


Università degli Studi di Napoli Federico II

Ph.D. Program in

Information Technology and Electrical Engineering

XXXVI Cycle

Thesis for the Degree of Doctor of Philosophy

A Cyber Threat Intelligence-driven Framework for Adversary Emulation

by

Vittorio Orbinato

Advisor: Prof. Domenico Cotroneo

Co-advisor: Prof. Roberto Natella

Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

To my grandmother Giuliana, my continuous source of inspiration.

A Cyber Threat Intelligence-driven Framework for Adversary Emulation

Ph.D. Thesis presented
for the fulfillment of the Degree of Doctor of Philosophy
in Information Technology and Electrical Engineering
by

Vittorio Orbinato

October 2023

Approved as to style and content by

Prof. Domenico Cotroneo, Advisor

Prof. Roberto Natella, Co-advisor

Università degli Studi di Napoli Federico II
Ph.D. Program in Information Technology and Electrical Engineering
XXXVI cycle - Chairman: Prof. Stefano Russo

http://itee.dieti.unina.it

Candidate's declaration

I hereby declare that this thesis submitted to obtain the academic degree of Philosophiæ Doctor (Ph.D.) in Information Technology and Electrical Engineering is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
Parts of this dissertation have been published in international journals and/or conference articles (see list of the author's publications at the end of the thesis).

Napoli, December 12, 2023

Vittorio Orbinato

Abstract

In the ever-evolving landscape of cybersecurity, the challenges faced by organizations in safeguarding their digital assets and data have grown exponentially. Traditional reactive security measures, which focus on responding to attackers after they have breached the perimeter, are no longer sufficient in the current dynamic threat environment [1], resulting in delayed detection and business damage.

To effectively protect against Advanced Persistent Threats (APTs), organizations must shift their paradigm toward proactive security measures. *Adversary emulation* is the pivotal strategy within this landscape, i.e., a strategy that emulates the Tactics, Techniques and Procedures (TTPs) employed by real-world threat actors to anticipate their moves and enhance defensive capabilities accordingly.

Unfortunately, adversary emulation also presents some drawbacks that limit its adoption. First, the scenarios emulated with this paradigm are *not representative* of real-world threat actors since adversary emulation lacks integration with Cyber Threat Intelligence (CTI) to provide insights into the TTPs employed by APTs. This happens since CTI still comes in unstructured forms, e.g., threat and incident reports written by security analysts, making it challenging to process this information automatically to replicate the attackers' behavior. Second, security practitioners cannot rely upon open-source adversary emulation tools to effectively emulate APTs. These solutions only provide educational emulation without being able to evade basic detection countermeasures in actual deployment scenarios.

To address these issues, this dissertation devises a **CTI-driven framework for adversary emulation**. The framework provides a pipeline to **automatically extract attack techniques** from CTI documents and **generate adversary emulation plans**. In addition, it offers a **novel solution for adversary emulation** (*Laccolith*) able to **perform ma-**

licious actions in a non-detectable way, to emulate the behavior of complex APTs realistically. Laccolith was tested against multiple Antivirus (AV)/Endpoint Detection and Response (EDR) solutions to assess its effectiveness for adversary emulation.

Keywords: Adversary Emulation, Cyber Threat Intelligence, Cybersecurity, Proactive Security.

Sintesi in lingua italiana

Nel panorama in continua evoluzione della sicurezza informatica, le sfide affrontate dalle organizzazioni nella salvaguardia dei propri asset e dati digitali sono cresciute in modo esponenziale. Le tradizionali misure di sicurezza reattive, che si concentrano sulla risposta agli attaccanti dopo aver violato il perimetro, non sono più sufficienti nell'attuale panorama dinamico delle minacce, con conseguente rilevamento ritardato e danni alle imprese. Per proteggersi efficacemente contro gli Advanced Persistent Threat (APT), le organizzazioni devono modificare il loro paradigma adottando misure di sicurezza proattive. L'*adversary emulation* è la strategia chiave all'interno di questo panorama, vale a dire, una strategia che emula le TTP impiegate dagli attaccanti del mondo reale per anticipare le loro mosse e migliorare le capacità difensive di conseguenza. Purtroppo, l'adversary emulation presenta anche alcuni svantaggi che limitano la sua adozione. In primo luogo, gli scenari emulati con questo paradigma non sono rappresentativi degli APT del mondo reale. Attualmente, l'adversary emulation è carente nell'integrazione con Cyber Threat Intelligence (CTI) per fornire informazioni sulle TTP impiegate dagli APT. Questo accade poiché la CTI è disponibile in forme non strutturate, ad esempio, report di minacce e incidenti scritti da analisti della sicurezza, rendendo difficile l'elaborazione automatica di queste informazioni per replicare il comportamento degli attaccanti. In secondo luogo, i professionisti della sicurezza non possono fare affidamento su strumenti di adversary emulation open-source per emulare efficacemente gli APT. Queste soluzioni forniscono solo emulazione a scopo didattico, senza essere in grado di eludere le misure di rilevamento di base negli scenari effettivi di dispiegamento. Per affrontare questi problemi, questa tesi elabora un framework guidato dalla CTI per l'adversary emulation. Il framework fornisce una pipeline per e**strarre automaticamente le tecniche di attacco** dai documenti di CTI e

generare piani di emulazione. Inoltre, offre una **nuova soluzione per l'adversary emulation** (*Laccolith*) in grado di **eseguire azioni dannose in modo non individuabile**, per emulare realisticamente il comportamento di APT complessi. Laccolith è stato testato contro diverse soluzioni AV/EDR per valutarne l'efficacia nell'adversary emulation.

Parole chiave: Adversary Emulation, Cyber Threat Intelligence, Cybersecurity, Proactive Security.

Acknowledgements

Another chapter comes to an end, the third of my university life, so it is time to acknowledge and thank everyone who supported me along the way.

First, I would like to thank my family for their unwavering support during my path, even when things got very complicated and I (probably) became the most insufferable person on Earth. Specifically, I would like to thank my brother Luca, witnessing how strong you were even in the darkest times allowed me to find the courage to persevere, always, no matter what. Thanks to my grandmother Giuliana, my "mentor" and source of inspiration, to whom I dedicate this thesis, hoping to reach even half of your successes and career.

I want to express my gratitude towards my advisor, Professor Domenico Cotroneo, for being a formidable guide throughout my Ph.D., and my co-advisor, Professor Roberto Natella, for the constant support and feedback.

The greatest praise probably goes to everyone in the DESSERT lab for being the best colleagues and mates to share this journey with. Thanks to all of you for everything you brought into my PhD path in terms of work and human relationships. Thanks to Gigi, Raf, Pietro, Antonio, Cristina, Barlo, Carmine, Luca, Giorgio, Daniele, and Simona. In particular, thanks to Cristina, for being "Cristina", I do not think there is any other way to describe you. Even though you started your PhD just one year ago, it would be impossible not to acknowledge your contribution to my path. Thanks to Gigi, for being an awesome "senior", (almost) always serious when you need to, and (actually) always ready to joke in front of a coffee to lighten up the tension.

The last shoutout in the department goes to my friends in the SECLAB group, for being a wonderful addition to this trip. Thanks to Franca, always there to support me and chat about work and personal life, or just

about SSC Napoli, the "bestest" topic. Thanks to Alessandra, for being an amazing colleague and friend, Maisto, a fantastic friend for many years now, Francesco, Stefano, and Debora.

I would also like to thank my abroad research period advisors, professors Marco Vieira and Nuno Antunes, for their support. A huge shoutout goes to everyone in the Laprie lab and the SSE group: Zé, Charles, Campos, Omid, João Ferreira, Rodolfo, Ivanilson, Qianying, and Anamta. Thanks for being amazing colleagues and friends during my time in Coimbra. I hope I did not forget anyone.

Thanks to the external reviewers, Professors Alina Oprea and Andrea Ceccarelli, for their invaluable feedback on my dissertation.

I would like to thank the coordinator of the ITEE Ph.D. program, Professor Stefano Russo, for his constant and inestimable supervision over these three years.

I extend my acknowledgments to everyone who supported me during my Ph.D. that I did not explicitly mention, whether it was with a comforting word or a piece of advice.

Finally, I would like to thank myself: for persevering even in the hardest times, when I did not believe in myself at all, for finding the strength to pursue my goals, giving the just relevance to outside feedback, and for understanding the importance of self-love and being enough for myself.

The research presented in this dissertation has been supported by the Italian Ministry of University and Research (MUR) under the programme "PON Ricerca e Innovazione 2014-2020 – Dottorati innovativi con caratterizzazione industriale".

The author's work has been supported by a Ph.D. scholarship funded by the Italian Ministry of University and Research (MUR) under the programme "PON Ricerca e Innovazione 2014-2020 – Dottorati innovativi con caratterizzazione industriale".


Unione Europea


MUR
Ministero dell'Università e della Ricerca


PON
RICERCA
E INNOVAZIONE

Contents

Abstract i
Sintesi in lingua italiana iii
Acknowledgements vii
List of Acronyms xiii
List of Figures xv
List of Tables xviii

1 Introduction **1**
1.1 Thesis structure 4

2 Related Work **7**
2.1 Automated CTI Analysis 7
2.2 Adversary Emulation Tools 9
2.3 Cyber Ranges 12

3 Experimental Study of Automatic Unstructured Cyber Threat Intelligence Classification **13**
3.1 Experimental Approach 15
3.2 Experimental Evaluation of Classification Models 21
3.3 Experimental evaluation on CTI documents 31
3.4 Lessons Learned 35

3.5 Threats to validity 36

4 From Cyber Threat Intelligence to Adversary Emulation 39
4.1 Pre-processing 39
4.2 TTP Extraction 40
4.3 Plan Generation 42
4.4 Experimental Evaluation 44

5 Detectability Evaluation of Adversary Emulation Solutions 57
5.1 Detectability evaluation of MITRE CALDERA 59
5.1.1 Integrating CALDERA with anti-detection solutions 63
5.2 Detectability evaluation of atomic tools 64
5.3 Lessons Learned 66

6 A Novel Solution for Anti-Detection in Adversary Emulation 69
6.1 Design 70
6.1.1 Emulation server 74
6.1.2 Emulation agent 77
6.1.3 Emulation manager 78
6.1.4 Implementation 78
6.2 Experimental Evaluation 79
6.3 Threats to validity 83

7 Conclusions 87

A MITRE ATT&CK 89
A.1 Enterprise Matrix 89
A.1.1 Tactics 90
A.1.2 Techniques 92
A.1.3 Procedures 93

B Introduction to CALDERA **95**
B.1 Command-and-Control server 96
B.2 Agents 96
B.3 Abilities 97
B.4 Adversary Profiles 98
B.5 Operations 98
B.6 Plugins 99

Bibliography **101**

Author's publications **117**

List of Acronyms

The following acronyms are used throughout the thesis.

AC@K	Top K Accuracy
AEL	Adversary Emulation Library
APT	Advanced Persistent Threat
AV	Anti-virus
CAPEC	Common Attack Pattern Enumerations and Classifications
CNN	Convolutional Neural Networks
CTI	Cyber Threat Intelligence
CTID	Center for Threat-Informed Defense
C2	Command-and-control
DL	Deep Learning
DLL	Dynamic-Link Library
EDR	Endpoint Detection and Response
FN	False Negative

FP	False Positive
IoT	Internet of Things
LM	Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MLM	Masked Language Modeling
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NSP	Next Sentence Prediction
OS	Operating System
RNN	Recurrent Neural Networks
STIX	Structured Threat Information eXpression
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
TRAM	Threat Report ATT&CK Mapping
TTPs	Tactics, Techniques and Procedures
UAC	User Account Control
VMI	Virtual Machine Introspection

List of Figures

1.1 Cyber Kill Chain. 2

3.1 OilRig Operational Steps. The red dotted arrows represent C2 communication, the black ones the operational steps. . . 15
3.2 Experimental approach. 16
3.3 Document-level evaluation. 34

4.1 CTI-driven adversary emulation pipeline. 39
4.2 Pre-processing stage. 40
4.3 TTP Extraction stage. 41
4.4 Plan Generation stage. 43

5.1 Experimental setup. 61

6.1 Traditional architecture of adversary emulation. 71
6.2 Architecture of Laccolith. 73
6.3 Injection method in Laccolith. 76

B.1 Architecture of CALDERA. 95

List of Tables

2.1 Adversary emulation solutions comparison. 11

3.1 Dataset of CTI samples in numbers. 18
3.2 Example of dataset entries. 19
3.3 Evaluation of the results with the novel CTI dataset. 26
3.4 Evaluation of the results with the TRAM dataset. 27
3.5 Evaluation of results in mixed dataset, novel CTI dataset combined with TRAM. 28
3.6 Examples of mispredictions. 30
3.7 Dataset of Cyber Threat Intelligence documents. 32

4.1 Anomaly detection dataset in numbers. 42
4.2 Emulation plans from the Adversary Emulation Library. . . 45
4.3 Document selection details. 46
4.6 SecBERT predictions for the MenuPass reports. 47
4.4 T5 summarization example. 49
4.5 Evaluation of the results for BERT. 50
4.7 Ordered techniques for the MenuPass reports. 51
4.8 Adversary Emulation Library plan for MenuPass. 52
4.9 Results of the technique-level comparison between the generated emulation plans and the Adversary Emulation Library. 54

4.10 Results of the tactic-level comparison between the generated emulation plans and the Adversary Emulation Library. . . . 55
4.11 Results of the tactic-level comparison for MenuPass. 56

5.1 Adversary Profile Execution Progress for MITRE CALDERA. **Bold** values represent incomplete progress. 61
5.2 Abilities of MITRE CALDERA detected by AV products. . 62
5.3 Technique detectability for Atomic Red Team. 65
5.4 Technique detectability for Invoke-Adversary. 65

6.1 Laccolith comparison. 70
6.2 Adversary profiles in Laccolith. 80
6.3 Coverage of commands of the Laccolith agent, with respect to the adversary profiles. 81
6.4 Adversary profile execution progress for Laccolith. 82
6.5 Injection success for Laccolith. 84

Chapter 1

Introduction

Dealing with security issues in IT systems has become a critical priority for businesses across all sectors. *Advanced Persistent Threats* (APTs) have become a severe threat in several domains where the impact can be exceedingly high (e.g., in terms of service outages, private data breaches, and intellectual property theft), such as healthcare, manufacturing, telecom, energy, and transportation. In APTs, attackers accomplish their goals through a carefully planned sequence of malicious actions while being careful to stay undetected. These attacks are getting even more challenging to counter as they are carried out by cybercriminal and state-sponsored groups. Well-known examples include the Stuxnet attack, which has been sabotaging Iran's nuclear centrifuges since 2005 and was uncovered in 2010 [2], GhostNet [3], and Carbanak [4]. The time an APT attack goes undetected ("dwell time") has been estimated to be up to 700 days [5]. With all this time to act, APTs can cause irreparable damage.

Traditional security measures, such as reactive security measures and penetration testing techniques, which involve identifying and exploiting vulnerabilities, are becoming ineffective [1]. By focusing solely on the initial stages of the Cyber Kill Chain (Figure 1.1) [6], analysts are unable to explore post-compromise scenarios by analyzing what might happen in the post-exploitation phases of an attack.

Adversary emulation is the most effective prevention against APTs [1], i.e., a proactive approach that reproduces the actions of an APT inside a target computer infrastructure from initial reconnaissance to lateral move-

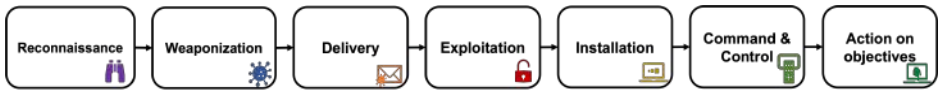


Figure 1.1. Cyber Kill Chain.

ment and data exfiltration. It grants several advantages for security training and assessment purposes, such as understanding threats, realistic and comprehensive testing, identification of weaknesses, and validation of security controls. This approach focuses on post-compromise scenarios, assuming that the APT has already gained access to the target infrastructure and is infecting hosts and devices silently, escalating privileges, and exfiltrating data. It empowers organizations to develop a proactive security mindset and cultivate a culture of continuous improvement. Rather than relying on static security measures that may become obsolete over time, organizations that embrace adversary emulation commit to staying one step ahead of adversaries by anticipating their moves and enhancing their defensive capabilities accordingly. Several tools aim to automate adversary emulation [7], which offer automated procedures that implement APT techniques, as learned from Cyber Threat Intelligence sources. These tools typically automate actions for information gathering, lateral movement across hosts, connections to command-and-control servers, and privilege escalation.

Nevertheless, adversary emulation presents some disadvantages that limit its adoption. Adversary emulation is a *resource-intensive* activity in terms of financial cost, time, and personnel. Organizations need to invest in specialized tools, licenses, and software, essential for simulating realistic attack scenarios and analyzing the results. Building and maintaining the necessary infrastructure for emulation exercises is costly, including isolated test environments and dedicated hardware. Moreover, emulation exercises are *time-consuming* activities that engage human personnel (*red teams*). They require careful planning, execution, and post-exercise analysis.

Adversary emulation also shows some intrinsic *complexities*. From a *technical* standpoint, accurately replicating the tactics and techniques employed by advanced threat actors is complex. These adversaries are often highly skilled and resourceful, making it challenging to emulate their behavior in controlled environments. Adversary emulation teams must continuously update their knowledge and skills to stay ahead of evolving

threats. This requires ongoing training, research, and staying informed about the latest attack techniques and tools. Therefore, adversary emulation should be *driven* by Cyber Threat Intelligence, i.e., information on the capabilities and intents of attackers and the techniques adopted in their campaigns. The analysis of CTI provides valuable insights into the TTPs employed by real-world threat actors. By incorporating CTI into emulation scenarios, organizations can ensure that their exercises accurately reflect the behaviors of actual adversaries. This realism is essential for testing the effectiveness of defensive measures in a way that mirrors the evolving threat landscape. To support adversary emulation, the industry is investing in standardizing CTI, such as the Structured Threat Information eXpression (STIX) [8] representation format and the TAXII sharing protocol [9]. Nevertheless, the majority of Cyber Threat Intelligence still comes in unstructured forms (i.e., text in natural language), such as incident reports [10, 11, 12] and documents leaked by insiders from attacker groups [13, 14, 15]. This heterogeneity hinders adversary emulation processes: unstructured CTI documents can vary significantly in format, content, and terminology. This inconsistency makes it challenging to extract meaningful information consistently across different sources. In addition, unstructured intelligence lacks critical context and provides only partial information about cyber threats. Emulation scenarios require detailed information about threat actors, their TTPs, and their targets. Without this context, emulation exercises may lack realism and relevance. In addition, extracting relevant information from unstructured CTI documents often requires manual effort. This process can be time-consuming and error-prone, making it challenging to maintain a consistent and up-to-date threat intelligence feed for emulation.

To effectively emulate the behavior of APTs, adversary emulation demands sophisticated solutions. In recent years, open-source adversary emulation tools have emerged as a promising solution to make security exercises easier [7]. These solutions automate the execution of individual, low-level malicious actions, such as stealing credentials, lateral movement, and data exfiltration [16]. Examples of solutions are MITRE CALDERA [1, 17], Atomic Red Team [18], Infection Monkey [19], and Invoke-Adversary [20]. Concerning overall criteria, CALDERA represents the most prominent and detailed solution. It has gained high popularity thanks to its ease of setup

and use. Moreover, it provides a high level of detail regarding coverage of ATT&CK TTPs.

Despite this support, relying solely on these tools proves inadequate for security practitioners. These solutions, including CALDERA, still need to be programmed to orchestrate multiple actions and emulate the behavior of an APT. Moreover, they primarily serve educational purposes, lacking the capability to evade fundamental detection countermeasures in real-world deployment contexts. Consequently, adversary emulation tools have to be customized with anti-detection capabilities for the specific AV/EDR to evade, which requires considerable skills and development efforts [21, 22], and is prone to become outdated and ineffective. Hence, emulating anti-detection techniques in automated ways in adversary simulations is still impractical. This critical drawback hinders the effectiveness and realism of adversary emulation, making it unfit for security assessment purposes.

1.1 Thesis structure

The remainder of this thesis is structured as follows:

- Chapter 2 provides a systematic review of related work.
- Chapter 3 presents an **experimental study** to explore the feasibility of **automatic analysis of unstructured CTI documents** to identify attack techniques [23]. The study investigates the use of machine learning techniques for Natural Language Processing (NLP) for this task and is accompanied by **a new dataset for CTI analysis**, providing CTI samples in natural language labeled with the TTPs from the well-known MITRE ATT&CK framework [24].
- Chapter 4 proposes a **pipeline for CTI-driven adversary emulation**. This pipeline encompasses the automatic analysis of unstructured CTI documents to identify and extract adversarial TTPs, which enable the automatic generation of emulation plans to replicate the behavior of well-known APTs leveraging adversary emulation tools.
- Chapter 5 illustrates a **comparative analysis of state-of-the-art solutions for adversary emulation**. We tested several tools for

adversary emulation against multiple AV/EDR products. We found that the malicious actions performed with these tools (and even the installation of their emulation agents) cannot evade detection, thus limiting the realism of emulated attacks. We also tested such tools in combination with state-of-the-art solutions for anti-detection to understand whether it was possible to hide their traces from AV/EDRs. However, adversary emulation tools do not benefit from the integration with anti-detection solutions, since their activities are still detected by the most common defense mechanisms.

- Chapter 6 presents *Laccolith*, a **novel solution for adversary emulation with anti-detection capabilities**. Laccolith introduces a new approach to inject an emulation agent into the target machine based on virtualization, an enabling technology for cybersecurity assessment and training infrastructures. Moreover, this solution can execute non-detectable malicious actions to emulate the behavior of complex APTs realistically. This enables flexible design and setups for cybersecurity emulations, allowing the emulation managers to choose which actions should be performed in a detectable way or not. Along with this novel framework, this dissertation presents an experimental evaluation against multiple AV/EDR solutions to assess Laccolith's effectiveness for adversary emulation and compare it to the state-of-the-art.

Chapter 2

Related Work

This chapter provides a systematic literature review to overview previous and related works.

2.1 Automated CTI Analysis

Automated analysis of Cyber Threat Intelligence is an emerging and active research area in the cybersecurity domain. It ensures consistency in extracting TTPs from CTI, mitigating potential variations that may arise due to different interpretations by human analysts. Automated processes adhere to predefined rules and patterns, ensuring uniform extraction of TTPs across all analyzed data. Ideally, the extracted TTPs should be mapped against a standardized taxonomy, such as MITRE ATT&CK, to facilitate information sharing and interoperability.

TIM [25] is a framework that mines information about attack techniques from unstructured CTI using the Threat Context Enhanced Network (TCENet), a customized model developed for this purpose. It conducts a coarse-grained analysis, categorizing CTI into only 6 categories: Phishing, Scheduled Task/Job, Obfuscated Files or Information, Deobfuscate/Decode Files or Information, Collection, and Application Layer Protocol.

EXTRACTOR [26] is a tool for analyzing sentences to extract information on attack behavior. Specifically, EXTRACTOR identifies system entity names (e.g., file or process names, IP, and registry keys) and actions

(read, write, send, receive, connect, fork) described in the sentences. This approach involves a low-level analysis without abstracting the actions to ATT&CK techniques. While EXTRACTOR has achieved high F-Score values, it has limitations in identifying low-level system entity names or actions when their descriptions span multiple sentences or paragraphs.

TTPDrill [27] is a system that extracts threat actions from unstructured CTI sources. It leverages a dedicated ontology to classify the mined malicious actions, combining NLP and Information Retrieval (IR) techniques. The ontology is built upon MITRE ATT&CK and Common Attack Pattern Enumerations and Classifications (CAPEC) to ensure alignment with the continuously evolving threat landscape. TTPDrill generates STIX shareable artifacts for the identified threat actions to facilitate CTI sharing. However, the tool's scope is confined to identifying a limited set of threat actions, such as data exfiltration, add registry, and DLL injection.

ActionMiner [28] is another framework for extracting threat actions from CTI documents. It utilizes entropy and mutual information to identify low-level actions represented as verb-object couples. The strength of this framework lies in accurately identifying the actions described in the CTI reports. However, the threat actions are not mapped to the techniques in the MITRE ATT&CK taxonomy (or any other standardized taxonomy).

Ampel *et al.* [29] developed a model for analyzing Common Vulnerabilities and Exposures (CVEs) and labeling them with one of 10 MITRE ATT&CK tactics (Defense Evasion, Discovery, Privilege Escalation, Collection, Lateral Movement, Impact, Credential Access, Initial Access, Exfiltration, Execution). For this purpose, Ampel *et al.* leverage transformer-based architectures combined with self-distillation techniques.

Vulcan [30] is a tool for CTI data extraction from unstructured sources. For this purpose, it leverages model-based named entity recognition and relation extraction models specifically trained for cybersecurity-related topics. Vulcan can identify 6 different threat entities (ransomware, attack-vector, vulnerability, platform, algorithm, and tool) and extract the semantic relationship between them. This solution is helpful in monitoring how the behavior of specific threat actors varies through time.

The limitations of these works lie in the fact that they extract coarse-grained information, such as the attackers' tactics (i.e., high-level goals) or low-level information (e.g., system entity names/actions), which are not

enough to define an operational flow to replicate the attacker's behavior. Moreover, most of these works do not map the extracted information against any standardized taxonomy. Using such taxonomies (e.g., MITRE ATT&CK) is a helpful solution to define a common representation for adversarial behavior and facilitate information sharing.

This dissertation differs from previous work: the goal is to automatically analyze CTI to extract meaningful information and map it into the techniques provided by the ATT&CK framework (*de facto* standard in this field). This contribution is relevant for adversary emulation because of the need to define a precise and realistic plan for the threat actor to emulate. To this end, this work introduces a novel dataset for automatic CTI analysis and an extensive experimental study on classification models integrated into a CTI-driven pipeline for adversary emulation.

2.2 Adversary Emulation Tools

Zilberman *et al.* [7] analyze and classify several adversary emulators. The authors defined a set of criteria and a methodology to evaluate the threat emulators; a taxonomy of their qualities; and guidelines for choosing an appropriate adversary emulator, given the specific environment and the security assessment tasks. Among the criteria defined in this work are Operating System (OS) compatibility, changes needed in the security array, ATT&CK TTPs coverage, procedures configuration, and required security expertise. In this section, we further analyze and discuss the most popular and mature adversary emulation tools according to the survey [7].

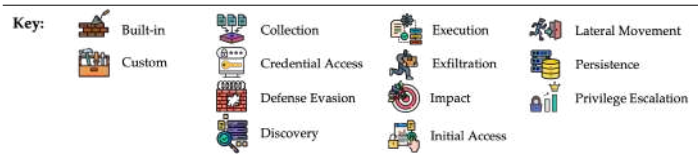
Atomic Red Team [18] is a library of scripts to emulate adversary behavior. Every script implements a single ATT&CK technique or sub-technique (277 out of 719, 231 out of 507 for Windows). It is helpful for specific/atomic tests but unfit for emulating complex scenarios. *Red Team Automation* [31] is a script framework that implements single techniques from the ATT&CK framework (around 50) for security assessment purposes. It does not offer built-in scripts for multi-procedure attacks. *APTSimulator* [32] is a batch script-based tool for Windows that offers around 30 attack techniques to emulate post-compromise scenarios. These techniques leverage external tools such as Mimikatz and PowerSploit. *Infection Monkey* [19] is an adversary emulation tool composed of two main

elements, the server (Monkey Island) and the agents (Monkeys). Infection Monkey implements a few ATT&CK techniques, mainly for Initial Access and Lateral Movement. As CALDERA, Infection Monkey does not offer anti-detection capabilities to hide malicious actions, and AVs easily detect its agents' activities [7]. *Metta* [33] is an adversary emulator developed by Uber Technologies to assess endpoint security. It provides techniques for various tactics, such as Discovery, Credential Access, and Defense Evasion. However, it does not emulate complete attacks but focuses on assessing specific targets. *DumpsterFire* [34] offers a collection of actions (fires) to chain together into complex attacks (dumpster fires). Fires are not based on ATT&CK framework techniques and are organized into categories: Network Scans, File Downloads, Websurfing, Account Bruteforcing, Filesystem Activities, Malware, Custom OS Commands, and Shenanigans. It can only execute on Linux. *Invoke-Adversary* [20] is a PowerShell script to test security mechanisms. It offers 39 techniques grouped by ATT&CK tactics. It does not offer multi-procedure scripts and does not support Lateral Movement. Moreover, its procedures are usually detected by AVs, making it not suitable for usage in real-world scenarios. *Sliver* [35] is a C2 framework for adversary emulation. It offers multi-platform agents that communicate with the C2 server using different protocols (e.g., HTTP, DNS, Mutual TLS). It also provides attacker capabilities through plugins called armories. Sliver does not offer any AV-evasion capability as explicitly stated in its documentation.

Table 2.1 summarizes the comparison among these adversary emulation solutions. It is possible to notice how the tools that provide an advanced architecture with a Command-and-control (C2) server (i.e., CALDERA, Infection Monkey, and Sliver) need to install an agent on the target system. None of the other tools, which do not use an agent, offer command-and-control capabilities. Regarding ATT&CK tactics coverage, Atomic Red Team is the best-performing tool, covering 11 tactics. CALDERA has a similar coverage to Atomic Red Team and is the most complete among the C2 frameworks. CALDERA does not cover Initial Access because it is irrelevant in adversary emulation and the Impact tactic. Moreover, CALDERA surpasses the other C2 frameworks regarding complex attacks, providing built-in attacks and capabilities to develop custom ones. CALDERA also provides a plugin to leverage Atomic Red Team tactics and techniques,

Table 2.1. Adversary emulation solutions comparison.

Tool	C2 Server	Complex Attacks	ATT&CK Tactics Coverage	Needs Pre-Installed Agent	Anti-detection
CALDERA	✓			✓	✗
Atomic Red Team	✗	✗		✗	✗
Red Team Automation	✗			✗	✗
APTSimulator	✗			✗	✗
Infection Monkey	✓			✓	✗
Metta	✗			✗	✗
DumpsterFire	✗			✗	✗
Invoke-Adversary	✗	✗		✗	✗
Sliver	✓	✗		✓	✗



making it the best-performing tool among the existing ones. It is worth noting that none of the analyzed tools provide anti-detection techniques to evade AV/EDR products, a critical feature to achieve realistic emulation.

The solution proposed in this dissertation (Laccolith) differs from the previous ones. Laccolith represents a C2 framework that does not need to explicitly install an agent on the victim machine, unlike CALDERA, Infection Monkey, and Sliver. It also offers profiles based on real-world Advanced Persistent Threat (APT) and mapped to the ATT&CK matrix, covering the same tactics of Atomic Red Team. Most importantly, Laccolith enables the emulation of malicious actions without being detected by AV and EDR solutions. Moreover, it is portable across different versions of the target guest OS and does not require any customization related to the specific AV/EDR. To achieve all these abilities, Laccolith leverages virtualization, widely used in infrastructures for cybersecurity exercises, as illustrated in the following.

2.3 Cyber Ranges

Adversary emulation is typically exercised within controlled virtualized environments, known as *cyber ranges*. These purpose-built platforms provide a simulated infrastructure for organizations to replicate and respond to complex cyber threats realistically, enabling comprehensive security assessments and training exercises. Yamin *et al.* [36] performed a literature review on Cyber Range platforms and security testbeds, highlighting that most solutions leverage virtualization to set up security training and assessment environments. The use of virtualization spans multiple domains, such as Internet of Things (IoT), autonomous systems, SCADA systems, and critical infrastructures. Beuran *et al.* [37] proposed *CyTrONE*, a framework for cybersecurity training. CyTrONE is equipped with a Learning Management System (LMS) for the trainees and offers a Cyber Range that relies on virtualization technologies. The Cyber Range is instantiated using a variable number of VMs, depending on the specific training scenario. *KYPO* [38] is a platform for cyber defense exercises developed by Masaryk University. It aims to emulate attacks on critical infrastructures in a controlled environment. For this purpose, KYPO leverages cloud computing technologies (e.g., OpenStack), and relies on virtualization for hosts and networks. *DETERlab* [39] is a solution for cybersecurity experimentation developed in the context of the *DETER* project. It provides a flexible and realistic environment for security training and assessment. These goals are fulfilled by leveraging virtualization technologies, which enable easy reconfiguration and scalability. The *National Cyber Range* (NCR) [40] is a facility for cybersecurity testing established by the Defense Advanced Research Projects Agency (DARPA). Its purpose is to provide an environment to design and test new ways to respond to the most recent threat actors. The range combines physical and virtualized resources and networks according to the nature of the simulation.

Chapter 3

Experimental Study of Automatic Unstructured Cyber Threat Intelligence Classification

The planning and execution of adversary emulation exercises are expensive activities in terms of skills and man-hours to invest. They include teaming as well as other activities: scenario definition, monitoring, and results evaluation. Adversary emulation entails a set of activities divided into two macro-phases: *Attack modeling* and *Attack emulation*.

Attack modeling. This phase concerns Cyber Threat Intelligence and gathers information on the threats and attacks to emulate. A significant effort is required to retrieve these documents because of their heterogeneous nature and sources. It is then necessary to analyze and systematize this data to map an attack onto a set of operational steps to define an execution flow for the emulation exercise. To this goal, many cybersecurity frameworks provide taxonomies of attack TTPs, with MITRE ATT&CK [24] rapidly becoming the *de facto* standard in the cybersecurity landscape. Currently, human operators perform the activities required by this phase, hence the need for highly specialized personnel.

As an example, we will refer to the OilRig APT [15]. OilRig is a cyber threat actor operating since 2014, interested in several domains, such as

finance, government, energy, chemical, and telecommunications [41]. It primarily leverages social engineering as an initial attack vector. The Center for Threat-Informed Defense (CTID) [42] defined the model of its behavior and actions by collecting and analyzing threat and incident reports from several sources, such as Cyware [43], Mandiant [44], and Malwarebytes Labs [45]. Figure 3.1 shows OilRig's operational steps for *exfiltrating data from a targeted server*. The first tactical goal is *Initial Access*, to identify the target to exploit: OilRig achieves it using *spearfishing attachment* (T1566[1] [46]) (step ① in Figure 3.1). Once the target opens the malicious attachment (T1204.002 [47]), a backdoor is installed on the host machine (step ②). Then, it performs *enumeration* (T1087 [48]) (step ③), leading to the discovery that the user is a member of the administrator group on an Exchange Web Server (EWS). Using the aforementioned backdoor, OilRig *obtains credentials* to EWS (T1003 [49]) (step ④). Leveraging these credentials, the attackers connect to EWS and install a new backdoor (step ⑤) to perform further *enumeration* (T1087 [48]) (step ⑥): this leads to the discovery of an SQL server. Using the backdoor, OilRig *dumps the credentials* (T1003 [49]) (step ⑦) to access the SQL server. The next step is *lateral movement* towards the server (step ⑧) by passing the hash (T1550.002 [50]), where the attackers will copy the database backup files (step ⑨) and exfiltrate them to a controlled mailbox (T1048 [51]) (step ⑩).

Attack emulation. The emulation phase entails implementing the attack using insights gained in the preceding phase. Specific adversary emulation tools facilitate the execution of attack techniques, automating low-level procedures aligned with the ATT&CK framework. Foremost among these tools is MITRE CALDERA [1, 17], renowned for its feature-richness, orchestration of complex attacks, extensive coverage of MITRE ATT&CK TTPs, and ease of setup and use [7]. In contrast, some emulation tools comprise only a limited set of scripts, automating individual malicious actions and leaving human red teams to construct comprehensive APT campaigns manually. Configuration of an adversary emulation tool, such as CALDERA, precedes the emulation process. Initially, the setup involves establishing a C2 server to coordinate emulations. The next step

[1] **Txxxx** indicates a technique from the MITRE ATT&CK framework, further detailed in Appendix A.

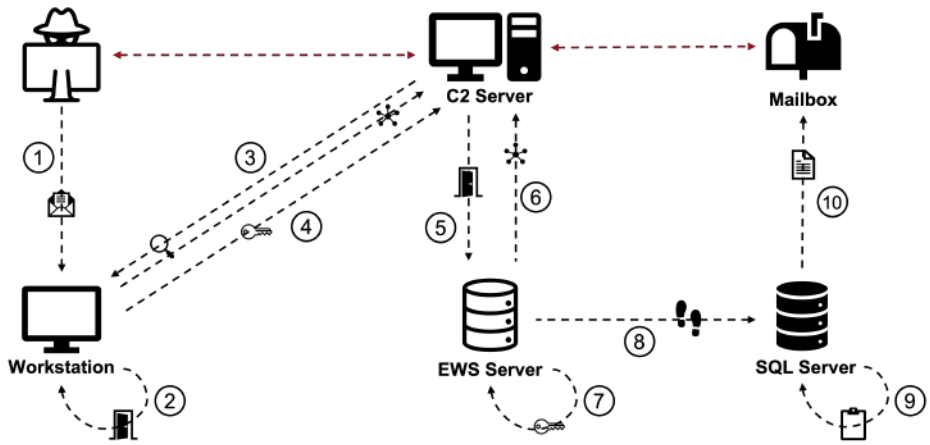


Figure 3.1. OilRig Operational Steps. The red dotted arrows represent C2 communication, the black ones the operational steps.

is to deploy the remote agent on the victim machine. The remote agent connects to the C2 server through a predefined communication channel [52]. The server then orchestrates the agents by transmitting instructions. Upon deployment, the choice of an appropriate *adversary profile* becomes crucial. This profile represents a collection of abilities encompassing attack techniques commonly associated with a specific threat actor [52].

In the case of OilRig, a new adversary profile for CALDERA needs to be configured by combining multiple techniques from CALDERA into an orchestrated attack. Therefore, the emulation of an attack relies on accurate planning of adversarial activities, currently performed by human operators. To overcome this issue, it is helpful to automate CTI analysis by extracting adversary techniques and mapping them to a standard taxonomy (e.g., MITRE ATT&CK) to feed the emulation tool.

3.1 Experimental Approach

Given the cumbersome nature of the manual threat intelligence analysis, this study aims to explore the feasibility of fitting CTI into a defined set of categories, specifically MITRE ATT&CK techniques. We addressed this

problem as a multi-class *classification* task. Consequently, we built novel datasets for the training and evaluation of machine learning algorithms, enabling the execution of diverse experiments across multiple classification techniques.

The experimental approach revolves around three fundamental steps: *Dataset Construction*, *Pre-processing*, and *Classification*, illustrated in Figure 3.2. The goal is to produce a series of machine learning models trained to extract techniques within the MITRE ATT&CK framework from natural language documents.

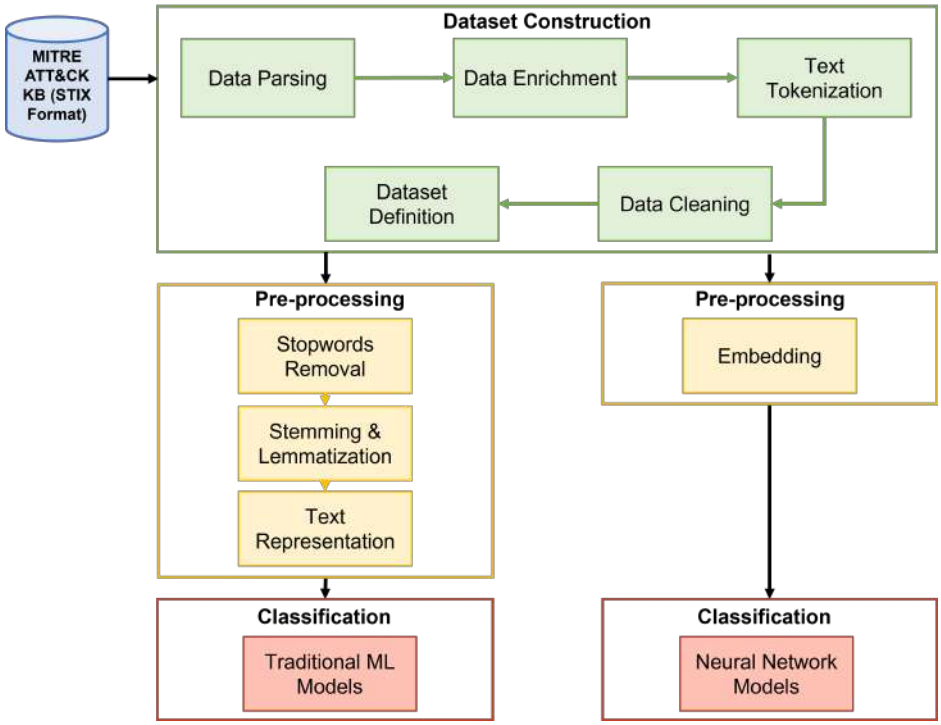


Figure 3.2. Experimental approach.

The *Dataset Construction* phase involves the creation of a new dataset comprising unstructured descriptions of adversarial techniques in natural language. Each entry describes a single malicious technique and is labeled with a specific technique from the MITRE ATT&CK framework taxonomy.

The dataset capitalizes on the publicly available *knowledge base* of the MITRE ATT&CK framework, which provides comprehensive natural language descriptions of various threat actors and their malware campaigns. These descriptions are systematically aligned with MITRE ATT&CK techniques. Therefore, the dataset is compiled by extracting the descriptions and their correlations to the MITRE ATT&CK taxonomy. Notably, the knowledge base was disclosed by MITRE utilizing the STIX language [53].

STIX serves as a representation of CTI in the form of serializable JSON. It characterizes cyber threats through an extensive collection of elements, encompassing cybercriminal groups, their campaigns, targeted sectors and companies, vulnerabilities targeted, employed software tools, and related artifacts such as hash values of malicious executables, IPs, domains involved in their campaigns, and other indicators of compromise (IOCs). These elements are depicted as nodes (*Domain Objects*) within a graph and interconnected by links (*Relationship Objects*) to represent associated concepts (e.g., a malware campaign linked to its IOCs). These objects incorporate attributes like a unique ID, a name, a natural language description, and references to external documents containing additional information (e.g., online incident reports, and other attack taxonomies such as CAPEC [54]).

The Dataset Construction process begins with the *parsing* of the knowledge base in the STIX format. Specifically, information is extracted from the *attack pattern* objects, which represent attack techniques from the MITRE ATT&CK framework and provide a general description of the technique in natural language. Furthermore, we extract the links (i.e., Relationship Objects) between the attack patterns and the threat actors and campaigns that employ them (e.g., *Intrusion Set* objects). These Relationship Objects encompass an additional description of the technique within the context of a specific attack. Subsequently, for attack pattern objects that reference the CAPEC taxonomy externally, we utilize these references to acquire more descriptions from CAPEC (also available in STIX format) about the attack techniques.

The subsequent step in our methodology involves *cleaning* the enriched data by eliminating redundant information, including empty lines, links, and references to external sources. After the data cleaning process, we proceeded with *Text Tokenization*, whereby the text was tokenized at the

sentence level using the Natural Language Toolkit (NLTK) [55] sentence tokenizer. Finally, we merged all this information to *define* the dataset. Each entry includes the following data: ATT&CK technique ID, name, and a description of the technique in natural language, as exemplified in Table 3.2.

Table 3.1. Dataset of CTI samples in numbers.

Categories	188
Samples	12,945
Unique words	7,881
Total # of words	193,453

The dataset is available on Github[2]. Table 3.1 provides additional details on the dataset. It encompasses all 188 techniques within the MITRE ATT&CK framework (at the time of this work). Each technique may be present in the dataset multiple times with distinct descriptions, as it might have appeared in various attack campaigns, albeit in different manners. In total, the dataset encompasses 12,945 entries.

Once the dataset has been created, the next step is its *pre-processing*. The Pre-processing stage encompasses different steps contingent upon the type of machine learning model.

For traditional Machine Learning (ML) models, this phase commences with *Stopwords Removal*. Following this, the process entails *Stemming and Lemmatization*: stemming involves the removal of prefixes and suffixes from words, whereas lemmatization establishes links between inflected words and their corresponding lemma (e.g., *exploits* and *exploiting* are both connected to the lemma *exploit*). The ultimate step in this phase pertains to *Text Representation*: the content within a sentence is transformed into a bag of words, with each sentence encoded as a one-hot vector. It is important to note that using bags of words can result in extensive feature vectors, potentially leading to the loss of word context and sentence position. In the case of a sizable corpus, it is common to encounter more frequently

[2] https://github.com/dessertlab/cti-to-mitre-with-nlp

Table 3.2. Example of dataset entries.

Text Description	Technique	Sub-technique	Sub-technique name
Hydraq creates a backdoor through which remote attackers can adjust token privileges	T1134	T1134	Access Token Manipulation
XCSSET attempts to discover accounts from various locations such as a user's Evernote, AppleID, Telegram, Skype, and WeChat data	T1087	T1087	Account Discovery
PoisonIvy creates a Registry key in the Active Setup pointing to a malicious executable	T1547	T1547.014	Active Setup
Kobalos can write captured SSH connection credentials to a file under the /var/run directory with a .pid extension for exfiltration	T1074	T1074	Data Staged
It has also disabled Windows Defender's Real-Time Monitoring feature and attempted to disable endpoint protection services	T1562	T1562.001	Disable or Modify Tools
APT29 has exfiltrated collected data over a simple HTTPS request to a password-protected archive staged on a victim's OWA servers	T1048	T1048.002	Exfiltration Over Asymmetric Encrypted Non-C2 Protocol

occurring words related to language patterns that may not contribute valuable features to the classifier, as well as less frequent yet informative words. To address this imbalance in representation, we employed Term-Frequency Inverse Document-Frequency (TF-IDF) term weighting, thereby converting count features into floating-point values, preserving significant but less common information throughout the corpus.

Conversely, for neural network models, the pre-processing phase involves the utilization of word *embeddings*. This form of representation encodes each word while considering the surrounding context. This approach enables the definition of similarity between two words based on their distance.

After this stage, the data is ready for *classification*. We consider several classification models, categorized into two sets:

- *Conventional ML models*. This collection encompasses widely studied classifiers employed for NLP tasks for several decades, including Naive Bayes, Logistic Regression, Support Vector Machines (SVM), and Multi-Layer Perceptron (MLP).

- *Deep Neural Networks*. Reflecting the latest advancements in NLP research, this set consists of contemporary classifiers built upon complex neural network architectures, such as Recurrent Neural Networks (RNN), particularly Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and Transformers [56].

In particular, within the domain of deep learning-based classifiers, the *Transformer-based* architecture represents the most recent advancement in NLP research. It introduces the *self-attention* mechanism, which assigns weights to the tokens of the input data based on their significance. As part of this, we also construct a classifier leveraging *SecBERT*, a Language Model (LM) pre-trained on cybersecurity terms [57]. SecBERT is based on BERT, a multi-layer bidirectional Transformer encoder comprising 12 layers (transformer blocks), 12 attention heads, and 110 million parameters in its *base* version. Typically pre-trained on Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) tasks, SecBERT exclusively features an MLM head layer.

Our challenge in attack classification confronts the ML classifiers with various intricacies. Considering a large number of classes (i.e., the 188

MITRE ATT&CK techniques), we are tackling a multi-class classification problem with a considerably extensive range of categories. Notably, it is worth mentioning that text classification problems usually tend to be binary or encompass a small number of categories (typically around 10).

Furthermore, we observed a significant imbalance within the attack data: the most prevalent class is the *Command and Scripting Interpreter* (T1059 [58]) technique, represented by 698 instances, while the least frequent class is the *Cloud Service Dashboard* (T1538 [59]), occurring only 4 times. The varied frequencies of the techniques reflect their actual adoption by threat actors and their chronological introduction in the ATT&CK framework, with more recent techniques tending to be less represented. These complexities motivate our experimental exploration of ML classifiers for the CTI mapping problem to understand their relative strengths and limitations.

3.2 Experimental Evaluation of Classification Models

We performed a preliminary evaluation on the novel dataset by adopting a cross-validation approach. In the following, we elaborate on the evaluation methods and experimental results.

Evaluation metrics and baseline

To compare the performance of the classification models, we considered four different metrics. Before introducing metrics, we briefly recap some key concepts: *True Positive (TP)* and *True Negative (TN)* represent a correct prediction made by the classifier, respectively for positive and negative classes, thus the predicted label matches the actual one. On the other hand, *False Positive (FP)* and *False Negative (FN)* respectively represent an entry from a positive/negative class that has been incorrectly classified. Based on these definitions, we analyze the following metrics for multi-class classification [60]: *precision* is the number of correct positive predictions (TP) divided by the total number of positive predictions (TP + FP) made by the model, while *recall* is the number of true positives (TP) divided by the total number of positive instances (TP + FN). Pre-

cision indicates how much the model can be trusted when it predicts an instance as positive, and recall shows the ability of the model to find all the positive instances in the dataset.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Given the imbalanced nature of our dataset and the skewed distribution of classes, we considered the *F-Measure* statistic instead of the accuracy. Also known as *F1-Score*, the F-Measure is the harmonic mean of precision and recall and expresses how well the model can classify all classes, including minority ones.

$$\text{F-Measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

In the case of multi-class classification problems, we compute these metrics for each class and then average on the number of classes: for this purpose, we can leverage *macro* and *micro* average. The former assigns the same weight to each class, while the latter considers the weight of each class, i.e., the support.

In some scenarios, ML models can serve as a "recommender", presenting the analyst with a selection of potential attack mappings rather than providing a single definitive prediction. Therefore, our analysis included the assessment of the *Top K Accuracy (AC@K)*, specifically with K set to 3, indicating the likelihood of the correct class being ranked within the top 3 predictions. To account for the frequency of each class, we considered the weighted versions of these metrics.

For our evaluation baseline, we refer to the Threat Report ATT&CK Mapping (TRAM) project [61], an open-source platform developed by the CTID, which maps ATT&CK techniques from unstructured text. However, the TRAM project currently supports only three classifiers (Multinomial Naive Bayes, a basic Logistic Regression model, and MLP), with no comprehensive experimental assessment documented in the scientific literature. Additionally, the TRAM project provides a limited dataset of CTI-to-ATT&CK mappings, covering only 80 attack techniques with more

than 1 sample, comprising a total of 1,482 samples of attack descriptions. In contrast, our research encompasses a broader array of ML algorithms and introduces a new, more extensive dataset for evaluation purposes.

Model Training

We constructed a processing pipeline through the SKLearn Library [62, 63], leveraging its diverse ML resources. To account for the dataset imbalance, we fine-tuned the hyperparameters of the models. Initial data analysis revealed a vocabulary comprising approximately 7,800 unique words, resulting in the generation of significantly large feature vectors. We employed standard NLP pre-processing techniques, i.e., stemming, lemmatization, and stopwords removal. Subsequently, we applied the TFIDFVectorizer, a method based on TF-IDF statistic, to transform the textual data into a numerical representation, capturing the significance of each word within the corpus. Furthermore, we configured the *ngram_range* to encompass both unigrams and bigrams (set to $(1, 2)$) and restricted the *max_features* to 10,000 for enhanced efficiency.

For the Naive Bayes and Logistic Regression models, we presented two different versions: the LogisticRegression with the *class_weight* parameter set to *balanced* and the ComplementNaiveBayes, which take into account the imbalanced representation of classes, and the basic library version of LogisticRegression and MultinomialNaiveBayes. For SVM, we consider both the case of One versus One (OvO), which trains $n(n-1)/2$ binary classifiers to split a multi-class classification into multiple binary classification problems; and One versus the Rest (OvR), which instantiates n classifiers to classify a sample between a single category versus all of the other choices united. We chose a linear kernel and set the *class_weight* parameter to *balanced* for both of them. Finally, for MLP, we used the model from SKLearn by setting up the maximum number of iterations for the learning algorithm and the early stopping parameter to prevent overfitting.

For the deep learning models, we leveraged the Keras library [64]. We focused on RNN and CNN models, using word embeddings as text representation. In these models, the first layer transforms words into their equivalent embedding vectors: each sentence has been previously split into tokens and subsequently padded or truncated to meet a maximum length

constraint. We chose the LSTM architecture, an RNN model commonly used for text-processing tasks. The hyper-parameters setup for the LSTM layer were: 150 for the number of units, defining the output dimension, and 0.2 for the dropout parameter, useful to prevent overfitting. Furthermore, we added a slightly different version of this model that initializes the embedding layer with a pre-trained cybersecurity-specific Word2vec model [65]. This model provides a word representation with hundred-size vectors: we extended its vocabulary and re-trained the model with an incremental approach to add new words.

On the other hand, for the CNN, we set up a single layer composed of 256 filters to extract features and a convolution window of 5, followed by a Global Max Pooling layer. Both models present a final fully connected layer of 188 units for the classification task. To avoid overfitting, we took advantage of Keras *EarlyStopping* callback to automatically define the number of epochs during the training of the models. The batch size was 64, while the validation-set percentage was 0.1.

Finally, we built a classifier based on Transformers, using the transfer learning technique and the pre-trained SecBERT model [57, 66]. Leveraging the Hugging Face [67] framework and the Pytorch [68] library, we fine-tuned the model on our dataset and re-trained it with a low learning rate, $1e-05$, to adapt pre-trained features to our data. SecBERT presents a BERT core followed by an MLM head layer. We extracted the pre-trained BERT core and built a classifier on top of that adding two layers: a dropout layer, with a fraction of 0.3, to prevent overfitting, and a linear layer for the classification into 188 different categories. We set the batch size to 16 and the number of epochs to 10.

To train each model, we split the dataset into two parts: 80% for the training set and 20% for the test set. We used a stratified split to preserve the same proportion of examples in the training and test sets.

Results

The analysis begins with a review of the results obtained from our dataset, as presented in Table 3.3. SecBERT achieves the best performance in terms of F-Measure and AC@3, followed by MLP and the OvR version of the SVM model, respectively. Both the Naive Bayes and Logistic Regression models present multiple variant results, detailed in Section 3.2.

This demonstration aims to highlight how the models exhibit improved performance when considering the frequency of each class. Concerning the SVM models, we observe no notable disparity between the performance of the OvO and OvR strategies. This finding supports the preference for the OvR strategy over the OvO strategy, as it constructs a linear number of classifiers rather than a quadratic one. SVM demonstrates promising results for both the F-Measure and top K accuracy, with values closely approaching the top-performing model for both metrics. On the other hand, the DL-based classifiers fail to match the performance of the traditional ML algorithms, with a discernible 10% discrepancy between them. It is worth mentioning that the LSTM model displays slightly enhanced performance when the embedding layer is initialized with pre-trained cybersecurity Word2vec weights.

Subsequently, we assessed the ML algorithms using the dataset from the TRAM project. This comparison with the TRAM project establishes a baseline for the evaluation and demonstrates the industry's engagement with the problem addressed in this study. The TRAM dataset comprises a smaller absolute number of samples compared to our proposed dataset. The metrics in Table 3.4 reveal that the DL-based models, utilizing word embeddings for text representation and benefiting from extensive training datasets, display the poorest performance. These models are penalized when evaluated on the smaller TRAM dataset, which contains fewer sample sentences than our dataset. Moreover, while SecBERT maintains the top ranking for AC@K, the SVM classifier outperforms it in terms of F-Measure. Traditional models employing Bag-of-Words (BoW) demonstrate superior results: the SVM model emerges as the best model according to the F-Measure, whereas Logistic Regression strikes the best trade-off between F-Measure and AC@K.

To complete our evaluation, we combined the TRAM dataset with our dataset. Table 3.5 showcases the classification results for the extended dataset. To integrate our analysis, we encompassed GPT-3.5 [69], the popular transformer-based model behind ChatGPT [70]. Specifically, we leveraged the *gpt3-text-ada-001* model. Similar to the evaluation on our dataset, SecBERT remains the top-performing model, recording the highest score for both F-Measure and AC@3, while GPT-3.5 is the only model that comes close to SecBERT's performance. Most traditional models

Table 3.3. Evaluation of the results with the novel CTI dataset.

Model	Metrics			
	F-Measure	**Precision**	**Recall**	**AC@3**
Complement Naive Bayes	63.9%	65.9%	66.3%	66.6%
Multinomial Naive Bayes	36.8%	49.2%	41.5%	20.2%
Logistic Regression	55.6%	59.1%	60.5%	43.6%
Logistic Regression (balanced)	64.6%	69.3%	64.5%	79.6%
SVM (OvO)	69.9%	71.8%	70.2%	77.2%
SVM (OvR)	69.9%	71.8%	70.2%	77.3%
MLP	70.4%	71.9%	71.6%	77.3%
LSTM	57.7%	59.1%	58.5 %	54.7%
LSTM (Word2vec)	61.0%	62.2%	62.3%	64.4%
CNN	61.4%	63.0%	62.7%	59.5%
SecBERT	**72.5%**	72.5%	72.5%	**86.9%**

Table 3.4. Evaluation of the results with the TRAM dataset.

Model	Metrics			
	F-Measure	**Precision**	**Recall**	**AC@3**
Complement Naive Bayes	53.7%	59.1%	61.6%	57.0%
Multinomial Naive Bayes	23.5%	29.6%	31.9%	16.7%
Logistic Regression	40.9%	50.6%	46.4%	31.0%
Logistic Regression (balanced)	57.7%	66.8%	58.9%	65.2%
SVM (OvO)	60.9%	63.8%	64.6%	44.0%
SVM (OvR)	**60.9%**	63.8%	64.6%	42.6%
MLP	50.5%	56.0%	54.2%	42.0%
LSTM	34.3%	36.0%	36.3%	32.4%
LSTM (Word2vec)	36.5%	37.2%	39.7%	40.2%
CNN	42.4%	42.1%	47.1%	38.5%
SecBERT	52.5%	52.5%	52.5%	**68.6%**

Table 3.5. Evaluation of results in mixed dataset, novel CTI dataset combined with TRAM.

Model	Metrics			
	F-Measure	**Precision**	**Recall**	**AC@3**
Complement Naive Bayes	62.3%	64.0%	64.8%	63.2%
Multinomial Naive Bayes	36.5%	50.5%	41.3%	36.5%
Logistic Regression	56.8%	60.0%	61.2%	42.2%
Logistic Regression (balanced)	64.2%	68.6%	63.9%	79.6%
SVM (OvO)	68.5%	70.3%	69.0%	73.1%
SVM (OvR)	68.5%	70.3%	69.0%	73.0%
MLP	68.9%	69.8%	70.0%	69.9%
LSTM	57.3%	58.6%	58.2%	52.9%
LSTM (Word2vec)	59.8%	60.4%	61.0%	59.8%
CNN	62.4%	62.7%	64.1%	58.0%
SecBERT	**77.3%**	77.3%	77.3%	**86.1%**
GPT-3.5	76.2%	-	-	-

continue to yield commendable results: the balanced Logistic Regression model outperforms the others, ranking as the second best for AC@3, while MLP secures the second-highest F-Measure. The performance remains consistent with the initial analysis, with a slight decline in the metrics resulting from the cross-validation process and the increased data variety in the combined dataset.

Although the classifier based on SecBERT demonstrates promising performance and delivers satisfactory results, it is important to note that this ML model, founded on the Transformer architecture, requires a prolonged training process, relying on GPU capacity to manage an extensive number of parameters. Consequently, the resultant model is considerably more intricate in terms of memory usage.

Qualitative Error Analysis

Considering the encouraging results obtained from the prior analysis, we conducted a qualitative analysis of mispredictions to assess their divergence from human analyst choices. Following the intuition that DL-based classifiers could offer the potential for further enhancements, we gathered all CNN predictions from the test set and manually reviewed several instances of misprediction. A representative subset is detailed in Table 3.6. In the first case, the actual technique involves an attacker's intervention in the system configuration to initiate a malicious program during startup. The predicted class concerns the adversary's manipulation of the Windows registry to conceal a malicious configuration. This prediction aligns with the correct technique, as autostart execution for persistence is achieved through Registry Key modification. Conversely, the classifier erred in the second case, making an incorrect prediction due to mentioning the Registry Key in the description. We also observed that roughly 46% of mispredictions corresponded to the same tactic as the correct technique, as seen in the third instance in Table 3.6, where both labels relate to the *Exfiltration* tactic of MITRE ATT&CK. In these instances, the ML model successfully extracted relevant information about the objective of the attack technique. Given the complexity of the classification task, specifically identifying an attack technique from a comprehensive set of techniques, this finding holds significant relevance. Furthermore, we discovered that several sentences were ambiguously phrased and could be mapped to mul-

tiple ATT&CK techniques, even by a human analyst. The fourth, fifth, and sixth examples in Table 3.6 could potentially be linked to both the predicted and actual labels. In the final example, the description appears more closely related to the predicted label than the actual one. In conclusion, we recognized the difficulty in precisely evaluating the effectiveness of the classifier's prediction based on a single sentence. Consequently, there is a need for a more comprehensive evaluation based on real-world CTI documents.

Table 3.6. Examples of mispredictions.

Text to Evaluate	Actual Technique	Predicted Technique	Actual Name	Predicted Name
SharpStage has the ability to create persistence for the malware using the Registry autorun key and startup folder.	T1547	T1112	Boot or Logon Autostart Execution	Modify Registry
Stuxnet can create registry keys to load driver files.	T1112	T1547	Modify Registry	Boot or Logon Autostart Execution
A Gamaredon Group file stealer can transfer collected files to a hardcoded C2 server.	T1041	T1020	Exfiltration Over C2 Channel	Automated Exfiltration
FIN6 has used has used Metasploit's named-pipe impersonation technique to escalate privileges.	T1134	T1036	Access Token Manipulation	Masquerading
APT29 added their own devices as allowed IDs for active sync using Set-CASMailbox, allowing it to obtain copies of victim mailboxes.	T1098	T1210	Account Manipulation	Exploitation of Remote Services
Sowbug identified and extracted all Word documents on a server by using a command containing *.doc and *.docx.	T1083	T1119	File and Directory Discovery	Automated Collection

3.3 Experimental evaluation on CTI documents

In the previous evaluation, we used ML models to classify short descriptions of the ATT&CK techniques in natural language. The samples were short sentences about a technique engaged by an attacker. Nevertheless, in a real-world scenario, a human cybersecurity analyst deals with entire CTI documents instead of individual sentences and techniques. In this case, the description in natural language spans all of the multiple techniques adopted in an attack campaign. For these reasons, we performed a document-level analysis, in which we would like to observe how many ATT&CK techniques the classifiers can identify from a document. In this case, the output of the ML model is a set of techniques. In the previous case, every sample described exactly one ATT&CK technique. We do not focus on mapping individual sentences to techniques; instead, we evaluate how many of the techniques presented in the document were correctly identified by the model.

We built an additional new dataset for this evaluation. For each document describing a threat actor or attack, we identified a list of ATT&CK techniques engaged in it. We collected the documents from the knowledge base of the MITRE ATT&CK framework. ATT&CK documents notorious malware campaigns and threat groups by structuring information from incident reports and cybersecurity experts. These sources typically provide a broad description of real-world attacks, including a general introduction, a description of the targeted industry sector, the general methodology of the threat actors, technical information about attack techniques, and suggested mitigations against the threat.

Differing from the first dataset (in which we looked at descriptions of individual techniques), in this case, we followed references from the MITRE ATT&CK to external information sources and collected these sources as documents. Detailed information is provided in Table 3.7: threat-related articles/reports, the number of techniques described, and the total number of sentences within each source. From the MITRE ATT&CK knowledge base, we also derived for each document a "ground-truth" vector of techniques presented in the document. Then, we evaluated how many techniques our models can correctly predict and the percentage of correct prediction overall.

Table 3.7. Dataset of Cyber Threat Intelligence documents.

APT	Reference	# Techniques	# Sentences
Follow The Money: Dissecting the Operations of the Cyber Crime Group FIN6	[10]	17	81
Pick-Six-Intercepting a FIN6 Intrusion, an Actor Recently Tied to Ryuk and LockerGoga Ransomware	[71]	50	101
MenuPass: Japan-Linked Organizations Targeted in Long-Running and Sophisticated Attack Campaign	[11]	32	73
MenuPass: United States v. Zhu Hua Indictment	[72]	23	46
WizardSpider: Ransomware Activity Targeting the Healthcare and Public Health Sector	[12]	99	275
WizardSpider: Ryuk's Return	[73]	72	76

To leverage ML models to identify multiple techniques from a document, we applied a threshold on the classifier outputs, i.e., a vector of probabilities, with one probability value for each ATT&CK technique. If the probability exceeds the threshold, we include the corresponding technique in the ATT&CK techniques chosen for the document. This approach also allows the ML model to distinguish between pertinent and non-pertinent sentences in the document (i.e., sentences not related to any technique) since a document typically includes sentences that are not strictly related to attack patterns. Finally, we use the predictions to compute metrics (precision, recall, and F-measure) to compare the results of different ML models. For this type of analysis, we defined *precision* as:

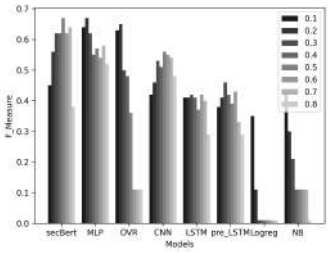
$$\frac{N_{CU}}{N_U}$$

where N_{CU} represents, given a classifier and a document, the number of unique techniques correctly predicted by the classifier (i.e., regardless of how many times the techniques are detected across the sentences of the document), and N_U indicates the total number of unique techniques predicted. Similarly, *recall* was defined as:

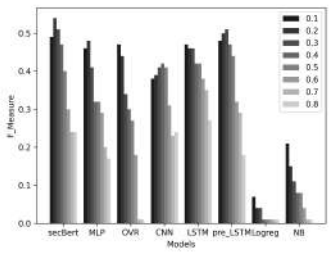
$$\frac{N_{CU}}{N_{GT}}$$

where N_{CU} is the same number used to compute the precision, and N_{GT} represents, for each document, the number of unique techniques in the ground truth.

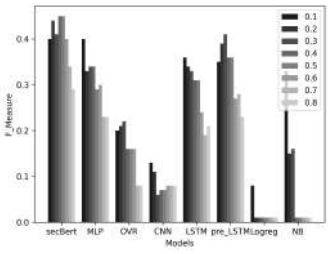
Figure 3.3 shows the results of our analysis. Given a document, we reported the value of the F-measure for each classifier for values of the threshold from 0.1 to 0.8, with a 0.1 step. As expected, the results tend to be worse for higher threshold values, because a smaller number of predictions is accepted. Indeed, the highest accuracy is reached with lower threshold values, with 0.2 emerging as the best value for it. Intuitively, since the probability is split among all 188 techniques, a 0.2 probability for a given technique would be much higher than the other 187. Overall, it is possible to notice that the Deep Learning (DL) models achieve better performance than the traditional ML models. The results of this analysis are lower than the sentence-level evaluation of the previous section.



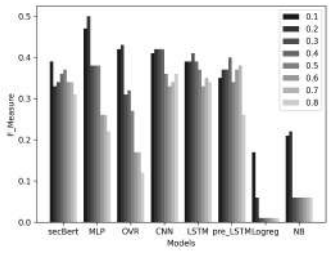
(a) FIN6 [10].



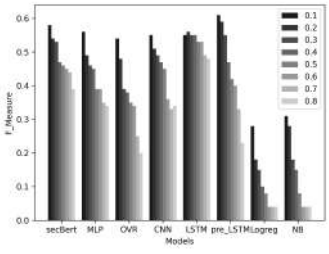
(b) FIN6 [71].



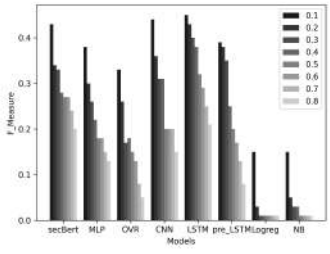
(c) MenuPass [72].



(d) MenuPass [11].



(e) WizardSpider [12].



(f) WizardSpider [73].

Figure 3.3. Document-level evaluation.

This can be traced back to the nature of the documents, which describe a large number of ATT&CK techniques. Moreover, in a given document, the number of sentences that describe the techniques is usually lower than the number of non-pertinent ones. However, the average F-measure achieved by the models is around 50%: this is a good starting point to help a human operator who needs to analyze one or more documents because the models will be capable of identifying at least half the techniques described in it.

3.4 Lessons Learned

In the experimental study, we investigated the use of machine learning to automatically classify unstructured CTI into the corresponding attack techniques from the MITRE ATT&CK framework. The achievements and open issues highlighted by this study provide us with several lessons learned, as summarized in the following.

Applicability of ML for mapping unstructured CTI to attack techniques. NLP to analyze unstructured text is a common procedure in research and the industry sector. The range of applicability of NLP is widening to different domains, including Cybersecurity in recent work. The experimental results provided by the first analysis with cross-validation support the idea that ML is applicable to CTI documents. The classifiers achieved an accuracy (in terms of F-Measure) up to 72% even when challenged by a large set of classes (188 MITRE ATT&CK techniques). This result is relevant since multi-class classification is a difficult task. Moreover, the ambiguity in natural language and between classes (as discussed in Section 3.2) makes this problem even more challenging for ML algorithms, posing challenges for their experimental evaluation.

Selecting a ML algorithm. From the initial analysis, we gained information on the performance of several different classifiers, spanning from traditional to deep learning-based ones. The best-performing was SecBERT, followed by MLP, SVM, Logistic Regression, and CNN. In particular, the analysis on several datasets of different sizes (i.e., TRAM and our new dataset) suggests that deep learning-based models, such as CNN, have the potential to improve their accuracy when trained with larger datasets. The document-level analysis confirms this finding, where we found that traditional, well-known ML models, such as Logistic Regression and Naive

Bayes, perform poorly compared to the others.

Applying classification on real-world documents. Document-level evaluation provides a different point of view on the performance of ML algorithms. In this scenario, we deal with large unstructured texts that describe campaigns with several attack techniques and include additional non-pertinent sentences. Due to these additional challenges, the classification accuracy was generally lower than the previous evaluation. Moreover, we found that several traditional ML algorithms were less accurate than the deep learning-based ones, pointing out that sentence-level evaluation is insufficient for a thorough evaluation since it can lead to over-optimistic results. Finally, our sensitivity analysis showed that the best accuracy is achieved when using low threshold values to select output classes. Indeed, the best value for the threshold is 0.2. Calibrating this threshold is critical since the classifier needs to deal with sentences that describe multiple attack techniques and others that do not detail any attack technique, i.e., *non-pertinent* sentences. The predominant presence of non-pertinent sentences in real-world documents requires accurate planning on how to train the classification models to identify (and discard) such sentences. For instance, a possible solution may be adding a new class for non-pertinent sentences. Such a solution would further complicate the multi-class classification problem since the number of additional samples should balance that of pertinent ones (around $13,000$ as shown in Table 3.1). Furthermore, to perform a more thorough evaluation of the models' performance for the document-level analysis, it would be better to analyze a set of documents related to the target APT to have a more comprehensive view of its intents, capabilities, and activities (since each document provides only a partial view of the attacker and the attack scenario). Specifically, the issue lies in the fact that the ground truth for the techniques described in the document may not reflect the actual content of the single document, i.e., the specific document may describe fewer techniques than those needed to replicate the whole attack.

3.5 Threats to validity

Documents. To build the datasets, we selected threat and incident reports publicly available online. Consequently, the choice of the CTI for

the dataset could affect our study. To mitigate this issue, we focused on documents gathered by MITRE for the ATT&CK framework.

Models. For the classification of attack techniques, we analyzed several traditional ML and DL models. We selected the most popular models for NLP tasks, choosing both traditional and recent ones. We also considered a Transformer-based model, *SecBERT*, to include the most recent advancements in NLP research. To the best of our knowledge, our study covers the largest number of ML models in this field.

Metrics. To evaluate the performance of the classification models, we adopted the traditional metrics for multi-class classification tasks. In particular, we decided to consider the *F1-Score* instead of the *accuracy* to consider the imbalance of our datasets and the distribution of classes. We also selected the *top K accuracy*, with K set to 3, which indicates if the correct class is in the top 3 predictions.

Chapter 4

From Cyber Threat Intelligence to Adversary Emulation

In light of the results of the experimental study presented in Chapter 3, this dissertation introduces a pipeline to automatically analyze CTI documents to extract adversary TTPs, thus allowing the generation of emulation plans to execute on open-source adversary emulation tools. The approach aims to enable the execution of red team exercises with a reduced need for highly specialized personnel. Figure 4.1 shows the stages of the pipeline: *Pre-processing*, *TTP Extraction*, *Plan Generation*.

Figure 4.1. CTI-driven adversary emulation pipeline.

4.1 Pre-processing

CTI documents are primarily composed of unstructured narratives in natural language and describe intricate APT behaviors, tactics, and tech-

niques. Their heterogeneity, stemming from diverse sources and structural disparities, makes retrieving actionable intelligence from such documents a significant challenge, demanding robust analytical frameworks and sophisticated processing techniques.

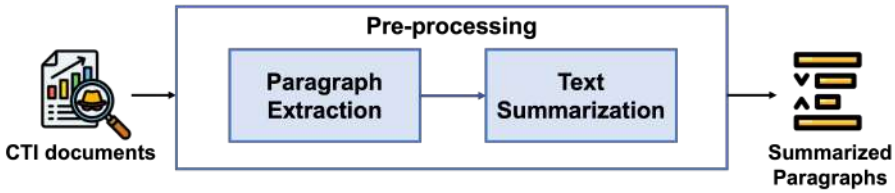


Figure 4.2. Pre-processing stage.

For this analysis, we considered the CTI documents from the *Adversary Emulation Library (AEL)* [74, 75] provided by the CTID [42]. The documents were retrieved using a custom scraper. Given the heterogeneity of their structure, before identifying and extracting the adversary TTPs described in these reports, the first stage of the pipeline is *Pre-processing*, shown in Figure 4.2. The first operation is *paragraph extraction*, according to the original structure of each document. The next step is *text summarization*, i.e., the procedure of condensing text reducing its length while preserving relevant information and meaning [76]. To uniform the paragraphs' length before feeding them to the classification model, we applied *automatic text summarization* [76] leveraging *T5* [77, 78], a transformer-based model developed by Google. Raffel *et al.* [77] performed a comparative analysis between T5 and state-of-the-art solutions, showing that their model achieves great performance in this task.

4.2 TTP Extraction

The next stage is *TTP Extraction*: the goal is to extract information about the TTPs used by known malware and APTs [79], together with the operational flow underneath their strategy. The first operation is *filtering out* non-pertinent sentences from the summarized paragraphs: CTI documents typically contain extensive descriptions of marginal context,

resulting in many non-pertinent sentences, i.e., those sentences that do not describe attack techniques or the operational flow of the attack. This task is accomplished using an *anomaly detector* model that performs a binary classification task. We performed this operation *a priori* instead of adding a "non-pertinent" class to the subsequent TTP Classification problem to avoid the explosion of the complexity of the problem. Moreover, we avoided increasing the dataset size since adding a non-pertinent class would have meant adding several samples for such a class, potentially leading to overfitting problems.

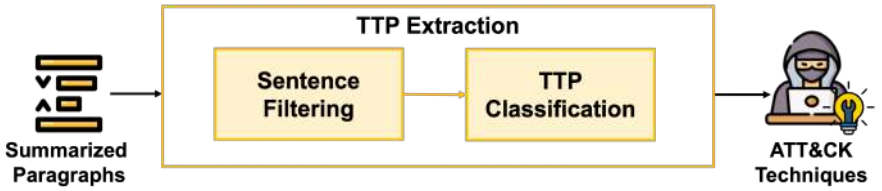


Figure 4.3. TTP Extraction stage.

For this task, we selected BERT in its base uncased version [80]. To fine-tune the model, we built a new dataset starting from the CTI dataset presented in Section 3.1. In particular, given the need for samples of non-pertinent sentences, we *undersampled* the original dataset and performed *data augmentation* with data from other datasets. Undersampling is necessary because of the binary classification task (pertinent versus non-pertinent), with 13,000 samples (original dataset size) representing an excessive number of samples for one class. Specifically, we sampled 15 entries for each class (ATT&CK technique) in the CTI dataset. For data augmentation, we selected two popular datasets for text classification [81, 82]. The datasets contain sentences extracted from news articles (labeled according to the topic), similar to CTI documents in terms of content and structure. We selected all the entries associated with the *tech* topic to maintain semantic consistency with the non-pertinent sentences of CTI documents. The selected sentences were pre-processed before adding them to the dataset, following the steps performed for the CTI dataset construction in Section 3.1. In this case, the dataset is balanced in terms of

samples for the two classes. Table 4.1 displays additional details about the new dataset.

Table 4.1. Anomaly detection dataset in numbers.

Categories	2
Samples	4,330
Unique words	27,691
Total # of words	185,664

The ultimate step of this stage is *TTP Classification*. This task is accomplished through a classification model that associates the extracted information to the TTPs described in MITRE ATT&CK [24]. Since we filtered input sentences in the previous phase of this stage, the classification model must only classify pertinent sentences according to the ATT&CK taxonomy. As observed in Section 3.1, the size of the classification problem (188 classes) is far greater than the usual one for text classification problems, making this task even more challenging for the model.

Specifically, TTP Classification leverages SecBERT [57, 66], the best-performing classification model from the experimental evaluation presented in Chapter 3.

4.3 Plan Generation

The output techniques of the TTP Extraction stage constitute the input for the last stage of the pipeline, *Plan Generation*. This stage aims to tie all of the information produced during the TTP Extraction stage together to build an emulation plan to run on adversary emulation tools.

The generation of an adversary emulation plan is a two-phase process. The first phase is *Technique Sorting*: based on the ATT&CK techniques extracted during the previous stage, we need to infer an ordering among them to define an execution flow for the APT under study. For this purpose, it is possible to leverage ATT&CK tactics as a sorting criterion: the

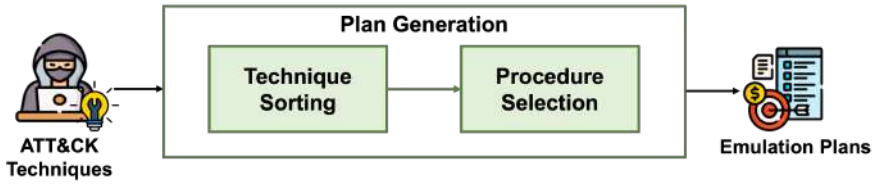


Figure 4.4. Plan Generation stage.

ATT&CK matrix orders tactics according to the logical and operational flow followed by APTs. Another valid solution could be sorting techniques following Cyber Kill Chain phases [6]. However, using ATT&CK tactics represents a more practical solution since each technique is already labeled with the corresponding tactic.

Once the techniques are sorted, we associate them with *procedures*, i.e., implementations of the techniques for target platforms, to define a proper emulation plan. We refer to CALDERA for this phase since it represents the most prominent open-source solution for adversary emulation, offering a wide range of *abilities* (the term for procedures in CALDERA) to emulate various adversaries. There are two possible approaches to mapping techniques to abilities:

- *Exact*: each technique is mapped with an ability from CALDERA.
- *Best fit*: tries to identify the adversary profile offered by CALDERA that better suits the set of predicted techniques.

Exact mapping grants complete customizability for the emulation plan without adapting the APT profile to pre-existing adversary profiles. However, identifying the ability corresponding to a given technique is a challenging task: for each technique, CALDERA provides several abilities that differ in implementation details or target platform. On the contrary, best fit restricts the capabilities of the emulation profile (which can still be modified manually in CALDERA). Despite this drawback, this criterion is still adequate since CALDERA lacks documentation about APIs to add functionalities/capabilities or retrieve information about the existing ones. This limitation also hinders exact mapping, in particular, its automation.

The final emulation plan is now ready to execute on the specific adversary emulation tool. For this reason, Chapter 5 illustrates an experimental

evaluation of state-of-the-art adversary emulation solutions.

4.4 Experimental Evaluation

To assess the effectiveness of the proposed pipeline, we performed an experimental evaluation using real-world CTI documents collected from a public knowledge base in the context of the MITRE ATT&CK framework: the AEL [75]. The AEL is a project developed by the CTID [42] that encompasses intelligence summaries, operations flow, and emulation plans for 8 APTs (at the time of this work). These artifacts were produced by security experts through manual analysis of CTI documents. The AEL offers two types of emulation plans, listed in Table 4.2:

- *Full Emulation Plans*: entail a thorough strategy for mimicking a specific APT, e.g., OilRig, encompassing the entire attack flow from initial access to exfiltration. These strategies replicate an extensive array of ATT&CK tactics and techniques, meticulously crafted to replicate an actual breach by the specified adversary.
- *Micro Emulation Plans*: replicate specific behaviors observed across multiple adversaries, such as remote code execution or log clearing. These strategies emulate a limited set of ATT&CK techniques usually executed within the context of a single adversary action.

Full Emulation Plans include the following artifacts:

- *Intelligence Summary*: is a summary of the CTI sources related to the target APT. It provides an overview of the adversary.
- *Operations Flow*: defines an execution flow for the techniques adopted by the APT in a typical attack scenario.
- *Emulation Plan*: implements the Operations Flow for a specific adversary emulation tool (CALDERA) to replicate the APT behavior.

To evaluate the pipeline, we retrieved the CTI references analyzed by the CTID to produce the Intelligence Summaries. This choice is motivated by the limitations of the evaluation criterion presented in the previous experimental study (Section 3.4). Table 4.3 displays the details about how

Table 4.2. Emulation plans from the Adversary Emulation Library.

Plan Type	Title
Full	APT29
	Carbanak
	FIN6
	FIN7
	menuPass
	OilRig
	Sandworm
	Wizard Spider
Micro	Active Directory Enumeration
	Data Exfiltration
	DLL Sideloading
	File Access and Modification
	Log Clearing
	Process Injection
	Remote Code Execution
	Web Shells
	Windows Registry

many and which documents were selected. We left out of the analysis the unavailable documents along with pdf files.

Table 4.3. Document selection details.

APT	# Selected documents	# Documents in AEL	References
Carbanak	10	19	[83] [84] [85] [86] [87] [88] [89] [90] [91] [92]
FIN6	8	15	[71] [93] [94] [95] [96] [97] [98] [99]
FIN7	12	26	[86] [87] [100] [101] [102] [103] [104] [105] [106] [107] [108] [109]
MenuPass	19	32	[11] [110] [111] [112] [113] [114] [115] [116] [117] [118] [119] [120] [121] [122] [123] [124] [125] [126]
Sandworm	7	10	[127] [128] [129] [130] [131] [132] [133]
WizardSpider	5	9	[100] [101] [102] [134] [135]

The example illustrated in the following refers to the CTI documents describing the *MenuPass* APT [11, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126]. MenuPass, also known as APT32, is a sophisticated APT group attributed to cyber espionage activities primarily targeting entities in Southeast Asia, including organizations in Vietnam, the Philippines, Laos, and Cambodia. This threat actor has been active since at least 2012, and it is believed to operate under the direction of the Vietnamese government [136]. MenuPass has demonstrated a high level of sophistication and has been involved in various espionage campaigns, with a particular focus on sectors such as media, research, and construction. The group is known for its utilization of multiple TTPs, including spear-phishing, watering hole attacks, and the deployment of custom malware, such as the Windshield backdoor. MenuPass is recognized for its adaptability and continuous evolution, often incorporating new tools and techniques to carry out its espionage activities while maintaining a low profile and evading detection by security measures.

As introduced in Section 4.1, we scraped these documents to retrieve the paragraphs, then summarized them to align their lengths leveraging T5. Specifically, Table 4.4 illustrates an example of text summarization for a specific MenuPass CTI report from Symantec [11].

The summarized paragraphs were then fed to the detector model (BERT) to filter out non-pertinent sentences. Table 4.5 presents the results for the evaluation of BERT. It is possible to notice how the model demonstrates flawless performance in distinguishing whether a sentence is pertinent, i.e., if it describes an attack technique. In addition to the overall results, we noticed that the single predictions of the model were accurate: the probability that BERT provided for each sentence was always close to 0%/100%, without any in-between predictions.

The pertinent sentences identified by the detector feed the crucial step of the pipeline: TTP classification. As illustrated in Section 4.2, we leverage SecBERT for this task since it represents the best-performing model in the explorative experimental study on automatic TTP extraction (Chapter 3). From the pertinent sentences extracted from the MenuPass report, SecBERT identifies 28 ATT&CK techniques, shown in Table 4.6.

Table 4.6. SecBERT predictions for the MenuPass reports.

Technique ID	Technique Name
T1003	OS Credential Dumping
T1036	Masquerading
T1589	Gather Victim Identity Information
T1140	Deobfuscate/Decode Files or Information
T1071	Application Layer Protocol
T1068	Exploitation for Privilege Escalation
T1518	Software Discovery
T1547	Boot or Logon Autostart Execution
T1059	Command and Scripting Interpreter

T1082	System Information Discovery
T1027	Obfuscated Files or Information
T1070	Indicator Removal
T1596	Search Open Technical Databases
T1505	Server Software Component
T1104	Multi-Stage Channels
T1016	System Network Configuration Discovery
T1055	Process Injection
T1499	Endpoint Denial of Service
T1018	Remote System Discovery
T1047	Windows Management Instrumentation
T1136	Create Account
T1560	Archive Collected Data
T1069	Permission Groups Discovery
T1033	System Owner/User Discovery
T1573	Encrypted Channel
T1095	Non-Application Layer Protocol
T1132	Data Encoding
T1008	Fallback Channels

To generate the emulation plan, the techniques need to be reordered. For this purpose, we retrieved the corresponding tactics to infer an operational flow. The ordered list of techniques is reported in Table 4.7. Before associating each technique with an implementation from an adversary emulation tool, we assessed the effectiveness of the pipeline by comparing the generated plans with those provided by the AEL. For this analysis, we

Table 4.4. T5 summarization example.

Paragraph	We observed the attackers using a wide variety of living-off-the-land, dual-use, and publicly available tools and techniques in these attacks. The threat actors also use a legitimate cloud file-hosting service for exfiltration. The attackers also use DLL side-loading at multiple stages during the attack, including using it to load Backdoor.Hartip. DLL side-loading occurs when attackers are able to replace a legitimate library with a malicious one, allowing them to load malware into legitimate processes. Attackers use DLL side-loading to try and hide their activity by making it look legitimate, and it also helps them avoid detection by security software. It is a tactic that is commonly used by APT groups and has often been observed being used by nation-state-backed actors. Monitoring networks for unusual activity, as Symantec 2019 Cloud Analytics technology does, is key for detecting this kind of malicious activity. The attackers were also seen deploying a tool capable of exploiting the ZeroLogon vulnerability (CVE-2020-1472). The critical elevation-of-privilege vulnerability was first disclosed and patched on August 11, 2020, and can allow attackers to spoof a domain controller account and then potentially use it to steal domain credentials, take over the domain, and completely compromise all Active Directory identity services. It has been exploited by multiple malicious actors since its disclosure, leading both Microsoft and the Department of Homeland Security 2019 (DHS) Cybersecurity and Infrastructure Security Agency (CISA) to issue warnings to Windows users to patch the issue as quickly as possible.
Summary	attackers used living-off-the-land, dual-use, and publicly available tools and techniques. they also used a legitimate cloud file-hosting service for exfiltration. the attackers also used a tool capable of exploiting the ZeroLogon vulnerability. the vulnerability can allow attackers to spoof a domain controller account and then potentially use it to steal domain credentials. the vulnerability has been exploited by multiple malicious actors since its disclosure.

Table 4.5. Evaluation of the results for BERT.

Class	Metrics		
	F-Measure	**Precision**	**Recall**
Pertinent	100%	100%	99.0%
Non-pertinent	100%	99.0%	100%

defined the following metrics:

- The set of correctly predicted techniques (TP) over the set of actual techniques
- The set of incorrectly predicted techniques (FP) over the set of predicted techniques

Table 4.8 reports the emulation plan provided by the AEL for MenuPass. It is possible to notice how the pipeline generates a larger emulation plan than the actual one (28 techniques versus 12). Table 4.9 illustrates the results of the comparison for all the APTs under study. Looking at the results, it is evident that the pipeline is not as effective as expected in the emulation plan generation. This can be attributed to the following reasons:

- *Information omission*: the AEL emulation plans omit the initial reconnaissance operations, described in the CTI documents. This hinders the evaluation of the automatically generated plans since the classification model correctly identifies those techniques in the documents, drastically increasing the number of false positives over predicted techniques.
- *Redundancy of the ATT&CK framework*: the ATT&CK framework represents an extensive knowledge base for adversarial TTPs, with hundreds of techniques that further specialize in more sub-techniques.

Table 4.7. Ordered techniques for the MenuPass reports.

Technique ID	Technique Name	Tactic ID	Tactic Name
T1589	Gather Victim Identity Information	TA0043	Reconnaissance
T1596	Search Open Technical Databases	TA0043	Reconnaissance
T1059	Command and Scripting Interpreter	TA0002	Execution
T1047	Windows Management Instrumentation	TA0002	Execution
T1136	Create Account	TA0003	Persistence
T1505	Server Software Component	TA0003	Persistence
T1055	Process Injection	TA0004, TA0005	Privilege Escalation, Defense Evasion
T1068	Exploitation for Privilege Escalation	TA0005	Privilege Escalation
T1547	Boot or Logon Autostart Execution	TA0003, TA0004	Persistence, Privilege Escalation
T1036	Masquerading	TA0005	Defense Evasion
T1027	Obfuscated Files or Information	TA0005	Defense Evasion
T1140	Deobfuscate/Decode Files or Information	TA0005	Defense Evasion
T1070	Indicator Removal	TA0005	Defense Evasion
T1003	OS Credential Dumping	TA0006	Credential Access
T1016	System Network Configuration Discovery	TA0007	Discovery
T1018	Remote System Discovery	TA0007	Discovery
T1069	Permission Groups Discovery	TA0007	Discovery
T1033	System Owner/User Discovery	TA0007	Discovery
T1082	System Information Discovery	TA0007	Discovery
T1560	Archive Collected Data	TA0009	Collection
T1104	Multi-Stage Channels	TA0011	Command and Control
T1573	Encrypted Channel	TA0011	Command and Control
T1095	Non-Application Layer Protocol	TA0011	Command and Control
T1132	Data Encoding	TA0011	Command and Control
T1008	Fallback Channels	TA0011	Command and Control
T1071	Application Layer Protocol	TA0011	Command and Control
T1499	Endpoint Denial of Service	TA0040	Impact

Table 4.8. Adversary Emulation Library plan for MenuPass.

Technique ID	Technique Name	Tactic ID	Tactic Name
T1105	Ingress Tool Transfer	TA0011	Command and Control
T1135	Network Share Discovery	TA0007	Discovery
T1018	Remote System Discovery	TA0007	Discovery
T1046	Network Service Discovery	TA0007	Discovery
T1016	System Network Configuration Discovery	TA0007	Discovery
T1003	OS Credential Dumping	TA0006	Credential Access
T1569	System Services	TA0002	Execution
T1560	Archive Collected Data	TA0009	Collection
T1104	Multi-Stage Channels	TA0011	Command and Control
T1537	Transfer Data to Cloud Account	TA0010	Exfiltration
T1047	Windows Management Instrumentation	TA0002	Execution
T1553	Subvert Trust Controls	TA0005	Defense Evasion

This abundance may interfere with the classification process since many techniques in the ATT&CK framework represent nuances of the same attack technique.

- *Extreme heterogeneity of the sources*: as previously stated in the experimental study of Chapter 3, CTI documents show high variability of structure, terminology, and content. This further complicates the task of defining a completely representative dataset for this problem and, consequently, the possibility of achieving high performance with classification models.

- *Difficult generalization of the APT behavior*: CTI sources typically describe different attack scenarios for a target APT. This happens because cybercriminal groups adapt their *modus operandi* to the specific target infrastructure and attack scenario. Consequently, different threat reports may describe situations where the APT caused damage or even scenarios in which its activity was detected in time. For instance, in one of the CTI reports for MenuPass, Symantec describes an instance of MenuPass detected by their systems [11], which may differ in behavior and attack techniques from other successful attacks. Moreover, APTs stay undetected for large time windows before any evidence of their activity is detected and traced back to them, which gives them the opportunity to perform several different kinds of attacks and breaches.

For these reasons, we decided to perform a higher-level comparison, abstracting the techniques to the respective tactics. In this way, we can identify whether the automatically generated plans achieve the same tactical goals as those from the AEL. The results of this analysis are illustrated in table 4.10. It is possible to notice an improvement in the results at the tactic level. Again, some properties of the adversary emulation plans from the AEL influenced the analysis. First, these plans do not encompass the Reconnaissance and Resource Development tactics whose techniques are still described in the documents, as pointed out in the technique-level comparison. In addition, looking at the analysis for MenuPass (Table 4.11), we notice how the AEL profile does not cover the Impact tactic. This represents a "contradiction" in the emulation plan since most CTI documents

Table 4.9. Results of the technique-level comparison between the generated emulation plans and the Adversary Emulation Library.

APT	TP / Actual	FP / Predicted
Carbanak	73%	54%
FIN6	20%	75%
FIN7	11%	91%
MenuPass	50%	77%
Sandworm	20%	33%
WizardSpider	30%	50%

describe typical attack scenarios in which the APT "impacted" the target infrastructures, causing damages such as service outages.

Table 4.10. Results of the tactic-level comparison between the generated emulation plans and the Adversary Emulation Library.

APT	$\frac{\text{TP}}{\text{Actual}}$	$\frac{\text{FP}}{\text{Predicted}}$
Carbanak	75%	40%
FIN6	62%	50%
FIN7	28%	66%
MenuPass	85%	40%
Sandworm	40%	0%
WizardSpider	45%	42%

Table 4.11. Results of the tactic-level comparison for MenuPass.

Tactic	Pipeline	Adversary Emulation Library
Reconnaissance	✔	
Execution	✔	✔
Persistence	✔	
Privilege Escalation	✔	
Defense Evasion	✔	✔
Credential Access	✔	✔
Discovery	✔	✔
Collection	✔	✔
Exfiltration		✔
Command and Control	✔	✔
Impact	✔	

Chapter 5

Detectability Evaluation of Adversary Emulation Solutions

The emulation of APT behavior, through the plans generated with the pipeline, leverages adversary emulation tools, such as CALDERA. This chapter presents an experimental analysis to evaluate the detectability of adversary emulation tools against multiple AV/EDR solutions. As adversaries continuously enhance their TTPs to evade detection, understanding the effectiveness of these tools in replicating real-world attack scenarios becomes essential for strengthening cybersecurity defenses. This study seeks to shed light on the strengths and weaknesses of state-of-the-art adversary emulation tools, offering valuable insights into their *detectability*, i.e., their capacity to execute malicious actions on target systems while remaining undetected by modern AV/EDR solutions.

The evaluation focuses on three state-of-the-art solutions for adversary emulation: MITRE CALDERA [17], Atomic Red Team [18], and Invoke-Adversary [20]. Compared to other adversary emulation tools, CALDERA provides higher coverage of APT tactics and techniques, a complete client-server architecture, and the ability to orchestrate complex APT campaigns. In contrast, Atomic Red Team and Invoke-Adversary belong to the *atomic* tools category, i.e., those tools that enable the execution of single (atomic) adversarial actions from the MITRE ATT&CK framework.

To evaluate detectability, we test the adversary emulation tools against a set of 5 AV products: Windows Defender, Avast, AVG, Kaspersky, and Avira. These products qualify as modern EDR solutions and cover state-of-the-art detection techniques [137]. For instance, they monitor system calls using user-space and kernel-space hooks; they use both signature-based and real-time behavioral detection; and they implement self- and system-protection techniques. An example of such protections is preventing filesystem access to specific directories, such as the main system directories and the AV directory itself. As a metric to measure detectability, we consider the **adversary profile execution progress**. An adversary profile is a sequence of atomic steps that represent malicious activities. The metric represents the execution progress of an adversary profile in terms of atomic steps successfully executed until the AV raises an alarm. The *successful execution* of an atomic step means that the emulation agent can execute the action without being detected by the AV under test. The steps that compose an adversary profile are called *abilities* for CALDERA. This metric is expressed as a fraction:

$$\text{Adversary Profile Execution Progress} \quad = \quad \frac{N_{EA}}{N_{PA}}$$

N_{EA} is the number of executed actions of an adversary profile, while N_{PA} is the total number of its actions. For example, suppose a profile has 10 actions, and the AV detects the execution of its eighth action. In that case, the adversary profile execution progress will be 7/10 since the profile managed to execute 7 actions out of 10 before being detected.

To gain additional insights into the detectability, we also analyze the initial loading of the emulation agent in the victim machine. Since AV products perform rigorous checks before an executable launches, the emulation agent must also evade these checks. For this purpose, we introduce an additional metric, the **injection success**. This metric represents the probability of successfully injecting the agent into the victim without triggering detection. The metric is defined as a fraction:

$$\text{Injection Success} \quad = \quad \frac{N_{SI}}{N_I}$$

where N_{SI} is the number of successful injections attempts and N_I is the

total number of attempts.

5.1 Detectability evaluation of MITRE CALDERA

To study the detectability of CALDERA, we selected its 12 default adversary profiles. These profiles can be divided into two categories: *reconnaissance & information gathering* and *advanced*. The first category encompasses the first eight profiles listed below, which perform basic operations: user identification, process enumeration, anti-virus discovery, screenshot capture, and file search. The advanced profiles perform more invasive actions, like process injection, lateral movement, and malicious payload execution. Therefore, their activities will be noisy and more likely to be flagged by the anti-viruses. We report a short description for each profile:

- *Discovery*: collects detailed information from a host, such as local users, user processes, admin shares, and anti-virus programs;
- *Hunter*: performs Discovery operations, then tries to exfiltrate files from the working directory;
- *Check*: collects information about the configuration of the host (e.g., installation of common software packages, such as Chrome, Go, and Python), and the configuration of its network interfaces;
- *Collection*: collects personal information from the host, such as company emails, IP addresses, and personal files;
- *Enumerator*: enumerates the presence of different types of processes on the host, such as WMIC, PowerShell, and SysInternals utilities;
- *Nosy Neighbor*: finds the preferred Wi-Fi networks and tries to disrupt the Wi-Fi connection;
- *Signed Binary Proxy Execution*: executes malicious code through signed and trusted binaries;
- *Super Spy*: monitors the active user by capturing screenshots, copying data from the clipboard, and scanning preferred Wi-Fi networks;

- *Undercover*: swaps the built-in PowerShell with PowerShell Core to stop PowerShell processes;
- *Stowaway*: injects Sandcat (the default emulation agent in CALDERA) into another process;
- *Worm*: runs PowerKatz to steal user credentials, then moves laterally in multiple ways;
- *You Shall (Not) Bypass*: bypasses User Account Control (UAC).

In addition to these profiles, we implemented a custom *Ransomware* profile for completeness. This profile was developed from scratch, similar to the implementation of the predefined profiles. It encompasses the following abilities:

- Find files;
- Stage sensitive files;
- Compress the staged directory;
- Exfiltrate the staged directory;
- Encrypt sensitive files (e.g., *docx* and *pdf* files).

The first four abilities re-use the ones of the predefined profiles to find the target files. The last ability has been implemented with a PowerShell script [138] to encrypt the sensitive files by silently skipping files if it does not have permission to rewrite them. Using a PowerShell script with standard APIs for encryption makes the process more similar to "legitimate" programs (as in fileless malware), similar to other profiles in CALDERA. In this way, the Ransomware profile should be as detectable as the reconnaissance & information gathering adversary profiles.

We used the following configuration for the experimental analysis, shown in Figure 5.1:

- Victim Windows 10 VM (VM1): the target of the adversary emulation;
- Additional Windows 10 VM (VM2): is a machine necessary to perform some malicious actions, e.g., lateral movement.

Both VMs are equipped with 4 CPU cores and 2 GB RAM and run Microsoft Windows 10.

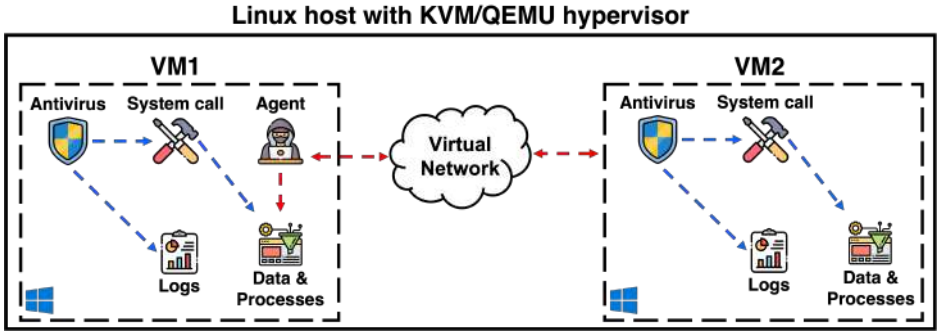


Figure 5.1. Experimental setup.

Table 5.1. Adversary Profile Execution Progress for MITRE CALDERA. **Bold** values represent incomplete progress.

Profile	Windows Defender	Avast	AVG	Kaspersky	Avira
Discovery	9/9	9/9	9/9	9/9	9/9
Hunter	14/14	14/14	14/14	14/14	14/14
Check	6/6	6/6	6/6	6/6	6/6
Collection	2/2	2/2	2/2	2/2	2/2
Enumerator	5/5	5/5	5/5	5/5	5/5
Nosy Neighbor	7/7	7/7	7/7	7/7	7/7
Signed Binary Proxy Execution	3/3	3/3	3/3	3/3	3/3
Super Spy	11/11	11/11	11/11	11/11	11/11
Undercover	**1/2**	**1/2**	**1/2**	**1/2**	2/2
Stowaway	**1/2**	**1/2**	**1/2**	**1/2**	2/2
Worm	**1/9**	**1/9**	**1/9**	**1/9**	9/9
You Shall (Not) Bypass	**2/4**	**2/4**	**2/4**	**1/4**	**1/4**
Ransomware	5/5	5/5	5/5	5/5	5/5

Table 5.1 shows the execution progress for each adversary profile of

MITRE CALDERA against the five anti-viruses. It is possible to notice that the reconnaissance & information gathering profiles execute without being detected since they do not perform harmful actions. For the advanced profiles, the results of the experiments are quite different. During the execution, all the anti-viruses, except for Avira, flagged the activities of the profiles as suspicious. Moreover, the detected abilities are always the first ones in the atomic order: once an ability has been detected, it is impossible to complete the operation since the prerequisites for the following ones will not be satisfied. As a consequence, the last abilities of the advanced profiles could not be executed. For instance, the Stowaway profile uses two abilities: since the execution of the first one is stopped by the anti-virus, the second one cannot be tested. Therefore, this profile scored 1/2 in terms of detectability. We reported the detected abilities for each profile in Table 5.2. This experimental analysis shows that the emulation agent of MITRE CALDERA can stealthily execute most Discovery, Collection, and Exfiltration techniques but is easily blocked when it tries to perform more intrusive actions, such as Credential Access and Privilege Escalation.

Table 5.2. Abilities of MITRE CALDERA detected by AV products.

Profile	Ability	Windows Defender	Avast	AVG	Kaspersky	Avira
Undercover	Install PowerShell Core 6	✓	✓	✓	✓	
Stowaway	Inject Sandcat into Process	✓	✓	✓	✓	
Worm	Run PowerKatz	✓	✓	✓	✓	
You Shall (Not) Bypass	Wow64log DLL Hijack	✓	✓	✓	✓	✓
You Shall (Not) Bypass	Bypass UAC Medium	✓	✓	✓		

It is worth noting that the AVs always detect the injection of CALDERA's agent. After performing 20 repetitions of the injection, we conclude that the injection success of CALDERA is 0/20 against each AV. Consequently, the results in Table 5.1 only refer to the execution of the adversary profiles, assuming that the agent has already been loaded on the target machine.

5.1.1 Integrating CALDERA with anti-detection solutions

To overcome the limitations of MITRE CALDERA in terms of anti-detection, we combined it with Inceptor [139], a state-of-the-art evasion framework. Inceptor is a template-driven framework: a template is a generic, customizable loader with placeholders for evasion techniques and the actual payloads. There are many templates for three different types of payloads: .NET, PowerShell, and native code. Inceptor offers the possibility of chaining encoding techniques to evade static code analysis. It is also possible to plug in additional source code writing techniques to evade the Windows Antimalware Scanning Interface's (AMSI) dynamic analysis [140]. AMSI also analyses in-memory artifacts, for example, text areas the code will jump to. For this reason, Inceptor includes AMSI bypass techniques [141]. In this experimental analysis, we focus on Windows Defender and the injection stage of the emulation agent of CALDERA (i.e., the initial installation of the agent). We enhance the agent with multiple anti-detection techniques from Inceptor.

The injection needs to bypass three layers of detection: User Account Control (UAC), signature-based static analysis, and dynamic sandbox analysis. UAC is a protection that involves user interaction by displaying a message through a GUI frontend (Microsoft SmartScreen [142]). Sandbox analysis is a technique used by anti-virus software to analyze potentially malicious files and programs by running them in a safe and isolated environment. It is notably helpful for detecting new and unknown malware, which can often evade static signature-based detection. In Microsoft Windows, *Windows Defender Application Guard* [143] is responsible for performing the sandbox analysis using a virtual container with a specialized version of the Windows OS.

We used the following combination of anti-detection techniques to make the injection stealthier: a native binary template, an encoding chain composed of *Shikata-Ga-Nai* [144], a popular polymorphic binary encoder [145], XOR encoding, 120 seconds of execution delay, and an unhooking technique for EDR bypass. Moreover, we signed the resulting binary with a Microsoft signature using CarbonCopy [146].

We experimentally evaluate the injection of the CALDERA agent with anti-detection by performing 20 repetitions. The injection is unable to evade UAC. This would require the exploitation of known vulnerabilities

in Microsoft Windows (*UAC bypass*), which are regularly fixed by updates of the OS. Therefore, Inceptor is unable to provide a reliable solution to escape UAC, and we manually allow the execution of the CALDERA agent through the UAC. Even neglecting the problem with UAC, the agent was successfully injected approximately 5 times out of 20, thus with an injection success equal to 5/20 and 25% probability. The encoding chain allows the binary to bypass static analysis since it does not match any known signature. The execution delay and unhooking technique heuristically help to elude the dynamic sandbox analysis: the delay is helpful to make the behavior seem benign while unhooking prevents EDRs from inspecting function calls. The binary signature helped to evade the dynamic analysis: when the binary has a valid signature, Windows Defender performs less detailed dynamic checks. However, since the binary needs to decode itself from the encryption chain at some point, it is still challenging to bypass the sandbox analysis, making the injection detectable. The need for additional UAC bypass techniques further complicates the process of making CALDERA stealthier.

5.2 Detectability evaluation of atomic tools

Our experimental analysis of state-of-the-art adversary emulation tools also encompasses *atomic tools*. In particular, we focused on *Atomic Red Team* [18] and *Invoke-Adversary* [20], two popular atomic frameworks [7]. Their atomic nature allows us to overcome the "problem" of not testing techniques that would be skipped when running an entire campaign with multiple actions, which could be detected and stopped by the AVs. Table 5.3 and 5.4 show the detectability of the techniques for Atomic Red Team and Invoke-Adversary respectively. In this case, for each technique, the detectability is expressed as a binary metric (detected/not detected).

The analysis shows that these tools exhibit the same drawbacks as CALDERA, where the most intrusive actions all raise alerts from the AVs. Looking at Tables 5.3 and 5.4, it is possible to notice how the non-intrusive actions are the only ones not detected by any of the AVs, namely *GetCurrent User with PowerShell Script*, *Prompt User for Password*, *Activate Guest Account* for Atomic Red Team, and *System Owner Discovery* and *Screen Capture* for Invoke-Adversary. It is worth noting that even the

deployment of the tools triggers an alert from the AVs, as in the case of CALDERA. Consequently, it is not possible to use them for realistic emulations without turning AVs off during the deployment phase.

Table 5.3. Technique detectability for Atomic Red Team.

Technique	Windows Defender	Avast	AVG	Kaspersky	Avira
Execute base64-encoded PowerShell from Windows Registry	✔	✔	✔	✔	
GetCurrent User with PowerShell Script					
Thread Execution Hijacking	✔	✔	✔	✔	✔
Prompt User for Password					
Mimikatz	✔	✔	✔	✔	✔
Clear Logs	✔	✔	✔		
Activate Guest Account					
Disable Windows Security Center Notifications	✔	✔	✔	✔	✔
Bypass UAC using Event Viewer (PowerShell)	✔	✔	✔		
Disable Microsoft Defender Firewall	✔	✔	✔	✔	

Table 5.4. Technique detectability for Invoke-Adversary.

Technique	Windows Defender	Avast	AVG	Kaspersky	Avira
System Owner Discovery					
PowerShell Encoded Mimikatz	✔	✔	✔	✔	✔
Screen Capture					
Add local firewall rule exceptions	✔	✔	✔	✔	
Create local administrator	✔	✔	✔	✔	✔
Capture Lsass Memory Dump	✔	✔	✔	✔	✔
Clear Security Log	✔	✔	✔		
PSExec	✔	✔	✔	✔	✔

5.3 Lessons Learned

The previous analysis showed that traditional adversary emulation is cumbersome and unable to perform intrusive actions without being detected. CALDERA cannot evade detection by AV/EDR solutions, limiting its adoption for security assessment and training purposes. We also integrated CALDERA with anti-detection solutions, namely Inceptor, to find out that this combination was still unable to evade defense mechanisms. The analysis of atomic tools, i.e., Atomic Red Team and Invoke-Adversary, led to the same conclusion: adversary emulation tools are not mature enough to be used in realistic security training emulations. The main issues to address are the following:

- **Inability to emulate attackers with configurable evasion capabilities**: attackers can exhibit different degrees of expertise, from inexpert script kiddies to cybercriminals that perform advanced targeted attacks, which result in increasing difficulty for the trainees (e.g., incident response teams) at detecting the attacks. Ideally, adversary emulation tools should be able to simulate several kinds of attackers, where anti-detection can be selectively interwoven with attacker actions. For example, to emulate APTs, the organizers of simulations may want to configure the adversary emulation to hide most of the actions and have the trainees start from only a few alarms.

- **Inability to hide from several different EDRs**: enterprise systems and networks typically adopt a broad set of EDRs, which leverage diverse techniques to collect data (e.g., hooking APIs, injecting code into processes, collecting data from existing tracing systems) [147]. Moreover, EDRs from different vendors may implement the same technique in different ways (e.g., by hooking APIs at different levels in the software stack). Therefore, it takes a significant effort for attackers to implement anti-detection techniques and to stay ahead of EDRs. Similarly, it is difficult for Red Teams to adopt these techniques in security emulations. Moreover, it is an open challenge for adversary emulation tools to implement all these anti-detection techniques for several EDR products. Therefore, new adversary emulation techniques are needed to perform anti-detection from several

EDRs, which could be implemented and maintained over time with a limited effort.

Chapter 6

A Novel Solution for Anti-Detection in Adversary Emulation

As introduced in Chapter 5, a fundamental characteristic of APTs is the adoption of *anti-detection* techniques to hide their traces from AV products and EDRs, a modern evolution of AVs, and to persist in the target infrastructure as long as possible. Therefore, adversary emulation also needs to apply anti-detection techniques to perform realistic and fruitful security assessments and training activities. Examples of anti-detection techniques include un-hooking probes used by EDRs to instrument and monitor Dynamic-Link Library (DLL) and API uses [148]; disabling or hampering event tracers, such as the Event Tracing for Windows (ETW) subsystem [149]; using malicious kernel modules to hide processes and files [150]; obfuscating malicious payloads (e.g., shellcodes) [151, 152].

The experimental evaluation on adversary emulation tools presented in Chapter 5 showed that state-of-the-art adversary emulation tools fail to replicate such anti-detection techniques, hindering the realism of adversary emulation for security assessment and training purposes. Adversary emulation tools must be customized for the specific AV/EDR to evade, which requires considerable skills and development efforts [21, 22], and is prone to become outdated and ineffective. Therefore, emulating anti-detection techniques in automated ways in adversary simulations is still impractical.

This is a significant limitation in realistically emulating APTs.

To fill this gap, we introduce *Laccolith*, a novel solution for adversary emulation with anti-detection capabilities. Laccolith builds on a novel *hypervisor-based* architecture for adversary emulation. This design choice was led by the fact that cybersecurity exercises typically happen in virtualized environments [36, 37, 38, 39, 40]. Laccolith enables the non-detectable execution of malicious actions by injecting them from the lowest layers of the software stack. We also experimented with Laccolith against several AV solutions for Microsoft Windows. The results showed that Laccolith could execute all of the malicious actions, which evaded all of the tested AV products. Our solution does not require customizations for the specific version of the target system, as it can reliably execute non-detectable actions across different versions of the guest OS and AV products. Table 6.1 classifies Laccolith according to the criteria presented in Chapter 2.

Table 6.1. Laccolith comparison.

Tool	C2 Server	Complex Attacks	ATT&CK Tactics Coverage	Needs Pre-Installed Agent	Anti-detection
Laccolith	✓			✗	✓

6.1 Design

We based the design of Laccolith on a set of assumptions applied in general for adversary emulation. The assumptions include:

- Adversary emulation focuses on post-compromise scenarios where the attacker has already gained a foothold inside the system (e.g., through phishing or exploiting a vulnerability) and is performing more malicious actions, such as gaining more privileges and stealing information.
- Adversary emulation is performed in the context of cybersecurity exercises, which are authorized and overseen by a "white team" in

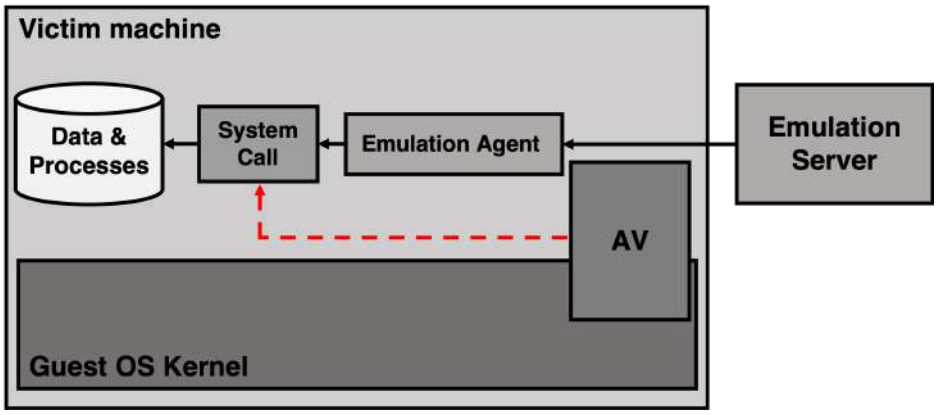


Figure 6.1. Traditional architecture of adversary emulation.

the organization (e.g., system administrators). Adversary emulation can be warranted system privileges as needed.

- Laccolith is designed to perform adversary emulation in a virtualized environment, where hosts are deployed using virtual machines. This is typically the case in cybersecurity exercises, where dedicated networks and hosts are deployed using virtualization infrastructures [37, 153, 154]. For example, all the most popular cyber ranges run in the cloud, leveraging virtualization technologies [36, 40, 38]. Moreover, the solution applies to organizations that run their private data centers, which is often the case for high-criticality domains such as defense, critical infrastructures, and healthcare [155, 156]. Instead, the proposed solution is not meant for networks and services running on bare-metal hardware.

- Endpoints in the environment can be equipped with AV products, as in the case of real computer networks. This assumption is not met by traditional adversary emulation tools, which require turning off AV products to install and run the tools [7], as demonstrated in the evaluation of Chapter 5. Laccolith is designed to overcome this limitation.

The driving idea for our design is to leverage kernel- and hypervisor-

level[1] privileges to execute malicious actions, which would otherwise be detected if performed from user level (i.e., from an application process). AV solutions typically use "hook" functions to intercept invocations of the system calls. When applications (including malicious ones) invoke system calls, the hook functions are executed instead (e.g., by replacing pointers in the system call table), which can check the invocation and detect suspicious activities. In traditional adversary emulation (Figure 6.1), an agent process executes malicious actions on behalf of a red team by issuing system calls that access OS resources, such as files, processes, connections, and others. Without any anti-detection technique, AV products can detect these actions by checking system calls.

To perform non-detectable actions it is necessary to use system calls not monitored by the AVs, hence from the kernel level. Consequently, the hypervisor level is the ideal choice to accomplish that, leveraging Virtual Machine Introspection (VMI) techniques. In our design, we circumvent the anti-virus checks by introducing an emulation agent from the hypervisor. Since the hypervisor has full privileges on a physical machine, it can warrant full read and write access to the state of a virtual machine. We use these privileges to install an emulator into the kernel of the guest OS of the virtual machine. This way, it is possible to directly access guest memory without issuing user-space system calls. From the agent installed in the guest OS kernel, we can perform malicious actions by calling kernel-level APIs. Such calls cannot be detected by AV products since they are not subject to security checks.

Moreover, we designed the kernel-level agent with the ability to run user-level commands, as in traditional adversary emulation tools. Therefore, red teams can combine kernel-level and user-level actions to perform detectable and non-detectable actions. This flexibility enables red teams to perform realistic training exercises for security teams, where the emulated APT leaves only a few selected traces of malicious activity.

Laccolith offers a complete architecture for adversary emulation: a central *emulation manager* orchestrates *emulation agents* across the network

[1]The name *Laccolith* reflects the injection of malicious actions from the lower layers of the software stack. A laccolith is a volcanic phenomenon where magma rises through the Earth's crust, forcing rock strata upward.

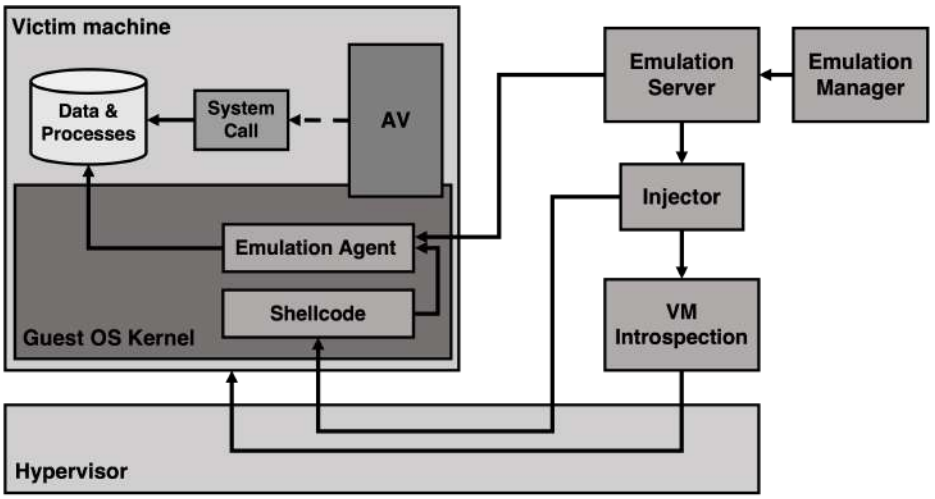


Figure 6.2. Architecture of Laccolith.

and issues actions to execute on the victim hosts. Therefore, the red team controls the emulation agents to reproduce the actions of an APT campaign. The red team can interact with the emulation manager through web and command-line interfaces. Laccolith is neutral in terms of post-facto analysis: the emulation designer will decide which artifacts should be left inside the system for emulation purposes, thanks to the possibility of configuring which actions are executed undetectably or not. For instance, the designer may want to poison some entries in the system registry for forensic analysis. In alternative, the designer may launch processes or connections in a non-detectable fashion in kernel space. In such a case, the participant will leverage event correlation techniques and SIEM to analyze the system state and understand what is happening. On each physical machine, Laccolith installs an *emulation server*, which runs at hypervisor-level and manages the emulation agents on each physical machine. The emulation server installs the emulation agents on each virtual machine by leveraging VMI techniques to modify guest memory and create a new execution flow inside the guest kernel. Moreover, the emulation server forwards communication between the emulation agents and the manager. The communication traffic between the components of Laccolith is invisible to the participants of the cybersecurity exercise since this traffic flows

at the physical level, beyond the virtual networks of the virtual machines.

6.1.1 Emulation server

The *emulation server* is the trickiest component in our architecture. It is responsible for injecting *emulation agents* inside the VMs, by leveraging read/write (R/W) access permissions to modify their state. It relies on installing a kernel-level agent to avoid detection from AV products.

Overview of the injection method

The emulation agent needs to run in a dedicated area of the virtual memory of the VM. However, the allocation of virtual memory cannot be directly performed from the hypervisor but needs to be performed by the guest OS kernel, which is aware of the current layout of memory allocations. Consequently, the injection of the emulation agent consists of two stages. The first stage consists of injecting a small program (a "shellcode") in kernel space, responsible for allocating the code region for the emulation agent, while the second stage entails bootstrapping the asynchronous execution of the agent in that area. The injector overwrites a piece of existing kernel code with the shellcode to make it executed by the kernel. Differently from the *injector* (which runs in the hypervisor and can only work with "guest physical" memory addresses), the shellcode can allocate and access the "guest virtual" memory addresses since it runs within the guest OS kernel. Before selecting the target kernel code to overwrite, we need to address two requirements for the shellcode:

1. *Space*: the shellcode should be small enough to fit into the target kernel code;

2. *Behavior*: if the overwritten kernel code is invoked, the shellcode needs to handle the call without hanging the calling process. Moreover, it should also handle the case of concurrent calls from different processes.

We consider the code of system calls for the injection of the shellcode since their position in the VM virtual memory can be identified with VMI techniques. VMI is an approach to gain visibility and control over VMs

without modifying the guest OS [157]. Moreover, system calls are regularly invoked by applications, thus assuring that the shellcode will be eventually executed. The previously mentioned requirements are the reason why we cannot trivially overwrite a system call with the code of the *emulation agent*: the first one would restrict the size of the agent, which would hamper the implementation of malicious actions; the second requirement implies that the system call code needs to be restored at some point, thus removing the injected code.

To choose a target system call to overwrite, we look for a *linear region of code*, i.e., a memory region that meets the following requirements:

1. The memory region is from the code area of the guest OS kernel; this code is only executed by starting from the initial address of the memory region. For example, this requirement is satisfied if the memory region exactly matches the code of an individual kernel function; in this case, other kernel code only jumps to the initial address of the memory region (i.e., there are no jumps to addresses in the middle of the memory region).

2. The code in the memory region does not call other functions. This requirement prevents code not belonging to the memory region from returning in the middle of the region.

3. The region fits inside a memory page without crossing page boundaries.

The first and second requirements allow us to inject arbitrary code in the memory region without risking the kernel jumping in the middle of the region, which would likely raise CPU exceptions (e.g., executing an invalid opcode). The shellcode will execute when the kernel executes the memory region, in place of the original code. The third requirement is essential to make it easier to modify the VM memory from the hypervisor since code over multiple pages in virtual memory does not necessarily map to contiguous pages in physical memory. An example of an eligible memory region in Microsoft Windows OS is the *MmQueryVirtualMemory* function (about $3,800$ bytes), which is called by the *NtQueryVirtualMemory* system call. In general, linear regions of code are plentiful and easy to find: given the function call graph of the kernel code (e.g., *ntoskrnl.exe* in Microsoft

Windows), a linear region of code is a function without fan-out that fits within a single memory page.

Injection method

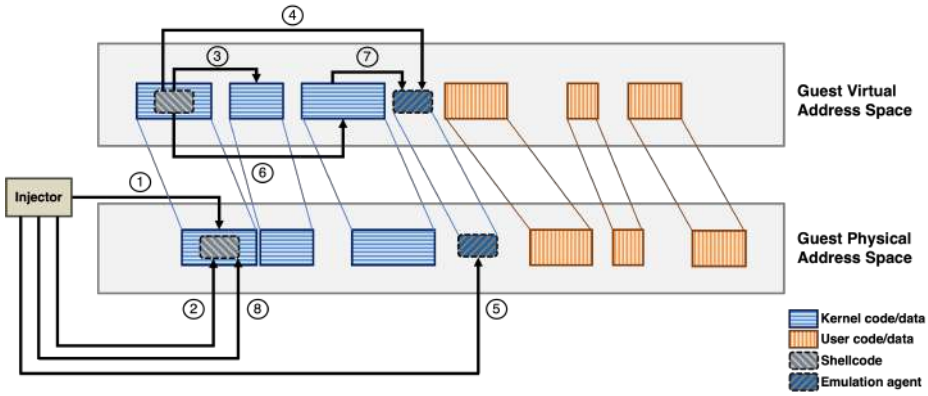


Figure 6.3. Injection method in Laccolith.

The injection method is shown in Figure 6.3 and summarized as follows. First, the *injector* searches the code of the target system call in memory (step ①). Since the *injector* can only access VM memory through guest physical addresses, it scans memory by looking for code that matches the initial unique bytes of the target system call. The injector leverages VMI to make this process more portable and efficient: it determines from VMI the unique bytes of the system call from the kernel binary and the relative offset of the system call within the kernel virtual address space. Then, it only scans memory pages on that relative offset. Once the position of the target system call has been found, the *injector* overwrites it with the shellcode (step ②).

Then, the shellcode will eventually execute when the system call is invoked. The shellcode will execute for a limited time, long enough to allocate a memory area and communicate to the emulation server that the memory area is ready for loading the emulation agent (the last step of the method). The shellcode uses the APIs of the guest OS kernel (step ③) to allocate a contiguous area of guest virtual memory (for example, the *MmAllocateContiguousMemory* function from the Windows Driver Kit

(WDK) [158]). The shellcode then obtains pages of contiguous virtual memory (e.g., 16KB in our implementation) from the guest virtual address space (step ④), without the constraints that previously applied for the shellcode.

During this time window, any other concurrent call to the target system call must immediately return since the shellcode should execute only once. For this purpose, the shellcode acquires a spinlock in a *non-blocking* way, without busy-waiting if the spinlock is already held (e.g., using the *xchg* instruction on Intel architectures [159]). Once acquired, the shellcode holds the spinlock to avoid concurrent executions.

Next, the shellcode writes a pre-defined value ("egg"), also known by the injector, within the allocated memory area. This way, the injector can find the allocated memory area using only guest physical addresses. The injector overwrites the allocated memory with the code of the emulation agent (step ⑤). Finally, the shellcode sets up an execution context to run the emulation agent in the kernel. Again, the shellcode uses APIs of the guest OS kernel to create a kernel thread (step ⑥), such as using the *PsCreateSystemThread* function in Windows. The new kernel thread will be configured to run the code of the emulation agent in the allocated memory area (step ⑦), e.g., using the *StartRoutine* of the Windows kernel. As a result, the emulation agent will execute with high privileges since it will run in kernel mode. After the agent starts, the injector restores the original system call code to leave no traces in the target VM other than the emulation agent (step ⑧).

6.1.2 Emulation agent

The *emulation agent* is responsible for receiving and executing commands from the emulation server. Since it runs with high privileges, it can call kernel APIs as if it were a kernel module. The agent can also access the OS resources, such as the process descriptors and the system registry. Through these resources, the agent can access the memory of user-level processes since a process descriptor indicates the memory regions where the code and data of a process are. It can both read (e.g., process dumping to steal passwords, tokens, and other data) or write (e.g., process/DLL injection to hide malicious code in system processes) the processes' memory and modify the system configuration, e.g., changing the registry for persis-

tence purposes. In addition, the agent can read/write the file system, e.g., navigating through folders, creating new files, and deleting existing ones. It can also create new user-level processes to execute commands, such as simulating fileless malware by running system binaries (e.g., PowerShell, WMI) and creating new OS resources. For example, the execution of user-mode commands can be accomplished using worker factories [160, 161], a Windows mechanism that allows the kernel to create user-space thread pools and let them execute specific tasks.

6.1.3 Emulation manager

The *emulation manager* communicates with the agents. Its role is critical to orchestrating the low-level actions performed by the agents into emulating the complex behavior of APTs, as in the example of Chapter 3. It also offers a user interface to manage Laccolith. In particular, it offers interfaces to inject agents into specific targets, to access C2 functions (e.g., listing connections to agents, sending a command to an agent, reading its output, starting an autonomous operation), to customize parameters (e.g., choosing which payload to inject), and upload and download files (e.g., to exfiltrate data and information gathering). The emulation manager is also responsible for managing the *facts*. A fact is a piece of information about the target system, helpful to run an ability. The manager communicates the fact values to emulation agents when they need specific information to perform an action. For example, suppose the goal of the emulation is to perform lateral movement to another machine in the local network. The emulation agent will perform a network scanning action, whose outcome will be a fact containing the username and IP address of the target machine. The agent will then use this fact to perform lateral movement.

6.1.4 Implementation

We implemented the design of Laccolith for the QEMU hypervisor for x86_64 [162], running on a Linux host managed by Libvirt [163], and with hardware-supported virtualization based on KVM [164]. Laccolith runs as a privileged process on the Linux host and accesses VM memory through a virtual device file. Our implementation targets the Microsoft Windows OS as the guest OS for the victim machine. We selected Windows

as the target OS since it is a common target for APTs and is widely used in the adversary emulation landscape. For example, the MITRE ATT&CK framework started as a project to gather information about TTPs against Windows-based systems [165]. We used Volatility [166], a VMI framework, to get Windows kernel symbols. We chose Volatility since it is a widespread and well-known framework that offers several profiles to inspect the memory of several OSes. Using a popular VMI framework facilitates the portability of Laccolith to new versions of the guest OS (e.g., builds of Microsoft Windows) and new guest OSes since the community continuously provides multiple profiles to align VMI with new releases of the OSes.

6.2 Experimental Evaluation

To evaluate Laccolith, we replicated the experimental evaluation performed on adversary emulation tools in Chapter 5. We performed this analysis using the same experimental setup presented in Section 5.1. Before our experiments, we tested the portability of Laccolith across different versions of the guest OS. We deployed Laccolith on 7 Windows 10 versions, which span over three years, with significant changes across the versions [167]. Laccolith leverages Volatility to apply the injection method in a portable way. Indeed, Laccolith could correctly inject the emulation agent in all the tested versions of Windows without the need for specific customizations. In our experiments, we focus on Microsoft Windows 10 build 19044.

The analysis presented in Chapter 5 showed that traditional adversary emulation is cumbersome and unable to perform intrusive actions without being detected. We present an experimental analysis of detectability for Laccolith. For this analysis, we implemented four adversary profiles with Laccolith: *Thief*, *Op-2*, *Ransomware*, and *Shares Hunter*, as described in Table 6.2. Since we had to develop adversary profiles from scratch for Laccolith, we did not aim for a verbatim reimplementation of the adversary profiles of CALDERA. Instead, the adversary profiles in Laccolith match the CALDERA profiles in terms of high-level strategy of the attackers. Furthermore, we relate the adversary profiles for Laccolith with real-world APTs (threat-informed adversary emulation). The new profiles in Laccol-

ith cover all the tactics covered by CALDERA profiles, except Privilege Escalation (covered by the *You Shall (Not) Bypass* profile) since Laccolith already has high privilege. The Ransomware profile also covers the Impact tactic, which is not covered by the default profiles of CALDERA. We introduced this profile to show the ability of Laccolith to support complex operations and cover a relevant cybersecurity threat.

Table 6.2. Adversary profiles in Laccolith.

Profile	Description	Tactics	Commands	High-level actions	Referenced APTs
Thief	Exfiltrate files from local user desktop	Discovery, Collection, Exfiltration	Directory listing (to find local users and to list desktop files), Read file	(1) Find local users, (2) List user desktop, (3) Exfiltrate a list of staged files	APT1, OilRig, APT3
Op-2	Upload a Powershell script in a system folder and install a scheduled task that executes that script at boot, get system version and dump LSASS memory	Persistence, Credential access	Write file, Write to registry, Version, Dump process memory	(1) Write file on remote file system, (2) Install a scheduled task on the remote Windows target, (3) Get system version, (4) Dump lsass credentials	Remsec (Strider), Ke3chang
Ransomware	Discover and exfiltrate sensitive files, encrypt them and leave a message	Discovery, Collection, Exfiltration, Impact	Directory listing, Read file, Write file	(1) Find local users, (2) Find sensitive files, (3) Exfiltrate a list of staged files and encrypt them, (4) Encrypt remote files, (5) Write ransom message	APT3, Bad Rabbit (multiple APTs)
Shares Hunter	Read ARP cache to find neighbors, scan them to see which has Netbios/SMB sharing enabled, enumerate shares	Discovery, Lateral Movement	User-mode commands: ipconfig, arp, nbtstat, net view	(1) Find local IP address, (2) Read ARP cache, (3) Scan hosts for SMB/NetBIOS, (4) Enumerate network shares	Conti, APT32

Table 6.3 provides details on which commands of the emulation agent are involved in each profile, to get more insight into the adversary profiles of this experiment. In total, the profiles cover 7 commands implemented by the emulation agent, including actions to access the filesystem, the system registry and to execute user-space commands. Moreover, the Laccolith agent provides additional commands for managing adversary emulation campaigns, not shown in the table for the sake of brevity. Overall, the newly implemented profiles allowed us to test the functionalities of Laccolith in depth.

To compare Laccolith to state-of-the-art solutions, we performed the same experimental analysis for detectability. Table 6.4 shows the adversary profile execution progress for the four profiles against all the chosen AVs. It is possible to notice that all profiles achieve complete execution progress, meaning that the AVs do not detect any of their actions. It is worth noting that the Shares Hunter profile has seven actions according to Table 6.4, instead of the four mentioned in Table 6.2. This happens because this profile does not have an *a priori* planning, but a dynamic one in which some actions depend on the outcome of the previous ones. In particular, since this profile performs network discovery operations, the result of its actions depends on whether there are any neighbors in the network. We assumed that the target VM interacted with the other VM recently, so it has the IP address of VM2 in its ARP cache alongside the gateway one. Both VM1 and VM2 have the NetBIOS sharing option active.

Table 6.3. Coverage of commands of the Laccolith agent, with respect to the adversary profiles.

Profile	Commands
Thief	dir, read
Op-2	write, setkey, version, dump
Ransomware	dir, read, write
Shares Hunter	read, usermode

The injection relies on overwriting the code of a system call or a kernel function called by a system call. Then, the shellcode executes only once, and the other concurrent calls return, as described in Section 6.1.1. After loading the emulation agent, the original code is restored. This injection method can fail since the original code may be restored concurrently with a thread executing that function, which could encounter invalid opcodes due to misalignment or valid code that returns errors because of invalid values in the CPU registers. This failure results in an assertion failure (e.g., *bug check*) within the kernel. Moreover, a critical system process

Table 6.4. Adversary profile execution progress for Laccolith.

Profile	Windows Defender	Avast	AVG	Kaspersky	Avira
Thief	■ 3/3	■ 3/3	■ 3/3	■ 3/3	■ 3/3
Op-2	■ 4/4	■ 4/4	■ 4/4	■ 4/4	■ 4/4
Ransomware	■ 5/5	■ 5/5	■ 5/5	■ 5/5	■ 5/5
Shares Hunter	■ 7/7	■ 7/7	■ 7/7	■ 7/7	■ 7/7

(e.g., *SYSTEM svchost.exe*) may crash because the system call does not exhibit the expected behavior.

We performed preliminary experiments to gain insights into these events. In these initial experiments, we executed the injection after an increasing amount of time after the boot of the victim machine. We found that these events are more likely if the injection is performed within a few minutes right after the boot. If the injection is performed after a few minutes have passed since the boot, it becomes more reliable. This behavior can be explained by the higher activity of system processes in the early phases of the start-up, which make high use of system calls and can expose these processes to failures.

Therefore, the success of the injection method depends on the fraction of time the system spends executing the injected system call, and the percentage of cases that the system call is invoked by system processes. To quantify the effectiveness of the injection method, we performed more experiments by focusing on the favorable case of injection after the boot had been completed and the system had stabilized. The experiments consisted in repeating the injection method multiple times, each time from a clean condition (e.g., a full restart of the victim machine), and measuring how many times the injection was successful.

We conducted 20 experiments for each AV, re-booting the target VM each time to have independent samples. The timing of the injection was set to one minute after the Windows login prompt appeared. To verify if the injection was successful, we sent the agent an *echo* and a *close* command to

check if it could execute commands and terminate gracefully. The injection fails if the connection is received, but it is impossible to perform these operations. If there is some minor error from user applications, that cannot be attributable to a security alarm (e.g., a generic "unknown exception in explorer.exe"), we still consider the injection successful.

Table 6.5 summarizes the experimental results. The AV does not impact the injection success since all the setups with the different AVs exhibit similar behavior. The overall success rate of the injection method has been 90/100, hence 90% of experiments. Considering a margin of error of $1/\sqrt{N}$, where N is the overall number of repetitions [168], the success rate of the injection method ranges between 80% and 100%. Even in the worst case, the success rate is still higher than the probability of running CALDERA and Inceptor without being detected, which had approximately a 25% probability of success, neglecting the issue of bypassing UAC which makes the process even more uncertain.

It is worth noting that the injection success for Laccolith can be further improved by injecting into a target linear region of code that is executed less frequently, to reduce the probability that the injection clashes with the execution of the target system call. This can be achieved by a preliminary profiling of the execution frequency of system calls. Since the profiling depends on the workload of the specific system under evaluation, and since this represents an engineering problem, we consider this beyond the scope of our research work.

6.3 Threats to validity

This section illustrates the threats to the validity of Laccolith.

Threats to external validity

Victim OS. We targeted Microsoft Windows as **guest OS** for the victim machine, which may affect the generality of our solution. Different OS environments may exhibit unique vulnerabilities and behaviors, making it challenging to generalize findings to diverse operating systems. We focus on Windows since it is a common target for APTs and is widely used in the adversary emulation landscape. For example, the MITRE ATT&CK

Table 6.5. Injection success for Laccolith.

Anti-virus	Injection Success
Windows Defender	19/20
Avast	17/20
AVG	17/20
Kaspersky	19/20
Avira	18/20
Overall	90/100

framework started as a project to gather information about TTPs against Windows-based systems [165]. It exhibits a diverse attack surface, encompassing a wide range of services, applications, and configurations, which enables a comprehensive exploration of attack vectors. This complexity makes Windows a realistic target for enterprise systems. Other guest OSes can also be targeted by our solution, as discussed later in the paper.

Adversary emulation tools. We performed experiments with **three state-of-the-art open-source adversary emulation tools**: MITRE CALDERA, Atomic Red Team, and Invoke-Adversary. We are aware that the choice of specific tools or techniques for adversary emulation can impact the validity of the experimental study. According to a recent survey [7], these tools are the most mature and are aligned to the MITRE ATT&CK matrix, which allow us to reproduce threat behavior reliably.

Anti-viruses. We evaluated the detectability of adversary emulation against five popular AV products. The **choice of antivirus solutions** can be a limitation of our evaluation. We focused on antivirus solutions that represent those commonly used in real-world scenarios [137], considering the ones that aligned better with the typical deployment context of adversary emulation.

Threats to internal validity

Inconsistent configuration settings for the AVs may threaten the internal validity of the study. We used standard configurations of the five AV solutions to avoid variations of results not due to the effectiveness of the adversary emulation tools.

Threats to construct validity

The **choice of detection metrics** can impact the validity of the study. We adopted three different metrics for the evaluation of detectability, which encompass all the key aspects of detectability: perimeter breach detection (*injection success*), malicious actions detection (*atomic technique detection*), and timeliness in the APT campaign detection (*adversary profile execution progress*).

Chapter 7

Conclusions

Throughout this dissertation, the primary objective was to bridge the gap between the dynamic landscape of cybersecurity and the limitations in existing adversary emulation strategies. The aim was to develop a comprehensive framework for adversary emulation that leverages Cyber Threat Intelligence and integrates anti-detection capabilities, thereby providing a robust solution for proactive security measures against APTs.

The devised CTI-driven framework for adversary emulation represents a relevant contribution to the field. By automating the extraction of attack techniques from unstructured CTI documents, this framework streamlines the emulation planning process, enabling a more accurate replication of real-world threat actors' behaviors. Furthermore, the integration with Laccolith has demonstrated the feasibility of conducting realistic APT emulations with enhanced anti-detection capabilities, addressing the limitations of existing open-source tools. The extensive comparative analysis of state-of-the-art solutions for adversary emulation highlighted critical gaps in the effectiveness of existing tools, particularly in evading detection from AV and EDR systems. This analysis underscored the need for a more sophisticated approach to adversary emulation, emphasizing the significance of integrating anti-detection measures seamlessly into the emulation process.

The findings of this research have profound implications for practical applications in the realm of cybersecurity. By emphasizing the importance of CTI integration and anti-detection capabilities, this research urges organizations to adopt a more proactive approach to security measures, en-

abling them to stay ahead of the evolving threat landscape. While the CTI-driven framework has demonstrated its potential in enhancing the emulation process, the identified limitations necessitate a more nuanced and adaptive approach to ensure the comprehensive replication of APT behaviors. Organizations must recognize the challenges posed by the heterogeneity and dynamic nature of CTI sources, thereby emphasizing the need for continuous refinement and advancement in the field of adversary emulation. Laccolith, in particular, offers a versatile and comprehensive framework for conducting adversary emulation exercises that closely mirror real-world APT behaviors, facilitating robust security assessments and training.

While this research represents a significant step forward in addressing the limitations of current adversary emulation techniques, there is still room for further exploration and improvement. Future research endeavors should focus on refining the automation of CTI extraction processes to overcome the limitations identified in the process of emulation plan generation, such as the difficulty in generalizing APT behaviors and the extreme heterogeneity of CTI sources, which substantiate the complexities involved in achieving a holistic and representative emulation of APT activities. Moreover, expanding the scope of Laccolith to encompass a broader range of APT behaviors will increase its effectiveness in emulating real-world threat actors realistically.

In conclusion, the comprehensive framework for CTI-driven adversary emulation, along with the innovative Laccolith solution, marks a pivotal advancement in the cybersecurity field. By emphasizing the significance of proactive security measures and the integration of sophisticated emulation techniques, this research paves the way for a more resilient and adaptive approach to safeguarding digital assets and data against the ever-evolving threat of APTs.

Appendix A

MITRE ATT&CK

The MITRE ATT&CK framework is a comprehensive knowledge base that outlines tactics, techniques, and procedures (TTPs) that adversaries leverage during cyber attacks. It is designed to provide organizations with a deeper understanding of the different stages of an attack and how adversaries operate within a network. The framework represents a valuable resource for security teams to improve their defense strategies, develop effective detection and response capabilities, and enhance overall cybersecurity posture. By understanding the tactics and techniques employed by real-world threat actors, organizations can better prepare for, detect, and respond to cyber-attacks. ATT&CK is continually updated to incorporate new techniques and behaviors observed in the evolving landscape of cyber threats. This ensures that the framework remains relevant and up-to-date in the face of emerging and evolving attack techniques.

A.1 Enterprise Matrix

The *Enterprise Matrix* is a visualization of the MITRE ATT&CK framework that provides a comprehensive view of the TTPs used by adversaries across different enterprise environments. It helps security professionals and organizations understand how threat actors may employ specific tactics and techniques to compromise various parts of an enterprise's network, systems, or data. The matrix is organized into columns representing the various tactics, while the rows represent the specific techniques as-

sociated with each tactic. Within the matrix, individual cells denote the relationship between specific tactics and techniques, providing insights into how adversaries typically progress through different stages of an attack.

A.1.1 Tactics

Tactics are the high-level entities in the Enterprise Matrix and represent the adversary goals at different steps of the attack [169]. ATT&CK encompasses 14 tactics, detailed in the following:

- *Reconnaissance*: encloses various approaches that allow adversaries to gather information by active engagement or passive observation. This data can include detailed insights into the target organization, its underlying systems, and the people associated with it. This acquired intelligence represents a valuable asset for the adversaries as they progress through different phases of their activities, like orchestrating initial access, assessing and ranking objectives post-compromise, and steering their subsequent reconnaissance efforts [170].
- *Resource Development*: involves procedures through which hostile entities create, purchase, compromise, or steal resources to facilitate their targeted objectives. These resources may span infrastructure, accounts, or various capabilities. Leveraging these resources, adversaries can support different stages of their operations, like utilizing bought domains for facilitating Command and Control, deploying email accounts for phishing during initial access, or illicitly obtaining code signing certificates to streamline Defense Evasion [171].
- *Initial Access*: comprises methods to secure an initial foothold within a network using various entry paths. Strategies for gaining this foothold include targeted spearphishing and exploiting vulnerabilities on publicly accessible web servers. Initial access points obtained by the adversary can provide continuous access, such as through valid accounts and external remote services, or may have limited functionality due to password changes [172].
- *Execution*: the attacker aims to deploy adversary-controlled code on local or remote systems. Malicious code execution is often linked

with techniques from other tactics to achieve broader objectives, like network exploration or data theft [173].

- *Persistence*: the adversary seeks to maintain access to systems despite restarts, updated credentials, or interruptions that might cut off their connection. Techniques for persistence include accesses, actions, or configuration changes that allow them to retain their presence inside systems [174].

- *Privilege Escalation*: encompasses practices to acquire higher-level permissions within a system or network. While adversaries might gain initial access and explore a network with limited privileges, elevated permissions are necessary to fulfill their objectives. Common strategies involve exploiting system weaknesses, misconfigurations, and vulnerabilities to achieve escalated access rights, such as root level, local administrator, or accounts with specific system access or functionalities [175].

- *Defense Evasion*: the aim is to elude detection during compromise. Defense evasion techniques comprise disabling security software, encrypting data and scripts, and leveraging trusted processes to conceal and camouflage malware [176].

- *Credential Access*: consists of techniques to obtain credentials, including account names and passwords. Methods for stealing credentials range from keylogging to credential dumping. Unauthorized access with legitimate credentials not only grants adversaries access to systems but also makes their activities more challenging to detect, providing the opportunity to create additional accounts to accomplish their objectives [177].

- *Discovery*: entails techniques to gather insights about the system and internal network. These strategies allow adversaries to survey the environment and understand the surroundings before determining their course of action. Native operating system tools are frequently employed to gather this post-compromise information [178].

- *Lateral Movement*: includes strategies to infiltrate and control remote systems. Navigating the network to locate and access the de-

sired target is often crucial for achieving their primary objectives. Lateral Movement typically involves traversing multiple systems and accounts, with adversaries either installing their remote access tools or utilizing legitimate credentials and native network and operating system tools for stealthier maneuvers [179].

- *Collection*: the adversary strives to gather relevant information for their following objectives. Collection helps get data to exfiltrate from several sources, such as storage devices, browsers, and emails. Collection techniques may involve capturing screenshots and keystrokes [180].
- *Command and Control*: refers to all the techniques to communicate with systems under the attacker's control within a target network. To evade detection, adversaries typically seek to mimic regular network traffic. The techniques for establishing Command and Control may vary in stealthiness depending on the victim's network structure and defensive measures [181].
- *Exfiltration*: adversaries aim to steal data from the target network. Attackers often package data in ways that evade detection, including compression and encryption. Exfiltration techniques typically involve transferring data over command and control channels or alternative channels, sometimes with limitations to the transmission size [182].
- *Impact*: encloses actions to disrupt system availability or compromise integrity by controlling business and operational processes. Impact strategies may involve data destruction or tampering. Adversaries often aim to make business processes appear normal even when altered to benefit their goals. These techniques may serve as a means to achieve their ultimate objectives or as a cover for a confidentiality breach [183].

A.1.2 Techniques

Techniques are a specialization of tactics, representing how an attacker achieves the goal defined by the specific tactic. Techniques are the methods and approaches attackers leverage to carry out the various stages of

an attack. For instance, within the Execution tactic, the techniques may include using malicious software or scripts to run commands on a target system [184]. Several techniques specialize into sub-techniques, which elaborate on specific variations or nuances of a particular technique. These sub-techniques provide more granular insights into the different ways to execute an attack. They help organizations understand the subtler details of how adversaries may carry out an attack and the various approaches they may take to achieve their objectives.

A.1.3 Procedures

Procedures are the detailed step-by-step sequences or methodologies adversaries follow to execute specific techniques during a cyber attack. These steps often include specific commands, tools, or actions adversaries take to achieve their desired outcomes. Procedures provide a deeper understanding of how threat actors implement various tactics and techniques to achieve their objectives within a targeted system or network environment. While techniques represent the general methods used by adversaries, procedures offer a more granular view of the specific actions taken to execute those techniques.

Appendix B

Introduction to CALDERA

CALDERA is an open-source adversary emulation framework developed by MITRE [1, 17]. The main use case for CALDERA is *autonomous red team engagements* [52], to allow security practitioners to replicate the behavior of cyber threats. CALDERA leverages the client-server architecture typical of adversary emulation tools, illustrated in Figure B.1. Specifically, the server is the command-and-control (C2) server, while the client is represented by the emulation agent installed on the victim machine.

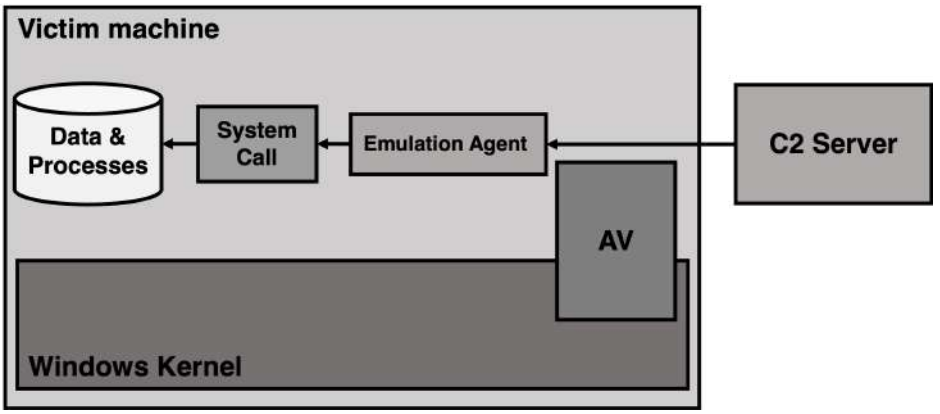


Figure B.1. Architecture of CALDERA.

B.1 Command-and-Control server

CALDERA's *command-and-control server* is a critical component of the emulation framework. It represents the centralized nerve center for managing and orchestrating cyber threat simulations and red teaming exercises. The C2 server provides security professionals with the capability to emulate sophisticated attack scenarios, test defensive measures, and assess an organization's readiness to combat real-world threats. It also allows security teams to remotely control and coordinate various agents and toolsets, mimicking the adversaries' tactics, techniques, and procedures (TTPs). The command and control server empowers organizations to enhance their cybersecurity posture by simulating, evaluating, and ultimately improving their defenses against evolving cyber threats.

B.2 Agents

Agents play a pivotal role in enabling the emulation of adversary behavior and conducting cyber threat simulations within an organization's network. CALDERA's agents can simulate various types of malicious activities, providing valuable insights into an organization's security posture and the effectiveness of its defense mechanisms. There are several types of agents within the CALDERA framework, each serving a specific purpose and possessing unique capabilities. Agents can be classified according to the high-level goals or their implementation and communication protocols.

- *Implant Agents*: designed to be implanted into target systems to carry out specific tasks or actions, such as lateral movement, data exfiltration, or privilege escalation. They mimic the behavior of real-world attackers, allowing security professionals to assess the effectiveness of their detection and response mechanisms.

- *Beacon Agents*: are lightweight, low-profile agents that aim to establish communication with the C2 server while remaining relatively stealthy. They help maintain persistent access to compromised systems and facilitate ongoing monitoring and data collection without raising suspicion.

- *Pivot Agents*: enable lateral movement within a network, mimicking the techniques used by advanced persistent threats (APTs) and other sophisticated adversaries. They help assess the security infrastructure's ability to detect and prevent unauthorized internal access and movement.

- *Exfiltration Agents*: focused on extracting sensitive data from the target environment. They offer various data exfiltration techniques, such as file transfer, network communication, or other covert channels, to evaluate the effectiveness of data loss prevention measures and incident response procedures.

According to implementation and communication protocol, it is possible to deploy the aforementioned agents through:

- *Sandcat Agents*: developed in GoLang, they can communicate through a diverse array of command and control (C2) channels, such as HTTP, GitHub GIST, and DNS tunneling. Its versatility in leveraging different communication protocols makes Sandcat an effective tool for executing stealthy operations and evading detection [52].

- *Manx Agents*: are GoLang agents that enable seamless and reliable contact with the C2 server through the TCP protocol. With its primary functionality as a reverse shell, Manx empowers cybersecurity professionals to execute intricate maneuvers within target systems, facilitating exploration, monitoring, and remote management of endpoints with precision and control [52].

- *Ragdoll Agents*: coded in Python, they operate as dynamic agents specialized in establishing communication via HTML contact. Through its tailored capabilities, Ragdoll facilitates seamless integration within web-based environments, enabling comprehensive interaction and data transmission with minimal footprint [52].

B.3 Abilities

In the context of CALDERA, an *ability* denotes a precise implementation of an ATT&CK technique, meticulously tailored for execution on

active agents. These abilities encompass a comprehensive set of directives, encompassing specific commands, compatible platforms, and corresponding executors (e.g., Windows as the target platform and PowerShell as the executor). They also include payloads, ensuring optimal performance and efficacy. Additionally, these abilities feature a reference to a dedicated module on the CALDERA server, configured to parse the generated output, enabling in-depth analysis and informed decision-making throughout the simulated cyber threat operations [52].

B.4 Adversary Profiles

Adversary profiles represent groups of abilities assembled to emulate the tactics, techniques, and procedures (TTPs) accessible to a potential threat actor. These profiles are instrumental during active operations to determine the precise sequence and selection of abilities to execute within the simulation environment. By encapsulating a diverse range of potential attack vectors and methodologies, adversary profiles facilitate a comprehensive understanding of the potential strategies and threat landscapes, allowing security professionals to fortify their defensive measures and overall cybersecurity resilience against several adversarial scenarios [52].

B.5 Operations

In CALDERA, operations represent the execution of abilities on specific agent groups, while the selection of which abilities to run is determined by the utilization of adversary profiles. The planners within CALDERA dictate the execution order of the abilities. CALDERA offers the following default planners:

- *Atomic*: executes abilities in the adversary profile following the atomic ordering designated by the adversary.
- *Batch*: runs all abilities in the adversary profile simultaneously.
- *Buckets*: runs abilities in the adversary profile categorized by their ATT&CK tactic.

When an operation triggers an ability, a link is generated for each respective agent under specific conditions:

1. All link facts and fact requirements have been satisfied.
2. The agent possesses an executor compatible with the ability's designated platform.
3. The agent has not yet executed the particular ability or the ability is marked as repeatable.

Facts are identifiable information related to a specific computer system, referenced by their names in ability files, with their values replacing the placeholders when a link is created. Link commands can be obfuscated to enhance covert operations depending on the operational stealth settings. The resulting links are incorporated into the operation chain, which encompasses all generated links associated with the ongoing operation. When agents check-in, they retrieve their designated instructions, which are subsequently executed based on the selected executor. The resulting outcomes are then transmitted back to the C2 server. Upon receiving these results, CALDERA employs parsers to identify and incorporate any collected facts into the operation. These parsers analyze the output of executed abilities, extracting potential facts. If these potential facts comply with the predefined fact rules, they are incorporated into the operation for utilization in subsequent links [52].

B.6 Plugins

It is possible to extend CALDERA's functionalities through *plugins*. Plugins are essential to extend the basic client-server architecture. The most relevant plugins are detailed in the following:

- *Atomic*: allows users to leverage the TTPs provided by Atomic Red Team [18], enriching CALDERA's knowledge base with several additional procedures.
- *Builder*: facilitates the creation and customization of adversary emulation scenarios and campaigns. It allows users to build and configure

specific attack sequences, techniques, and behaviors through operations.

- *Debrief*: supports the post-execution analysis and assessment of adversary emulation activities conducted within CALDERA. It enables users to review and analyze the results, outputs, and data generated during operations.
- *Emu*: provides the adversary emulation plans defined in the Adversary Emulation Library by the CTID. Emu is a helpful plugin for emulating real-world APTs.
- *Human*: enables the deployment of agents that perform user actions on a target system to obfuscate red team actions. Each human is built for a specific operating system and leverages the Chrome browser along with other native OS applications to perform a variety of tasks.
- *Manx*: provides all the functionalities related to Manx agents, such as reverse shell capabilities.
- *Sandcat*: as for the Manx plugin, Sandcat supplies the functionalities related to the homonymous agent.
- *Stockpile*: is the most critical plugin for CALDERA. It provides key components for CALDERA, i.e., abilities, adversary profiles, and planners.

Bibliography

[1] Andy Applebaum, Doug Miller, Blake E. Strom, Chris Korban, and Ross Wolf. Intelligent, automated red team emulation. *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016.

[2] Symantec Security Response. W32.Stuxnet Dossier. https://docs.broadcom.com/doc/security-response-w32-stuxnet-dossier-11-en.

[3] Citizen Lab. Tracking GhostNet: Investigating a Cyber Espionage Network. https://citizenlab.ca/wp-content/uploads/2017/05/ghostnet.pdf.

[4] Kaspersky. Carbanak APT: The Great Bank Robbery. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064518/Carbanak_APT_eng.pdf.

[5] Mandiant. M-Trends 2021. https://www.mandiant.com/resources/m-trends-2021.

[6] and others. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[7] Polina Zilberman, Rami Puzis, Sunders Bruskin, Shai Shwarz, and Yuval Elovici. Sok: A survey of open-source threat emulators. *arXiv preprint arXiv:2003.01518*, 2020.

[8] MITRE. Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression. https://www.mitre.org/sites/default/files/publications/stix.pdf.

[9] MITRE. The Trusted Automated eXchange of Indicator Information. https://taxii.mitre.org/about/documents/Introduction_to_TAXII_White_Paper_May_2014.pdf.

[10] FireEye. Follow The Money: Dissecting the Operations of the Cyber Crime Group FIN6. https://www2.fireeye.com/rs/848-DID-242/images/rpt-fin6.pdf.

[11] Symantec. Japan-Linked Organizations Targeted in Long-Running and Sophisticated Attack Campaign. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/cicada-apt10-japan-espionage.

[12] Cybersecurity & Infrastructure Security Agency. Ransomware Activity Targeting the Healthcare and Public Health Sector. https://www.cisa.gov/uscert/ncas/alerts/aa20-302a.

[13] The Record. Inside Conti leaks: The Panama Papers of ransomware. https://therecord.media/conti-leaks-the-panama-papers-of-ransomware/.

[14] The Free Library. MuddyWater APT Group's Operations Leaked On The Dark Web. https://www.thefreelibrary.com/MuddyWater+APT+Group%27s+Operations+Leaked+On+The+Dark+Web.-a0584888340.

[15] CTID. OilRig Adversary Plan. https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/oilrig.

[16] Andy Applebaum, Doug Miller, Blake E. Strom, Henry Foster, and Cody Thomas. Analysis of automated adversary emulation techniques. In *SummerSim*, 2017.

[17] MITRE. CALDERA. https://github.com/mitre/caldera.

[18] Red Canary. Atomic Red Team. https://atomicredteam.io/.

[19] Guardicore. Infection Monkey. https://www.guardicore.com/infectionmonkey/.

[20] CyberMonitor. Invoke-Adversary. https://github.com/CyberMonitor/Invoke-Adversary.

[21] Spot the Planet. Bypassing Cylance and other AVs/EDRs by Unhooking Windows APIs. https://www.ired.team/offensive-security/defense-evasion/bypassing-cylance-and-other-avs-edrs-by-unhooking-windows-apis.

[22] CyberStruggle. FireEye EDR Bypassed with Basic Process Injection. https://cyberstruggle.org/fireeye-edr-bypassed-with-basic-process-injection/.

[23] Vittorio Orbinato, Mariarosaria Barbaraci, Roberto Natella, and Domenico Cotroneo. Automatic mapping of unstructured cyber threat intelligence: An experimental study:(practical experience report). In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pages 181–192. IEEE, 2022.

[24] MITRE Corporation. MITRE ATT&CK. https://attack.mitre.org/.

[25] Yizhe You, Jun Jiang, Zhengwei Jiang, Peian Yang, Baoxu Liu, Huamin Feng, Xuren Wang, and Ning Li. TIM: threat context-enhanced TTP intelligence mining on unstructured threat data. *Cybersecurity*, 5(1):1–17, 2022.

[26] Kiavash Satvat, Rigel Gjomemo, and VN Venkatakrishnan. Extractor: extracting attack behavior from threat reports. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 598–615. IEEE, 2021.

[27] Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. TTPDrill: Automatic and Accurate Extraction of threat actions from unstructured text of cti sources. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 103–115, 2017.

[28] Ghaith Husari, Xi Niu, Bill Chu, and Ehab Al-Shaer. Using entropy and mutual information to extract threat actions from cyber threat intelligence. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2018.

[29] Benjamin Ampel, Sagar Samtani, Steven Ullman, and Hsinchun Chen. Linking common vulnerabilities and exposures to the mitre att&ck framework: A self-distillation approach. *arXiv preprint arXiv:2108.01696*, 2021.

[30] Hyeonseong Jo, Yongjae Lee, and Seungwon Shin. Vulcan: Automatic extraction and analysis of cyber threat intelligence from unstructured text. *Computers & Security*, 120:102763, 2022.

[31] Endgame Inc. Red Team Automation. https://github.com/endgameinc/RTA.

[32] Nextron Systems GmbH. APTSimulator. https://github.com/NextronSystems/APTSimulator.

[33] Uber Technologies Inc. Metta. https://github.com/uber-common/metta.

[34] TryCatchHCF. DumpsterFire. https://github.com/TryCatchHCF/DumpsterFire.

[35] Bishop Fox. Sliver. https://github.com/BishopFox/sliver.

[36] Muhammad Mudassar Yamin, Basel Katt, and Vasileios Gkioulos. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. *Computers & Security*, 88:101636, 2020.

[37] Razvan Beuran, Dat Tang, Cuong Pham, Ken-ichi Chinen, Yasuo Tan, and Yoichi Shinoda. Integrated framework for hands-on cybersecurity training: Cytrone. *Computers & Security*, 78:43–59, 2018.

[38] Pavel Čeleda, Jakub Čegan, Jan Vykopal, Daniel Tovarňák, et al. Kypo–a platform for cyber defence exercises. *M&S Support to Operational Tasks Including War Gaming, Logistics, Cyber Defence. NATO Science and Technology Organization*, 2015.

[39] John Wroclawski, Terry Benzel, Jim Blythe, Ted Faber, Alefiya Hussain, Jelena Mirkovic, and Stephen Schwab. Deterlab and the deter project. *The GENI Book*, pages 35–62, 2016.

[40] Bernard Ferguson, Anne Tall, and Denise Olsen. National cyber range overview. In *2014 IEEE Military Communications Conference*, pages 123–128. IEEE, 2014.

[41] CTID. OilRig Intelligence Summary. https://github.com/center-for-threat-informed-defense/adversary_emulation_library/blob/master/fin6/Intelligence_Summary.md.

[42] MITRE Engenuity. Center for threat-informed defense. https://ctid.mitre-engenuity.org/.

[43] Cyware. APT34: The Helix Kitten Cybercriminal Group Loves to Meow Middle Eastern and International Organizations. https://cyware.com/blog/apt34-the-helix-kitten-cybercriminal-group-loves-to-meow-middle-eastern-and-international-organizations-48ae.

[44] Mandiant. New Targeted Attack in the Middle East by APT34, a Suspected Iranian Threat Group, Using CVE-2017-11882 Exploit. https://www.mandiant.com/resources/blog/targeted-attack-in-middle-east-by-apt34.

[45] Malwarebytes Labs. APT34 targets Jordan Government using new Saitama backdoor. https://www.malwarebytes.com/blog/threat-intelligence/2022/05/apt34-targets-jordan-government-using-new-saitama-backdoor.

[46] MITRE. Phishing, T1566. https://attack.mitre.org/techniques/T1566/.

[47] MITRE. User Execution: Malicious File, T1204.002. https://attack.mitre.org/techniques/T1204/002/.

[48] MITRE. Account Discovery, T1087. https://attack.mitre.org/techniques/T1087/.

[49] MITRE. OS Credential Dumping, T1003. https://attack.mitre.org/techniques/T1003/.

[50] MITRE. Use Alternate Authentication Material: Pass the Hash, T1550.002. https://attack.mitre.org/techniques/T1550/002/.

[51] MITRE. Exfiltration Over Alternative Protocol, T1048. https://attack.mitre.org/techniques/T1048/.

[52] MITRE. CALDERA documentation. https://caldera.readthedocs.io/en/latest/.

[53] CTID. STIX project. https://stixproject.github.io/.

[54] MITRE Corporation. CAPEC. https://capec.mitre.org/.

[55] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc.", 2009.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[57] SecBERT: A Pretrained Language Model for Cyber Security Text. https://huggingface.co/jackaduma/SecBERT.

[58] MITRE. Command and Scripting Interpreter, T1059. https://attack.mitre.org/techniques/T1059/.

[59] MITRE. Cloud Service Dashboard, T1538. https://attack.mitre.org/techniques/T1538/.

[60] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. *ArXiv*, abs/2008.05756, 2020.

[61] MITRE. TRAM. https://github.com/center-for-threat-informed-defense/tram/.

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[63] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[64] François Chollet et al. Keras, 2015. https://keras.io.

[65] Ankur Padia, Arpita Roy, Taneeya Satyapanich, Francis Ferraro, Shimei Pan, Youngja Park, Anupam Joshi, and Tim Finin. UMBC at SemEval-2018 task 8: Understanding text about malware. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 878–884, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[66] SecBert Hugging Face Model. https://huggingface.co/jackaduma/SecBERT.

[67] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2019.

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[69] OpenAI. GPT-3.5. https://platform.openai.com/docs/models/gpt-3-5.

[70] OpenAI. ChatGPT. https://openai.com/chatgpt.

[71] Mandiant. Pick-Six: Intercepting a FIN6 Intrusion, an Actor Recently Tied to Ryuk and LockerGoga Ransomware. https://www.mandiant.com/resources/pick-six-intercepting-a-fin6-intrusion.

[72] The United States Department of Justice. United States v. Zhu Hua Indictment. https://www.justice.gov/opa/press-release/file/1121706/download.

[73] The DFIR Report. Ryuk's Return. https://thedfirreport.com/2020/10/08/ryuks-return/.

[74] MITRE. Introducing the all-new Adversary Emulation Plan Library. https://medium.com/mitre-engenuity/introducing-the-all-new-adversary-emulation-plan-library-234b1d543f6b.

[75] CTID. Adversary emulation library. https://github.com/center-for-threat-informed-defense/adversary_emulation_library.

[76] Divakar Yadav, Jalpa Desai, and Arun Kumar Yadav. Automatic text summarization methods: A comprehensive review. *arXiv preprint arXiv:2204.01849*, 2022.

[77] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[78] HuggingFace. T5 HuggingFace. https://huggingface.co/docs/transformers/model_doc/t5.

[79] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *Communications and Multimedia Security*, 2014.

[80] HuggingFace. BERT Base Uncased. https://huggingface.co/bert-base-uncased.

[81] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[82] SetFit. Bbc news topic classification. https://huggingface.co/datasets/SetFit/bbc-news.

[83] threatpost. Alleged Mastermind Behind Carbanak Crime Gang Arrested. https://threatpost.com/alleged-mastermind-behind-carbanak-crime-gang-arrested/130831/.

[84] CrowdStrike. Arrests Put New Focus on CARBON SPIDER Adversary Group. https://www.crowdstrike.com/blog/arrests-put-new-focus-on-carbon-spider-adversary-group/.

[85] Trustwave. Operation Grand Mars: a comprehensive profile of Carbanak activity in 2016/17. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/operation-grand-mars-a-comprehensive-profile-of-carbanak-activity-in-201617/.

[86] Securelist. The Great Bank Robbery: the Carbanak APT. https://securelist.com/the-great-bank-robbery-the-carbanak-apt/68732/.

[87] Mandiant. Behind the carbanak backdoor. https://www.mandiant.com/resources/blog/behind-the-carbanak-backdoor.

[88] Trustwave. New carbanak anunak attack methodology. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/new-carbanak-anunak-attack-methodology/.

[89] Mandiant. Fin7 phishing lnk. https://www.mandiant.com/resources/blog/fin7-phishing-lnk.

[90] Forcepoint. Carbanak group uses google malware command and control. https://www.forcepoint.com/blog/x-labs/carbanak-group-uses-google-malware-command-and-control.

[91] RSA. Rsa blog. https://www.rsa.com/blog/.

[92] Securelist. Financial predictions 2020. https://securelist.com/financial-predictions-2020/95388/.

[93] Cyware. FIN6 Group Goes from Compromising POS Systems to Deploying Ransomware. https://cyware.com/news/fin6-group-goes-from-compromising-pos-systems-to-deploying-ransomware-3e9d0691.

[94] Morphisec. New Global Attack on Point of Sale Systems. https://blog.morphisec.com/new-global-attack-on-point-of-sale-systems.

[95] Security Intelligence. ITG08 AKA FIN6 Partners With Trickbot Gang, Uses Anchor Framework. https://securityintelligence.com/posts/itg08-aka-fin6-partners-with-trickbot-gang-uses-anchor-framework/.

[96] Security Intelligence. More Eggs Anyone? Threat Actor ITG08 Strikes Again. https://securityintelligence.com/posts/more_eggs-anyone-threat-actor-itg08-strikes-again/.

[97] Trend Micro. FIN6 Compromised E-commerce Platform via Magecart to Inject Credit Card Skimmers Into Thousands of Online Shops. https://www.trendmicro.com/en_us/research/19/j/fin6-compromised-e-commerce-platform-via-magecart-to-inject-credit-card-skimmers-into-thousands-of-online-shops.html.

[98] Proofpoint. Fake Jobs Campaigns Delivering MoreEggs Backdoor Fake Job Offers. https://www.proofpoint.com/us/threat-insight/post/fake-jobs-campaigns-delivering-moreeggs-backdoor-fake-job-offers.

[99] Mandiant. Tactics, Techniques, Procedures Associated with Maze Ransomware Incidents. https://www.mandiant.com/resources/blog/tactics-techniques-procedures-associated-with-maze-ransomware-incidents.

[100] CrowdStrike. Big Game Hunting with Ryuk: Another Lucrative Targeted Ransomware. https://www.crowdstrike.com/blog/big-game-hunting-with-ryuk-another-lucrative-targeted-ransomware/.

[101] Intel471. Ransomware as a Service 2020: Ryuk, Maze, REvil, Egregor, Doppelpaymer. https://intel471.com/blog/ransomware-as-a-service-2020-ryuk-maze-revil-egregor-doppelpaymer.

[102] Trend Micro. Trickbot Adds Remote Application Credential Grabbing Capabilities to Its Repertoire. https://www.trendmicro.com/en_us/research/19/b/trickbot-adds-remote-application-credential-grabbing-capabilities-to-its-repertoire.html.

[103] Mandiant. Mahalo FIN7: Responding to New Tools and Techniques. https://www.mandiant.com/resources/blog/mahalo-fin7-responding-to-new-tools-and-techniques.

[104] Securelist. FIN7 5: The Infamous Cybercrime Rig FIN7 Continues Its Activities. https://securelist.com/fin7-5-the-infamous-cybercrime-rig-fin7-continues-its-activities/90703/.

[105] Flashpoint. FIN7 Revisited: Inside Astra Panel and SQLRat Malware. https://flashpoint.io/blog/fin7-revisited-inside-astra-panel-and-sqlrat-malware/.

[106] Deepwatch. Profile of an Adversary: FIN7. https://www.deepwatch.com/labs/profile-of-an-adversary-fin7/.

[107] Mandiant. FIN7: Pursuing an Enigmatic and Evasive Global Criminal Operation. https://www.mandiant.com/resources/blog/fin7-pursuing-an-enigmatic-and-evasive-global-criminal-operation.

[108] Gigamon. Footprints of FIN7: Tracking Actor Patterns Part 1. https://blog.gigamon.com/2017/07/26/footprints-of-fin7-tracking-actor-patterns-part-1/.

[109] Gigamon. Footprints of FIN7: Tracking Actor Patterns Part 2. https://blog.gigamon.com/2017/07/26/footprints-of-fin7-tracking-actor-patterns-part-2/.

[110] Mandiant. APT10 MenuPass Group. https://www.mandiant.com/resources/blog/apt10-menupass-group.

[111] Mandiant. APT10 Targeting Japanese Corporations Using Updated TTPs. https://www.mandiant.com/resources/blog/apt10-targeting-japanese-corporations-using-updated-ttps.

[112] Trend Micro. ChessMaster Cyber Espionage Campaign. https://www.trendmicro.com/en_us/research/17/g/chessmaster-cyber-espionage-campaign.html.

[113] CISA. Intrusions Affecting Multiple Victims Across Multiple Sectors. https://www.cisa.gov/news-events/alerts/2017/04/27/intrusions-affecting-multiple-victims-across-multiple-sectors.

[114] BlackBerry. Threat Spotlight: MenuPass QuasarRat Backdoor. https://blogs.blackberry.com/en/2019/06/threat-spotlight-menupass-quasarrat-backdoor.

[115] CrowdStrike. Two Birds, One Stone: Panda. https://www.crowdstrike.com/blog/two-birds-one-stone-panda/.

[116] Trend Micro. ChessMaster Adds Updated Tools to Its Arsenal. https://www.trendmicro.com/en_us/research/18/c/chessmaster-adds-updated-tools-to-its-arsenal.html.

[117] IntrusionTruth. APT10 Was Managed by the Tianjin Bureau of the Chinese Ministry of State Security. https://intrusiontruth.wordpress.com/2018/08/15/apt10-was-managed-by-the-tianjin-bureau-of-the-chinese-ministry-of-state-security/.

[118] JPCERT. RedLeaves - Malware Based on Open Source RAT. https://blogs.jpcert.or.jp/en/2017/04/redleaves---malware-based-on-open-source-rat.html.

[119] VMware. Carbon Black Threat Research Dissects Red Leaves Malware, Leverages DLL Side-Loading. https://blogs.vmware.com/security/2017/05/carbon-black-threat-research-dissects-red-leaves-malware-leverages-dll-side-loading.html.

[120] Fortinet. Uncovering New Activity by APT. https://www.fortinet.com/blog/threat-research/uncovering-new-activity-by-apt-.

[121] Cybereason. Operation Soft Cell: A Worldwide Campaign Against Telecommunications Providers. https://www.cybereason.com/blog/research/operation-soft-cell-a-worldwide-campaign-against-telecommunications-providers.

[122] JPCERT. TA410: Group Behind Lookback Attacks Against US Utilities Sector Returns With New Malware. https://www.proofpoint.com/us/blog/threat-insight/ta410-group-behind-lookback-attacks-against-us-utilities-sector-returns-new.

[123] JPCERT. Quasar Family. https://blogs.jpcert.or.jp/en/2020/12/quasar-family.html.

[124] JPCERT. Malware Leveraging PowerSploit. https://blogs.jpcert.or.jp/en/2017/03/malware-leveraging-powersploit.html.

[125] JPCERT. Chches Malware. https://blogs.jpcert.or.jp/en/2017/02/chches-malware--93d6.html.

[126] Trend Micro. Rising Trend: Attackers Using LNK Files to Download Malware. https://www.trendmicro.com/en_us/research/17/e/rising-trend-attackers-using-lnk-files-download-malware.html.

[127] CISA. IR Alert H-16-056-01. https://www.cisa.gov/news-events/ics-alerts/ir-alert-h-16-056-01.

[128] Talos Intelligence Blog. Olympic Destroyer. https://blog.talosintelligence.com/olympic-destroyer/.

[129] VMware. Carbon Black Threat Research: Technical Analysis of Petya (NotPetya) Ransomware. https://blogs.vmware.com/security/2017/06/carbon-black-threat-research-technical-analysis-petya-notpetya-ransomware.html.

[130] United States Department of Justice. Six Russian GRU Officers Charged in Connection with Worldwide Deployment of Destructive Malware and Other Disruptive Actions in Cyberspace. https://www.justice.gov/opa/pr/six-russian-gru-officers-charged-connection-worldwide-deployment-destructive-malware-and.

[131] ReliaQuest. Mapping MITRE ATT&CK to Sandworm APT's Global Campaign. https://www.reliaquest.com/blog/mapping-mitre-attck-to-sandworm-apts-global-campaign/#:~:text=SandWorm%20is%20an%20APT%20group,aggressive%20and%20sometimes%20destructive%20cyberattacks.

[132] Securelist. ExPetr/Petya/NotPetya is a Wiper, Not Ransomware. https://securelist.com/expetrpetyanotpetya-is-a-wiper-not-ransomware/78902/.

[133] Mandiant. Ukraine and the Sandworm Team. https://www.mandiant.com/resources/blog/ukraine-and-sandworm-team.

[134] UPI. Ransom hackers hit Georgia courts after cities pay $1M. https://www.upi.com/Top_News/US/2019/07/08/Ransom-hackers-hit-Georgia-courts-after-cities-pay-1M/4111562116580/.

[135] Cybersecurity Dive. Ryuk: FBI, DHS warn of ransomware targeting healthcare. https://www.cybersecuritydive.com/news/Ryuk-FBI-DHS-ransomware-healthcare/587961/.

[136] CTID. menuPass Intelligence Summary. https://github.com/center-for-threat-informed-defense/adversary_emulation_library/blob/master/menu_pass/Intelligence_Summary.md.

[137] Marcus Botacin, Felipe Duarte Domingues, Fabrício Ceschin, Raphael Machnicki, Marco Antonio Zanata Alves, Paulo Lício de Geus, and André Grégio. Antiviruses under the microscope: A hands-on perspective. *Computers & Security*, 112:102500, 2022.

[138] Markus Fleschutz. Collection of Powershell scripts. https://github.com/fleschutz/PowerShell/blob/master/Scripts/encrypt-file.ps1.

[139] klezVirus. Inceptor. https://github.com/klezVirus/inceptor.

[140] Danny Hendler, Shay Kels, and Amir Rubin. Amsi-based detection of malicious powershell code using contextual embeddings. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 679–693, 2020.

[141] klezVirus. Inceptor – Bypass AV-EDR solutions combining well-known techniques. https://github.com/klezVirus/inceptor/blob/main/slides/Inceptor%20-%20Bypass%20AV-EDR%20solutions%20combining%20well%20known%20techniques.pdf.

[142] Microsoft. Microsoft Defender SmartScreen. https://learn.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-smartscreen/microsoft-defender-smartscreen-overview.

[143] Microsoft. Microsoft Defender Application Guard overview. https://learn.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-application-guard/md-app-guard-overview.

[144] Ege Balci. Shikata-Ga-Nai. https://github.com/EgeBalci/sgn.

[145] Mandiant. Shikata Ga Nai Encoder Still Going Strong. https://www.mandiant.com/resources/blog/shikata-ga-nai-encoder-still-going-strong.

[146] Paranoid Ninja. CarbonCopy. https://github.com/paranoidninja/CarbonCopy.

[147] Team Ascend. The complete guide to EDR. https://blog.teamascend.com/the-complete-guide-to-edr.

[148] Hoang Bui. Bypass EDR's memory protection, introduction to hooking. https://medium.com/@fsx30/bypass-edrs-memory-protection-introduction-to-hooking-2efb21acffd6.

[149] Binarly. Design issues of modern EDRs: bypassing ETW-based solutions. https://www.binarly.io/posts/Design_issues_of_modern_EDRs_bypassing_ETW-based_solutions/index.html.

[150] Cornelis De Plaa. Bypass EDR's memory protection, introduction to hooking. https://outflank.nl/blog/2019/06/19/red-team-tactics-combining-direct-system-calls-and-srdi-to-bypass-av-edr/.

[151] Evan Pena and Casey Erikson. Staying Hidden on the Endpoint: Evading Detection with Shellcode. https://www.mandiant.com/resources/staying-hidden-on-the-endpoint-evading-detection-with-shellcode.

[152] InfoSec. Evade EDR with Shellcode Injection and gain persistence using Registry Run Keys. https://infosecwriteups.com/evade-avs-edr-with-shellcode-injection-159dde4dba1a.

[153] Kevin Schoonover, Eric Michalak, Sean Harris, Adam Gausmann, Hannah Reinbolt, Daniel R Tauritz, Chris Rawlings, and Aaron Scott Pope. Galaxy: a network emulation framework for cybersecurity. In *11th USENIX Workshop on Cyber Security Experimentation and Test (CSET 18)*, 2018.

[154] Daniel Kouril, Tomáš Rebok, Tomas Jirsik, Jakub Cegan, Martin Drasar, Martin Vizváry, and Jan Vykopal. Cloud-based testbed for simulation of cyber attacks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–6. IEEE, 2014.

[155] R Dhaya, R Kanthavel, and Kanagaraj Venusamy. Dynamic secure and automated infrastructure for private cloud data center. *Annals of Operations Research*, pages 1–21, 2021.

[156] Nathan Regola, Nitesh V Chawla, et al. Storing and using health data in a virtual private cloud. *Journal of medical Internet research*, 15(3):e2076, 2013.

[157] Tal Garfinkel, Mendel Rosenblum, et al. A virtual machine introspection based architecture for intrusion detection. In *Ndss*, volume 3, pages 191–206. San Diego, CA, 2003.

[158] Microsoft. API reference docs for Windows Driver Kit (WDK). https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/.

[159] Intel. XCHG - Exchange Register/Memory with Register. https://www.cs.princeton.edu/courses/archive/spr18/cos217/reading/x86-64-2.pdf.

[160] Zerosum0x0. "Heresy's Gate": Kernel Zw*/NTDLL Scraping + "Work Out": Ring 0 to Ring 3 via Worker Factories. https://zerosum0x0.blogspot.com/2020/06/heresys-gate-kernel-zwntdll-scraping.html#workout.

[161] Zerosum0x0. Zerosum0x0 GitHub repository. https://github.com/zerosum0x0-archive/archive.

[162] Qemu. Qemu. https://www.qemu.org/.

[163] LibVirt. LibVirt. https://libvirt.org/.

[164] RedHat. Linux-KVM. https://www.linux-kvm.org/page/Main_Page.

[165] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. Mitre att&ck: Design and philosophy. In *Technical report*. The MITRE Corporation, 2018.

[166] Volatility Foundation. Volatility. https://www.volatilityfoundation.org/.

[167] Pavel Yosifovich, David A Solomon, and Alex Ionescu. *Windows Internals, Part 1: System architecture, processes, threads, memory management, and more*. Microsoft Press, 2017.

[168] Brian Caffo. Statistical inference for data science. *British Columbia, UK: Leanpub*, 2016.

[169] MITRE. Tactics. https://attack.mitre.org/tactics/enterprise/.

[170] MITRE. Reconnaissance, Tactic TA0043. https://attack.mitre.org/tactics/TA0043/.

[171] MITRE. Resource Development, Tactic TA0042. https://attack.mitre.org/tactics/TA0042/.

[172] MITRE. Initial Access, Tactic TA0001. https://attack.mitre.org/tactics/TA0001/.

[173] MITRE. Execution, Tactic TA0002. https://attack.mitre.org/tactics/TA0002/.

[174] MITRE. Persistence, Tactic TA0003. https://attack.mitre.org/tactics/TA0003/.

[175] MITRE. Privilege Escalation, Tactic TA0004. https://attack.mitre.org/tactics/TA0004/.

[176] MITRE. Defense Evasion, Tactic TA0005. https://attack.mitre.org/tactics/TA0005/.

[177] MITRE. Credential Access, Tactic TA0006. https://attack.mitre.org/tactics/TA0006/.

[178] MITRE. Discovery, Tactic TA0007. https://attack.mitre.org/tactics/TA0007/.

[179] MITRE. Lateral Movement, Tactic TA0008. https://attack.mitre.org/tactics/TA0008/.

[180] MITRE. Collection, Tactic TA0009. https://attack.mitre.org/tactics/TA0009/.

[181] MITRE. Command and Control, Tactic TA0011. https://attack.mitre.org/tactics/TA0011/.

[182] MITRE. Exfiltration, Tactic TA0010. https://attack.mitre.org/tactics/TA0010/.

[183] MITRE. Impact, Tactic TA0040. https://attack.mitre.org/tactics/TA0040/.

[184] MITRE. Enterprise Techniques. https://attack.mitre.org/techniques/enterprise/.

Author's publications

The following previously published material has been, in parts verbatim, included in this thesis.

1. V. Orbinato. A next-generation platform for Cyber Range-as-a-Service. *2021 IEEE 32nd International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 314-318, 2021. DOI: 10.1109/ISSREW53611.2021.00094.
2. P. Liguori, E. Al-Hossami, V. Orbinato, R. Natella, S. Shaikh, D. Cotroneo, B. Cukic. EVIL: Exploiting Software via Natural Language, *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 321-332, 2021. DOI: 10.1109/ISSRE52982.2021.00042.
3. V. Orbinato, M. Barbaraci, R. Natella, D. Cotroneo. Automatic Mapping of Unstructured Cyber Threat Intelligence: An Experimental Study, *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 181-192, 2022. DOI: 10.1109/ISSRE55969.2022.00027.
4. V. Orbinato, M. C. Feliciano, D. Cotroneo, R. Natella, Laccolith: Hypervisor-Based Adversary Emulation with Anti-Detection, *IEEE Transactions on Dependable and Secure Computing (TDSC)* (**under review after revision**)

The following publications are related to different aspects than those covered in this thesis and have not been included:

1. V. Casola, A. De Benedictis, C. Mazzocca, V. Orbinato, Secure Software Development and Testing: a Model-based Methodology, *Computers & Security, Elsevier*, 2023, DOI: 10.1016/J.COSE.2023.103639.

2. V. Orbinato, F. C. Grasso, R. Natella, D. Cotroneo, Vulnerability Prediction on Binary Code via Neural Decompilation, *54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (**under review**)