70]. Collaboration is crucial to the information security community, and it appeared necessary to develop a common taxonomy systematically categorising adversary behaviour. It would enable researchers and analysts to share experiences, document and improve the detection of malicious activity [116]. MITRE developed *ATT&CK* in response to the need to implement a structured categorisation and to model adversary behaviour using a common knowledge base.

2.2.1 Background and history

In 2010, MITRE started the FMX (Fort Meade eXperiment) research project, which focuses on conducting structured and systematic adversary emulation exercises inside a “living lab”. MITRE began analysing data sources and analytic procedures within FMX in order to detect *Advanced Persistent Threats* (APT). They studied whether APT could be detected faster using a “assume breach” mentality, which assumes the adversary is already present within the system [116]. The purpose of FMX was to investigate the use of endpoint telemetry data and analytics to improve post-compromise detection of adversaries operating in an enterprise network. The MITRE ATT&CK framework served as the foundation for testing the effect of sensors and analytics under FMX [116]. ATT&CK acted as the standard language for both the offence and the defence to improve over time [34]. Categorising observed behaviour across relevant real-world adversary groups was proven to be effective. Additionally, the information was valuable in conducting controlled exercises to emulate adversaries within the FMX environment.

In September 2013, the first ATT&CK model was created. It primarily targeted Windows Enterprise environments. The framework was used to document commonly used tactics, techniques and procedures (TTPs) by different APT groups [34]. ATT&CK kept adjusting through internal research and development until its release in May 2015. The framework comprised 96 techniques organised under nine tactics at the time [116].

With the support and contributions of the cybersecurity community, ATT&CK has grown tremendously. In 2017, the framework was updated to include macOS and Linux platforms, and it became known as *ATT&CK for Enterprise*. It covers the techniques implemented throughout the attack life cycle on enterprise platforms [116].

In 2017, MITRE released ATT&CK for Mobile, which focus on mobile domains [116]. Attackers use the tactics and techniques described in the matrix to gain access to the device or network-based effects on Android and iOS systems [42]. Because of their varied targets, we now distinguish between the two sections *ATT&CK for Enterprise* and *ATT&CK for Mobile*.

In addition to operating system (OS) domains, ATT&CK for Enterprise now contains matrices for cloud environments, and container technologies [40]. Some cloud environments are Amazon Web Services (AWS), Microsoft Azure and Microsoft Office 365. Malware and network access are common objectives with OS environments; cloud attacks focus on leveraging native features in order to access the target victim’s account, elevate privileges, move laterally, and exfiltrate data [85].

In 2020, *ATT&CK for ICS* was published, documenting adversary behaviour against Industrial Control Systems (ICS) [116]. There was a need to analyse effectively, understand, collect and share knowledge about adversary behaviour in ICS environments [1]. Both ATT&CK for ICS and ATT&CK

for Mobile are separate matrices available on the ATT&CK web page [36].

ATT&CK has a mapping of groups, which tracks known adversary groups, more specifically APTs. Information on these groups are gathered from intelligence reports from public and private organisations [116]. Information about each group can be accessed, including various names used to describe the group in intelligence reports, as well as observed campaigns and techniques [47]. The mapping of groups and techniques are primarily focused on APT, but it may also include other advanced groups such as financially motivated actors [116].

MITRE updates the ATT&CK matrix twice a year, and the most recent version at the time of writing is v11, released on April 25th, 2022. This version added new techniques, groups and software for Enterprise, Mobile and ATT&CK matrices. ATT&CK for Enterprise was extended with 12 new techniques, examples including Wordlist scanning, DHCP spoofing and debugger evasion [47].

2.2.2 Use Cases

ATT&CK may be a resource when developing threat models, and methodologies in the private sector, government, cybersecurity and service community [35]. ATT&CK describes how attackers can penetrate networks and move laterally, elevate privileges and circumvent defences from the perspective of an adversary [38].

Applications for ATT&CK include assisting cyber defenders in developing analytics that detect adversary techniques or providing analysts and red-team groups with a common language throughout their respective areas. ATT&CK allows analysts to more efficiently structure, compare and analyse threat intelligence. At the same time, red-team groups benefit from having a shared vocabulary as a foundation for emulating specific threats and planning operations. ATT&CK can also be used to assess the capabilities of an organisation and drive engineering decisions, e.g. what tools to use or what kind of logging to implement [35].

2.2.3 The ATT&CK Model

The description of actions that adversaries can take to accomplish *objectives*, as expressed in a *behavioural model*, is the foundation of ATT&CK [116]. It is made up by the following components:

- **Tactics**, denoting short-term, tactical adversary goals during an attack. All tactics have an identifier on form TXXXXX, e.g. TA0002.
- **Techniques**, describing the means of how adversaries achieve tactical goals. All techniques have an identifier on form TXXXX, e.g. T1006.
- **Sub-techniques**, describing in more detail the specific means in which adversaries achieve tactical goals. All sub-techniques have an identifier on the form TXXXX.YYY, e.g. T1564.001.

Reconnaissance 10 techniques	Resource Development 7 techniques	Initial Access 9 techniques	Execution 12 techniques	Persistence 19 techniques	Privilege Escalation 13 techniques
Active Scanning (2)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (4)	Abuse Elevation Control Mechanism (4)
Gather Victim Host Information (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs	Access Token Manipulation (5)
Gather Victim Identity Information (3)	Compromise Infrastructure (6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (15)	Boot or Logon Autostart Execution (15)
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)
Gather Victim Org Information (4)	Establish Accounts (2)	Phishing (3)	Inter-Process Communication (2)	Browser Extensions	Create or Modify System Process (4)
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary	Domain Policy Modification (2)
Search Closed Sources (2)	Stage Capabilities (5)	Supply Chain Compromise (3)	Scheduled Task/Job (6)	Create Account (3)	Escape to Host
Search Open Technical Databases (5)		Trusted Relationship	Shared Modules	Create or Modify System Process (4)	Event Triggered Execution (15)
Search Open Websites/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)	Exploitation for Privilege Escalation
Search Victim-Owned Websites			System Services (2)	External Remote Services	Hijack Execution Flow (11)
			User Execution (3)	Hijack Execution Flow (11)	Process Injection (11)
			Windows Management Instrumentation	Implant Internal Image	Scheduled Task/Job (6)
				Modify Authentication Process (4)	Valid Accounts (4)

Figure 2.1: Part of ATT&CK Matrix for Enterprise, v10.1.

- Documented adversary usage of techniques, mapping to which adversary groups are known to deploy the techniques, platforms and other metadata.

The ATT&CK Matrix shown in Figure 2.1 depicts the relationship between these components. The figure displays a section of the Enterprise Matrix that details six tactics and 64 techniques that an adversary might perform to compromise and operate within an enterprise network. As previously mentioned, this matrix is applicable for the platforms Windows, macOS, Linux, AWS (Amazon Web Services), Microsoft Azure and Office 365, to name a few.

The ATT&CK version shown in Figure 2.1 is v10.01, captured March 20th, 2022¹. The figure provides an overview of the TTPs, with tactics represented as the heading above each column, their respective techniques in columns below, and a grey space to the right of the column cell where sub-techniques are available. In total, ATT&CK contains 222 techniques grouped into 14 different tactics.

Figure 2.2 show eight of the first techniques for the tactic *Initial Access* (TA0001). The tactic contains the techniques *Drive-by Compromise*, *Exploit Public-Facing Application*, *External Remote Services*, *Hardware Additions*,

¹Unless otherwise noted, all matrix images are from this version.

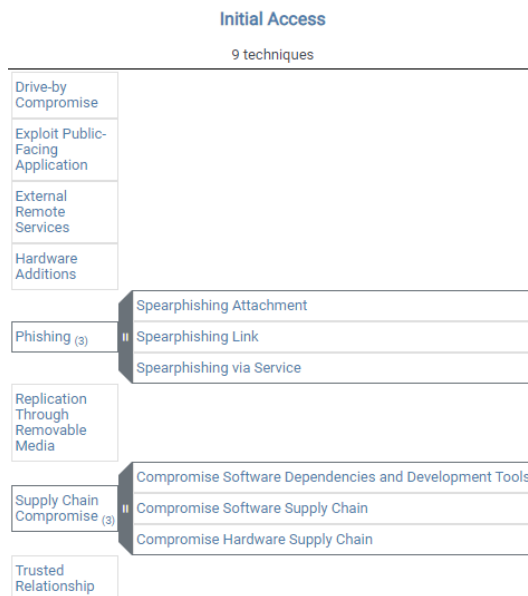


Figure 2.2: Expanded the tactic "Initial Access", ATT&CK v10.1.

Phishing, Replication Through Removable Media, Supply Chain Compromise and Trusted Relationship. There are no sub-techniques in the first four techniques. The Phishing technique (T1566) contains three sub-techniques, each of which details various phishing methods that adversaries are known to employ. For example, attackers have performed phishing attacks via spearphishing attachments, links or spearphishing via Service to gain Initial Access to an enterprise system.

2.3 Advanced Persistent Threats

We are faced with new attacks and malware daily, and the trend has seen to shift more towards a stealthier behaviour over the last decades [2]. Slow and low movements characterise this new class of attacks, which usually has the goal of exfiltrating or stealing target data while remaining undetected [2]. This class has been given the term *Advanced Persistent Threat*, shortened APT. These are organised threat groups and often well-funded by organisations or national governments that have a goal of attaining information about target organisations or governments [2]. National Institute of Standards and Technology (NIST) describes APT as the following:

"An adversary with sophisticated levels of expertise and significant resources, allowing it through the use of multiple different attack vectors (e.g., cyber, physical, and deception) to generate opportunities to achieve its objectives, which are typically to establish and extend footholds within the information technology

infrastructure of organisations for purposes of continually exfiltrating information and/or to undermine or impede critical aspects of a mission, program, or organisation, or place itself in a position to do so in the future; moreover, the advanced persistent threat pursues its objectives repeatedly over an extended period of time, adapting to a defender's efforts to resist it, and with determination to maintain the level of interaction needed to execute its objectives." [97]

Myneni et al. [90] provide an additional definition by describing the phrases contained in the APT term: *Advanced* describe the threat actors' capabilities in terms of attack tools, expertise and attack methods, which often are customised to the target and organised into multiple stages. *Persistent* represent the actor's determination to achieve the attack objective, which usually involves the use of evasive techniques to avoid detection. *Threat* stands for the potential loss of sensitive data or mission-crucial component that the actor represents to the target organisation [90].

Well-funded APT groups create sophisticated, customised tools such as new types of malware that are not usually detected by signature-based anti-virus software or IDS and IPS [2]. In order to obtain access to an organisation's network, this targeted malware is typically distributed via (spear) phishing emails. Once the malware has reached the target network, it can exploit vulnerabilities that the group has discovered before infiltrating the network [2].

We differentiate between APT attacks and traditional types of cyberattacks [2]. Traditional attacks are usually unspecified and mostly target single systems, whereas APT attacks target specific organisations, governmental institutions, and commercial enterprises [23]. Furthermore, traditional and APT attack approaches differ, with traditional attacks being noisier and are generally accomplished in a single run. In contrast, APT attacks consist of repeated attempts, long term persistence, and adapts to remain undiscovered [23][2]. Targeted attacks are often easier to prevent, whereas APT attacks require defenders to change their detection systems or methods since APT uses methods that they have never encountered [23].

APT attacks are well planned and custom-built to increase the probability of a successful attack. In order to achieve this, they are often performed in multiple stages, shown in Figure 2.3.

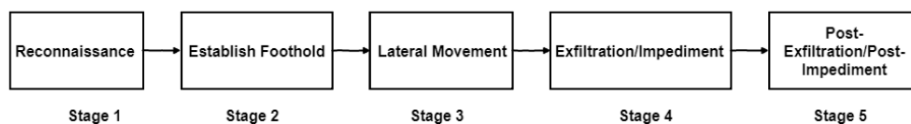


Figure 2.3: APT attack steps [2].

The *Reconnaissance* phase marks the initial step of any successful attack. This phase is important for the attackers, as they gather the necessary in-

formation before later launching an attack. The group studies the target organisation, collecting as much information about the technical environment and relevant personnel. This information is often gathered via *open-source intelligence* (OSINT), and social engineering techniques [23].

The *Establish Foothold* phase represents the attackers' accomplished entry into their target's computer or network [2]. The typical way for establishing a foothold on the target computer is to execute malicious code that exploits a vulnerability, but obtaining access through social engineering is also an observed technique [23, 114]. When the exploit successfully executes, the intrusion is complete, and they can work further towards their end goal [23][2].

Lateral movement refers to the phase when the attackers start to move inside the network to expand their control over the targeted organisation and discover and collect valuable data [23]. This phase often takes place over a more extended period to collect as much information as possible. Lateral movement usually involves "*internal reconnaissance, compromising additional systems in order to harvest credentials and gain escalated privileges, identifying, and collecting valuable digital assets*" [23]. The activities are planned to operate quietly and slowly to avoid detection. Furthermore, APT actors may use the same tools as IT administrators, making them more challenging to detect as they move deeper into the network – their activities could go undetected or even untraced [23].

Exfiltration / Impediment indicate the phase in which the adversary has reached their goal of obtaining organisational data and proceeds to transmit it to their C&C (Command&Control) centre. If the attackers' goal is to sabotage components, disabling or destroying the critical components is part of this attack phase [2]. Some literature, including the ATT&CK matrix, split Exfiltration and C&C into separate steps [36].

The fifth and final phase is *Post-Exfiltration / Post-Impediment*, representing activities such as continuing to exfiltrate, disabling additional critical components, or deleting evidence in order to have a clean exit from the network [2]. This usually involves transporting the stolen data to an internal staging server, compressing or encrypting it before sending it to the attackers' external locations. Traffic from post-exfiltration actions is often hidden using SSL/TLS or leverages the features of the Tor network [23].

The attack steps described are comparable to the attack steps in Lockheed Martin Cyber Kill Chain² identifies. The Cyber Kill Chain is a component of the Intelligence Driven Defence model for identifying and preventing cyber-attacks [28]. The model describes the steps that the attacker must complete to achieve their goal, summarily to Figure 2.3. The steps included in Cyber Kill Chain are *reconnaissance, weaponisation, delivery, exploitation, installation, C&C* and *actions on objectives*. The reconnaissance and weaponisation steps are similar to Stage 1 Reconnaissance in Figure 2.3. These steps include gathering information about the target organisation and preparing a payload of malware and exploit(s) [28]. Delivery and Installation in Cyber Kill Chain are included in Stage 2 Establish foothold in 2.3, referring

²<https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>

to delivery and successful installation of the malware, and hence foothold on the target. C&C is a separate step in the Cyber Kill Chain, an example of literature or references that divide C&C and Exfiltration as two steps. Stage 5 in Figure 2.3 is similar to the final step in Cyber Kill Chain. However, in this final step of the Cyber Kill Chain, lateral movement and data exfiltration are included [28]. Figure 2.3 differentiate between these two steps. There are various methods and models to categorise the steps of an APT attack, and these were two comparable examples.

APTs are sophisticated, specific and evolving threats, indicating that traditional countermeasures are necessary, though not sufficient, to protect against APT [23]. Organisations that have been victims of APT attacks have suffered significant, if not irreversible, damage [2]. The speed at which the attack tools and techniques evolve requires solutions that adapt to the changing behaviour of APTs. Defenders must understand the stages and strategies involved in the attacks and develop new capabilities that address the threat actors' specifications, even if distinct patterns can be recognised.

2.3.1 Adversary emulation plans

Emulating adversaries provides an approach to assessing a network's resilience against advanced attackers, but it is often a manual, time-consuming and complex process [4]. While some security assessment tools, such as Metasploit³ can automate specific red team tasks and hence reduce total time cost, automated adversary emulation (EMU) aims to extend such emulation to the tactical level [4].

MITRE Engenuity's Center for Threat-Informed Defense (CENTER)⁴ introduced the first *EMU plan* in 2017. Each EMU plan focuses on a single APT and provides a curated summary that captures and describes their publicly attributed breaches and campaigns based on intelligence reports and other artefacts [10]. Each EMU plan includes an operational flow that gives a high-level description of specific threat scenarios observed by the APT group. The material in the EMU plan is broken down into step-by-step procedures that defenders can run from start to finish or as individual tests [120]. Organisations can further adapt the scenarios and behaviours within each plan to better fit their environment and priorities [121].

The initial EMU plan focused on APT3, signalling the start of CENTER's mission to improve global threat-informed practice [56]. Since then, their methodology has involved capturing an adversary's publicly attributed techniques, chaining them together into a logical series of actions inspired by the attacker's past behaviour [10]. FIN6 was the first group in a public library of new EMU plans released by CENTER in September 2020, followed by APT29 a few months later. According to Jon Baker at CENTER, the plans require substantial time, expertise, and effort to build [10]. This process includes:

- cyber threat intelligence (CTI) research;

³<https://www.metasploit.com/>

⁴<https://ctid.mitre-engenuity.org/>

- custom tool development where needed;
- TTP analysis;
- ATT&CK mapping;
- test range setup;
- emulation plan development;
- testing and final quality review [10].

In addition, CENTER focused not only on *what* the attackers do, but also on *how* and *when* they do it. In other words, “*how does this threat group obtain credentials from victims, which tools do they use, and at what stage of a breach does it happen?*” [10]. The team at CENTER identified some common features for a plan template based on the APT29 plan. The plans needed to be easily updatable, follow a set of CTI-informed scenarios and offer a human-readable, command-by-command version for organisations and teams to follow the implementation. Finally, it must have a machine-readable form in order to enable the key feature of automated execution or parsing [10]. CENTER published the EMU plan for menuPass in February 2021 and plans for FIN7 and Carbanak in April 2021 [120]. New plans will be released in the future, as they have successfully defined a common methodology with a specific point. Organisations will reduce overall costs because the EMU plans are consistent and easily accessible [10].

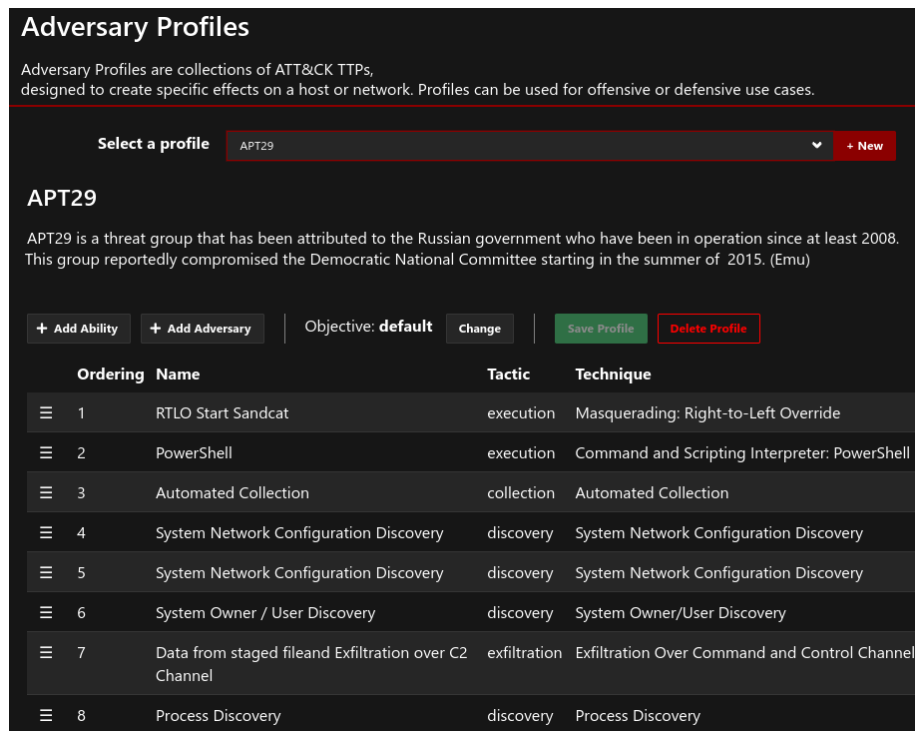
The first set of EMU plans was introduced to CALDERA in February 2021 [29]. At the time, they included APT29 and FIN6, but as CENTER has continued to improve, EMU plans for menuPass, Carbanak, and FIN7 are now available. Figure 2.4 shows the EMU plan for APT29 in CALDERA, first presenting the group with a description from ATT&CK [37]. Next, the steps in the EMU plan are listed, with an overview of execution order, names, tactics and techniques. Intelligence reports on APT29 are used to select and order attacks [10]. Only the first 8 steps of the EMU plan are shown in Figure 2.4, which involve execution, collection, exfiltration and discovery tactics. The APT29 EMU plan has 79 attack steps in total.

2.4 MITRE CALDERA™

Red teams play a critical part in assessing the security of a system or network, and their importance cannot be underestimated [5]. These red teams are usually a group of security specialists who emulate attackers to test all aspects of an organisation’s security posture [5]. Red teaming requires staff training, experience and design. It is both costly and time-consuming to implement.

To address these challenges, MITRE began a research initiative to create the framework CALDERA⁵ [45]. It is a fully automated red teaming system with adversary emulation capabilities, focusing on post-compromise

⁵<https://caldera.readthedocs.io/en/latest/>



Adversary Profiles

Adversary Profiles are collections of ATT&CK TTPs, designed to create specific effects on a host or network. Profiles can be used for offensive or defensive use cases.

Select a profile: APT29 + New

APT29

APT29 is a threat group that has been attributed to the Russian government who have been in operation since at least 2008. This group reportedly compromised the Democratic National Committee starting in the summer of 2015. (Emu)

+ Add Ability | + Add Adversary | Objective: default | Change | Save Profile | Delete Profile

Ordering	Name	Tactic	Technique
1	RTLO Start Sandcat	execution	Masquerading: Right-to-Left Override
2	PowerShell	execution	Command and Scripting Interpreter: PowerShell
3	Automated Collection	collection	Automated Collection
4	System Network Configuration Discovery	discovery	System Network Configuration Discovery
5	System Network Configuration Discovery	discovery	System Network Configuration Discovery
6	System Owner / User Discovery	discovery	System Owner/User Discovery
7	Data from staged file and Exfiltration over C2 Channel	exfiltration	Exfiltration Over Command and Control Channel
8	Process Discovery	discovery	Process Discovery

Figure 2.4: Adversary Emulation Plan for APT29 in CALDERA.

activities. CALDERA acts as an automated and intelligent red team, probing the target network for weaknesses and training defenders [5].

One of the essential aspects of CALDERA regarding this thesis is that it leverages ATT&CK TTPs to drive atomic activities, which means it comprises tactics and techniques that map to ATT&CK [5]. CALDERA can be customised to add new features or enable various *plugins* to extend the framework with functionality, such as simulating human behaviour, adding encrypted traffic with HTTPS, or running adversary emulation (EMU) plans [45]. As previously stated, the EMU plans focus on individual APT groups addressed in ATT&CK.

CALDERA itself is available to run on Linux and macOS operating systems. Agents can be deployed on computers running Windows, Linux and macOS, working as targeted or infected hosts. CALDERA requires Python v3.7+ (with pip3), GoLang 1.17+, Google Chrome browser, 8 GB RAM and 2 CPUs. MITRE has provided documentation for the installation process and how to get started using CALDERA on its website [45].

2.4.1 Architecture and example usage

Virtual Red Team System (ViRTS) and *Logic for Advanced Virtual Adversaries (LAVA)* are the two main components that comprise CALDERA. ViRTS

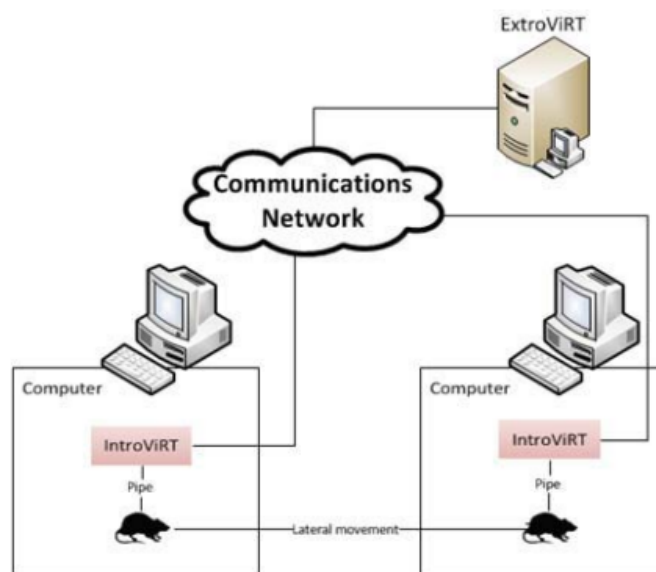


Figure 2.5: Visualisation of the CALDERA infrastructure [5].

is the software infrastructure used to create and emulate a red team adversary, and LAVA is the logical model used by CALDERA to decide which actions to take. The CALDERA infrastructure is simplified and illustrated in Figure 2.5, taken from [5].

The ViRTS infrastructure, referred to as the main framework, is made up of two parts:

1. *The core system*, which is the framework code that includes a Command&Control (C2/C&C) server with a REST API and a web interface. It consists of the *master server* (ExtroViRTS) and the *remote access tool* (RAT) clients (IntroViRTS), which are running on already infected hosts (IntroViRTS) [5]. These components are shown in the architecture in Figure 2.5.
2. *Plugins* that, among other things, provide additional functionality for collections of TTPs, agents and graphical user interfaces (GUI) [45, 5].

CALDERA is typically used to run offensive (red) or defensive (blue) operations. The master server (ExtroViRTS) is the C&C server with a web interface on Linux and macOS platforms. The RAT client (IntroViRTS) is deployed on the target host, now referred to as an agent. The infected host or target system establishes contact between its running RAT and the attacker's master C&C server [45]. The deployment of CALDERA and its usage can be summarised in the following steps [45]:

- Start the C&C server on the attacker platform;
- Deploy agent on the host to infect by launching the RAT;

- Choose an adversary profile that is currently in CALDERA or create a new one;
- Run an operation (attack) on the infected machine by either selecting an adversary profile or manually adding commands to execute;
- Examine the operation as it progresses, reviewing successful, failed or timed out attacks;
- Export the operation results to a CALDERA report that includes details about the operation, such as the ATT&CK techniques used in the attacks, the name of the infected host, the commands used in the attacks, and the attack times.

The RAT continuously reports relevant information about the attacks to the master server, which updates its internal knowledge base [5]. The LAVA engine runs on the master server and uses the knowledge to determine what actions CALDERA should take and which commands to execute on the agent [5].

2.4.2 CALDERA terminology

CALDERA uses various terminology to describe its functioning. The following list is taken from the CALDERA documentation [45][5]:

- **Agents:** software application that connects back to CALDERA at regular intervals to obtain instructions. Agent communicates with the CALDERA server, and each agent is assigned a unique ID called *paw*. **Sandcat** (54ndc47), **Manx** and **Ragdoll** are the three agent applications included in CALDERA.
- **Groups:** for assembling agents and executing commands in groups. The group determines if an agent is a "red agent" (offensive) or "blue agent" (defensive).
- **Abilities:** a particular ATT&CK tactic/technique to execute on running agents. Each ability includes the command(s) to run, the platform/executors for the command, payloads and a reference to output results on the CALDERA server.
- **Operations:** run abilities on agent groups.
- **Adversary profiles:** pre-defined threat actors that consist of a group of abilities (TTPs). Profiles may help determine which abilities should execute when conducting an operation.
- **Planner:** specifies which order to run which abilities. The default configuration is the *Atomic* planner, running abilities in their atomic ordering. *Batch* planner runs all abilities in the adversary profile once. *Buckets* planner runs all abilities, grouped by ATT&CK tactics, in the adversary profile.

- **Link:** a link is created for each agent when an operation is running. Certain conditions should match to create a link; most importantly, all link *facts* and *fact requirements* should be fulfilled. Depending on the stealth settings, link commands may be obfuscated. The operation creates a link *chain*, which holds all links generated for the operation.
- **Fact:** an identifiable piece of information about a specific computer.
- **Plugins:** extensions to CALDERA that introduce new capabilities, e.g. the ability to mimic human behaviour on agents, adding simulated agents, autonomous incident response and adding HTTPS.
- **ViRTS:** the software infrastructure used to create and emulate a red team adversary.
- **LAVA:** the logical model that CALDERA uses to decide which actions to perform.

Chapter 3

Background

This chapter presents concepts and approaches that the rest of the thesis builds on. The principles of simulation, network traffic capture, and labelling are defined. Building on this material, one section will provide a more in-depth review of the challenges that the cybersecurity community experiences in labelling datasets. The final sections will describe the solutions formed to address these issues.

3.1 Simulation

A simulation is the imitation of an operation; in technical terms, it uses a *model* to represent key characteristics or behaviours of the selected system or process the imitate [11, 137]. The simulation then represents the model's progression over time. Simulation is used in various situations, and computers are widely employed to execute simulations [137]. Simulation can be used to observe the model repeatedly, for example, to determine the real-world consequences of specific conditions and actions [112]. Additionally, simulations are often utilised when the real system can be damaged, which is often the case when generating network datasets [137] [77]. Simulations are widely used in dataset generation to emulate specific tasks or activities, such as simulating attacks, and typical behaviour [26][62].

3.2 Capturing network traffic

In a network, computers communicate by exchanging network packets that are composed of control information (the packet header) and the user information (the payload) [26]. The payload, which might be encrypted, transports data on behalf of an application. The header holds information such as the transmission protocol layer, IP addresses, and so on that is essential for proper packet transmission [26]. A *packet analyser* or a *packet sniffer* monitors network traffic by capturing packets as they traverse the network [124]. This process of collecting and logging traffic is called *packet capture*, and the API often used for this process is called *pcap* [135]. A capture file

is usually stored using the file extension `.pcap`, a format that packet analysers such as Wireshark¹ and tcpdump² can read [136, 26]. *IPFIX* is another common format for displaying or storing network logs or information, which is based on connection summaries or flow statistics [26]. A network flow is defined by RFC 3697 [18] as a sequence of packets with the same source and destination IP addresses (IP protocol). The sequences for TCP and UDP connections are summarised using the same source and destination ports [26]. This data is typically used to represent a network flow, along with additional information such as the start and end time of the connections, as well as the total number of packets and bytes transmitted [26].

3.3 Labelling

In principle, labelling is the process of providing context to data by adding one or more meaningful and informative tags [13, 62]. The definition of a label varies per application, and there is no general agreement on what constitutes a "correct" label [24]. *Label granularity* or how elaborate a label should be, is not precisely defined, but several authors agree on some common characteristics [62]. Almost every labelling process, regardless of the application of the final dataset, has two goals: accuracy and speed, i.e., effectiveness and efficiency [13].

In the context of anomaly detection, the datasets should provide all associated behavioural patterns for malicious and normal network traces, which is referred to as *representative labels* [62]. When labelling network traces for regular users, representativeness is crucial because timing patterns, frequency of use and work cycle must all be accurately represented in the dataset [62]. Representative labels for malicious network traces could include the chain of misuse actions or timing patterns of malicious behaviour. On the other hand, only those portions of a network trace containing the behaviour of interest should be given a precise label [62]. If a dataset is mislabelled and underrepresented, it will consequently make any model generated from it perform poorly [62]. Furthermore, assembling, cleaning and debugging large datasets can in some instances take months, which as led to an increase in the use of *weak labelling* or *weak supervision* [102][125]. Instead of gathering strongly enough labelled data, this approach programmatically generates weakly labelled data, proving effective when the task at hand is to label large datasets [102].

3.4 Challenges in labelling datasets

Datasets are essential in order to train and improve the algorithms used in anomaly-based network-intrusion detection systems (NIDS) [104]. They are also crucial for evaluating and comparing the performance of the IDS

¹<https://www.wireshark.org/>

²<https://www.tcpdump.org/>

[123]. For NIDS, machine learning (ML) algorithms could leverage *labelled* or annotated datasets, i.e. traffic traces with known and tagged anomalies and incidents [113]. The number of detected attacks or false alarms may act as an evaluation criterion, given a labelled dataset in which each data point is assigned to the class 'normal' or 'attack' [104]. Quality dataset labelling has emerged as a fundamental aspect in the application of ML models in NIDS in recent years due to their generalisation capabilities, and increased computational power [21][20]. The network security field has focused on the development of NIDS based on ML with the promising goal of achieving better detection performance [21]. Still, the main issue with developing better detection potentials in NIDS is the lack of labelled datasets. There are few publicly available datasets, and the quality of the available ones varies. Although there is no precise definition of what a quality dataset is [62], several authors agree that representative and accurate labels are the two main aspects for determining the quality of a labelled network traffic dataset [26][107][83]. A representative labelled dataset should provide all the relevant behavioural patterns for malicious and normal network traces. This is particularly important when labelling network traces from regular users, which should include time patterns, frequency of use, and work cycle [62].

Another challenge with labelling datasets originates from the growing use of encrypted traffic. The cryptographic protocol *TLS*, or *Transport Layer Security*, currently protects the majority of internet traffic. Malware authors have followed this trend to hide malicious network connections and evade detection [51] [25]. According to Cisco, more than 70% of malware campaigns in 2020 used some sort of encryption to conceal malware, C&C activity or data exfiltration [25]. Encryption poses some challenges to traditional means of detection, as the data cannot be inspected, and it is not feasible to decrypt it [25]. de Lucia et al. [51] propose that ML traffic analysis should be improved to avoid relying on pattern matching or the payload content for detecting malicious or suspicious communications.

In recent years, APT groups have become a growing concern in the security research field [2]. They are highly resourceful, state-sponsored adversary groups. The most common method of detecting APT activity is anomaly-based detection, which relies on datasets with realistic attack scenarios [2]. For a detection system to learn both abnormal and normal behaviour in order to detect APT, supervised ML employing labelled datasets, or semi-supervised learning, is required [2]. Myneni et al. [90] created a dataset, DAPT2020, that captures the various aspects of real-world APT attacks, including the main phases of an APT attack; reconnaissance, foothold establishment, lateral movement and data exfiltration. Aside from DAPT2020, several authors underline that there are currently no publicly available datasets containing APT datasets [114, 90]. Datasets are critical in the development of ML models capable of detecting sophisticated attacks such as APT [90]. Due to the complexity of the groups, detecting APT is challenging, and modelling and creating realistic datasets for training and evaluation has become a significant challenge [114].

Because of the challenges presented, there only exist four publicly available structured datasets for NIDS that incorporate real-world traffic and attacks [26]. Guerra et al. [62] conducted an in-depth analysis of the present challenges present in labelling network traffic. They concluded that all current labelling approaches have fundamental flaws in terms of quality, volume and speed of the generated dataset [62]. Clausen et al. mentioned similar difficulties [26]. Guerra et al. further suggested that the network security community should address the lack of consistent methods for continuously developing representative datasets, using an accurate and validated methodology [62].

3.5 Current solutions

The growing complexity of network systems, software, and services, as well as their rising integration and dependencies, has resulted in new types of cyberattacks [111]. Malicious behaviour is rapidly evolving, with new and "better" malware and attacks being introduced every day [2]. Thus, anomaly detection and IDS require datasets that are up to date, representative of current trends and attacks, and have low maintenance and modification costs [62][77]. There currently exists no such perfect dataset, but multiple studies have proposed methods for improving the challenges described in section 3.4. This section will include a few suggested approaches.

In the past years, the practice has been to use the few available datasets as benchmarks [26, 75, 118]. Some of the datasets are over a decade old. According to Hindy et al. [68], more than 85% of the published approaches use the DARPA1998 dataset from over 20 years ago [114]. The lack of suitable datasets constitutes one of the biggest challenges for anomaly-based intrusion detection [104, 27].

From the standpoint of creation, datasets are divided into three categories: *realistic*, *synthetic* and *semi-synthetic* [114]. *Realistic datasets* from real-world situations may be the preferred option. However, this raises concerns about user privacy, poor scalability, potential harm from attack simulation in a production environment and lack of labels [114]. The use of *synthetic data* has the advantage of data control and network setup, as well as avoiding the issues of managing unknown properties and false alarms caused by real network noise [114]. However, there are several limitations, such as a lack of realism, where an absence of noise simplifies the simulated attacks, potentially leading to unrealistically good detection results [26, 114]. *Semi-synthetic data* is a combination of the two previous categories. Drawbacks include failure of detection methods in realistic situations and biased datasets due to following an insufficiently accurate synthetic user model [114].

Several authors agree that simply having a dataset available is insufficient to evaluate IDSs; a ground truth that defines harmful behaviour in a quantifiable way is also required [77, 26, 6]. To address this, Landauer et

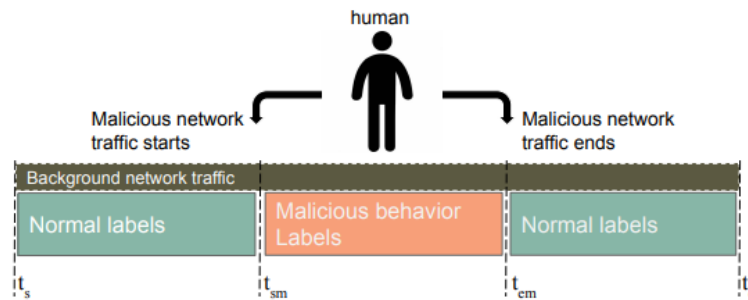


Figure 3.1: Injection Timing, a widely deployed labelling strategy [62].

al. [77] outline a few potential labelling strategies to achieve ground truth. The simplest yet most effective method is to label all traffic generated during attack time intervals as malicious [77]. Guerra et al. refers to this method as *Injection Timing* [62], with a descriptive illustration shown in Figure 3.1. The simulation runs isolated from a production environment so that no unknown processes can influence the events. Thus, one can consider all the events that occur outside of the malicious time interval benign. It is possible to create a labelled dataset with various network traffic behaviours because of the controlled environment, which adds authenticity to the generated dataset [62]. Anomalous time frames can be derived from attack scenario descriptions, making this method easy to implement. However, if normal events occur during attacks, they will also be labelled as malicious [77]. In order to acquire correct labels when using Injection Timing, a strict time control mechanism is required [62]. Garcia et al. [60] and Bhuyan et al. [14] used the Injection Timing as a labelling method in their work.

Behavioural profiles is another approach for automatically labelling network traffic [62]. Behavioural profiles provide the information to simulate a specific feature or aspect of the network, often implemented as computer programs that perform regular activities. The labelling process using this technique is straightforward; all traffic generated by a profile emulating normal traffic will be labelled as such, and similarly, with malicious behaviour [62].

A third method introduced by Guerra et al. [62] bases the labels on information provided by *network security tools* (NST). This strategy was applied to create the DARPA datasets in 1998-99 [87] and the KDD99 dataset [127], which has been used as a benchmark dataset in numerous research [27]. This method relies on network security tools such as sniffers, honeypots or NIDS. The DARPA and KDD99 datasets used sniffers to capture network traffic [62].

The methodologies presented so far are all forms of *automatic labelling*, in which the datasets originate from controlled and deterministic environments. These predictable conditions make it easier to distinguish anomalous traffic from regular traffic, hence eliminating the need for expert manual labelling [62]. On the contrary, many authors see humans as an es-

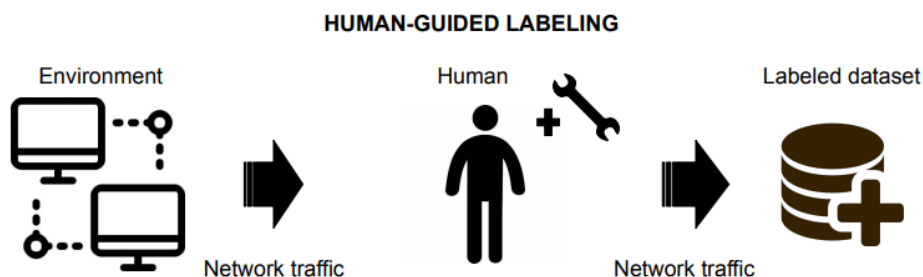


Figure 3.2: Human-guided labelling [62].

essential part of traffic analysis and the labelling process [115, 12, 58, 61]. *Human-Guided labelling* is a method in which expert users complete the work, and no user controls the network [62]. Figure 3.2 shows a simplified representation of the human-guided labelling method, where labels are the result of human domain expertise with the help of specific tools. A significant percentage of network analysis today is performed manually. However, many of these extensive processes are not published, leaving the research community with limited knowledge about it [62].

AL is a specific case of ML, an example that supports the widely accepted view that ML is the preferred approach to labelling network datasets [26]. ML has been used to label data for use in NIDS, allowing them to advance from simple rule-based systems to intelligent automated decision-making engines powered by modern ML algorithms [55, 68].

Several studies have explored the issues raised previously regarding increased encryption in malware and malicious traffic. One example showed that many vendors of security monitor devices have chosen to decrypt traffic content first, and afterwards applying a traditional signature-based detection method [51]. This method introduces a serious breach of confidentiality, effectively disqualifying encryption for ordinary users [51]. An alternative approach is to leverage modern traffic analysis methods, which previously depended on *deep packet inspection* (DPI), *TLS fingerprinting* and the use of URLs [51]. However, this is insufficient to detect malicious communication hidden within encrypted TLS traffic.

Information such as inter-packet arrival times, flow direction, TCP headers, TLS handshake fields and frequencies can help predict the application and data protected within TLS. de Lucia and Cotton demonstrated how modern traffic analysis methods rely on combining these features with ML models [51]. They proposed a malicious communication detection mechanism using a Support Vector Machine (SVM) and one alternative using a Convolutional Neural Network (CNN). In the field of ML, SVM is a supervised learning model, and CNN is a Deep Learning algorithm [48][51]. Anderson et al. took a similar approach in [3], utilising ML to decipher TLS in malware.

Reference	Learning Approach	Anomaly Method	Detection	Source of Data	APT stages
[57]	Semi-Supervised	Machine Learning		Network/Host logs	Establish Foothold and Lateral Movement
[58]	Supervised	Machine Learning		Network Traffic	Establish Foothold
[59]	Semi-Supervised	Statistical		Host-based logs	Accomplishing Foothold
[60]	Semi-Supervised	Machine Learning		Network Traffic	Accomplishing Foothold, Lateral Movement, Exfiltration
[61]	Supervised	Machine Learning		Malware Detection	Accomplishing Foothold
[62]	Supervised	Machine Learning		Network Traffic	Lateral Movement, Exfiltration
[63]	Supervised	Machine Learning		Network Traffic	Internal Exfiltration
[64]	Supervised	Machine Learning		Emails	Reconnaissance, Establishing Foothold
[40]	Supervised, Unsupervised	Machine Learning		Host/Network events	Not specific to a stage, established behavior profiling
[65]	Unsupervised	Machine Learning		Active Directory domain service logs	Lateral Movement, Exfiltration
[66]	Supervised	Statistical		Network traffic, Access Information	Maintaining Access, Lateral Movement, Data Exfiltration

Figure 3.3: Comparison of current APT anomaly detection methods [2].

3.5.1 APT detection

When it comes to APT groups and their complexity, it is complicated to effectively model attacks, detect APT, and create benchmark datasets for training and assessing IDS [114]. Current APT detection methods are classified as *anomaly based detection* and *detection by pattern matching* [2]. Anomaly-based detection uses different ML techniques, which need classifiers trained on datasets that include the characteristic and adaptive behaviour of APT groups [2]. Figure 3.3 (taken from [2]) depicts a high-level comparison of various anomaly-based attack defence methods, along with their learning methods. The figure show that ML remains the most effective method for detecting APT activity [2], with supervised learning being the approach commonly used with network traffic.

3.6 Summary and discussion

In conclusion, there has always been a lack of suitable datasets for evaluation in the field of IDS [27]. Detection of malicious network traffic often requires extensive human labelling and expert knowledge [62]. The dataset should contain both malicious and benign ground truth labels, allowing for a more in-depth evaluation of IDS. According to Landauer et al., datasets with labels that differentiate between different sorts of attacks or attack steps will be highly valuable for NIDS [77]. This issue is particularly relevant for identifying APT, which is adaptive, intelligent, and persistent. [2]. This thesis will describe my best efforts at contributing to the field of APT detection by presenting a dataset consisting of different labelled attack steps, where the lack of suitable datasets represents a significant challenge [90] [114].

Chapter 4

Approach and implementation

This chapter covers the approach to labelling network traffic and its implementation. Illustrations and descriptions are first introduced to give an overview of the work in this thesis. A description of the network architecture follows. The emulated threat actor APT29 is introduced, along with their emulation plan (EMU) in CALDERA. The subsequent sections will provide background information on the technologies used in the experiment, DetectionLab and GHOSTS. After the introductory sections have set the foundation, this chapter presents the main work of this thesis. A comprehensive section outlining the labelling approach comes before a description of LabelGen, the script that generates datasets. This chapter ends with a concluding section.

4.1 General plan

The work in this thesis is divided into three phases; *experiment*, *processing* and *evaluation*, illustrated by Figure 4.1.

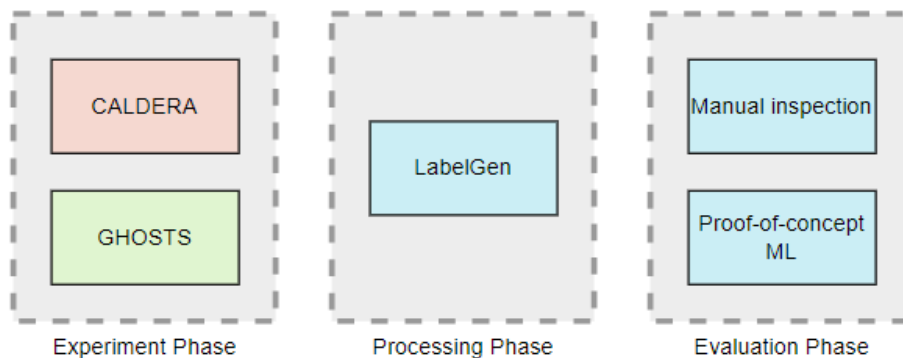


Figure 4.1: Phases of the work in this thesis.

The experiment phase covers the experiment, which involved simulating APT29 using CALDERA while generating benign traffic with the framework GHOSTS. The experiment used five virtual machines (VMs) to mimic an enterprise network. Figure 4.2 illustrates the concept of this arrangement, which DetectionLab built. Each VM has a particular function reflected by its various colours. These VMs communicate, as indicated by the wireless connection symbol and the rounded grey arrows. An additional black arrow stretches from the red VM (attacker) to the green VM (victim) to signal the CALDERA attacks.

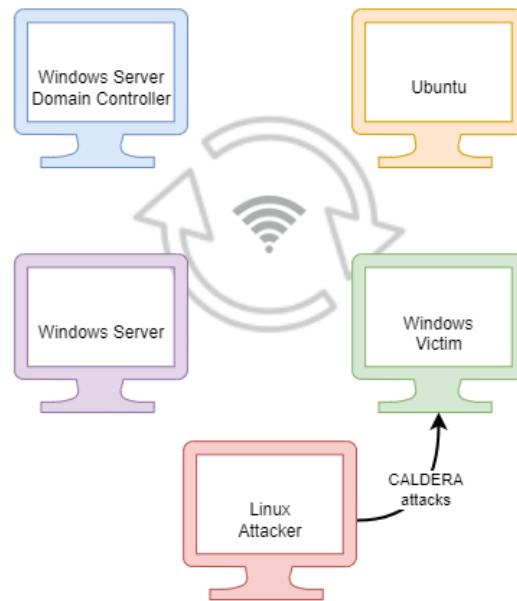


Figure 4.2: Simplified representation of the experiment environment.

The experiment generated a network logfile (.pcap) and a *CALDERA report*, which summarised the attacks. LabelGen used these two files as a basis for labelling in the processing phase. From the original logfile, LabelGen created two datasets and one labelled logfile.

The evaluation phase is discussed in Chapter 5, which will examine the LabelGen labelled logfile and train a proof-of-concept ML as an assessment.

The implementation of the general plan shown in Figure 4.1 will be covered later in this chapter. The network architecture will be explained first, followed by the tools, frameworks, and the APT29 EMU plan.

4.2 Network architecture

As previously mentioned, the experiment involved 5 VMs. They were connected to distinct networks in order to separate the various activities carried out during the experiment. Thus, three networks were created, each with different purposes. The overall network architecture is depicted in

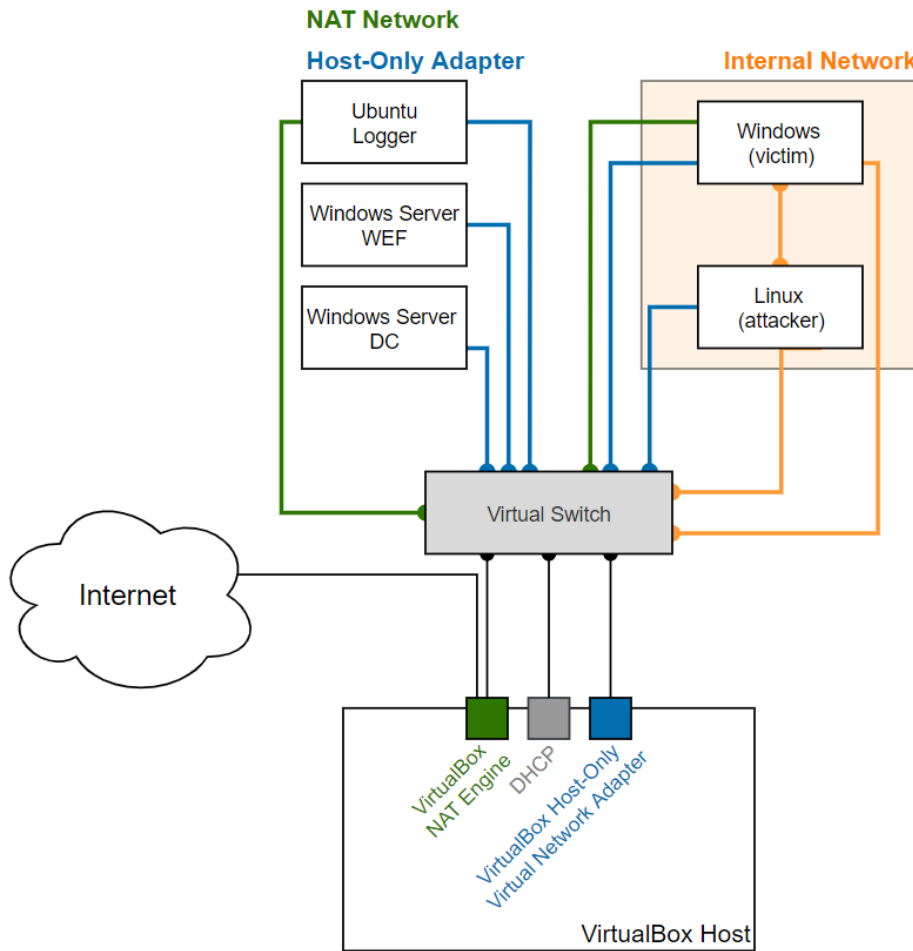


Figure 4.3: Architecture of the experiment setup.

Figure 4.3. It is based on illustrations from [16] that have been simplified and altered for this project.

Figure 4.3 shows the three networks or interfaces in distinct colours; NAT Network (green), Host-Only Adapter (blue) and Internal Network (orange). DetectionLab created the NAT Network and the Host-Only Adapter to provide a realistic enterprise environment. During the experiment, all of the VMs were connected to the Host-Only Network, as seen by the blue lines in Figure 4.3 linking to the virtual switch. The Host-Only network allows for inter-communication between the VMs. This network carries traffic from background processes such as event logs and various ARP, MDNS, and NTP requests. Because the attacker is also connected to this network, it has established a foothold in the target network. Presence in the target network is a requirement for emulating APT29. Because all the VMs on this network can interact, it implies that the attacker VM (Linux) can attack the victim VM (Windows).

The NAT Network supports outbound connections, indicating that it has Internet connectivity [30]. The NAT Engine on the VirtualBox Host has a line connecting to Internet, as seen in Figure 4.3. Only Ubuntu (Logger) and Windows (victim) were connected to this network during the experiment.

Internal networking can construct a software-based network that is only visible to the connected VMs, and not visible to the host OS or the outside world [30]. The purpose of the Internal Network was to carry the internal structure of GHOSTS. This network traffic was not captured, making the experiment more controlled. Figure 4.3 shows the internal network as an orange square beneath Windows and Linux. Only these two VMs are linked to this network and are visible to each other, as indicated by the square. An orange line linking the two devices accentuates their relation.

Network traffic from the NAT network and the Host-Only Adapter was captured during the experiment. The NAT network generated benign Internet traffic, whereas the Host-Only network generated background traffic. The attacker VM interacted with and attacked the victim VM over the Host-Only network.

4.3 APT29

APT29 is a Russian Foreign Intelligence Service (SVR)-affiliated threat group that has been active since at least 2008 [33]. Intelligence reports have also referred to the group as Cozy Bear, CozyDuke, Dark Halo, The Dukes, IRON HEMLOCK, IRON RITUAL, NobleBaron, NOBELIUM, StellarParticle, UNC2452, YTTRIUM [33]. Their attacks have often targeted government networks in Europe and NATO member countries, research institutes and think tanks [33]. The group has received allegations of substantial breaches targeting U.S. governments and organisations. Some examples are the attack the US Democratic National Committee in 2016 [119], attack on the Dutch government ministries in 2017 [71], attacks on the Norwegian government in 2017 [110], assault on Covid-19 vaccine development labs in 2020 [91] and breaching the systems of the Republican National Committee in 2021 [126]. On April 15, 2021, US and British authorities publicly announced their attribution and accusation toward APT29 and SVR as the attackers behind the SolarWinds supply chain compromise cyber operation [92][122].

The group is known for their commitment to stealth and use of sophisticated techniques [7]. Reports indicate that they have exploited zero-day vulnerabilities, a type of attack that exploits previously unknown hardware, firmware or software vulnerabilities [97]. MITRE reports that the group typically accomplish their goals using custom malware (binaries) combined with alternative execution methods such as PowerShell and WMI. Furthermore, depending on the perceived intelligence value of the target, observed attacks range from a rapid but noisy "smash-and-grab" approach

to a "slow-and-deliberate" approach more focused on persistent compromise and long-term intelligence gathering [57].

APT29 has gained initial access by exploiting public-facing applications, sending varying levels of targeted email campaigns and supply chain compromises [122, 86]. A joint analysis report between the Department of Homeland Security (DHS) and the Federal Bureau of Investigation (FBI) state that APT29 mainly focus on exfiltrating and analysing information to obtain intelligence value. They reportedly set up operational infrastructure to obfuscate their source infrastructure, host domains and malware to target organisations. They also establish C&C nodes and gather valuable information from their targets [96]. DHS and FBI also inform that the group leverages links to a malicious dropper within targeted spearphishing emails. The executed code has then delivered a Remote Access Tools (RAT) [96].

Infamous for major breaches and their characteristic sophistication has made the group an ideal object for emulation [52]. In January 2021, MITRE published the APT29 EMU plan, with public threat intelligence reporting used as the basis for mapping techniques to ATT&CK [53]. It is available as a plugin in CALDERA, which made for easy integration with both the framework and ATT&CK in this work.

4.3.1 APT29 EMU plan

The EMU plan for APT29 combines techniques that the group has been observed using across their operations, forming an *Operation Flow*. This flow is shown in Figure 4.4 from [7]. The EMU plan, and hence the operation flow, is separated into two categories [120]:

- **Scenario 1:** A scenario with a "smash-and-grab" approach for collection and exfiltration. A more general scenario in which widespread phishing attempts are used. After determining whether the target is valuable, the group deploys stealthier malware for long-term exploitation.
- **Scenario 2:** A scenario in which a "low and slow" approach is used to compromise a specific target via spearphishing. A more systematic scenario aimed at gaining control of the target and, eventually, the entire domain.

Figure 4.4 depicts these two scenarios, with red and blue lines indicating the differing approaches. The manner in which the initial compromise is made is what distinguishes these two scenarios in the figure. The operation flow is enhanced by the use of illustrative symbols.

The steps and techniques contained in the complete APT29 EMU plan are listed below, gathered from [120]. It has 20 steps with techniques mapping to ATT&CK. The steps are divided into two scenarios, which is noted in the list.

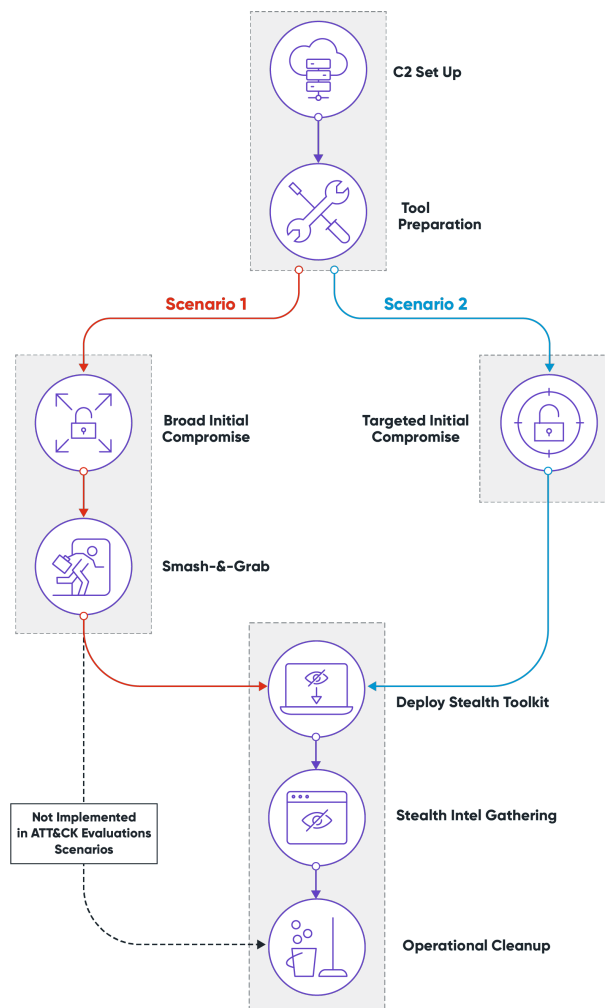


Figure 4.4: Operation flow of APT29 EMU plan [7].

Scenario 1 - "Smash-and-grab"

1. Initial Breach
 - 1.A - User Execution: Malicious File (T1204 / T1204.002)
 - 1.B - Command and Scripting Interpreter: PowerShell (T1086 / T1059.001)
2. Rapid Collection and Exfiltration
 - 2.A - Collection (T1119, T1005, T1002 / T1560.001)
 - 2.B - Exfiltration Over C2 Channel (T1041)
3. Deploy Stealth Toolkit
 - 3.A - Ingress Tool Transfer (T1105)
 - 3.B - Abuse Elevation Control Mechanism: Bypass User Access Control (T1088 / T1548.002)
 - 3.C - Modify Registry (T1112)
4. Defense Evasion and Discovery

- 4.A - Ingress Tool Transfer (T1105)
- 4.B - Indicator Removal on Host: File Deletion (T1107 / T1070.004)
- 4.C - Discovery (T1016, T1033, T1063 / T1518.001, T1069, T1082, T1083)
- 5. Persistence
 - 5.A - Create or Modify System Process: Windows Service (T1031 / T1543.003)
 - 5.B - Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder (T1060 / T1547.001)
- 6. Credential Access
 - 6.A - Credentials from Password Stores: Credentials from Web Browsers (T1003 / T1555.003)
 - 6.B - Unsecured Credentials: Private Keys (T1145 / T1552.004)
 - 6.C - OS Credential Dumping: Security Account Manager (T1003 / T1003.002)
- 7. Collection and Exfiltration
 - 7.A - User Monitoring (T1113, T1115, T1056 / T1056.001)
 - 7.B - Compression and Exfiltration (T1048, T1002, T1022 / T1560.001)
- 8. Lateral Movement
 - 8.A - Remote Services: Windows Remote Management (T1021 / T1021.006)
 - 8.B - Ingress Tool Transfer (T1105)
 - 8.C - System Services: Service Execution (T1035 / T1569.002)
- 9. Collection
 - 9.A - Ingress Tool Transfer (T1105)
 - 9.B - Collection and Exfiltration (T1005, T1041, T1002, T1022 / T1560.001)
 - 9.C - Indicator Removal on Host: File Deletion (T1107 / T1070.004)
- 10. Persistence Execution
 - 10.A - System Services: Service Execution (T1035 / T1569.002)
 - 10.B - Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder (T1060 / T1547.001)
- Scenario 2 - "low and slow"**
- 11. Initial Breach
 - 11.A - User Execution: Malicious File (T1204 / T1204.002)
- 12. Fortify Access
 - 12.A - Indicator Removal on Host: Timestamp (T1099 / T1070.006)
 - 12.B - Software Discovery: Security Software Discovery (T1063 / T1518.001)
 - 12.C - Software Discovery (T1518 / T1518.001)
- 13. Local Enumeration
 - 13.A - System Information Discovery (T1082)
 - 13.B - System Network Configuration Discovery (T1016)
 - 13.C - System Owner/User Discovery (T1033)
 - 13.D - Process Discovery (T1057)
- 14. Elevation
 - 14.A - Abuse Elevation Control Mechanism: Bypass User Access Control (T1088 / T1548.002)
 - 14.B - OS Credential Dumping: LSASS Memory (T1003 / T1003.001)

- 15. Establish Persistence
 - 15.A - Event Triggered Execution: Windows Management Instrumentation Event Subscription (T1084 / T1546.003)
- 16. Lateral Movement
 - 16.A - Remote System Discovery (T1018)
 - 16.B - System Owner/User Discovery (T1033)
 - 16.C - Remote Services: Windows Remote Management (T1028 / T1021.006)
 - 16.D - OS Credential Dumping (T1003 / T1003.001)
- 17. Collection
 - 17.A - Email Collection: Local Email Collection (T1114 / T1114.001)
 - 17.B - Data from Local System (T1005)
 - 17.C - Obfuscated Files or Information (T1027)
- 18. Exfiltration
 - 18.A - Exfiltration Over Alternative Protocol (T1048 / T1567.002)
- 19. Clean Up
 - 19.A - Indicator Removal on Host: File Deletion (T1107 / T1070.004)
 - 19.B - Indicator Removal on Host: File Deletion (T1107 / T1070.004)
 - 19.C - Indicator Removal on Host: File Deletion (T1107 / T1070.004)
- 20. Leverage Persistence
 - 20.A - Persistence Execution (T1085 / T1218.011, T1084 / T1546.003)
 - 20.B - Use Alternate Authentication Material: Pass the Ticket (T1097 / T1550.001, T1550.003)

Scenario 1 begins with noisy techniques before proceeding to a quick espionage mission for collecting data and exfiltration. In the end, the scenario transition to stealthier techniques for persistence, more data collecting, credential access, and lateral movement. The scenario ends when the previously established persistence method is implemented [120]. Scenario 2 involves compromising the initial target, establishing persistence, obtaining credentials, and then enumerating and compromising the entire domain in a stealthier and slower manner [120]. The scenario finishes by executing the previously established persistence mechanism [120].

4.4 DetectionLab

DetectionLab¹ is a tool created for offensive security practitioners to make testing, analysis, and research more convenient [81]. It is composed of scripts that automate the construction of an ActiveDirectory environment [81]. DetectionLab was used to achieve a realistic lab environment and network infrastructure in this project. The tool includes various endpoint security tools and recommended practices for logging [82]. DetectionLab is created by Chris Long, a security engineer at Material Security in the USA.

DetectionLab consists of a Windows 2016 Domain Controller (DC), a Windows 2016 server (WEF), a Windows 10 workstation (Win10) and an Ubuntu

¹<https://github.com/clong/DetectionLab>

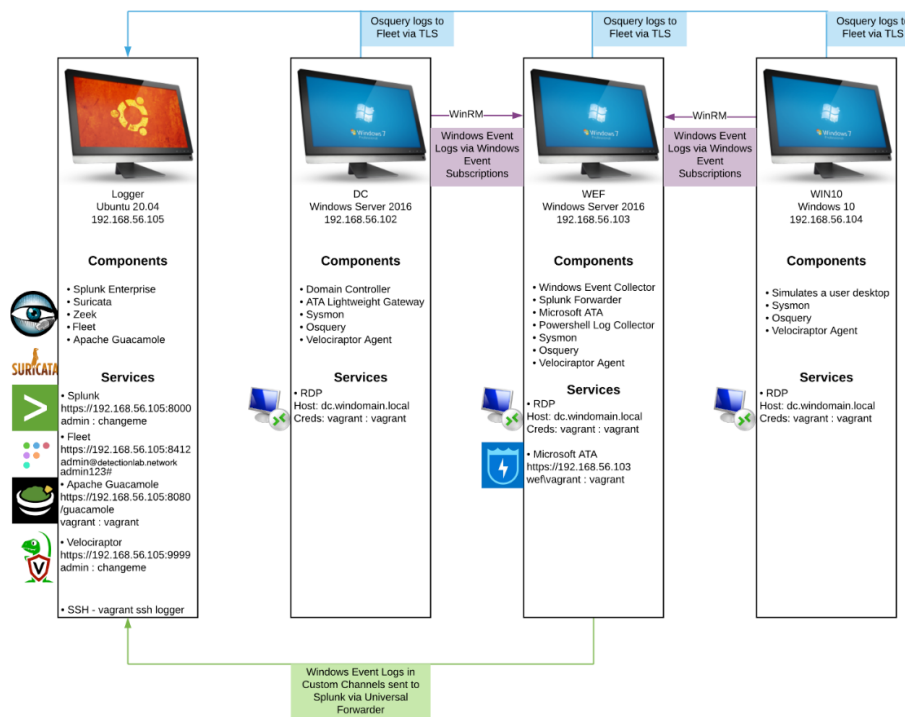


Figure 4.5: DetectionLab overview [81].

16.04 (Logger). DC has components such as Sysmon and Osquery, and it sends logs to WEF. WEF manages, amongst other things, Windows Event Logs, Microsoft Advanced Threat Analytics (ATA) and Powershell Log Collection. Win10 is a workstation or a host simulating an endpoint. Lastly, the Logger runs Splunk, and a Fleet server [82]. Figure 4.5 illustrates the lab setup with the different hosts and services, taken from [81].

Possible use-cases for DetectionLab includes:

- red team members interested to see what kind of logs and forensic artefacts their tools and methods will generate in a comparable environment,
- anyone seeking reference material to automate the installation and configuration of security tools,
- anyone needing a small staging environment to adjust security tool configurations.

DetectionLab makes setting up a lab environment more straightforward and faster through Packet and Vagrant features. Packer is responsible for producing a "box", a compressed version of each VM. DetectionLab has pre-built boxes that the user can directly employ, or they can modify or add custom ones [81]. Vagrant uses Vagrantfiles, which are comparable to Dockerfiles. They contain information on the VM, such as the OS, memory and

Capability	User Action	Methods
Web browsing	Browse Enter text Click link or button	Random, specific, looping
Terminal commands	Execute cmd commands Execute PowerShell commands	Random, specific, looping
Inter-NPC communication	Email creation and management	Specific, looping
Office document management	Common file formats for word processor, spreadsheet, and presentation documents created and saved locally or on a network drive	Random, specific, looping

Table 4.1: Capabilities in the GHOSTS framework from [128].

CPU specifications, network options, and any command scripts that the VM should run. Similar to how Docker can pull down images, Vagrant can pull down complete VMs [81].

DetectionLab is vulnerable by design and does not include any operating system hardening. The developer recommends running it in a virtualised environment such as VirtualBox or VMware [82].

4.5 GHOSTS

Researchers at Carnegie Mellon University (CMU) Software Engineering Institute (SEI) in Pennsylvania created the GHOSTS framework² [128, 69]. It is a technology for simulating complex, realistic *non-player characters (NPCs)* that mimics different roles in enterprise networks [128]. GHOSTS produces “*highly authentic, observable network traffic*”, according to its developers [128]. The framework allows cybersecurity researchers and professionals to construct a varied simulated player experience, e.g. creating documents, accessing systems, browsing the web, clicking and running commands [69]. No network activity can be traced back to the GHOSTS software; instead, activity is carried out by a *software agent*, represented via a character [128]. This agent represents what administrators might regularly encounter in real networks. GHOSTS also allows facilitators to recreate the scenario with a high level of realism [128]. Table 4.1 provides a full summary of the frameworks’ capabilities, fetched from [128].

GHOSTS clients (NPCs) are configurable for Windows and Linux operating systems, and they interact with a RESTful API server. The NPC itself runs as an executable file, *ghosts.exe*. The GHOSTS server comprises a PostgreSQL database, an API server, and a Grafana-based visualisation tool that gives an overview of the NPCs’ activity [93].

²<https://github.com/cmu-sei/GHOSTS>

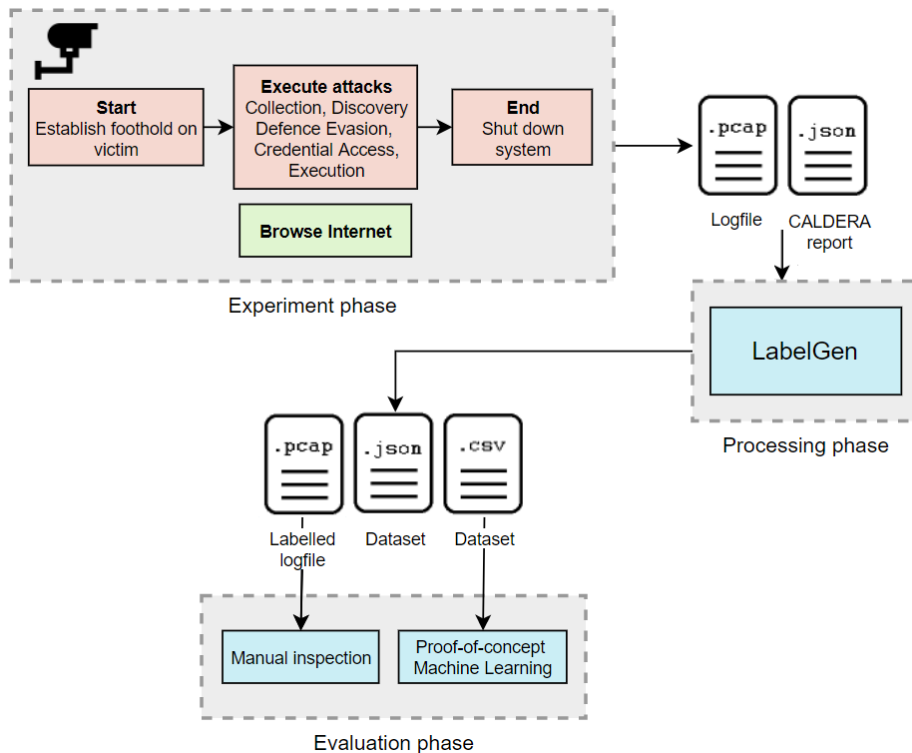


Figure 4.6: Overall workflow of the approach in this thesis.

4.6 Approach

As described in Section 4.1, the work in this thesis is divided into three phases. Figure 4.6 provides a more thorough view than Figure 4.1 in Section 4.1, putting each of the three stages of *experiment*, *processing*, and *evaluation* in context. In the figure, the simulation of malicious and benign activities (red and green squares) to generate network traffic is covered in the experiment phase. As previously stated, malicious traffic was generated using the APT29 EMU plan in CALDERA, while benign traffic was generated using GHOSTS to access the Internet. A small surveillance camera is placed in the upper right corner of the box denoting the experiment phase in Figure 4.6. The camera indicates that the experiment is being captured or recorded.

DetectionLab created two networks in which the VMs were connected during the experiment. During the experiment, the VMs had different responsibilities or tasks, which are depicted in Figure 4.7. The networks are distinguished by two colours, which correspond to the colours used in Section 4.2 to describe the network architecture. The Host-Internal network connects the 5 VMs, allowing them to communicate. The network traffic from the two networks, Host-Internal Network (blue) and NAT Network (green), is captured by Ubuntu (Logger). The use of coloured arrows pointing to each

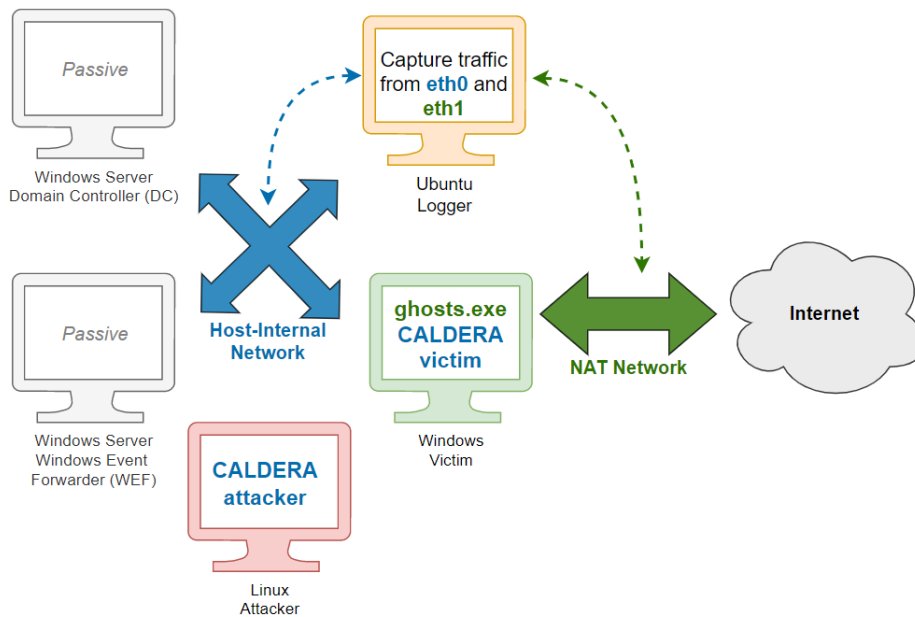


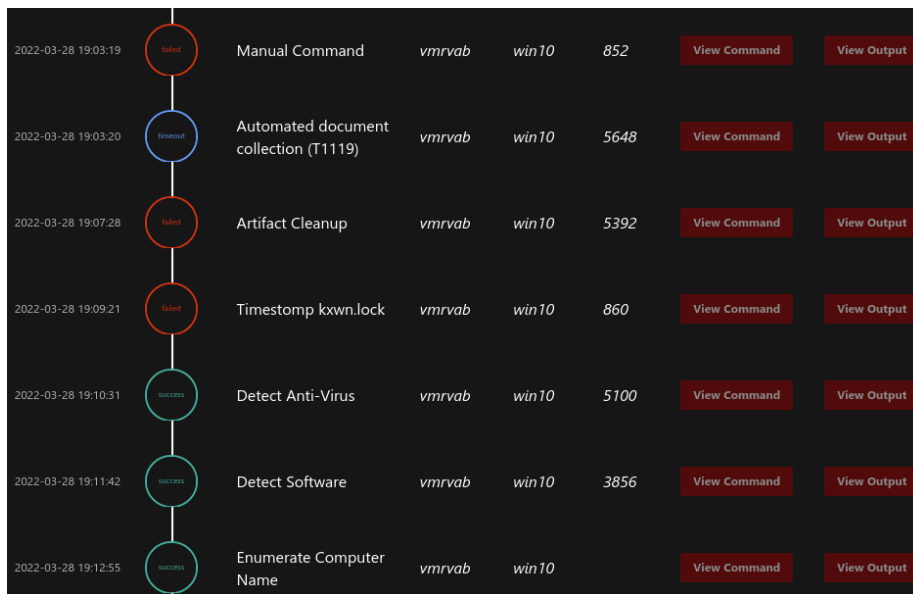
Figure 4.7: Responsibilities between the VMs during implementation.

of these networks in the figure emphasise this. During the experiment, two VMs were passive (DC and WEF), as shown by their grey colouring and the text "Passive" on the monitor. The main reason they were connected to the Host-Internal network was to generate background. Linux (attacker), emulating APT29, targets Windows (Victim). In addition to being a victim of the attack, Windows also executes GHOSTS, as shown by the file *ghosts.exe* in its monitor.

The experiment produced a network logfile (.pcap) and a *CALDERA report*, summarising the attacks that were executed. These two files were used to label with LabelGen. LabelGen assigned labels to network packets in the logfile that matched background, benign, or attacker-related activities or behaviour. LabelGen created two labelled datasets and one labelled logfile. The implementation of the approaches above will be detailed in the following section, complementing this overview. LabelGen will be described in-depth in a dedicated subsection (4.7.1).

4.7 Implementation

The packet analyser *tcpdump* was used to capture network traffic. The VM collecting traffic was configured in promiscuous mode, allowing it to capture from multiple interfaces or networks. The interfaces transporting DetectionLab and CALDERA traffic were captured, yielding two separate .pcap files. These files were merged prior to labelling. An internal network facilitated the connection between the GHOSTS application and the GHOSTS API. This communication would, however, interfere with the rel-



2022-03-28 19:03:19	failed	Manual Command	vmrvab	win10	852	View Command	View Output
2022-03-28 19:03:20	success	Automated document collection (T1119)	vmrvab	win10	5648	View Command	View Output
2022-03-28 19:07:28	failed	Artifact Cleanup	vmrvab	win10	5392	View Command	View Output
2022-03-28 19:09:21	failed	Timestomp kxwn.lock	vmrvab	win10	860	View Command	View Output
2022-03-28 19:10:31	success	Detect Anti-Virus	vmrvab	win10	5100	View Command	View Output
2022-03-28 19:11:42	success	Detect Software	vmrvab	win10	3856	View Command	View Output
2022-03-28 19:12:55	success	Enumerate Computer Name	vmrvab	win10		View Command	View Output

Figure 4.8: CALDERA operation, attack steps 42-48.

evant traffic. Hence it was excluded from logging.

In the implementation, the following versions of tools were used:

- CALDERA version 4.0.0-alpha for adversary emulation;
- DetectionLab master branch, commit 4318620a4dd279665fd11ae5b88217385047fe9d (28.11.21) from GitHub;
- GHOSTS version 6.0.0;
- LinuxLite version 5.6 on the attacker VM;
- Windows 10 version 19H2 on the victim VM;
- Firefox version 98.0.1 on Linux and Windows;
- PyCharm Community Edition, version 2021.3.3.

The VMs were given the following RAM specifications: 1024 MB RAM for WEF and DC, 2048 MB RAM for Logger, and 2500 MB RAM for Windows (victim) and Linux (attacker). A laptop with a total of 16 GB powered the VMs.

Both CALDERA and GHOSTS were run by the attacker VM (Linux), yet on separate networks, as previously mentioned. The default CALDERA settings were used, including beaconing from the agent to the server every 30 to 60 seconds and destroying the agent after 90 seconds if the server became unreachable. CALDERA displays operation results during attacks, which can be viewed in Figure 4.8. As the figure shows, the majority of the attacks were successful (green circle), but some were unsuccessful (red

circle) and timed out (blue circle). As the experiment progressed, I documented which steps of the EMU plan were successful, which were ignored and which failed. The findings are discussed in Chapter 6, Discussion.

The EMU plan's final step was to shut down the victim VM (Windows), bringing the experiment to an end. The entire experiment or operation lasted one hour and produced 1 679 967 network packets. CALDERA can provide a report that summarises the operation once it is completed. The report contains information such as the agent's name, the commands executed, base64 encoding of the command, timestamp of when CALDERA delivered the command to the agent, and timestamp of when the agent reported having executed the command. Most importantly, data from ATT&CK is included in this report, specifying which tactic, technique name, and technique ID each completed attack relates to. The attacks' time and technique IDs are used in the labelling.

4.7.1 LabelGen: implementation of labelling

LabelGen is the actual implementation of labelling in this thesis. It is a Python script that is comparable to a *main.py* file. Figure 4.9 illustrates the general flow or logic of LabelGen. The input and output for the various steps are displayed, with arrows showing their application and guiding the program flow. The figure presents the complete procedure in LabelGen, from the initial definition of variables to the final generation of datasets.

LabelGen requires the definition of specific variables in the beginning; the logfile to label, the CALDERA report and a *comments-file*. The latter is a .txt file that was created because the command for labelling a .pcap file, *Editcap*³, allows a limited number of arguments. Therefore, the *comments-file* is used throughout LabelGen to save packets and labels for later steps. Before LabelGen can proceed, additional variables must be provided, including the IP addresses of the attacker VM (Linux), the victim VM (Windows), and three other DetectionLab VMs. These addresses determine whether a packet belongs in the attack, background, or benign categories.

The algorithm *processDeclareLabels.py* takes the network logfile and the defined IP addresses as input. The output from processing and declaring labels is presented in Figure 4.6. The output is packets with categories attack, background, benign, and the *comments-file*. The CALDERA report and attack packets are then passed to a function that searches for packets related to CALDERA attacks, *findCaldera.py*.

The *comments-file* is the output of *findCaldera.py*, which is separated into multiple files in the next step. The *comments-file* had to be split because *Editcap* only accepted a certain number of arguments for labelling packets. *labelPackets.sh* takes these *comments-files* and the original network logfile as input. This function labelled the network packets, i.e. added frame comments. The function generated a number of small, labelled .pcap files, which were then merged into a single, labelled logfile. Two datasets were created from this logfile: one in JSON format and the other in CSV format.

³<https://www.wireshark.org/docs/man-pages/editcap.html>

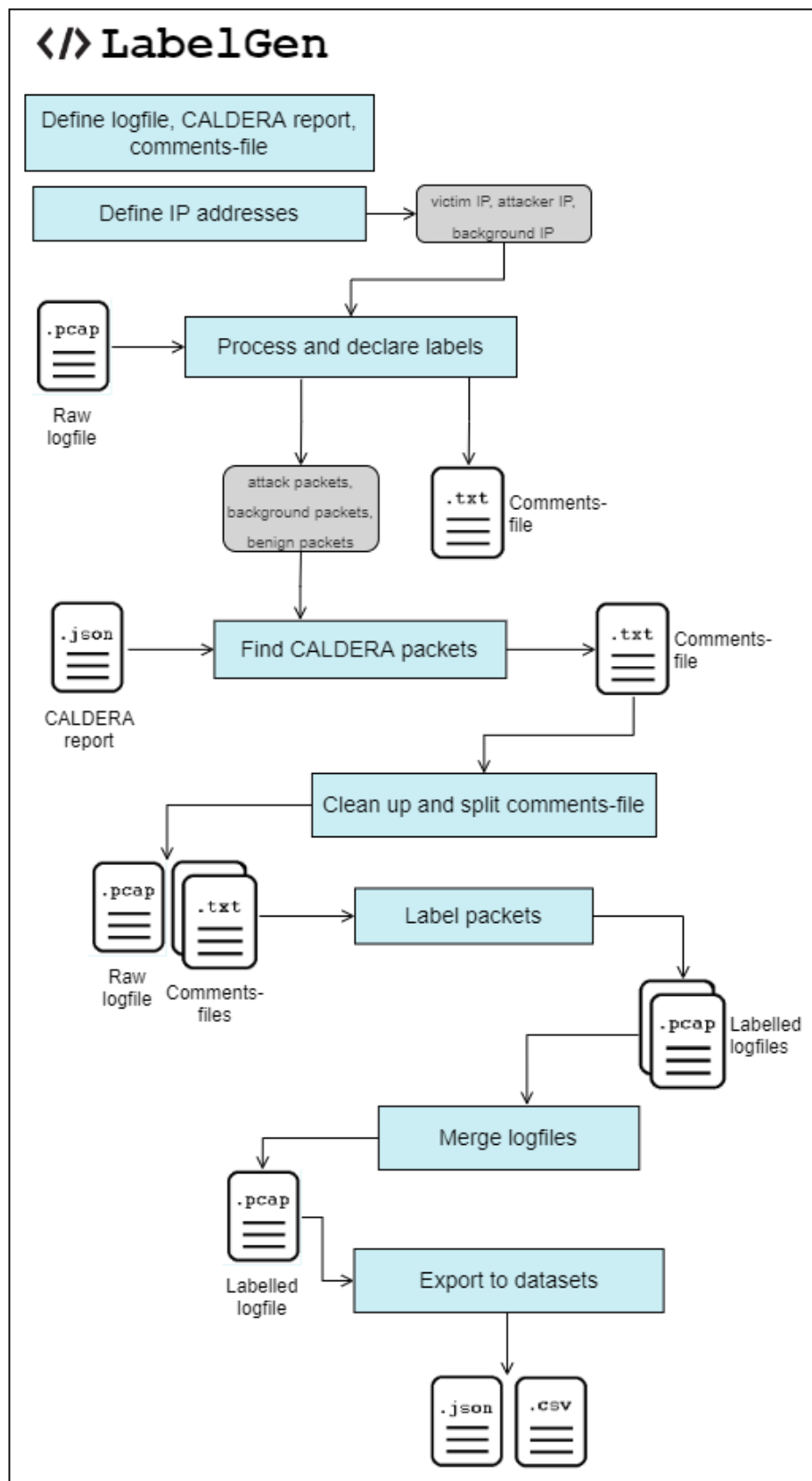


Figure 4.9: Order of implementation.

The following pages will present pseudo-code for some of the most essential steps of LabelGen, namely *processDeclareLabels.py*, *findCaldera.py* and *labelPackets.sh*. These functions are invoked from other scripts as per best practice principles.

Processing the logfile and declaring labels

Listing 1 processDeclareLabels.py

```

1  input = logfile
2  for packet in logfile:
3      if (packet.ip = attacker ip or victim ip):
4          attack_packets.append(packet)
5          commentsfile.write("-a packet_number:T1071.001")
6      elif (packet.protocol = background protocols) or
7          (packet.ip = background ip)
8          background_packets.append(packet)
9          commentsfile.write("-a packet_number:Background")
10     else:
11         benign_packets.append(packet)
12         commentsfile.write("-a packet_number:Benign")
13     return attack_packets, commentsfile

```

The pseudocode in Listing 1 iterates through all of the packets in the logfile from the experiment. The function determines which packets are related to attacks, background, and benign activity based on their IP addresses and protocols. If the source and destination IP addresses match the known IP addresses of the victim and the attacker, the packet is considered part of an attack, as shown on line 4 in Listing 1. The packet number and the string "T1071.001" are written in the comments-file to signify C&C communication. It qualifies as C&C communication because the agent beacons back to the server, though it could be considered metadata in this thesis. However, I decided on this labelling method because I do not consider C&C to be either background or benign operations; it is, according to MITRE, an attack⁴. The packets that receive a C&C label are later modified. The packets are also added to the list *attack packets*, later used to identify packets based on CALDERA attack techniques.

If the current packet matches the conditions shown in the elif blocks on lines 6 and 7 of Listing 1, it is added to the list *background packets*. These elif statements capture packets that use protocols known to contain background information and operations or if their IP addresses match the DetectionLab Host-Only network.

Finally, the benign traffic associated with GHOSTS and web browsing is caught in the final else statement on line 10 in Listing 1. The packets that do not meet the conditions above and therefore execute here may not be GHOSTS traffic. However, because of the previous statement in which the

⁴Tactic Command&Control, technique T1071.001, communication over Application Layer Protocol, Web protocols: <https://attack.mitre.org/tactics/TA0011/>.

packet is background or C&C, the packets that fall in the benign category are certainly not malicious.

The comments-file contains entries for each packet number and its corresponding traffic label when the code in Listing 1 has run. Each packet has one line in the comments-file, which is formatted as follows:

```
-a frame-number:comment, e.g.
-a 1190402:Background
-a 1190403:Benign
-a 3793021:T1071.011
```

Finding packets with CALDERA attacks

Listing 2 findCaldera.py

```
1 input = caldera report
2 for step in caldera report:
3     start time = step.agent_reported_time
4     end time = step.run
5     for packet in attack packets:
6         while start time <= packet.timestamp <= end time:
7             commentsfile.write("-a packet_number:technique")
8             caldera_packets.append(packet)
9     return commentsfile
```

Listing 2 executes as the subsequent step in LabelGen. The code in the listing analyses each packet in the list *attack packets* to identify those associated with certain attack techniques. The CALDERA report determines the timing of each attack stage, and the algorithm checks to see if each packet is within that interval.

There were 59 attack steps in the CALDERA report, with 36 unique techniques. The outer for-loop on line 2 in Listing 2 is therefore executed 59 times. The timestamps *agent_reported_time* and *run* from the CALDERA report represent the time the agent executed the attack, and the time where the agent submitted execution results, respectively [45]. Accordingly, these properties define the start and end times of the current attack step. If the packet's timestamp falls within this range, the current technique ID is assigned to it, and the packet number and technique ID are written to the comments-file.

When Listing 2 finished, the comments-file contained entries for both attack packets with C&C labels and technique-labels. Some smaller scripts removed redundant C&C lines to prevent certain packets from being labelled twice. As a result, the comments-file comprised CALDERA packets where applicable, C&C packets elsewhere, and the other background and benign packets.

Labelling the packets

Listing 3 labelPackets.sh

```

1  #!/bin/bash
2  input=logfile, commentsfiles
3  for file in $commentsfiles; do
4      concatString=""
5      while read line; do
6          concatString += $line
7      done < file
8      editcap $concatString -r logfile n.pcapng
9          ↪ first_packet-last_packet
9  done
10 return n commentsfiles

```

Listing 3 displays pseudo-code for the labelling done with LabelGen. *labelPackets.sh* is a Bash script that reads lines from smaller comments-files as input, as demonstrated in Figure 4.6 earlier. It also takes the logfile as input.

The for-loop reads the smaller comments-files in line 3 of the algorithm. As indicated on line 6, each line read is concatenated to the string *concatString*. This string is used as the parameter for packet comments when labelling with Editcap on line 8. Each of the comments-files followed the same procedure. The following Editcap command was used to label packets or add frame comments:

```

editcap -a 1234:Background -a 1235:Background -a 1236:Benign
-r originalLogfile.pcap 1.pcapng 1234-1236

```

labelPackets.sh generated multiple .pcap files, one for each comments-file. After this algorithm was completed, the smaller logfiles were merged into one final, labelled logfile. This .pcap file was converted to JSON and CSV format datasets using a separate script, including some final adjustments for the datasets.

4.8 Summary and discussion

This chapter has presented the tools and frameworks that are used in the conducted experiment and the labelling approach, LabelGen. The experiment generated a logfile, which was labelled in LabelGen using Editcap. All of the methods in LabelGen are called from other scripts, following best practise principles. It also contains the declarations of the required variables, making it simple to maintain.

The initial raw logfile contained 1 679 967 packets, while the labelled logfile contained 1 679 966. These numbers demonstrate that only one packet got lost during the the process of running LabelGen. There were also 1 679 037 packet comments or labels in the final logfile.

LabelGen can automatically perform the labelling process when the values for the logfile, CALDERA report, and IP addresses contain the user's specifications. During the labelling, numbers are printed, indicating which line or file it is currently processing. The script also prints some feedback, such as telling the user whether the current method ran successfully or not. In other words, the labelling process is automatic and covers the whole process, from identifying packets to creating labelled datasets with the correct formats.

Chapter 5

Results and evaluation

This chapter presents the labelling results, including the labelled .pcap file and the two datasets generated with LabelGen. The methods of evaluation are as follows:

- Manual inspection, which is the main method of evaluation;
- Evaluation of LabelGen by labelling a different logfile;
- Proof-of-concept machine learning model.

This chapter will begin with a description of the datasets. The subsequent section will be a manual inspection of the labelled logfile. Furthermore, the generalisability of LabelGen is evaluated with a distinct logfile. This evaluation will determine whether LabelGen is tailored to the experiment described in this thesis or if it can also label other logfiles. A trained proof-of-concept ML model will add to the dataset's and LabelGen's evaluation. Finally, there will be a concluding section on whether these generated datasets could be used in the future to detect an APT attack or these attack techniques.

5.1 The datasets

The final datasets are in JSON and CSV format. Because the JSON dataset was exported directly from the logfile, it contains somewhat more information than the CSV dataset. For the CSV dataset, the columns to export had to be specified, resulting in a smaller dataset. Because the dataset would be utilised for ML later, the following columns were selected for the CSV dataset:

- frame.number
- frame.time_epoch
- frame.len
- frame.encap_type
- frame.time_
- frame.protocols
- frame.time
- relative
- label

- eth.dst
- eth.src
- eth.type
- ip.hdr_len
- ip.dsfield
- ip.len
- ip.id
- ip.flags
- ip.ttl
- ip.proto
- ip.checksum
- ip.checksum.status
- ip.src
- ip.addr
- ip.src_host
- ip.dst
- ip.dst_host
- tcp.srcport
- tcp.dstport
- tcp.port
- tcp.stream
- tcp.completeness
- tcp.len
- tcp.seq
- tcp.seq_raw
- tcp.nxtseq
- tcp.ack
- tcp.ack_raw
- tcp.hdr_len
- tcp.flags
- tcp.window_size_value
- tcp.checksum
- tcp.checksum.status
- tcp.urgent_pointer
- tcp.analysis
- tcp.analysis.bytes_in_flight
- tcp.analysis.push_bytes_sent
- tcp.payload
- tcp.segment_data
- tcp.segments
- tcp.segments
- tcp.segment.count
- tcp.reassembled.length
- udp.srcport
- udp.dstport
- udp.port
- udp.length
- udp.checksum
- udp.checksum.status
- udp.stream
- http.connection
- http.request.line
- http.content_type
- http.content_encoding
- http.user_agent
- http.content_length_header
- http.host
- http.request
- http.request_number
- _ws.col.Info

The dataset contains 1 679 966 rows, one for each packet. The field names containing "frame" refers packets. The label is also a frame comment, as mentioned previously. The column *_ws.col.Info* coincide to the information column in Wireshark, showing port numbers and POST/GET requests. Figure 5.1 shows a snippet of the CSV dataset in Excel. The columns included in the figure are frame.number, frame.encap_type, frame.time, frame.time_epoch, frame.time_relative, frame.len, frame.protocols, label, eth.dst and eth.src. The label column shows that the packets are labelled as background packets.

Listing 4 show an example of a single packet in the JSON dataset. The majority of the non-essential layer objects are collapsed for illustration purposes.

1	1	Mar 28, 2022 18:35:13	164848°0.00000	54	eth:ethertype:ip:tcp	Background	08:00:27:1e:ce:34	08:00:27:0e:39:f8
2	1	Mar 28, 2022 18:35:13	164848°0.00518	89	eth:ethertype:ip:tcp:da	Background	08:00:27:0e:39:f8	08:00:27:1e:ce:34
3	1	Mar 28, 2022 18:35:13	164848°0.00539	54	eth:ethertype:ip:tcp	Background	08:00:27:1e:ce:34	08:00:27:0e:39:f8
4	1	Mar 28, 2022 18:35:13	164848°0.00600	1270	eth:ethertype:ip:tcp:da	Background	08:00:27:0e:39:f8	08:00:27:1e:ce:34
5	1	Mar 28, 2022 18:35:13	164848°0.00613	54	eth:ethertype:ip:tcp	Background	08:00:27:1e:ce:34	08:00:27:0e:39:f8
6	1	Mar 28, 2022 18:35:13	164848°0.00659	185	eth:ethertype:ip:tcp:da	Background	08:00:27:0e:39:f8	08:00:27:1e:ce:34
7	1	Mar 28, 2022 18:35:13	164848°0.00672	54	eth:ethertype:ip:tcp	Background	08:00:27:1e:ce:34	08:00:27:0e:39:f8
8	1	Mar 28, 2022 18:35:13	164848°0.01417	89	eth:ethertype:ip:tcp:da	Background	08:00:27:0e:39:f8	08:00:27:1e:ce:34
9	1	Mar 28, 2022 18:35:13	164848°0.01417	1514	eth:ethertype:ip:tcp:da	Background	08:00:27:0e:39:f8	08:00:27:1e:ce:34
10	1	Mar 28, 2022 18:35:13	164848°0.01417	1172	eth:ethertype:ip:tcp:da	Background	08:00:27:0e:39:f8	08:00:27:1e:ce:34
11	1	Mar 28, 2022 18:35:13	164848°0.01431	54	eth:ethertype:ip:tcp	Background	08:00:27:1e:ce:34	08:00:27:0e:39:f8
12	1	Mar 28, 2022 18:35:13	164848°0.01480	185	eth:ethertype:ip:tcp:da	Background	08:00:27:0e:39:f8	08:00:27:1e:ce:34
13	1	Mar 28, 2022 18:35:13	164848°0.01498	54	eth:ethertype:ip:tcp	Background	08:00:27:1e:ce:34	08:00:27:0e:39:f8
14	1	Mar 28, 2022 18:35:13	164848°0.02026	89	eth:ethertype:ip:tcp:da	Background	08:00:27:0e:39:f8	08:00:27:1e:ce:34
15	1	Mar 28, 2022 18:35:13	164848°0.02040	54	eth:ethertype:ip:tcp	Background	08:00:27:1e:ce:34	08:00:27:0e:39:f8

Figure 5.1: A snippet of the dataset in CSV format.

Listing 4 One packet shown in JSON format.

```

1 {
2   "_index": "packets-2022-03-28",
3   [ ... ]
4   "_source": {
5     "layers": {
6       "pkt_label": {
7         "label": "T1036.005",
8         "label_tree": { ... }
9       },
10      "frame": {
11        "frame.time": "Mar 28, 2022 18:57:37.303315000 CEST",
12        "frame.protocols": "eth:ethertype:ip:tcp:data"
13        [ ... ]
14      },
15      "eth": {
16        "eth.dst": "08:00:27:4b:c7:37",
17        "eth.src": "08:00:27:5d:f0:36",
18        "eth.type": "0x0800",
19        [ ... ]
20      },
21      "ip": {
22        "ip.version": "4",
23        "ip.hdr_len": "20",
24        "ip.ttl": "128",
25        "ip.proto": "6",
26        "ip.checksum": "0x4faa",
27        "ip.src": "192.168.56.104",
28        "ip.dst": "192.168.56.106",
29        [ ... ]
30      },
31      "tcp": {
32        "tcp.srcport": "50424",
33        "tcp.dstport": "8888",
34        [ ... ],
35      },
36      "data": {
37        "data.data": "00",
38        "data.len": "1"
39      }
40    }
41  }
42 }, [ next packet ]

```

5.2 Manual inspection

While automated inspection approaches such as tools or scripts might speed up the inspection process, manual inspection will be the primary source of evaluation in this thesis. Examining the labelled .pcap file and datasets more closely makes it possible to determine whether LabelGen performed satisfactorily and whether CALDERA was an appropriate tool choice.

Figure 5.2 shows packets with the comment "T1036.005," which corresponds to the ATT&CK tactic defensive-evasion, technique Masquerading: Match Legitimate Name or Location [36]. The Wireshark window to the right counts seven packets. The command executed for this attack is short; hence the number of packets is few. To the left in Figure 5.2, the fields *agent_reported_time* and *run* are marked with red boxes. By looking at the arrows pointing to the first and the last packets in the .pcap file, it is evident that the timestamps match. It is important to note that the timestamps in Wireshark are two hours behind the timestamps in the CALDERA report. The reason for this is that Wireshark maintains timestamps in UTC [59], whereas network packets are stored in the local time zone of Norway, which is currently GMT+2. Thus, the evaluation should be based on minute- and second-numbers. Furthermore, the timing of the last packet and *run* are not an exact match. This is because no more packets were sent between the attacker and the victim throughout the attack time interval, resulting in a slightly off time stamp. I analysed this further by examining the next packets in the logfile, which contained neither the IP addresses of the attacker nor any other ATT&CK labels. To further inspect if the labelling was correct, a beacon request from the attacker to the victim is presented in Figure 5.3. This packet was sent before the labelled T1036.005 packets that were shown in the previous figure, Figure 5.2. It *should* therefore not be labelled with this technique, but rather as C&C. Figure 5.3 confirms that this is a C&C packet, as the output in the middle of the figure is decoded, showing an empty instructions-object.

Manual inspection of benign packets concluded that the traffic originated on the victim machine's NAT network and was directed to other IP addresses. Encryption (port 443) was frequently utilised, indicating that Benign was the accurate label for the packets. Wireshark's manual inspection allowed for packet comment filtering and statistics viewing. I could search for "Benign" and check which IP addresses were involved. The majority of the addresses were unknown, and the IP address of the victim system running GHOSTS was by far the most common. A few packets were labelled as benign that contained system calls such as "BROWSER" between the DetectionLab VMs. They should instead be labelled as background instead of benign. I would therefore conclude that some of the benign packets were mislabelled.

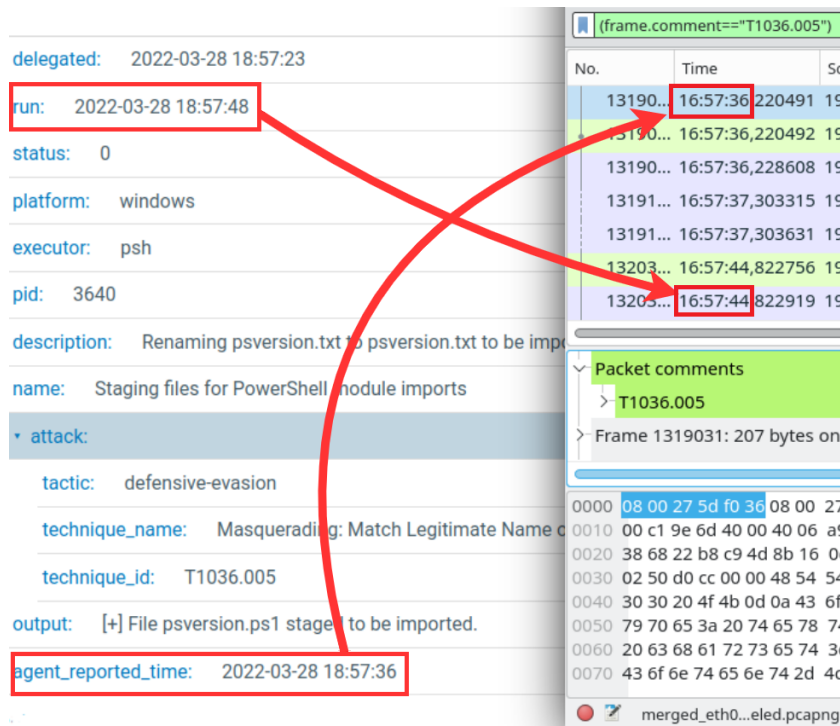


Figure 5.2: Packets related to ATT&CK technique T1036.005.

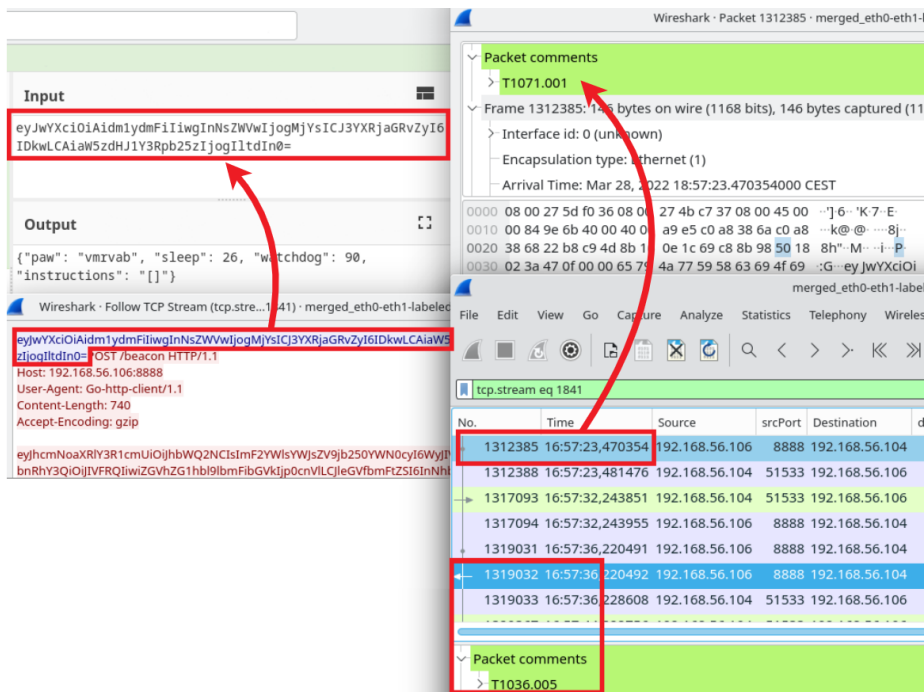


Figure 5.3: Confirmed beaconing packet prior to attack step T1071.001.

5.2.1 Packet statistics

Table 5.1 presents the number of packets in each of the categories in the labelled .pcap file and the datasets. Except for the second to last row in the table, which counts the number of missing labels, the techniques are sorted alphabetically. The majority of the packets are labelled as background and benign, as shown in Table 5.1. In a realistic scenario, attack traffic would be substantially less than benign or background traffic [90]. This makes the labelled .pcap file and datasets realistic in that regard. DetectionLab generated most of these background packets, which would be realistic in an enterprise environment. Therefore, this traffic was not filtered in an attempt to create a more balanced dataset. The vast majority of the rows in Table 5.1 display the CALDERA attack techniques and the number of packets associated with each approach in the dataset. There is not necessarily a correlation between a low number of packets and a successful attack. The amount of packets is mostly determined by the attack’s command or response. The number of packets on row 9 of technique T1036.002, for example, is 3492. This attack stage involves sending several large files, as I observed during a manual inspection. As a result, this attack contains many packets, compared to T1518.001 on row 7 from the bottom, which only has 6 packets.

Label	Technique name	No. of packets
Background	Background traffic	1 488 521
Benign	Benign Internet Browsing	183 037
T1003	Credential Dumping	877
T1007	System Service Discovery	79
T1012	Query Registry	11
T1016	System Network Configuration Discovery	30
T1018	Remote System Discovery	307
T1033	System Owner/User Discovery	47
T1036.002	Masquerading: Right-to-Left Override	3492
T1036.005	Masquerading: Match Legitimate Name or Location	7
T1041	Exfiltration over C2 Channel	3
T1049	System Network Connections Discovery	26
T1055	Process Injection	906
T1056.001	Input Capture: Keylogging	21
T1057	Process Discovery	123
T1059.001	Command and Scripting Interpreter: PowerShell	70
T1069	Permission Groups Discovery	27

T1070.004	Indicator Removal on Host: File Deletion	30
T1070.006	Indicator Removal on Host: Timestomp	7
T1071.001	Command & Control: Web Protocols	631
T1082	System Information Discovery	53
T1087	Account Discovery	14
T1112	Modify Registry	8
T1113	Screen Capture	283
T1114.001	Email Collection: Local Email Collection	14
T1115	Clipboard data	7
T1119	Automated Collection	16
T1134.001	Access Token Manipulation: Token Impersonation/Theft	45
T1134.002	Access Token Manipulation: Create Process with Token	8
T1218.011	Signed Binary Proxy Execution: Rundll32	7
T1518	Software Discovery	10
T1518.001	Software Discovery: Security Software Discovery	6
T1546.003	Event Triggered Execution: Windows Management Instrumentation Event Subscription	10
T1547.009	Boot or Logon Autostart: Shortcut Modification	28
T1552.004	Unsecured Credentials: Private Keys	11
T1561.001	Disk Wipe: Disk Content Wipe	265
Empty	No label	929
Total number of packets:		1 679 966

Table 5.1: Mapping of techniques and the corresponding number of packets in the labelled network file.

Figure 5.4 visualises the number of packets for each attack strategy to complement the representation of packets per technique presented in Table 5.1. Packet with missing labels, background and benign labels are excluded from this figure because they distort the proportions. The figure clearly illustrates that the attack T1036.002 had the most packets, which is due to the fact that this attack transfers huge files, as previously stated.

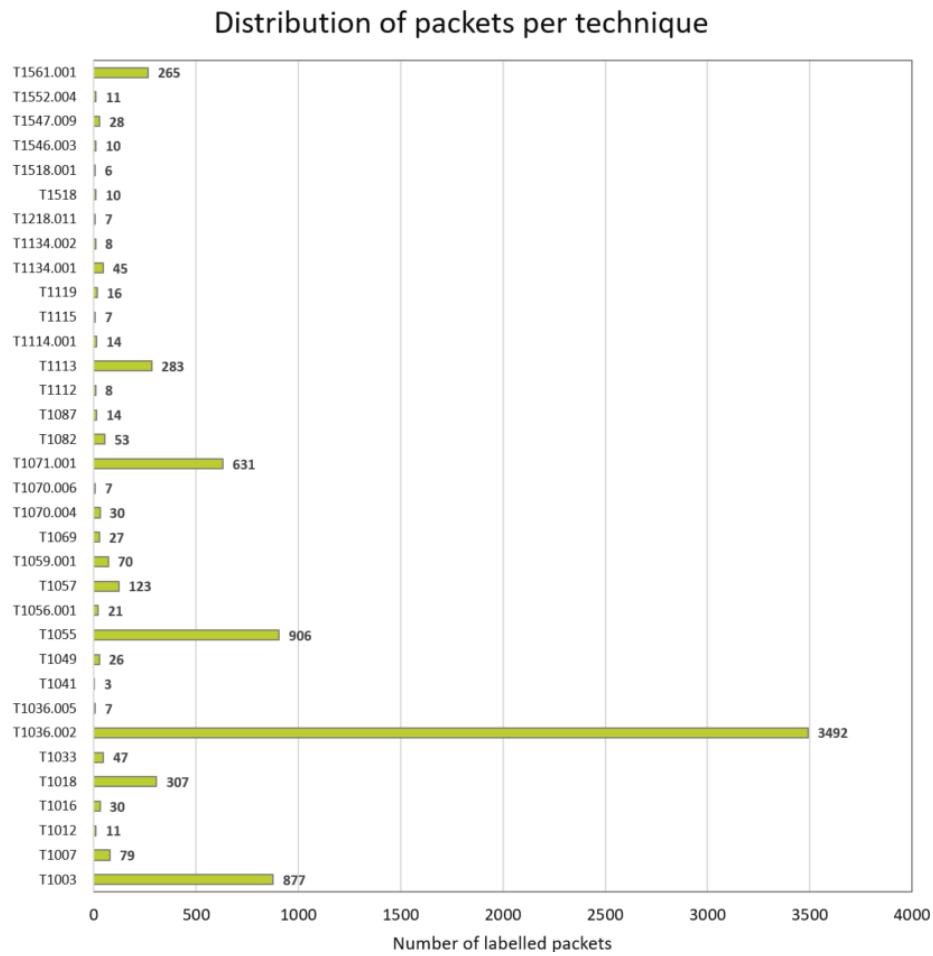


Figure 5.4: Number of packets for each technique

Table 5.2 provides additional statistics. This table shows the distribution between packets related to attacks, background and benign traffic, including packets without a label. Because it may be difficult to determine if the labelling was satisfactory or not based on the preceding table (5.2), this table summarises the details. Table 5.2 that the dataset contained 7479 packets related to attacks, accounting for 0.44% of the whole dataset. Background packets account for 88.60% of the dataset, whereas benign traffic accounts for 10.89%. We can also see how many per cent of the packages were unlabelled in this table, which was 0.06%. The last column summarises that 99.93% of the packets were labelled. This value, in my opinion, is satisfactory, showing that LabelGen labelled the majority of the packets correctly. After manual inspection, the unlabelled packets were revealed to be related to ARP queries, ICMP traffic, and other background operations.

	No. of packets	Percentage
Attack	7479	0.44%
Benign	183037	10.89%
Background	1488521	88.60%
Empty label	929	0.06%
Total no. labelled packets	1 679 037	99.94%
Total no. packets	1 679 966	100%

Table 5.2: Number of packets and percentage of each category.

5.3 Evaluating LabelGen

LabelGen could be used for other network logfiles, as long as the CALDERA report - in its current format - is the source for labelling on attack techniques. LabelGen is bound to CALDERA but not to the specific experiment that was conducted in this thesis. Some variables must be adjusted before LabelGen may be used to label other logfiles. The IP addresses of the machines that generate background traffic and the IP addresses of the victims and attackers must be adjusted to match the other experiment. Similarly, the experiment must be represented in the .pcap file and the CALDERA report.

5.3.1 Experimenting with a different attack

I conducted another short experiment to evaluate LabelGen, using the same approach as outlined in Section 4.7. In CALDERA, I created a new adversary profile and included ten attacks from different ATT&CK techniques that were not included in the APT29 EMU plan. The goal was to test the approach for a broader spectrum of attacks. The profile can be seen in Figure 5.5. The attacks shown are related to the tactics execution, privilege-escalation, defensive-evasion, discovery, exfiltration and persistence are shown.

The operation flow of the experiment is shown in Figure 5.6. Not all planned attacks were carried out effectively, as indicated by the various coloured circles. This was largely due to CALDERA failing to provide the necessary payloads or Powershell refusing to execute the given command. The goal of this experiment, however, was to primarily create a dataset to evaluate LabelGen rather than to carry out the most successful attacks.

Before LabelGen could be utilised, the CALDERA report had to be pre-processed. Although, in the future, LabelGen should account for insufficient data.

CustomProfile
Custom profile for evaluating LabelGen.

+ Add Ability + Add Adversary Objective: **default** Change Save Profile Delete Profile

Ordering	Name	Tactic	Technique
1	Execution through API - CreateProcess	execution	Native API
2	Process Injection	privilege-escalation	Process Injection
3	PowerSploit Named-Pipe Impersonation	privilege-escalation	Access Token Manipulation
4	Maldoc choice flags command execution	execution	User Execution: Malicious File
5	Potentially Unwanted Applications (PUA)	execution	User Execution: Malicious File
6	Registry Cleanup for UAC Bypass Technique	defensive-evasion	Modify Registry
7	Examine domain password policy - Windows	discovery	Password Policy Discovery
8	Data from staged file (T1074) and Exfiltration over C2 Channel (T1041)	exfiltration	Exfiltration Over C2 Channel

Figure 5.5: Custom adversary profile created in CALDERA.

Decide	Status	Link/Ability Name	Agent #paw	Host	pid	Link Command
2022-05-03 16:36:54	success	Execution through API - CreateProcess	vytjgh	win10	5544	View Command
2022-05-03 16:37:04	success	Process Injection	vytjgh	win10	2288	View Command
2022-05-03 16:37:54	failed	PowerSploit Named-Pipe Impersonation	vytjgh	win10	6120	View Command
2022-05-03 16:38:29	success	Maldoc choice flags command execution	vytjgh	win10	5832	View Command
2022-05-03 16:38:29	failed	Execution through API - CreateProcess	vtbjif	win10	3040	View Command
2022-05-03 16:38:39	success	Potentially Unwanted Applications (PUA)	vytjgh	win10	5784	View Command
2022-05-03 16:38:39	collect	Process Injection	vtbjif	win10	n/a	View Command

Figure 5.6: Custom adversary plan running in CALDERA.

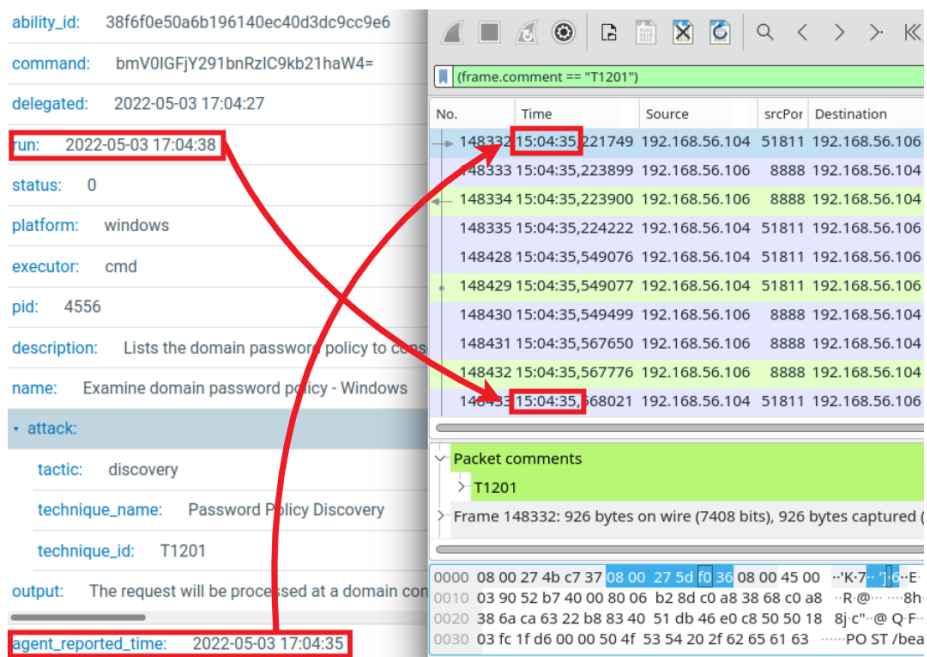


Figure 5.7: Corresponding timestamps in the new experiment.

Manual inspection

The labelled logfile from this experiment will be analysed in the same way as the prior manual inspection method. Wireshark will once more be used to display and document the findings.

The experiment took approximately 30 minutes and produced 193 078 packets. After processing the logfile from the experiment with LabelGen, the labelled file contained 192 993 packet comments. Similar the figures in Section 5.2, Figure 5.7 shows the mapping between *agent_reported_time* and *run*, where it is clear that the packet is labelled correctly within the time frame of the current attack step, T1201. As before, the total number of packets labelled to each attack can be seen in a table, namely Table 5.3. This table illustrates that, similar to the results of the original experiment, the majority of packets are benign and background, whereas attacks are in the minority. 85 packets were unlabelled, and this was shown in Wireshark to be ARP and other background processes.

Label	Technique name	No. of packets	Percentage
Background	Background traffic	142 571	73.84%
Benign	Benign Internet Browsing	43 990	22.78%
T1055	Process Injection	20	0.01%

T1071.001	Boot or Logon Autostart: Shortcut Modification	87	0.04%
T1106	Native API	5823	3.01%
T1112	Modify Registry	4	0.002%
T1134	Access Token Manipulation	14	0.007%
T1201	Password Policy Discovery	10	0.005%
T1204.002	User Execution: Malicious File	455	0.23%
T1505.002	Server Software Component: Transport Agent	15	0.007%
T1105.003	Server Software Component: Web Shell	4	0.002%
Empty	No label	85	0.04%
Total labelled:		192 993	99.95%
Total:		193 078	100%

Table 5.3: Mapping of techniques and the corresponding number of packets in the labelled logfile.

5.4 Machine Learning

Supervised machine learning algorithms requires labelled datasets. In this thesis, two labelled datasets were created. By training a proof-of-concept machine learning (ML) model, it could be demonstrated that the datasets are valuable for later usage with anomaly-based IDS. Anomaly-based IDS frequently use ML, which involves training a model on labelled data. This section will first describe the general concepts of ML as an introduction to the implementation of ML in the next section.

ML is a subset of *Artificial Intelligence (AI)*, which is defined as *intelligence displayed by machines* [100, 94]. The field of AI research, according to Legg et al. [78], is *"the study of any system that perceives its environment and takes actions to increase its probability of achieving its goals"*.

ML is the study of computer algorithms that can improve themselves automatically through experience and data usage [88]. Heung et al. [67] define ML as the automated process of discovering patterns in large datasets using computer-based statistical models. The two primary steps of this process, according to Liu et al. [80], are *training* and *testing*. By using learn-

ing algorithms, the training seeks to learn from known properties. Using the knowledge learned in the training step, testing attempts to predict the known properties [80]. Training and testing are in this context referred to as *learning* and *prediction* [80].

ML algorithms create a model based on training data to make predictions or decisions. Without being explicitly programmed, they adopt learning algorithms to create sample rules [76, 80].

Three major categories or paradigms are frequently used to classify ML methods, depending on the type of input or responses provided to the learning system. There are several categories, but the following are the most widespread:

- *Supervised learning*: a "teacher" or an expert presents the computer with example inputs and their expected outputs. By mapping the example input-output pairs, the objective is to learn a general rule [105]. Supervised learning is based on *labelled datasets*, where the learning algorithm is trained and builds a model that can predict the correct label for unlabelled, arbitrary input [80]. This is the most frequently used learning algorithm [64, 89].
- *Unsupervised learning*: the dataset given to the learning algorithm has no labels. The aim is that the learning algorithm itself should find structure and previously unknown patterns in its input [80]. Unsupervised learning methods include, for example, Neural Networks and Probabilistic Methods [138].
- *Reinforcement learning*: a computer program, known as an agent, should learn behaviour to accomplish a particular goal through trial-and-error interactions with a dynamic environment [74]. The training data contains information that is somewhere between supervised and unsupervised learning [80]. The training data does not indicate the correct input-output pairs but instead provides an indicator of whether an action is correct or not. In other words, the program is provided analogous feedback [74, 80].

Some other types of ML techniques to mention are *semi-supervised*, *self-learning*, *deep learning* and *active learning*. However, the techniques listed above are the categories into which methods are traditionally classified [134].

Each ML method has its own set of advantages and disadvantages. Depending on the dataset, a particular algorithm may perform well on some datasets while performing poorly on others [80]. It is also important to reduce errors caused by the algorithm (bias) and data-related inaccuracies. By scaling up the algorithm or scaling down the data, one can eliminate *overfitting*. On the algorithmic side, the former method decreases bias, while the latter method reduces variance on the data side [80]. If the training data is extensive, it can result in high computational costs and cause the learning algorithm to learn noise or coincidental patterns. The data, therefore, needs to be scaled down and processed.

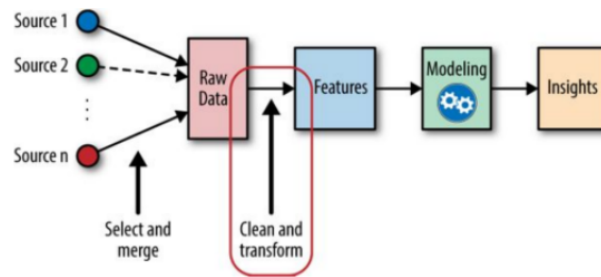


Figure 5.8: The workflow of ML with feature engineering [140]

The process of extracting attributes, properties or attributes that should be used in the ML model, based on domain knowledge, is called *feature engineering* or *feature selection* [133]. Figure 5.8 illustrates that this process takes place after the raw data is acquired, resulting in final features that are subsequently used in the ML modelling. Feature engineering represents “[...] the act of extracting features from raw data and transforming them into formats that are suitable for the machine learning model” [140]. Aside from labelling datasets, feature engineering is regarded as one of the most time-consuming phases of ML, especially when it comes to facilitating APT detection and IDS approaches [9, 50]. Network data must be pre-processed because its raw contents are incompatible with automated IDS approaches [114]. Practitioners agree that datasets need to be revised and that feature engineering and data cleaning is time-consuming [114][140]. Additionally, datasets often have numerous features which may impact the performance of the ML algorithms, and they are also likely to be correlated in some way [114]. Thus, it is necessary to decrease the dimensions of the dataset by discarding any redundant, respectively irrelevant features [50].

5.5 Proof-of-concept

A basic ML model was trained to evaluate if the created datasets would be useful for future use in anomaly-based IDS as a proof-of-concept. It should be emphasised that this section aims to demonstrate and review whether LabelGen generates suitable datasets. The procedure of feature engineering should ideally have been more thorough if time constraints were not a factor.

Support Vector Machine (SVM), a widely used supervised learning model [65]. SVM is a set of similar supervised learning algorithms for classification, and regression [48]. SVM learns by example and assigns labels to objects, and builds a model that predicts the assigned label [95]. An SVM algorithm is non-probabilistic, binary and a linear classifier [48]. A linear classifier will use the object’s characteristics to identify which class or group the object belongs to through a linear combination, $ax + by$ where a and b are constants. [139][8]. Given a set of labelled training data, the SVM learning algorithm builds a model that predicts which of two categories a

new example belongs to [48].

SVM is a powerful and robust data classification technique, but it is not suitable for large datasets. The training complexity of SVM is mainly dependent on the dataset size [22]. Although it has solid theoretical foundations and substantial classification accuracy, it is unsuitable for large datasets. If the training data is immense, there will be significant computing expenditures. In addition, it could also cause the learning algorithm to learn noise or coincidental patterns [80].

5.5.1 Implementation of SVM

The libraries Pandas¹ and scikit-learn² were used in the implementation of SVM. These are some of the most popular tools for ML in Python [99]. The SVM implementation will not be shown in pseudo-code because it follows a typical pattern that I consider irrelevant to the dataset's evaluation.

The generated datasets are imbalanced, as this thesis previously presented in Section 5.2.1, packet statistics. To perform better prediction, an ML algorithm should ideally use a balanced dataset with an even distribution of labels. Thus, a Pandas script was written to randomly select 3-30 rows within each label category, dependent on the quantity, to create a more balanced dataset. Admittedly, this dataset is not properly balanced, but given that several label categories included hundreds of packets, whereas others contained 3-10, narrowing the distribution to this number is useful.

When it comes to feature engineering, according to Stojanovic et al. [114], there are commonly three approaches for selecting features: *wrapper approach*, *filters approach* and *embedded approaches*. The wrapper approach focuses on searching through the space of possible features, with model training and cross-validation performed for each subset of features [114]. Similarly, the filter method also searches through the feature subsets, but it does so before training the model and acts independently of the chosen learning algorithm [114]. Finally, the embedded approach selects features throughout the training process and is often tailored to a single learning algorithm. According to Breiman et al. [17], embedded feature selection is available for most statistical regression and classification models.

The feature engineering approach in this thesis has been the embedded method. I initially removed columns from the dataset that I perceived as irrelevant, e.g. `ip.checksum.status`, `ip.completeness`, `tcp.reassembled`, `http.connection` and `http.user_agent`. These columns generally contained identical values across the dataset, which I presumed would not benefit the model. Once the dataset was reformed, the feature engineering was carried out according to the definition of the embedded approach. I experimented with giving different features to the learning algorithms, adding and removing features to find which combinations gave the best predictions. The final list of features is listed below:

¹<https://pandas.pydata.org/>

²<https://scikit-learn.org/stable/>

- frame.time
- frame.len
- frame.protocols
- eth.type
- ip.hdr_len
- ip.len
- ip.flags
- ip.ttl
- ip.proto
- ip.checksum
- tcp.srcport
- tcp.dstport
- tcp.port
- tcp.stream
- tcp.len
- tcp.seq
- tcp.ack
- tcp.flags
- tcp.window_size_value
- tcp.analysis.bytes_in_flight
- tcp.analysis.push_bytes_sent
- udp.srcport
- udp.dstport
- udp.port
- udp.length
- udp.checksum
- udp.checksum.status
- udp.stream

The columns containing IP and MAC addresses are intentionally excluded from the feature list. These included `ip.dst`, `ip.src`, `eth.dst` and `eth.src`. If these features were provided to the model, it might classify packets based on them, which is undesirable. IP addresses are the easiest properties for adversaries to change once they are detected [49]. Therefore, if an ML model is trained on addresses and applied in IDS, the detection would not be generalisable. Nevertheless, IP addresses associated with the attacks in this work were controlled. However, because the victim VM generated both background and attack traffic, its IP address would appear in numerous columns in the dataset. This led me to the decision to exclude IP addresses entirely, preferring to concentrate on creating features that would be adaptable in the future.

Parameters given to the SVM model were as follows:

```
clf_svm = SVC(C=10, kernel='rbf', gamma=0.0001,
             random_state=42)
```

As seen in the code above, the Radio Basis Function (RBF) was applied as the kernel. The dataset was scaled down to 500 samples and not normalised. A confusion matrix was plotted with the data from the SVM, which can be seen in Figure 5.9.

Confusion matrices are often used to define and visualise the performance of a classification algorithm [109]. They reflect the number of expected and actual values, as seen on the names of the X- and Y-axis in Figure 5.9, "Predicted Label" and "True Label". The model was trained to predict packets labels, and the confusion matrix displays the predicted label for each packet on the X-axis and their actual labels on the Y-axis. The number of places for which the predicted label equals the actual label is represented by the diagonal elements (top left to bottom right). Off-diagonal elements represent the elements that were mislabelled by the classifier [106]. This diagonal line may be seen in Figure 5.9, though most of the data points were in fact mislabelled as mislabelled by the classifier as T1082. Both

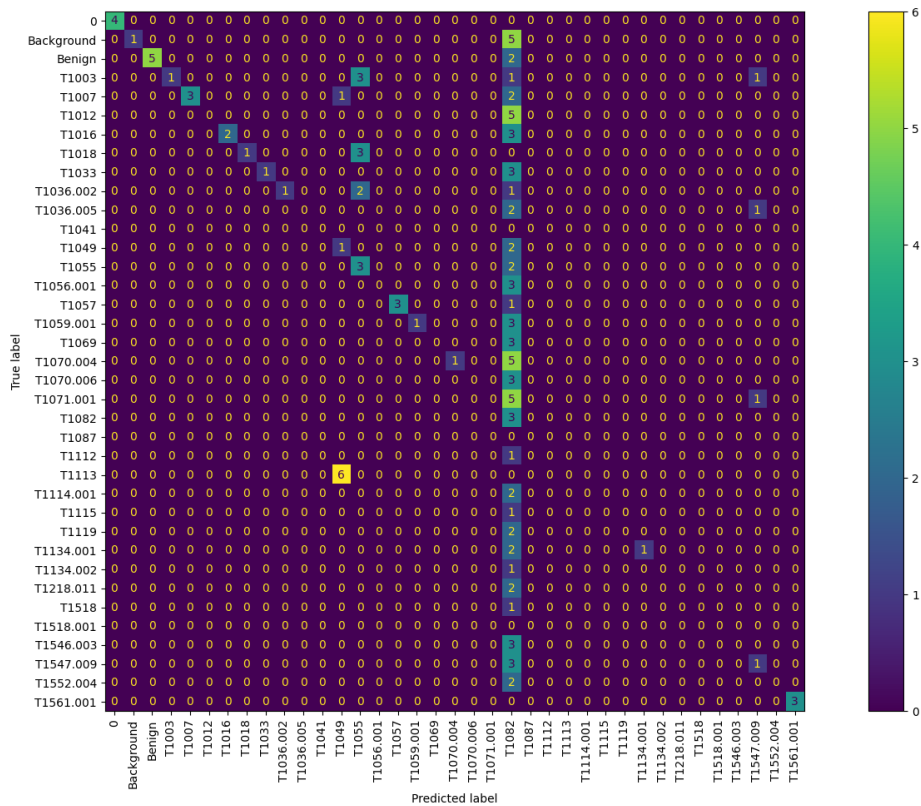


Figure 5.9: Confusion matrix with the new dataset.

background and benign packets were mislabelled. However, the diagonal line on the left side of the figure illustrates that some packets were correctly labelled, even though the majority were not.

Accuracy is a standard metric for evaluating the classifier's or the model's performance. The accuracy of the model trained in this proof-of-concept model was 28.80, representing the percentage of true positive and true negative to all data points. The F1 score, which is often valuable for calculating accuracy on imbalanced datasets such as this one, is 31.85. This indicates the predictive performance of the model. This learning model is the most frequently used

5.6 Summary and discussion

This chapter has presented and evaluated the labelled logfile and the datasets that LabelGen created. Manual inspection indicated that the majority of the packets were labelled, with a percentage of 99.94% labelled packets. LabelGen was further evaluated with a different logfile as input. The results of evaluation suggests that LabelGen correctly labelled these packets as well, with a labelling percentage of 99.95%. However, certain packets were discovered to be mislabelled, which did not reflect in the tables. On

one of the datasets, a proof-of-concept ML model was trained, demonstrating that it was indeed valuable and possible to utilise in ML, despite the poor final results. This is most likely due to the model itself. Several of the topics discussed in this chapter will be addressed in the next chapter.

Chapter 6

Discussion and related work

This chapter will reflect on the work done in this thesis compared to other approaches and alternatives. The first two topics are CALDERA and labelling approaches. LabelGen is discussed next. Following that are some reflections on an alternative approach considered, namely Docker containers. Finally, the tools and alternatives for generating benign traffic will be discussed and how realistic the experiment environment was.

6.1 CALDERA as a framework

CALDERA is primarily designed for autonomous simulation, manual red-team engagements, and automated incident response [5]. CALDERA was not developed for the purpose of capturing network traffic from simulated attacks, which was the focus of this thesis. It has, nevertheless, performed acceptable for this purpose and has proven to be a viable platform for future expansion.

The autonomous aspect of CALDERA is intuitive, and MITRE offers considerable documentation on the framework. The options for attacks to run are adjustable, in the way that the user can run operations from a pre-defined adversary profile, just like APT29, or create a new adversary profile.

The options for attacks are adjustable in that the user can choose to launch attacks from a pre-defined adversary profile, similar to APT29, or construct a new adversary profile. Additionally, operations can be executed without an adversary profile by applying individual commands that map to ATT&CK tactics and techniques. As the agent connects to the C&C server, the operation (attacks) can be executed manually or automatically. While the operation is running, the web interface on the CALDERA server updates the operation flow, displaying the executed commands and their output. I found CALDERA to be straightforward and effective as a framework. However, there are significant drawbacks that apply to the EMU plans.

There seems to be a discrepancy between what MITRE described the EMU plans as capable of and what they achieved in practice. Most of the APT29 EMU plan attacks are attempted, although many fail to execute due to missing payloads or because they were skipped from the EMU plan. I could not identify why these attack steps were missed, but I assume it is related to the LAVA engine in CALDERA, which determines what should be executed. This was described previously in Section 2.4 about CALDERA. Several attack steps were reliant on the preceding attack, requiring output or files provided by the previous attack. Thus, if the first attack failed or was skipped, that would also be the case for the second attack. Consequently, mainly the basic attacks in the APT29 EMU plan were successfully executed, which often comprised short Powershell commands. Furthermore, several of the more complex assaults in the EMU plan failed because of missing payloads.

During the experiment, I kept a record of which attacks were successful, unsuccessful, or skipped, as mentioned briefly Section 4.7 Implementation. The results of this logging are shown in Figure 6.1, as the Excel sheet where I documented the experiment. Each attack step in the EMU plan, including their related ATT&CK tactics and techniques, payloads and whether or not they were successfully executed, are shown in the figure. A significant portion of the attacks that needed payloads was skipped, as shown in the column to the right, colour-coded after their execution status. This figure supports my argument regarding the missing payloads. Furthermore, as seen in Figure 6.1 with yellow rows, I removed attacks 41, 50, 56, and 79 from the EMU plan. Attack 56, in particular, was a difficulty. This attack marks the separation between the two scenarios in the APT29 EMU plan, which were presented in Section 4.3.1. It included shutting down the target machine, which destroyed the agent running on it and terminated the CALDERA agent-server C&C connection. Therefore, the subsequent attacks would not be executed at all. When the machine restarted, the RAT that connects the agent to CALDERA would not automatically start; the C&C connection needed to be initiated manually. This was not reliable, as the whole plan was supposed to automatically execute from start to finish. Thus, this step was deleted from the APT 29 EMU plan. I question the fact that this attack is included in this EMU plan. CALDERA is a tool for autonomous red-team testing, whilst some of the most impacting plugins are faulty. It is indeed probable that the EMU plan is not intended to be performed all at once. To the best of my knowledge, that has never been stated anywhere.

Overall, I conclude that CALDERA provides a solid foundation for mapping to the ATT&CK framework, which was the main objective of this master thesis. Because the ATT&CK taxonomy is widely adopted in the cybersecurity industry [98], I would argue that a dataset with such labels would benefit the community. By mapping to both ATT&CK techniques and groups, particularly APT29, the datasets would contribute to a common understanding. Researchers can collaborate and understand each other's work better when they refer to the common taxonomy. Furthermore, CALDERA has proven to be extensible, allowing for the addition of new attacks and

Ordering	Name	Tactic	Tactic_ID	Technique	Tech_ID	Payload	Works?
1	RTLO Start Sandcat	execution	TA0002	Masquerading: Right-to-Left	T1036.002	cod.3aka3.s	Yes
2	PowerShell	execution	TA0002	Command and Scripting Interp	T1059.001		Yes
3	Automated Collection	collection	TA0009	Automated Collection	T1119		Yes
4	System Network Configuration	discovery	TA0007	System Network Configuration	T1016		Yes
5	System Network Configuration	discovery	TA0007	System Network Configuration	T1016		Yes
6	System Owner / User Discover	discovery	TA0007	System Owner/User Discover	T1033		Yes
7	Data from staged file and Exfiltr	exfiltration	TA0010	Exfiltration Over C2 Channel	T1041	upload.ps1	Skipped
8	Process Discovery	discovery	TA0007	Process Discovery	T1057	ps.ps1	Yes
9	Process Discovery	discovery	TA0007	Process Discovery	T1057		Yes
10	System Service Discovery	discovery	TA0007	System Service Discovery	T1007		Yes
11	System Service Discovery	discovery	TA0007	System Service Discovery	T1007		Yes
12	System Information Discovery	discovery	TA0007	System Information Discovery	T1082		Yes
13	System Information Discovery	discovery	TA0007	System Information Discovery	T1082		Yes
14	Permissions Groups Discovery	discovery	TA0007	Permission Groups Discovery	T1069		Yes
15	Permissions Groups Discovery	discovery	TA0007	Permission Groups Discovery	T1069		Yes
16	Permissions Groups Discovery	discovery	TA0007	Permission Groups Discovery	T1069		Yes
17	Account Discovery	discovery	TA0007	Account Discovery	T1087		Yes
18	Account Discovery	discovery	TA0007	Account Discovery	T1087		Yes
19	Query Registry	discovery	TA0007	Query Registry	T1012		Yes
20	Staging monkey PNG	defensive-evasion	TA0005	Masquerading: Match Legitim	T1036.005	monkey.png	Skipped
21	Bypass User Account Control	privilege-escalati	TA0004	Access Token Manipulation:	T1134.001	update.ps.1	Yes
22	UAC Bypass via Backup Utility	privilege-escalati	TA0004	Abuse Elevation Control Mech	T1548.002		Skipped
23	Registry Cleanup for UAC Byp	defensive-evasion	TA0005	Modify Registry	T1112		Yes
24	Process Injection	privilege-escalati	TA0004	Process Injection	T1055	update.ps.1	Timeout
25	Planting Modified Sysinternals	stage-capabilities	TA0026	Masquerading: Match Legitim	T1036.005	Modified-sys	Skipped
26	Remote System Discovery	discovery	TA0042	Remote System Discovery	T1018		Yes
27	Remote System Discovery	discovery	TA0007	Remote System Discovery	T1018		Yes
28	System Network Configuration	discovery	TA0007	System Network Configuration	T1016		Yes
29	Process Discovery	discovery	TA0007	Process Discovery	T1057		Yes
30	Artifact Cleanup - Delete Files	defensive-evasion	TA0005	Indicator Removal on Host: Fil	T1070.004		Yes
31	Loading Stage-2 & Performing	discovery	TA0007	System Information Discovery	T1082		Yes
32	4.C.2 - System Network Conne	discovery	TA0007	System Network Connections	T1049		Yes
33	Credential Dumping using Proc	credential-access	TA0006	Credential Dumping	T1003		Timeout
34	Persistent Service 1	persistence	TA0003	Boot or Logon Autostart Execu	T1547.009		No
35	Persistent Service 2	persistence	TA0003	Boot or Logon Autostart Execu	T1547.009		No
36	Access Token Manipulation	defensive-evasion	TA0005	Access Token Manipulation	T1134	stealtoken.p	Skipped
37	Credentials In Files-Chrome	defensive-evasion	TA0005	Credential Dumping	T1003		Yes
38	Query Registry	defensive-evasion	TA0005	Access Token Manipulation	T1134	stealtoken.p	Skipped
39	Credentials In Files (T1081) -	credential-access	TA0006	Unsecured Credentials: Privat	T1552.004	dmevals.loc	Yes
40	Staging files for PowerShell m	defensive-evasion	TA0005	Masquerading: Match Legitim	T1036.005		Yes
41	Screen Capturing	collection	TA0009	Screen Capture	T1113		Removed
42	Automated Collection (T1119)	collection	TA0009	Clipboard Data	T1115		Yes
43	Automated Collection (T1119)	collection	TA0009	Input Capture: Keylogging	T1056.001		Yes
44	Data from staged file (T1074)	exfiltration	TA0010	Exfiltration Over C2 Channel	T1041	upload.ps1	Skipped
45	Remote File Copy (T1105)	defensive-evasion	TA0005	Access Token Manipulation:	T1134.001	upload.ps1	Skipped
46	Scheduled Tasks (T1053)	defensive-evasion	TA0005	Access Token Manipulation:	T1134.001	stealtoken.p	Skipped
47	Remote System Discovery (T1	execution	TA0002	Command and Scripting Interp	T1059.001		Yes
48	Identifying current user on oth	execution	TA0002	Command and Scripting Interp	T1059.001		No
49	File and Directory Discovery (T	defensive-evasion	TA0005	Access Token Manipulation:	T1134.001	stealtoken.p	Skipped
50	Copy Sandcat File	lateral-movement	TA0008	Ingress (Lateral) Tool Transfe	T1570	sandcat.go-	Removed
51	Screen Capture (T1113)	collection	TA0009	Screen Capture	T1113	Get-screens	Yes
52	File and Directory Discovery (T	discovery	TA0007	File and Directory Discovery	T1083		Skipped
53	Automated document collector	execution	TA0002	Command and Scripting Interp	1059.001	rar.exe	Timeout
54	Data from staged file (T1074)	exfiltration	TA0010	Exfiltration Over C2 Channel	T1041	upload.ps1	Skipped
55	Artifact Cleanup - Delete Stage	defensive-evasion	TA0005	Indicator Removal on Host: Fil	T1070.004		No
56	Scheduled Task	impact	TA0040	System Shutdown/Reboot	T1529		Removed
57	Artifact Cleanup	defensive-evasion	TA0005	Indicator Removal on Host: Fil	T1070.004		Skipped
58	Startup Folder Persistence Exe	lateral-movement	TA0008	Boot or Logon Initialization Sc	T1037.005		Skipped
59	Click .LNK payload	execution	TA0002	User Execution: Malicious File	T1204.002		Skipped
60	Timestomp kxwn.lock	defensive-evasion	TA0005	Indicator Removal on Host: Fil	T1070.006	timestomp.ps	No
61	Detect Anti-Virus	discovery	TA0007	Software Discovery: Security	T1518.001	stepTwelve	Yes
62	Detect Software	discovery	TA0007	Software Discovery	T1518	stepTwelve	Yes
63	Enumerate Computer Name	discovery	TA0007	System Information Discovery	T1082	stepThirteen	Yes
64	Enumerate Domain Name	discovery	TA0007	System Information Discovery	T1082	stepThirteen	Yes
65	Enumerate Username	discovery	TA0007	System Owner/User Discover	T1033	stepThirteen	Yes
66	Enumerate Processes	discovery	TA0007	Process Discovery	T1057	stepThirteen	Yes
67	UAC Bypass via sdctl	defensive-evasion	TA0005	Access Token Manipulation:	T1134.002	stepFourteen	Yes
68	Credential Dumping	credential-access	TA0006	Credential Dumping	T1003	stepFourteen	Yes
69	Stage Mimikatz Binary	credential-access	TA0006	Credential Dumping	T1003	m.exe	Unsure
70	WMI Persistence technique	persistence	TA0003	Event Triggered Execution: W	T1546.003	stepFifteen	No
71	Enumerate Domain Controller	discovery	TA0007	Remote System Discovery	T1018	powerview.p	No
72	Enumerate Domain SID discov	discovery	TA0007	System Owner/User Discover	T1033	stepSixteen	Yes
73	Remote Connection (T1028) &	lateral-movement	TA0008	Ingress (Lateral) Tool Transfe	T1570	invoke_winnr	Skipped
74	Collect E-mails	collection	TA0009	Email Collection: Local Email	T1114.001	stepSevent	Yes
75	Collect Files & Compress Colle	collection	TA0009	Data from Local System	T1005	stepsevent	Skipped
76	Exfiltrate data to OneDrive	exfiltration	TA0010	Transfer Data to Cloud Accou	T1537		Skipped
77	Data Wiping of staged files	impact	TA0040	Disk Wipe: Disk Content Wipe	T1561.001	wipe.ps1	No
78	Execute Invoke-Mimikatz	credential-access	TA0006	Credential Dumping	T1003	Invoke-mimik	Skipped
79	Triggering Persistent	persistence	TA0003	Signed Binary Proxy Executio	T1218.011		Yes

Figure 6.1: APT29 Emulation plan: planned and executed steps.

the creation of new adversary profiles. Custom commands are also possible, suggesting that CALDERA might be used with other frameworks or tools to launch additional attacks. This way, one can adapt to using various tools while keeping CALDERA as the foundation. In conclusion, both CALDERA and the EMU plans are active research projects at MITRE, promising further improvements on both products and their integration [45][10].

6.1.1 Alternative tools for emulation

Because of its mappings to ATT&CK techniques, groups and EMU plans, CALDERA has demonstrated to be a practical choice of tool for simulating attacks. As MITRE continues to upgrade ATT&CK, the developers working on CALDERA are likely to update the framework. Frequent updates are one of the advantages of CALDERA. Other tools, however, could be employed in this thesis as well. Popular penetration testing tools such as Metasploit or Cobalt Strike¹ are among the alternatives. Such tools, which are designed primarily for penetration testing, are frequently employed in actual malware, according to a threat report from Recorded Future [103]. Cobalt Strike was discovered to be the most commonly used C&C software by adversaries in 2020 [103]. In addition to C&C, the tool has a wide range of other attacks, including reconnaissance, keylogging, and the development of Trojan horse malware [46]. If traffic from this tool could be captured, it would be highly beneficial for detection. However, because Cobalt Strike is closed source and requires a licence, there are certain limitations [66]. Metasploit is a tool that supports the complete attack scenario and is frequently used by attackers [63]. Metasploit offers the ability to scan for and exploit known vulnerabilities, and it has a wide array of attacks. However, there is no mapping to ATT&CK tactics nor APT groups.

Atomic Red Team² is a collection of short and convenient tests that has a mapping to ATT&CK techniques and hence might be used to simulate attacks. However, it does not provide the same level of continuity as CALDERA. It would require building a PowerShell module to run the tests automatically. Nevertheless, CALDERA automated this process with EMU plans.

6.2 Labelling network traffic

The labelling approach in this thesis was the combination of behavioural profiles and injection timing. Behaviour profiles employ the behaviour generated by profiles, which are typically computer programmes, to label datasets [62]. The behaviour profile for attack traffic, and thus general C&C labels, were applied to the IP addresses of the victim VM and the attacker VM. Based on the timestamps in the CALDERA report, the injection timing approach was also used to label attack traffic on a technique level. This method of labelling would not qualify as a behavioural profile because it is

¹<https://www.cobaltstrike.com/>

²<https://atomicredteam.io/>

merely based on the timing value [62]. This approach was used for labelling the widely used public dataset CICIDS2017 [107].

In principle, behavioural profiles and injection timing would not guarantee that no mislabelling would occur [55]. Labelling by IP addresses and the behaviour profiles that these addresses represented proved to be a suitable approach in this thesis. Applying labels based on IP addresses might not be the best approach. Attack traffic can (and often will) blend in with traffic originating from a supposedly benign IP address, i.e. if the machine is infected. However, the environment was controlled, and the malware was isolated in this work. The respective networks assigning IP addresses to the behavioural profiles were sufficiently controlled to make this method adequate. The manual inspection also indicated that IP-based labelling correctly labelled malicious traffic. No malicious characteristics in traffic labelled "background" or "benign" occurred during the same time as the attacks, indicating that the behavioural profile combined with timing injection was a suitable approach to labelling in this thesis.

I decided on a labelling approach in which a network logfile is labelled and subsequently exported to datasets in CSV and JSON formats. These datasets include both raw packet data and a label, as previously shown in Section 5.1. Unlike datasets derived from .pcap files, flow data has been the data format in several of the publicly accessible datasets that are often utilised by the cybersecurity community [26] [111]. In my datasets, I do not consider *IPFIX* or Netflow to be an appropriate format. *IPFIX* summarises the network packets by properties such as ports, protocols or timeslots [130]. Hence, all the packets between the victim VM and the attacker VM would be compressed into flow data. The CALDERA attacks use the same TCP and UDP ports for the entire operation, and execute the attacks sequentially. *IPFIX* would therefore summarise these packets. Consequently, most of the CALDERA attacks' timestamps would be missing from the flow data. The resulting datasets would not be fine-grained enough to include labels for each attack technique. Finally, timestamps and injection timing are the only options for technique-level labelling in this work, in my opinion.

Nonetheless, because LabelGen generated a labelled .pcap file, its raw format provides various options for later use. To display statistics or create IDS detection rules, the file could be exported to *IPFIX* data. Because .pcap can be converted to *IPFIX* but not the other way around, the .pcap format allows for more flexibility.

6.3 LabelGen

LabelGen was created to label network logfiles from CALDERA. As a result, it is limited to the framework and highly dependent on the CALDERA report's format. The labelling will fail if LabelGen is given a .pcap file and a technique specification source other than CALDERA. Therefore, I con-

clude that LabelGen is not generalisable. Because CALDERA is updated regularly, the developers will add more attacks when ATT&CK is updated. LabelGen would successfully label a logfile with a newer CALDERA attack, such as Log4j, as long as the CALDERA post-attack report follows its current format.

In terms of usage, I consider LabelGen to be clear and understandable. It only needs minor alterations to work with a logfile other than the one used in this thesis. As per best practice principles, variables that are likely to change frequently, such as IP addresses and logfiles, are conveniently assembled.

The purpose of LabelGen has been to investigate whether a concept worked or not. Based on the results presented in Chapter 5 one might suggest that the labelling was accurate. LabelGen generated labels for all of the attacks in the APT29 EMU plan and benign and background labels. I conclude that the general idea of LabelGen was achieved based on the findings in Chapter 5.

However, the results presented and the labelling approach can not guarantee 100% correct labelling. Because manual inspection of each packet is impractical for files of this size, only a sample of the packets formed the basis for evaluation.

The developed proof-of-concept ML model also shows that the dataset can be used to train a more complex model in the future, which might apply to the NIDS detection approach. However, the feature engineering and learning algorithms should be improved before being used in real-world situations.

6.4 Use of containers/DetGen

The framework DetGen created by Clausen et al. [26] was used in the work by Asprusten et al. in [6]³. DetGen generates various attacks using Docker containers, an approach originally discussed for this thesis. We discussed whether I could extend and improve in [6] by emulating APT attacks in containers. Container technology could be used to label attack, benign, and background traffic accurately. The logging and subsequent labelling control is the main advantage of executing attacks within containers. The containers isolate the attacks and network traffic, allowing for collecting ground truth information about the origin of the traffic [26]. Because of the controlled environment, ground truth information implies that one can be certain which traffic is malicious or not. Additionally, the dataset's labelling would gain a higher percentage of accuracy. While 'noise' is introduced in some contexts and this thesis's experiment, such features could be added to the containers as an option to create a more realistic environment. This was done by Clausen et al. with DetGen [26].

³My summer internship at FFI, June-August 2021.

As initial testing showed, the container approach was not practical for the experiment conducted. If I labelled traffic from each container as separate attacks, this required a significant amount of work to add each attack to different containers. This procedure was far too extensive, resulting in a loss of context between the various attack steps in the EMU plan.

Because containers only perform their assigned tasks and then stop, maintaining state between attack steps would be challenging. Attack steps dependent on input from previous steps would fail to execute. If the same container executed all the attacks, this approach would be no different from what I did in the experiment. The IP addresses could be compared to containers in this case because the networks were carefully regulated to isolate and control traffic and malware.

Although the CALDERA server can run in a Docker container, the Agent on the victim machine cannot. As a result, using Docker containers in this thesis would require a significant effort to get everything functioning properly, resulting in a trade-off between reward and work.

In addition to running benign and malicious scenarios in Docker containers, DetGen also addresses the challenges presented in Section 3.4 regarding unrealistic network traffic in current datasets. They solved this problem by adding latency and randomisation to the containerised traffic [26]. However, because the aim was to have the experiment as automated as possible, I chose GHOSTS. While writing this thesis, DetGen requires manual input for the scenarios to execute, which would be time-consuming and possibly distracting during the experiment. Instead, I wanted to concentrate entirely on the experiment and document how this proceeded. Even though I lost some elements of realism, I would argue that it was otherwise not that unrealistic to use DetectionLab and GHOSTS to generate the background and benign traffic. Both tools are developed for use in experiments, focusing on realism.

6.5 GHOSTS and its alternatives

I had experience with GHOSTS from the work I contributed within [6], and thus it was chosen as the framework for generating benign traffic. It is relatively simple to determine which tasks the GHOSTS Client (Non-Player Character, NPC) should perform, and it generates background noise that adversaries frequently blend in with [93]. On the other hand, more detailed documentation of GHOSTS would be beneficial. This is a weakness of using GHOSTS because it requires a complex setup. It runs three Docker containers and requires some prerequisites to function properly.

CALDERA has a *human* plugin that provides a similar feature to GHOSTS in terms of simulating benign users. Others have managed to successfully simulate benign user activity using this plugin [19]. However, I could not complete his task, which could be due to the CALDERA version, but the documentation did not guide how to resolve the issue.

DetGen could have been used to generate benign traffic as well, but there were some technical issues with running nested virtualised environments. Instead, with the GHOSTS approach, the synthetic realism that DetGen adds was for this experiment actual network congestion.

6.6 The environment

I did some testing on two different virtual machines in preparation for the experiment. I tried setting up a C&C connection by installing a CALDERA agent on the victim VM. Windows Defender immediately flagged this on the victim VM, which I had to turn off to establish a foothold and C&C contact. For the CALDERA attacks to work, the group policy for allowing to execute scripts was required, and the firewalls were turned off. Successful attacks were required to provide the most valuable results for the datasets. Although such settings are not ideal in real-world situations, this policy may apply to some organisations or users with higher privileges.

In the DetectionLab setup, Windows Defender is by default disabled. Because DetectionLab should assist defenders in testing products and experiments, the attacks must work. On the Ubuntu VM (Logger), DetectionLab comes with the SIEM tool Splunk that could configure rules to detect CALDERA events. Since CALDERA was flagged by Windows Defender during initial testing, similar rules could be added to Splunk to create event alerts.

Real environments may experience fluctuations and faults introduced by the complexity of modern networking. Examples include packet delays through network congestion, unexpected connection drops or resets, and out-of-order arrivals. All these factors result in variations in real network traffic, which is often a missing element in separated or virtualised environments [26]. This aspect was also missing in this experiment. However, I would argue that the focus of this thesis was on creating fine-grained labels for the dataset rather than creating realistic conditions in its environment.

One important aspect of the experiment was the attacker's presence in an enterprise environment. The attacker and victim VMs were supposed to be on the same network, similar to a real-world attack. They were distinct network entities with different IP addresses. Containerisation, as described in Section 6.4, would not contribute to a realistic environment. The VMs could not easily communicate, which was necessary for creating background traffic. DetectionLab was a better option because it automates the creation of the enterprise environment and the experiment background operations.

It is safe to conclude that the experiment environment in this work was not realistic. The security mechanisms normally in place were intentionally turned off. The objective of this project was to carry out successful attacks that generated network traffic and produced labelled datasets, and the created environment for this was built without major changes.

Chapter 7

Conclusion and future work

This thesis has presented the experiment conducted with CALDERA to generate an APT network logfile using the EMU plan for APT29. LabelGen processed the logfile and generated one fine-grained labelled logfile and two fine-grained labelled datasets. The labelling technique used by Landauer et al. inspired the labelling approach in LabelGen. The APT attacks were distinguishable on a technique level. A dedicated chapter presented and reviewed the labelled files, suggesting that accurate labelling was applied.

This chapter will summarise and analyse the results of this thesis with the research question. A final section identifies areas for future work.

7.1 Summary of results and findings

Can MITRE CALDERA be used to generate a dataset of APT behaviour with fine-grained labels that can identify different stages of the attack?

The goal of this thesis was to find out if CALDERA could be used to create a fine-grained labelled dataset in which the stages of APT attacks could be identified. The results presented in this thesis suggests CALDERA is appropriate for this purpose.

The EMU plan in CALDERA emulated the adversary group APT29 by attacking a host on a simulated enterprise network in a controlled experiment. The network traffic generated by the activities related to attacks, background and benign behaviour during the attacks were captured. The logfile contained 1 679 966 network packets. The resulting logfile proves that the APT29 EMU plan in CALDERA did successfully generate APT traffic.

From the logfile, LabelGen created one labelled logfile and two labelled datasets. LabelGen automates the labelling process, starting with an unlabelled .pcap file and ending with three labelled files in their respective formats. The labelled logfile was shown to contain 7479 packets related to

the APT29 emulation (0.44%), 1 488 521 packets labelled as background (88%) and 183 037 packets labelled as benign ($\approx 11\%$). 929 packets (0.06%) remained unlabelled. A close examination indicated that the packets labelled as APT29 attacks contained traffic or data from CALDERA. Thus, LabelGen concluded to be able to generate labelled datasets from the APT logfile.

LabelGen was evaluated by conducting a new experiment and using the generated logfile as input. Inspection of the labelled files indicated that LabelGen successfully managed to label packets related to different CALDERA attacks.

A simple ML model was trained to predict labels on a subset of the labelled dataset. As a confusion matrix illustrated, and accuracy scores revealed, the model did not perform satisfactory. It did, however, demonstrate the principle that the dataset can be used to train a ML for IDS.

According to the findings of this thesis, it could be concluded that CALDERA can be used to generate a dataset of APT behaviour with fine-grained labels that can identify stages of the attack. Packet statistics, manual inspection, proof-of-concept ML all provided indications that CALDERA could generate APT datasets.

7.2 Future work

Optimisation of LabelGen are among the suggestions for future work. LabelGen is only capable of reading attack information from the CALDERA report in the format it was created after the experiment was completed. Therefore, if multiple hosts are involved in the attacks, or if the report contains missing values, LabelGen will stop execution. To address these concerns, future work should include improvements to LabelGen to handle inconsistent report structures.

LabelGen could function as a framework in future usage. It might be extended to label from other sources than CALDERA. LabelGen gives the user a complete labelling process, from providing a raw network logfile, to three resulting labelled files. This saves time, as the whole process is automated. Furthermore, because CALDERA is mapped to ATT&CK, any changes to ATT&CK are eventually reflected in CALDERA. Therefore, LabelGen would be able to label new CALDERA attacks, which proved to be one of the main advantages of the framework. With extending the functionality of LabelGen it could be a framework for other labelling techniques, as the behavioural profiles and timing injection methods are included in the script. LabelGen would be able to label new CALDERA attacks in the future. This was one of the benefits of using a highly relevant tool as CALDERA in this thesis.

To properly determine the value of the SVM model, it should be tested without training first. In addition, to thoroughly evaluate the datasets, a more complex model needs to be trained. Because this thesis only used ML as

a proof-of-concept to demonstrate the potential of the dataset, a more sophisticated classifier would most certainly provide a better representation of the quality of the datasets.

Future work should make use of Docker containers, with a focus on the DetGen framework. The prospect of extending DetGen with CALDERA is an interesting option to consider, especially since CALDERA can already run in a container. Containerising the attacks isolates them and ensures ground truth labels because each container produces attack traffic with no background or noise. DetGen was skipped due to time constraints, but it would undoubtedly be a priority for future work.

The network logs could be integrated with host logs to provide insightful information. Fikret Kadiric, a student colleague, conducted the same experiment described in this thesis, but collected host logs instead of network logs [73]. Fikret used Sysmon, a Windows service that monitor and logs system activities, in combination with CALDERA to generate a labelled dataset. The logs generated from the attacks are therefore host-based. If we could merge our datasets together, it would give a bigger overview of how the attacks interact with the system; more context, information and data to develop detection capabilities. Combining the two approaches might make the attack kill chain more evident. Time constraints were again the reason why this integration did not happen, but it would be an interesting research topic for future work.

Bibliography

- [1] Otis Alexander, Misha Belisle, and Jacob Steele. *Mitre att&ck® for industrial control systems: Design and philosophy*, 2020.
- [2] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials*, 21(2):1851–1877, 2019.
- [3] Blake Anderson, Subharthi Paul, and David McGrew. Deciphering malware’s use of tls (without decryption). *Journal of Computer Virology and Hacking Techniques*, 14(3):195–211, 2018.
- [4] Andy Applebaum, Doug Miller, Blake Strom, Henry Foster, and Cody Thomas. Analysis of automated adversary emulation techniques. In *Proceedings of the Summer Simulation Multi-Conference*, pages 1–12, 2017.
- [5] Andy Applebaum, Doug Miller, Blake Strom, Chris Korban, and Ross Wolf. Intelligent, automated red team emulation. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 363–373, 2016.
- [6] Markus Asprusten, Julie Gjerstad, Gudmund Grov, Espen Kjellstadli, Robert Flood, Henry Clausen, and David Aspinall. A containerised approach to labelled c&c traffic: Short paper. In *Norsk IKT-konferanse for forskning og utdanning*, number 3, 2021.
- [7] ATT&CK Evaluations. Apt29 enterprise evaluation 2019. <https://attackevals.mitre-engenuity.org/enterprise/apt29>, 2019. [Online; last updated June 14, 2021], [Last accessed: April 19, 2022].
- [8] Sheldon Axler. *Linear algebra done right*. Springer Science & Business Media, 1997.
- [9] Sherenaz W Al-Haj Baddar, Alessio Merlo, and Mauro Migliardi. Anomaly detection in computer networks: A state-of-the-art review. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 5(4):29–64, 2014.
- [10] Jon Baker and Forrest Carver. Introducing the all-new adversary emulation plan library, September 2020. [Online; published September 10, 2020], [Accessed: April 5, 2022].

- [11] Jerry Banks. *Discrete event system simulation*. Pearson Education India, 2005.
- [12] Anaël Beaugnon, Pierre Chifflier, and Francis Bach. Ilab: An interactive labelling strategy for intrusion detection. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 120–140. Springer, 2017.
- [13] Jürgen Bernard, Marco Hutter, Matthias Zeppelzauer, Dieter Feller, and Michael Sedlmair. Comparing visual-interactive labeling with active learning: An experimental study. *IEEE transactions on visualization and computer graphics*, 24(1):298–308, 2017.
- [14] Monowar H Bhuyan, Dhruba K Bhattacharyya, and Jugal K Kalita. Towards generating real-life datasets for network intrusion detection. *Int. J. Netw. Secur.*, 17(6):683–701, 2015.
- [15] Elmarie Biermann, Elsabe Cloete, and Lucas M Venter. A comparison of intrusion detection systems. *Computers & Security*, 20(8):676–683, 2001.
- [16] Michael Bose. Virtualbox network settings: Complete guide. <https://www.nakivo.com/blog/virtualbox-network-setting-guide/>. Online; Published: July 16, 2019. Accessed: April 10, 2022.
- [17] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.
- [18] Nevil Brownlee, Cyndi Mills, and Greg Ruth. Traffic flow measurement: Architecture. Technical report, RFC 2722, 1999.
- [19] Molly Buchanan, Jeffrey W Collyer, Jack W Davidson, Saikat Dey, Mark Gardner, Jason D Hiser, Jeffrey Lang, Alastair Nottingham, and Alina Oprea. On generating and labeling network traffic with realistic, self-propagating malware. *arXiv preprint arXiv:2104.10034*, 2021.
- [20] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.
- [21] Carlos A Catania and Carlos García Garino. Automatic network intrusion detection: Current techniques and open issues. *Computers & Electrical Engineering*, 38(5):1062–1072, 2012.
- [22] Jair Cervantes, Xiaou Li, Wen Yu, and Kang Li. Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, 71(4-6):611–619, 2008.
- [23] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

- [24] Zhuo Chen, Ruizhou Ding, Ting-Wu Chin, and Diana Marculescu. Understanding the impact of label granularity on cnn-based image classification. In *2018 IEEE international conference on data mining workshops (ICDMW)*, pages 895–904. IEEE, 2018.
- [25] Cisco. Cisco encrypted traffic analytics. White paper, Cisco, 2021. Online; accessed May 12, 2022.
- [26] Henry Clausen, Robert Flood, and David Aspinall. Traffic generation using containerization for machine learning. *arXiv preprint arXiv:2011.06350*, 2020.
- [27] Henry Clausen, Gudmund Grov, and David Aspinall. Cbam: A contextual model for network anomaly detection. *Computers*, 10(6):79, 2021.
- [28] Lockheed Martin Corporation. Applying cyber kill chain@methodology. White paper, Lockheed Martin Corporation, 2015. Online; accessed May 11, 2022.
- [29] MITRE Corporation. Automated adversary emulation platform, 2022. Last accessed: 08 May 2022.
- [30] Oracle Corporation. Chapter 6. virtual networking. <https://www.virtualbox.org/manual/ch06.html>. Online; Last accessed: May 4th, 2022.
- [31] The Mitre Corporation. Common Vulnerabilities And Exposures (CVE) history. <https://cve.mitre.org/about/history.html>. Last accessed: February 23, 2021.
- [32] The MITRE Corporation. Common Weakness Enumeration cwe - about - cwe overview. <https://cwe.mitre.org/about/index.html>. Last accessed: May 10, 2022.
- [33] The Mitre Corporation. MITRE ATT&CK apt29. <https://attack.mitre.org/groups/G0016/>. Last accessed: May 9, 2022.
- [34] The Mitre Corporation. MITRE ATT&CK frequently asked questions. <https://attack.mitre.org/resources/faq/>. Last accessed: February 24, 2021.
- [35] The Mitre Corporation. MITRE ATT&CK getting started. <https://attack.mitre.org/resources/getting-started/>. Last accessed: February 24, 2021.
- [36] The Mitre Corporation. MITRE ATT&CK matrix - enterprise. <https://attack.mitre.org/matrices/enterprise/>. Last accessed: May 10, 2022.
- [37] The Mitre Corporation. MITRE ATT&CK®. <https://attack.mitre.org/>. Last accessed: May 10, 2022.

- [38] The Mitre Corporation. MITRE ATT&CK® Framework - youtube. <https://www.youtube.com/watch?v=Yxv1suJYMI8>. Date: January 25, 2021.
- [39] The Mitre Corporation. The MITRE Corporation corporate overview. <https://www.mitre.org/>. Last accessed: May 10, 2022.
- [40] The Mitre Corporation. The MITRE Corporation matrix - enterprise. <https://attack.mitre.org/matrices/enterprise/containers/>. Last accessed: May 30, 2021.
- [41] The Mitre Corporation. The MITRE Corporation media resources. <https://www.mitre.org/news/media-resources>. Last accessed: February 23, 2021.
- [42] The Mitre Corporation. The MITRE Corporation mobile - matrix. <https://attack.mitre.org/matrices/mobile/>. Last accessed: May 22, 2021.
- [43] The Mitre Corporation. The MITRE Corporation national cybersecurity ffrdc. <https://www.mitre.org/centers/national-cybersecurity-ffrdc/who-we-are>. Last accessed: February 23, 2021.
- [44] The Mitre Corporation. The MITRE Corporation our history. <https://www.mitre.org/about/our-history>. Last accessed: February 23, 2021.
- [45] The Mitre Corporation. Welcome to caldera’s documentation! <https://caldera.readthedocs.io/en/latest/index.html>. Last accessed: May 10, 2022.
- [46] The MITRE Corporation. Cobalt strike, software s0154, 2021. Last accessed: 17 August.
- [47] The MITRE Corporation. Command and control, tactic ta0011, 2021. Last accessed: 17 August.
- [48] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [49] David Bianco. The pyramid of pain. <https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html>, 2013. [Online; accessed 14-April-2022].
- [50] Jonathan J Davis and Andrew J Clark. Data preprocessing for anomaly based network intrusion detection: A review. *computers & security*, 30(6-7):353–375, 2011.
- [51] Michael J De Lucia and Chase Cotton. Detection of encrypted malicious network traffic using machine learning. In *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pages 1–6. IEEE, 2019.

- [52] Frank Duff. Round 2 of ATT&CK Evaluations is Now Open. <https://medium.com/mitre-attack/attack-evals-round-2-c3ea383ba55d>, May 1, 2019. Accessed: April 7, 2022.
- [53] Frank Duff. Att&ck evaluation apt29 emulation plan added to the emulation plan library, 2021. Last accessed: 01 May 2022.
- [54] Thomas Edgar and David Manz. *Research methods for cyber security*. Syngress, 2017.
- [55] Gints Engelen, Vera Rimmer, and Wouter Joosen. Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12. IEEE, 2021.
- [56] MITRE Engenuity. Center for threat-informed defense - home, 2022. Last accessed: 01 May 2022.
- [57] F-Secure. The dukes: 7 years of russian cyberespionage. Technical report, F-Secure, 2015.
- [58] Xin Fan, Chenlu Li, Xiaoru Yuan, Xiaoju Dong, and Jie Liang. An interactive visual analytics approach for network anomaly detection through smart labeling. *Journal of Visualization*, 22(5):955–971, 2019.
- [59] Wireshark Foundation. Wireshark manual, 7.7. time zones. https://www.wireshark.org/docs/wsug_html_chunked/ChAdvTime-zones.html. Online; last accessed: May 13, 2022.
- [60] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *computers & security*, 45:100–123, 2014.
- [61] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 46:235–262, 2013.
- [62] Jorge Guerra, Carlos Catania, and Eduardo Veas. Datasets are not enough: Challenges in labeling network traffic. *arXiv preprint arXiv:2110.05977*, 2021.
- [63] Himanshu Gupta and Rohit Kumar. Protection against penetration attacks using metasploit. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, pages 1–4. IEEE, 2015.
- [64] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
- [65] Mitsuhiro Hatada, Matthew Scholl, et al. An empirical study on flow-based botnet attacks prediction. *National Institute of Standards and Technology Technical Note*, 2111, 2020.

- [66] HelpSystems. Adversary simulation and red team operations software - cobalt strike, 2021. Last accessed: 17 August 2021.
- [67] Brandon Heung, Hung Chak Ho, Jin Zhang, Anders Knudby, Chuck E Bulmer, and Margaret G Schmidt. An overview and comparison of machine-learning techniques for classification purposes in digital soil mapping. *Geoderma*, 265:62–77, 2016.
- [68] Hanan Hindy, Elike Hodo, Ethan Bayne, Amar Seeam, Robert Atkinson, and Xavier Bellekens. A taxonomy of malicious traffic for intrusion detection systems. In *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, pages 1–4. IEEE, 2018.
- [69] Software Engineering Institute. cmu-sei/ghosts | github, 2021. Last accessed: 20 August 2021.
- [70] Manel Jerbi, Zaineb Chelly Dagdia, Slim Bechikh, and Lamjed Ben Said. On the use of artificial malicious patterns for android malware detection. *Computers & Security*, 92:101743, 2020.
- [71] Portswigger John Leyden at The Daily Swig. Who is behind apt29? what we know about this nation-state cybercrime group. <https://portswigger.net/daily-swig/who-is-behind-apt29-what-we-know-about-this-nation-state-cybercrime-group>, July 2020. [Online; published July 23, 2020], [Accessed: April 6, 2022].
- [72] VVRPV Jyothsna, Rama Prasad, and K Munivara Prasad. A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 28(7):26–35, 2011.
- [73] Fikret Kadiric. Generating labeled apt host datasets using caldera and sysmon. Master’s thesis, University of Oslo, 2022.
- [74] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.
- [75] H Günes Kayacik, A Nur Zincir-Heywood, and Malcolm I Heywood. Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. In *Proceedings of the third annual conference on privacy, security and trust*, volume 94, pages 1723–1722. Citeseer, 2005.
- [76] John R Koza, Forrest H Bennett, David Andre, and Martin A Keane. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In *Artificial Intelligence in Design’96*, pages 151–170. Springer, 1996.
- [77] Max Landauer, Florian Skopik, Markus Wurzenberger, Wolfgang Hotwagner, and Andreas Rauber. Have it your way: Generating customized log datasets with a model-driven simulation testbed. *IEEE Transactions on Reliability*, 70(1):402–415, 2020.

- [78] Shane Legg, Marcus Hutter, et al. A collection of definitions of intelligence. *Frontiers in Artificial Intelligence and applications*, 157:17, 2007.
- [79] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.
- [80] Han Liu, Alexander Gegov, and Mihaela Cocea. *Rule based systems for big data: a machine learning approach*, volume 13. Springer, 2015.
- [81] Chris Long. Detectionlab introduction. <https://detectionlab.network/>. Last accessed: May 9, 2022.
- [82] Chris Long. Introduction: Detection lab. <https://medium.com/@clong/introducing-detection-lab-61db34bed6ae>, December 2017. Online; Published: December 11, 2017. Accessed: April 9, 2022.
- [83] Gabriel Maciá-Fernández, José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and Roberto Therón. Ugr ‘16: A new dataset for the evaluation of cyclostationarity-based network idss. *Computers & Security*, 73:411–424, 2018.
- [84] Mohammad Masdari and Hemn Khezri. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Applied Soft Computing*, 92:106301, 2020.
- [85] LLC McAfee. McAfee what is the mitre att&ck framework? <https://www.mcafee.com/enterprise/en-us/security-awareness/cybersecurity/what-is-mitre-attack-framework.html>. Last accessed: May 22, 2021.
- [86] Microsoft Threat Intelligence Center (MSTIC) and Microsoft 365 Defender Threat Intelligence Team. New sophisticated email-based attack from nobelium. <https://www.microsoft.com/security/blog/2021/05/27/new-sophisticated-email-based-attack-from-nobelium/>, 2019. [Online; published May 27, 2021], [Last accessed: April 19, 2022].
- [87] MIT Lincoln Labs. 1998 darpa intrusion detection evaluation dataset, 1998. Available online:<https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset> (Last accessed April 16 2022).
- [88] Tom M Mitchell. Artificial neural networks. *Machine learning*, 45:81–127, 1997.
- [89] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

- [90] Sowmya Myneni, Ankur Chowdhary, Abdulhakim Sabur, Sailik Sengupta, Garima Agrawal, Dijiang Huang, and Myong Kang. Dapt 2020-constructing a benchmark dataset for advanced persistent threats. In *International Workshop on Deployable Machine Learning for Security Defense*, pages 138–163. Springer, 2020.
- [91] The National Cyber Security Centre (NCSC). Advisory: Apt29 targets covid-19 vaccine development. Technical report, The National Cyber Security Centre (NCSC), 2020.
- [92] The National Cyber Security Centre (NCSC). Uk and us call out russia for solarwinds compromise. <https://www.ncsc.gov.uk/news/uk-and-us-call-out-russia-for-solarwinds-compromise>, April 2021. [Online; published April 15, 2021], [Accessed: April 6, 2022].
- [93] Georgi Nikolov. Simulate user activity with the ghosts framework : Introduction, April 2020. Last accessed: April 15 2022.
- [94] Nils J Nilsson and Nils Johan Nilsson. *Artificial intelligence: a new synthesis*. Morgan Kaufmann, 1998.
- [95] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.
- [96] US Department of Homeland Security (DHS) and the Federal Bureau of Investigation (FBI). Grizzly steppe – russian malicious cyber activity. Technical report, US Department of Homeland Security (DHS), 2016. Available online; https://media.defense.gov/2021/Apr/15/2002621240/-1/-1/0/CSA_SVR_TARGETS_US_ALLIES_UOO13234021.PDF/CSA_SVR_TARGETS_US_ALLIES_UOO13234021.PDF (Last accessed April 19, 2022).
- [97] National Institute of Standards and Technology. Nist computer security resource center (csrc) - glossary. <https://csrc.nist.gov/glossary>, March 2022. [Online; last updated March 10, 2022], [Accessed: April 6, 2022].
- [98] Kris Oosthoek and Christian Doerr. Sok: Att&ck techniques and trends in windows malware. In *International Conference on Security and Privacy in Communication Systems*, pages 406–425. Springer, 2019.
- [99] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [100] David Poole, Alan Mackworth, and Randy Goebel. Computational intelligence. 1998.

- [101] Prem Maurya. Polymorphism in java — GeeksforGeeks. <https://www.geeksforgeeks.org/polymorphism-in-java/>, 2020. Accessed: May 30, 2021.
- [102] Alex Ratner, Stephen Bach, Paroma Varma, Chris Ré, and members of Hazy Research. Weak supervision: A new programming paradigm for machine learning. Accessed May 11, 2022.
- [103] Recorded Future by Insikt Group. Adversary infrastructure report 2020: A defender’s view. Technical Report CTA-2021-0107, Recorded Future, 2021.
- [104] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167, 2019.
- [105] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.
- [106] scikit-learn developers. Confusion matrix scikit-learn documentation. https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py, 2022. [Online; Last accessed 14-April-2022].
- [107] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.
- [108] Manmeet Singh, Maninder Singh, and Sanmeet Kaur. Issues and challenges in dns based botnet detection: A survey. *Computers & Security*, 86:28–52, 2019.
- [109] Pushpa Singh, Narendra Singh, Krishna Kant Singh, and Akansha Singh. Diagnosing of disease using machine learning. In *Machine Learning and the Internet of Medical Things in Healthcare*, pages 89–111. Elsevier, 2021.
- [110] Sunniva Rebekka Skjeggstad, Harald Stolt-Nielsen, Line Tomter, Ellen Omland, and Anja Strønen. Norge utsatt for et omfattende hackerangrep. <https://www.nrk.no/norge/norge-utsatt-for-et-omfattende-hackerangrep-1.13358988>, February 2017. [Online; published February 3, 2017], [Accessed: April 6, 2022].
- [111] Florian Skopik, Giuseppe Settanni, Roman Fiedler, and Ivo Friedberg. Semi-synthetic data set generation for security software evaluation. In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pages 156–163. IEEE, 2014.
- [112] John A Sokolowski and Catherine M Banks. *Principles of modeling and simulation: a multidisciplinary approach*. John Wiley & Sons, 2011.

- [113] Anna Sperotto, Ramin Sadre, Frank van Vliet, and Aiko Pras. A labeled data set for flow-based intrusion detection. In *International Workshop on IP Operations and Management*, pages 39–50. Springer, 2009.
- [114] Branka Stojanović, Katharina Hofer-Schmitz, and Ulrike Kleb. Apt datasets and attack modeling for automated detection methods: A review. *Computers & Security*, 92:101734, 2020.
- [115] Jay Stokes, John Platt, Joseph Kravis, and Michael Shilman. Aladin: Active learning of anomalies to detect intrusions. Technical Report MSR-TR-2008-24, March 2008.
- [116] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. Mitre att&ck: Design and philosophy. *Technical report*, 2018.
- [117] Aurobindo Sundaram. An introduction to intrusion detection. *Crossroads*, 2(4):3–7, 1996.
- [118] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.
- [119] Editorial team. Crowdstrike’s work with the democratic national committee: Setting the record straight. <https://www.crowdstrike.com/blog/bears-midst-intrusion-democratic-national-committee/>, June 2020. [Online; published June 5, 2020], [Accessed: April 6, 2022].
- [120] The Center for Threat-Informed Defense. Adversary emulation library | github. https://github.com/center-for-threat-informed-defense/adversary_emulation_library, 2021. [Online; Last accessed 8-May-2022].
- [121] The MITRE Corporation. Emu plugin | github. <https://github.com/mitre/emu>, 2022. [Online; Last accessed 8-May-2022].
- [122] Cybersecurity The National Security Agency (NSA), Infrastructure Security Agency (CISA), and Federal Bureau of Investigation (FBI). Russian svr targets u.s. and allied networks. "https://media.defense.gov/2021/Apr/15/2002621240/-1/-1/0/CSA_-SVR_TARGETS_US_ALLIES_UOO13234021.PDF", April 2021. [Online; published April, 2021], [Accessed: April 6, 2022].
- [123] Ciza Thomas, Vishwas Sharma, and N Balakrishnan. Usefulness of darpa dataset for intrusion detection system evaluation. In *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008*, volume 6973, pages 164–171. SPIE, 2008.

- [124] Ryan Trost. *Practical Intrusion Analysis: Prevention and Detection for the Twenty-First Century: Prevention and Detection for the Twenty-First Century*. Pearson Education, 2009.
- [125] Nicolas Turpault, Romain Serizel, and Emmanuel Vincent. Limitations of weak labels for embedding and tagging. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135. IEEE, 2020.
- [126] William Turton and Jennifer Jacobs. Russia ‘cozy bear’ breached gop as ransomware attack hit. <https://www.bloomberg.com/news/articles/2021-07-06/russian-state-hackers-breached-republican-national-committee>, July 2021. [Online; published July 6, 2021; updated July 7, 2021], [Accessed: April 6, 2022].
- [127] University of California, Irvine. Kdd cup 1999 dataset, 1999. Available online:<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> (Last accessed April 16 2022).
- [128] D. Updyke, G. Dobson, T. Podnar, B. Earl, and A. Cerini. Technical report: Ghosts in the machine: A framework for cyber-warfare exercise npc simulation. Technical Report CMU/SEI-2018-TR-005, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, December 2018.
- [129] Wikipedia contributors. Common weakness enumeration — Wikipedia, the free encyclopedia, 2020. Accessed: May 21, 2021.
- [130] Wikipedia contributors. Ip flow information export — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=IP_Flow_Information_Export&oldid=987546541, 2020. [Online; accessed 10-May-2022].
- [131] Wikipedia contributors. Common vulnerabilities and exposures — Wikipedia, the free encyclopedia, 2021. Accessed: May 21, 2021.
- [132] Wikipedia contributors. Metamorphic code — Wikipedia, the free encyclopedia, 2021. Accessed: May 30, 2021.
- [133] Wikipedia contributors. Feature engineering — Wikipedia, the free encyclopedia, 2022. [Online; accessed 14-April-2022].
- [134] Wikipedia contributors. Machine learning — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=1086035935, 2022. [Online; accessed 10-May-2022].
- [135] Wikipedia contributors. Packet analyzer — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Packet_analyzer&oldid=1085620838, 2022. [Online; accessed 2-May-2022].
- [136] Wikipedia contributors. Pcap — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=Pcap&oldid=1085247125>, 2022. [Online; accessed 2-May-2022].

- [137] Wikipedia contributors. Simulation — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=Simulation&oldid=1084588840>, 2022. [Online; accessed 13-May-2022].
- [138] Wikipedia contributors. Unsupervised learning — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Unsupervised_learning&oldid=1083933993, 2022. [Online; accessed 27-April-2022].
- [139] Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584–2603, 2012.
- [140] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018.