Detecting Abuse of Cloud and Public Legitimate Services as Command and Control Infrastructure Using Machine Learning

A thesis submitted in partial fulfilment

of the requirement for the degree of Doctor of Philosophy

Turki Zuher Al lelah

May 2024

Cardiff University
School of Computer Science & Informatics

Abstract

The widespread adoption of Cloud and Public Legitimate Services (CPLS) has inadvertently created new opportunities for cybercriminals to establish hidden and robust command-and-control (C&C) communication infrastructure. This abuse represents a major cybersecurity risk, as it allows malicious traffic to seamlessly disguise itself within normal network activities. Traditional detection systems are proving inadequate in accurately identifying such abuses. Therefore, this thesis is motivated by emphasizing the urgent need for more advanced detection techniques that are capable of identifying the C&C activity hidden within legitimate CPLS traffic.

To assess the extent of the cyber threat of abusing CPLS, this thesis presents an extensive Systematic Literature Review (SLR) encompassing academic and industry literature. The review provides a comprehensive categorization of the attack techniques utilized to abuse CPLS as C&C infrastructure. The open problems uncovered through the SLR motivate this thesis to propose a novel Detection System (DS) capable of identifying malware that abuse CPLS as C&C communication channels. Furthermore, to evaluate our system robustness against attempts to evade detection, this thesis introduces the Replace Misclassified Parameter (RMCP) adversarial attack. The proposed detection system leverages Artificial Intelligence (AI) techniques, combining static and dynamic malware analysis methods to accurately identify CPLS abuse. The effectiveness of the proposed system is validated through extensive experiments, demonstrating its ability to detect novel and sophisticated attacks that evade traditional security measures. The outcomes of this thesis have significant implications for enhancing the security of cloud environments, contributing valuable knowledge and practical solutions to the field of cloud security.

Contents

List of Figures

List of Tables

List of Acronyms

CPLS Cloud and Public Legitimate Services

C&C Command and Control

DS Detection System

RMCP Replace Misclassified Parameter

AI Artificial Intelligence

DDoS Distributed Denial of Service

IRC Internet Relay Chat

HTTP Hypertext Transfer Protocol

P2P Peer-to-Peer

DNS Domain Name System

TLS Transport Layer Security

SFSF Sequential Feature Selector Forward

SFSB Sequential Feature Selector Backward

RFE Recursive Feature Elimination

CFE Custom Feature Elimination

ML	Machine Learning
PE	Portable Executable
OS	Operating System
DLL	Dynamic-Link Library
COFF	Common Object File Format
API	Application Programming Interface
URL	Uniform Resource Locator
CSV	Comma-Separated Values
IAT	Import Address Table
J48	J48 Decision Tree
RF	Random Forest
NB	Naïve Bayes
KNN	K-Nearest Neighbors
SVM	Support Vector Machine

Acknowledgements

First and foremost, I am sincerely grateful to the Almighty God, "ALLAH" for His mercy, never-ending sustenance, and generosity upon me. No amount of gratitude or act of worship can truly reciprocate these blessings. I am thankful for endowing me with the fortitude and providing me with the strength to pursue this PhD research.

My deepest gratitude and love are extended to my beloved parents. Your unwavering love, encouragement, and sacrifices have been the cornerstone of my success. This achievement is as much yours as it is mine. I hope this accomplishment makes you proud.

To my dear wife Amani, you have been a constant source of inspiration and support throughout my academic and daily life. I am grateful for your presence in my life and express my deep love, gratitude, and respect for you. To the roots of my happiness during this journey, my children Beisan, Aleen, and Badr, I would like to express my deepest gratitude for your patience. Your sacrifice of family time for my work is immensely appreciated.

The successful completion of this research would not have been possible without the support of my academic supervisor, Dr George Theodorakopoulos. I am deeply indebted to you for your remarkable assistance, mentorship, and expertise at every stage of the research project. I am grateful for your patience, encouragement, and dedication to my research. I am truly honored to have had the opportunity to collaborate with you.

A special thanks also goes to Dr Philipp Reinecke for his supervision during his time at Cardiff University as my second supervisor.

Furthermore, I am grateful to Dr Amir Javed and Dr Eirini Anthi for their contributions and support. Your guidance and feedback have greatly enriched my research journey.

I would also like to express my gratitude to the School of Computer Science and Informatics at Cardiff University for providing the resources, specifically the Cyber Range environment, which was essential for conducting my research.

I would like to extend a special thank you to the Royal Commission for Jubail and Yanbu (RCJY) and Jubail Industrial College (JIC) for their generous sponsorship of my PhD program. Their support has been invaluable throughout my academic journey.

List of Publications

The work introduced in this thesis is based on the following publications.

Previous Publications

- Al lelah, T.; Theodorakopoulos, G.; Reinecke, P.; Javed, A.; Anthi, E. Abuse of Cloud-Based and Public Legitimate Services as Command-and-Control (C&C) Infrastructure: A Systematic Literature Review. J. Cybersecur. Priv. 2023, 3, 558-590. https://doi.org/10.3390/jcp3030027 (Based on research described in Chapter 2 of this thesis.)
- Al lelah T, Theodorakopoulos G, Javed A, Anthi E. Machine Learning Detection of Cloud Services Abuse as C&C Infrastructure. Journal of Cybersecurity and Privacy. 2023; 3(4):858-881. https://doi.org/10.3390/jcp3040039 (Based on research described in Chapters 3, 4, and 6 of this thesis.)

Submit for Publications

- Al lelah T, Theodorakopoulos G, Javed A, Anthi E. Detecting the Abuse of Cloud Services for C&C Infrastructure Through Dynamic Analysis and Machine Learning," is currently under review for publication at the International Symposium on Networks, Computers and Communications 2024 (ISNCC 2024) in

Washington, D.C., USA. (Based on research described in Chapters 5 and 6 of this thesis.)

Chapter 1

Introduction

In the modern digital landscape, individuals and organisations face a constant barrage of threats from various types of **mal**icious soft**ware** (malware), including bots, ransomware, trojans, and worms. These malicious entities have become the predominant tools employed by cybercriminals to carry out their nefarious activities. Among these threats, bots are particularly concerning as they can form networks, known as botnets, which comprise compromised computers remotely controlled by a botmaster or multiple controllers via a command-and-control (C&C) infrastructure. Botnets have emerged as one of the most dominant threat vectors, posing severe risks to global Internet security. The increasing use of Cloud and Public Legitimate Services (CPLS) has unintentionally opened new avenues for cybercriminals to create concealed and resilient C&C communication channels. This misuse of CPLS poses a significant threat to cybersecurity, as it enables malicious traffic to effortlessly blend in with regular network activities, making it difficult to detect. Conventional detection systems, such as signature-based antivirus software and traditional intrusion detection systems, are struggling to effectively identify CPLS abuse. These systems rely on known malware signatures or predefined rules, which are less effective against malware leveraging trusted cloud services for C&C communication. Malicious actors blend their traffic with legitimate cloud service usage, making it challenging for these systems to distinguish between benign and malicious activities. This emphasises the need for more advanced detection methods to identify C&C activity masked within legitimate CPLS traffic.

This thesis aims to address this critical issue by first conducting a thorough systematic review of the literature (SLR) to assess the extent of the cyber threat posed by the abuse of CPLS as C&C infrastructure. The SLR encompasses both academic and industry literature, providing a comprehensive categorization of the attack techniques utilized by malicious actors. The open problems uncovered through the SLR motivate the development of a novel Detection System (DS) capable of identifying malware that abuses CPLS as C&C communication channels.

Therefore, this thesis introduces two distinct DSs that leverage machine learning (ML) techniques. The first DS employs a static analysis technique and achieves a remarkable detection accuracy rate of 98.26%. The second DS, on the other hand, utilizes the dynamic analysis technique and demonstrates a detection accuracy rate of 97.95%.

To evaluate the robustness of the proposed detection system, this thesis introduces the Replace Misclassified Parameter (RMCP) adversarial attack. The effectiveness of the proposed system is validated through extensive experiments, with 3,067 malicious and 3,067 benign samples, demonstrating its ability to detect novel and sophisticated attacks that evade traditional security measures.

The outcomes of this thesis have significant implications for enhancing the security of cloud environments, contributing valuable knowledge and practical solutions to the field of cloud security. By providing a comprehensive understanding of the CPLS abuse threat landscape, developing a taxonomy of attack techniques, and proposing an innovative AI-driven detection system that is robust to adversarial manipulation, this research aims to equip organisations with the tools and insights necessary to safeguard their critical assets and data hosted in cloud environments.

1.1 Background

1.1.1 Botnet Components

To facilitate a comprehensive understanding of botnet operations within the context of this thesis, it is essential to introduce the fundamental elements that constitute a botnet: bots, the botmaster, and the command and control (C&C) channel. Figure 1.1 provides a visual representation of these components and their interactions.

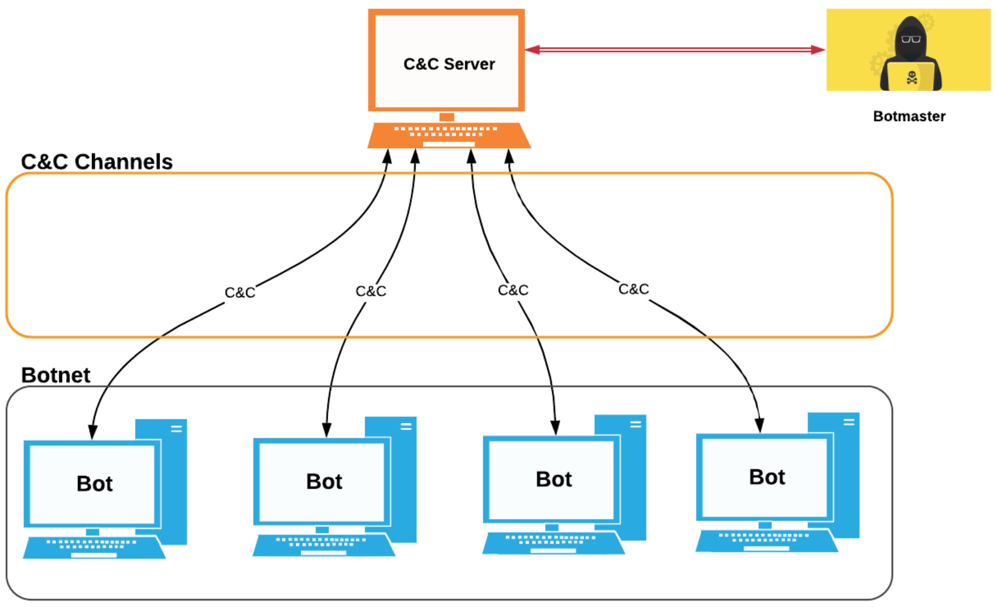


Figure 1.1: Botnet elements.

Bot

A bot is a software program that is often malicious in nature. It is installed on a compromised host and can perform a wide range of activities, including low-level operations such as creating or modifying system-level processes, as well as disabling security tools like Anti-Virus (AV) software, and modifying registry keys. For instance, the Advanced Persistent Threat (APT) group Kimsuky [23, 19] is known for employing bots to create or modify system-level processes as a means of maintaining persistence on compromised systems. This is a method of maintaining persistence on compromised systems. These bots repeatedly execute malicious payloads, ensuring that the malware remains active and continues its malicious activities. Another common activity of bots is disabling security tools, such as Anti-Virus (AV) software. Sophisticated bots like Agent Tesla [15] and Brave Prince [14] possess the ability to terminate any active analysis processes, including AV software, thus ensuring their undetected continuity in malicious activities. Modifying registry keys for persistence is another activity performed by bots. For example, Amadey [24] changes the Startup folder by overwriting the registry keys,

The installation of bots on victims' machines can occur through various malware spreading mechanisms, such as accessing infected websites or installing trojans.

Botnet

A botnet is a collection of bots connected to a C&C channel, forming a cohesive unit that awaits commands from the botmaster to carry out malicious activities. The coordinated nature of botnets allows the execution of large-scale attacks and the amplification of the impact of malicious campaigns.

Botmaster

Botmasters are malicious actors who exercise control over botnets by issuing commands to bots, enabling them to engage in various malicious operations. These operations can include theft of financial assets, exfiltrating confidential data, degrading systems, launching distributed denial of service (DDoS) attacks, or sending spam [154, 130]. The botmaster serves as the central authority, orchestrating the actions of the botnet to achieve their malicious objectives.

C&C Communication Channels

Botnets often leverage pre-existing or established communication protocols for their internal communication, rather than developing new network protocols. The most common protocols utilised by botnets include:

- Internet Relay Chat (IRC)-based C&C channels: These channels utilise a push-based model, where the botmaster issues new commands to the botnet and the bots respond promptly to these commands.
- HTTP-based C&C channels: These channels adopt a pull-based model, where bots are configured to periodically check in with the C&C server and retrieve any new commands.
- Peer-to-peer (P2P)-based C&C channels: These channels make use of peer-to-peer communication to either relay commands or locate a C&C server [103].
- Domain Name System (DNS)-based C&C channels: These channels employ DNS tunnelling, a technique that allows the encapsulation of non-DNS traffic within DNS packets [98, 18].

C&C communication channels play a crucial role in facilitating communication between the botmaster and bot clients, as well as among bot clients themselves. The evolution of botnet C&C communication has witnessed several generations, each adopting different protocols and architectures to enhance their resilience and evade detection.

The first generation of botnets relied heavily on Internet Relay Chat (IRC) for C&C communication. In this setup, bots would connect to IRC channels created by the botmaster on the C&C server to await commands to perform malicious activities. However, this approach had a significant drawback, it relied on a single point of failure. If the IRC servers were taken down or identified, the entire army of bot clients would become ineffective.

To address this vulnerability, the second generation of botnets transitioned to peer-to-peer (P2P) protocols [103, 110]. This decentralized approach eliminated the single-point-of-failure weakness. However, managing and controlling P2P-based botnets proved to be a challenge, leading to the adoption of HTTP for implementing C&C communication in subsequent generations [3, 131, 110].

In recent years, a new type of botnet has emerged, known as hybrid botnets, which combine both P2P and HTTP protocols [130]. This hybrid approach leverages the strengths of both protocols to enhance the resilience and flexibility of the botnet's C&C infrastructure. Additionally, DNS-based C&C channels have been observed in recent attacks, where APT groups such as Wekby [18] and APT41 [2, 68] use DNS TXT records to establish a two-way command and control communication channel. This technique, known as DNS tunnelling, allows the encapsulation of non-DNS traffic within DNS packets. It enables malware to receive commands and exfiltrate data using DNS queries and responses, making it more difficult to detect and block malicious traffic.

C&C Server

The C&C server is an essential component that acts as the communication hub, facilitating the interaction between the botmaster and the bots. Often referred to as the coordinator server, the C&C server is utilized by the botmaster to issue commands, maintain and update bot programs. Bots establish connections to the C&C servers to receive commands or download bot binaries [130], enabling the botmaster to maintain control over the botnet and orchestrate malicious activities.

1.1.2 Abuse of Cloud and Public Legitimate Services (CPLS)

In the context of this thesis, the term "abuse" refers to the malicious exploitation of CPLS by cybercriminals to facilitate C&C operations for botnets. This abuse involves using these services in ways that were not intended by their providers, often violating terms of service and exploiting the trust placed in these platforms. Specifically, abuse of CPLS as C&C infrastructure can include:

1. Using CPLS to receive and host stolen data.
2. Exploiting CPLS to disseminate commands to infected machines.
3. Leveraging CPLS APIs for covert communication between botnet components.
4. Misusing CPLS for sending commands or exfiltrating data.

1.2 Motivation and Problem Definition

The rising global adoption of Cloud and Public Legitimate Services (CPLS) has unintentionally opened new avenues for cybercriminals to create concealed and resilient C&C communication channels. This misuse of CPLS, including popular services like Google Drive, Dropbox, and OneDrive, poses a significant threat to cybersecurity. Malicious actors exploit these services using sophisticated techniques, allowing their traffic to blend seamlessly with legitimate network activities. Our comprehensive understanding of this landscape, derived from systematic literature review and analysis of real-world malware samples, reveals the increasing complexity of botnet attacks leveraging CPLS as C&C infrastructure. Conventional detection systems struggle to effectively identify this abuse, emphasising the pressing need for more advanced detection methods that can recognise C&C activity masked within legitimate CPLS traffic. This evolving threat landscape and the challenges in detecting CPLS abuse serve as the primary motivations for this study.

Abusing CPLS as C&C infrastructure

Understanding the life cycle of a botnet that abuses Cloud Public Legitimate Services (CPLS) as its C&C infrastructure is crucial for the successful analysis and development of effective botnet detection systems. By comprehending each phase of this cycle, it becomes possible to enhance and create more efficient detection mechanisms. Figure 1.2 illustrates how threat actors abuse CPLS as centralized C&C servers to control bots. After compromising a victim's machine, the bot requires a remote control mechanism to communicate with the C&C server and receive further malicious instructions. To establish this remote C&C channel, bot connection requests are routed through CPLS instead of direct connections [141, 93]. This approach leverages the legitimate and trusted nature of CPLS to evade detection and blend in with normal network traffic.

The life cycle of abusing CPLS as a C&C channel generally involves the following steps:

(a) The botmaster issues commands to the bots in the botnet through the CPLS.

(b) Bots continuously monitor the designated CPLS for new commands from the botmaster.

(c) Bots execute the received malicious commands.

(d) Bots report the results of the executed commands back to the botmaster through the CPLS.

To illustrate this lifecycle with a concrete, real-world example, the following sections examine how different threat actor groups have exploited various CPLS platforms as C&C infrastructure for their malware campaigns.

Abuse Google Drive as C&C infrastructure The Pawn Storm cyber espionage group, also known as Fancy Bear or APT28, has been observed abusing Google Drive as a C&C infrastructure for their malware campaigns, which allowed the attackers to send commands and receive data from compromised systems using the Google Drive API [104].

The lifecycle steps for abusing Google Drive as a C&C infrastructure by the Pawn Storm cyber espionage group can be outlined as follows:

- **Command Execution**: Every time the malware runs a command using cmd.exe, the standard output (STDOUT) of the executed command is piped and written to a Google Drive account.
- **Credential**: The credential of the attacker's Google Drive account are hardcoded within the malware itself.
- **Command and Control**: Every 20 minutes, the bot checks for a file in Google Drive. If a file with a corresponding filename format exists, it downloads that file and runs the contents as a batch file. The STDOUT of the commands is then written back to Google Drive as a result, effectively creating a reverse shell back to the attacker with Google Drive acting as the C&C server.

Abuse OneDrive as C&C Infrastructure The POLONIUM threat actor group has been abused OneDrive for C&C) infrastructure for for their malware campaigns [81], by leveraging OneDrive to retrieve malicious payloads and exfiltrate data from compromised systems.

The lifecycle steps for abusing OneDrive as a C&C infrastructure by the POLONIUM threat actor group can be outlined as follows:

- **Initial Compromise**: POLONIUM gains initial access to the target system through various means, such as phishing emails or exploiting vulnerabilities in web-facing applications.
- **Malware Deployment**: Once the initial compromise is achieved, POLONIUM deploys their custom malware, which is designed to communicate with the OneDrive-based C&C infrastructure.

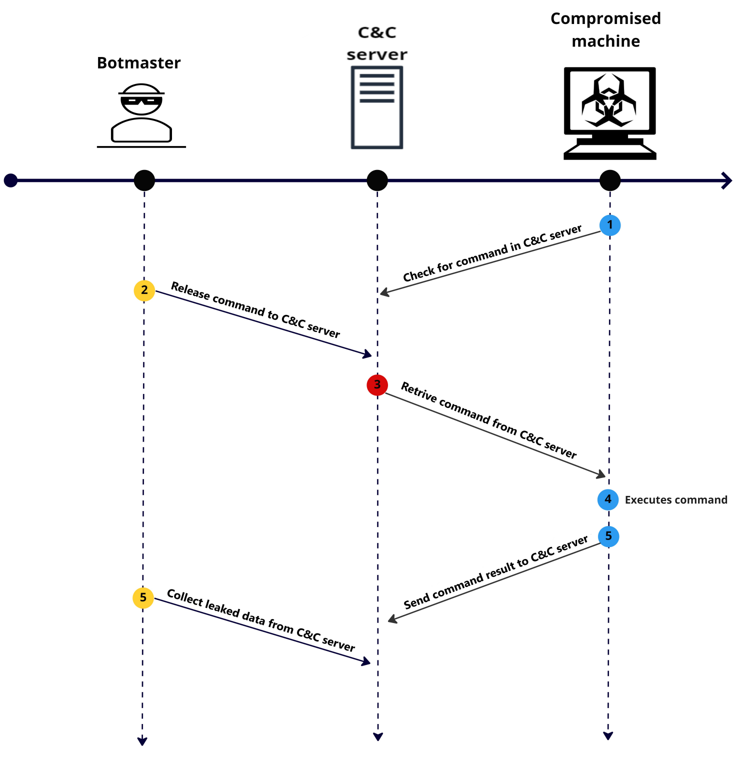
- **Payload Retrieval**: The malware connects to a specific OneDrive account controlled by the attackers and retrieves additional malicious payloads. These payloads are stored in OneDrive as innocuous-looking files to evade detection.
- **Command Execution**: The malware periodically checks the OneDrive account for new commands from the attackers. When a command file is detected, the malware downloads and executes it on the compromised system.
- **Data Exfiltration**: The malware collects sensitive data from the compromised system, such as user credentials, files, and keystrokes. The collected data is then uploaded to the OneDrive account, making it accessible to the attackers.

Abuse Dropbox as C&C Infrastructure The IndigoZebra APT group has been observed abusing Dropbox as a C&C infrastructure for their malware campaigns [64]. The group's malware, named "BoxCaon," uses Dropbox to receive commands from the attackers and exfiltrate data from compromised systems. The lifecycle steps for abusing Dropbox as a C&C infrastructure by the IndigoZebra APT group can be outlined as follows:

- **Initial Configuration**: BoxCaon is configured with hardcoded Dropbox account credentials and a specific directory for C&C communication.
- **Command Retrieval**: The malware continuously monitors the designated Dropbox directory for new files containing commands from the attackers. When a new file is detected, BoxCaon downloads it and saves it to a local directory.
- **Command Execution**: If a file named "c.txt" is found in the local directory, the malware reads its content and executes.
- **Data Exfiltration**: After executing the commands, the malware collects the output and saves it to a file named "p.txt" in the local directory. It then uploads this file to the designated Dropbox directory, making it accessible to the attackers.

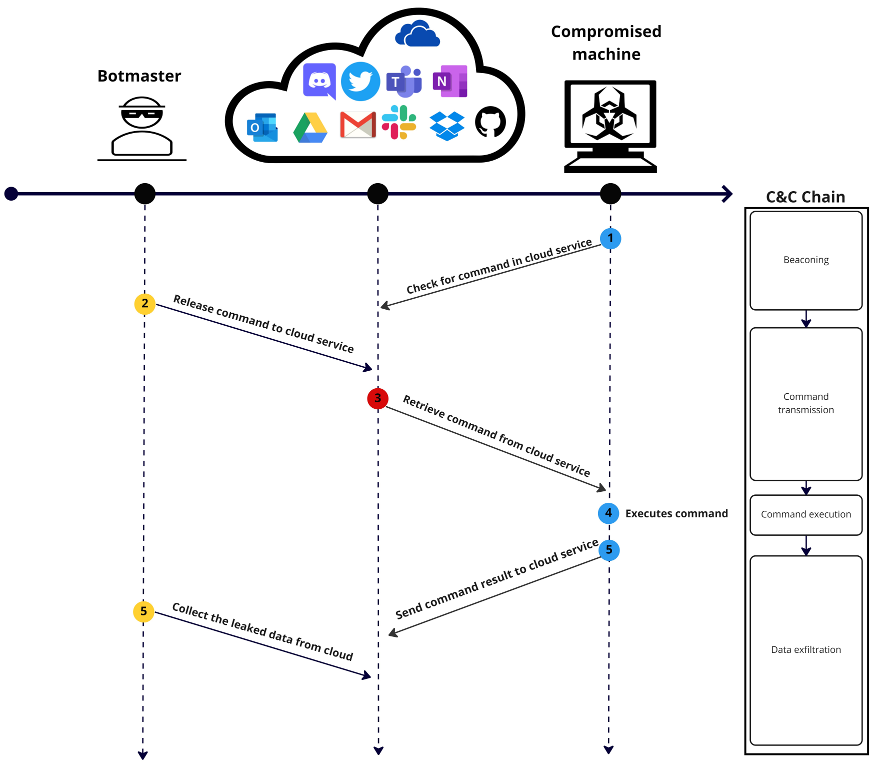
- **Persistence**: To maintain persistence on the infected system, BoxCaon creates a scheduled task that runs the malware every 10 minutes. This ensures that the malware remains active and continues to communicate with the C&C server.

By leveraging trusted and widely-used CPLS such as Google Drive, OneDrive, and Dropbox, APT groups like Pawn Storm, POLONIUM, and IndigoZebra can evade detection by security solutions and ensure the resilience of their C&C infrastructure. This technique allows them to blend their malicious activities with regular ones. This structure is highly stealthy and difficult to detect, as the bot retrieves commands and exfiltrates data through legitimate CPLS. This approach makes it challenging for network defenders and security solutions to differentiate between malicious and benign network traffic, as the malicious activities are hidden within the normal communication patterns of the CPLS.



Botmaster
C&C server
Compromised machine
1
Check for command in C&C server
2
Release command to C&C server
3
Retrive command from C&C server
4
Executes command
5
Send command result to C&C server
5
Collect leaked data from C&C server

(a)



Botmaster
Compromised machine
C&C Chain
Beaconing
Command transmission
Command execution
Data exfiltration
1
Check for command in cloud service
2
Release command to cloud service
3
Retrieve command from cloud service
4
Executes command
5
Send command result to cloud service
5
Collect the leaked data from cloud

(b)

Figure 1.2: Illustration of (a) traditional C&C server communication and (b) CPLS platforms being abused as C&C.

The MITRE ATT&CK framework [17], a globally accessible knowledge base of adversary tactics and techniques, recognises the use of CPLS for C&C as a specific technique under the "Command and Control" tactic. The technique, identified as "Bidirectional Communication" with ID T1102.002 [27], involves the use of CPLS such as Google Drive [60] Onedrive [12] Dropbox [64] to establish bi-directional communication between the compromised system and adversaries.

In the following sections, we will discuss how different threat actor groups abuse various CPLS as C&C infrastructure for their malware campaigns.

Abuse Google Drive as C&C infrastructure The Pawn Storm cyber espionage group, also known as Fancy Bear or APT28, has been observed abusing Google Drive as a C&C infrastructure for their malware campaigns, which allowed the attackers to send commands and receive data from compromised systems using the Google Drive API [104].

The lifecycle steps for abusing Google Drive as a C&C infrastructure by the Pawn Storm cyber espionage group can be outlined as follows:

- **Command Execution**: Every time the malware runs a command using cmd.exe, the standard output (STDOUT) of the executed command is piped and written to a Google Drive account.
- **Credential**: The credential of the attacker's Google Drive account are hardcoded within the malware itself.
- **Command and Control**: Every 20 minutes, the bot checks for a file in Google Drive. If a file with a corresponding filename format exists, it downloads that file and runs the contents as a batch file. The STDOUT of the commands is then written back to Google Drive as a result, effectively creating a reverse shell back to the attacker with Google Drive acting as the C&C server.

Abuse OneDrive as C&C Infrastructure The POLONIUM threat actor group has been abused OneDrive for C&C) infrastructure for for their malware campaigns [81], by leveraging OneDrive to retrieve malicious payloads and exfiltrate data from compromised systems.

The lifecycle steps for abusing OneDrive as a C&C infrastructure by the POLONIUM threat actor group can be outlined as follows:

- **Initial Compromise**: POLONIUM gains initial access to the target system through various means, such as phishing emails or exploiting vulnerabilities in web-facing applications.
- **Malware Deployment**: Once the initial compromise is achieved, POLONIUM deploys their custom malware, which is designed to communicate with the OneDrive-based C&C infrastructure.
- **Payload Retrieval**: The malware connects to a specific OneDrive account controlled by the attackers and retrieves additional malicious payloads. These payloads are stored in OneDrive as innocuous-looking files to evade detection.
- **Command Execution**: The malware periodically checks the OneDrive account for new commands from the attackers. When a command file is detected, the malware downloads and executes it on the compromised system.
- **Data Exfiltration**: The malware collects sensitive data from the compromised system, such as user credentials, files, and keystrokes. The collected data is then uploaded to the OneDrive account, making it accessible to the attackers.

Abuse Dropbox as C&C Infrastructure The IndigoZebra APT group has been observed abusing Dropbox as a C&C infrastructure for their malware campaigns [64]. The group's malware, named "BoxCaon," uses Dropbox to receive commands from the attackers and exfiltrate data from compromised systems. The lifecycle steps for abusing Dropbox as a C&C infrastructure by the IndigoZebra APT group can be outlined as follows:

- **Initial Configuration**: BoxCaon is configured with hardcoded Dropbox account credentials and a specific directory for C&C communication.
- **Command Retrieval**: The malware continuously monitors the designated Dropbox directory for new files containing commands from the attackers. When a new file is detected, BoxCaon downloads it and saves it to a local directory.
- **Command Execution**: If a file named "c.txt" is found in the local directory, the malware reads its content and executes.
- **Data Exfiltration**: After executing the commands, the malware collects the output and saves it to a file named "p.txt" in the local directory. It then uploads this file to the designated Dropbox directory, making it accessible to the attackers.
- **Persistence**: To maintain persistence on the infected system, BoxCaon creates a scheduled task that runs the malware every 10 minutes. This ensures that the malware remains active and continues to communicate with the C&C server.

By leveraging trusted and widely-used CPLS such as Google Drive, OneDrive, and Dropbox, APT groups like Pawn Storm, POLONIUM, and IndigoZebra can evade detection by security solutions and ensure the resilience of their C&C infrastructure. This technique allows them to blend their malicious activities with regular ones. This structure is highly stealthy and difficult to detect, as the bot retrieves commands and exfiltrates data through legitimate CPLS. This approach makes it challenging for network defenders and security solutions to differentiate between malicious and benign network traffic, as the malicious activities are hidden within the normal communication patterns of the CPLS.

Botnet creators find CPLS extremely useful for establishing hidden and robust C&C communication channels due to the following factors:

- **Evasion and Stealth**: CPLS are often whitelisted in corporate environments. By using CPLS, malware can bypass network filters and firewalls, making detection more challenging.
- **Detection Challenge**: By using CPLS, attackers can blend malicious traffic with regular traffic, making it harder for analysts to distinguish between benign and malicious activities.
- **Reliability & Availability**: CPLS usually have high availability, ensuring that the C&C infrastructure remains operational. This contrasts with malicious domains, which can be quickly taken down or blacklisted.
- **Cost-Effective**: CPLS are free or available at low-cost tiers for attackers compared to setting up and maintaining their own infrastructure.
- **Fast Deployment**: With CPLS, attackers can quickly deploy and scale their C&C infrastructure as needed. This speed is especially useful when launching large-scale campaigns.
- **Difficult to Block**: Since CPLS are legitimate and widely used, blocking them entirely would disrupt normal operations for many businesses.
- **Anonymity**: Using CPLS can add a layer of abstraction between the malware authors and their C&C infrastructure, making it harder for researchers to trace back to the actual operators.
- **Data Storage and Exfiltration**: CPLS can be used to store stolen data or payloads. Malware can upload sensitive data to these services without raising alarms, especially if the data is encrypted or obfuscated.

- **Adaptable Infrastructure**: With CPLS, malware authors can easily switch between different services or accounts, making their infrastructure more adaptable and resilient against takedowns.

These stealthy communication channels leverage the trust placed in enterprise-authorized services, the reputable names of CPLS vendors, and secure communication protocols like Transport Layer Security (TLS). This combination provides adversaries with an additional layer of protection, making detection and mitigation efforts more challenging. Moreover, the user-friendly nature of most CPLS, demanding minimal technical skills, lowers the barrier to entry for botnet creators. This ease of use has contributed to the proliferation of botnets abusing CPLS as their C&C infrastructure. Furthermore, users implicitly trust CPLS providers to protect their data and provide secure access. However, malware authors exploit this trust, blending their malicious traffic within the legitimate traffic flow of these services.

Despite the range of defence mechanisms proposed in the literature, the analysis of existing detection mechanisms in Chapter 2 reveals a deficiency in effectively identifying the abuse of CPLS as C&C infrastructure. This gap in the current state of the art highlights the need for novel approaches and solutions to address this emerging threat. These factors have made CPLS a fertile ground for botnet abuse, presenting substantial challenges for most intrusion detection systems. These systems often struggle to differentiate between legitimate and malicious C&C activities, leading to delayed detection and response. This delay in identifying and countering threats substantially increases the likelihood of successful cyberattacks, potentially leading to catastrophic consequences such as data breaches, significant financial losses, and damage to organizations' reputations. To address these gaps and enhance the security of cloud environments, this thesis aims to answer the following research questions:

RQ1: What techniques are utilized to abuse the CPLS as C&C communication channels, and how frequently are these attack techniques employed?

RQ2: Which types of CPLS are targeted for abuse?

RQ3: What countermeasures have been proposed to detect the abusive use of CPLS as C&C infrastructure?

RQ4: How can we develop and validate a comprehensive dataset and labeling strategy for Portable Executable (PE) files to support the detection of CPLS abuses?

RQ5: How can statically extracted features from Portable Executable (PE) files be leveraged through artificial intelligence or machine learning methodologies to enhance the detection and mitigation of CPLS abuses?

RQ6: In what ways can dynamic feature extraction from Portable Executable (PE) files, integrated with machine learning models, improve the detection of CPLS abuse as C&C infrastructure?

RQ7: How can the robustness of detection models against adversarial attacks be enhanced for both statically and dynamically extracted features from Portable Executable (PE) files in CPLS abuse?

1.3 Objectives

To address the issues mentioned previously, this thesis will focus on the following key objectives:

- **Objective 1: Conduct a comprehensive literature review to establish a taxonomy of attack techniques employed by threat actors to abuse CPLS as C&C infrastructure.** This objective will involve a thorough examination of existing research, industry reports, and case studies to identify and categorize the various methods, tools, and strategies used by malicious actors to exploit CPLS for C&C purposes. By systematically organising and classifying these attack techniques, this thesis aims to provide a clear understanding of the current threat landscape and identify potential gaps in existing defence mechanisms, thus laying the foundation for developing a novel detection system.

- **Objective 2: Identify gaps in existing defense mechanisms of abuse the CPLS as C&C infrastructure.** Building upon the insights gained from the literature review and the developed taxonomy of attack techniques, this objective will involve a critical analysis of current defense mechanisms and their limitations in effectively detecting and mitigating CPLS abuse.

- **Objective 3: Investigate the use of static and dynamic malware analysis methods through Artificial Intelligence (AI) techniques to detect CPLS abuse.** This objective will explore the potential of AI-driven approaches to enhance the effectiveness and efficiency of malware analysis in the context of CPLS abuse detection. Taking advantage of the strengths of static and dynamic analysis methods, this thesis aims to develop an approach that can accurately identify and classify malicious artifacts, behaviours, and communication patterns indicative of CPLS abuse. The investigation will involve feature engineering techniques and evaluation of machine learning models to determine the most suitable combination to achieve high detection accuracy while minimizing false positives and computational overhead. The ultimate goal is to develop an intelligent, scalable, and adaptable system that can effectively detect and respond to the ever-evolving threat of CPLS abuse in real-world cloud environments.

By successfully addressing these objectives, this thesis aims to contribute to the field of cloud security by providing a comprehensive understanding of the CPLS abuse threat landscape, developing a taxonomy of attack techniques, and proposing a robust AI-driven detection system that combines static and dynamic malware analysis methods. The results of this research will have significant implications for improving the security posture of organisations using cloud services and will help to safeguard the confidentiality, integrity, and availability of data and resources hosted within these environments.

1.4 CPLS Abuse Detection System Architecture and Intended Users

The proposed detection system is designed to be implemented by cybersecurity teams within organizations, cloud service providers, or managed security service providers (MSSPs). The system architecture consists of several key components:

1. Data Collection Module: Gathers PE files from various sources within the organization's network or cloud infrastructure.
2. Feature Extraction Module: Applies static and dynamic analysis techniques to extract relevant features from the collected PE files.
3. Machine Learning Module: Utilizes the extracted features to classify files as benign or malicious using the trained Random Forest model.
4. Alert Generation Module: Produces alerts for files classified as potentially abusing CPLS for C&C infrastructure.
5. Reporting Interface: Provides a dashboard for security analysts to review and investigate detected threats.

The primary users of this system would be:

1. Security Operations Center (SOC) analysts who monitor and respond to security incidents.
2. Threat hunters who proactively search for hidden threats within the network.
3. Incident response team who investigates and mitigates detected abuses.
4. Cloud security team who continuously monitor cloud environments to identify potential abuses.

By integrating this detection system into existing security infrastructure, organisations can enhance their capability to identify and mitigate the abuse of CPLS as C&C infrastructure, thereby improving their overall security posture against sophisticated cyber threats.

1.5 CPLS Abuse Detection System Deployment Scenarios

The system can be deployed in various scenarios, depending on the organisation's infrastructure and security needs:

1. **On-premises Deployment:**
    - **Installation:** The system would be installed on dedicated servers within the organisation.
    - **Data Collection:** Network traffic analysers and endpoint agents would collect PE files and send them to the central analysis server.
    - **Integration:** The system would integrate with existing Security Information and Event Management (SIEM) systems for centralised alert management.
2. **Managed Security Service Provider (MSSP) Deployment:**
    - **Installation:** The MSSP would host the system in their infrastructure and offer it as a service to multiple clients.
    - **Data Collection:** Secure channels would be established between the client's infrastructure and the MSSP's system to collect relevant PE files.
    - **Integration:** The system would integrate with the MSSP's existing multi-tenant security management platform.

3. **Deployment by Cloud Service Providers:** Cloud service providers such as Microsoft (OneDrive), Google (Google Drive), or Dropbox could implement our CPLS abuse detection system directly into their infrastructure to detect potential abuse. The integration could be as follows:

    - **Installation:**
        - The system would be integrated directly into the cloud provider's existing infrastructure.
        - Machine learning models and analysis engines would be deployed across the provider's global data centers for distributed processing.
        - A centralized management console would be set up for the provider's security team to monitor and respond to alerts.
    - **Data Collection:**
        - The system would automatically analyze PE files upon upload, modification, or based on scheduled scans of existing files.
        - Metadata about file access patterns, and behaviors would be collected to provide context for the analysis.
    - **Integration:**
        - The system would be integrated with the provider's existing security infrastructure and threat intelligence platforms.
        - Alerts would be fed into the provider's security information and event management (SIEM) system for correlation with other security events.
        - Automated response mechanisms could be implemented to quarantine suspicious files, limit sharing, or suspend potentially compromised accounts.

    By implementing the CPLS abuse detection system in this manner, cloud service providers could significantly enhance their ability to detect and prevent the abuse of their platforms as C&C infrastructure.

1.6 Practical Deployment Considerations

While the previous section outlines general deployment scenarios, each case requires careful evaluation due to unique challenges and considerations. Here, we provide a more detailed discussion on the practical aspects of deploying our CPLS abuse detection system in different environments:

1. **On-premises Deployment:**
    - **Performance Impact:** Evaluate the computational load on existing infrastructure. Organizations may need to allocate dedicated hardware resources to ensure optimal performance without impacting other operations.
    - **Network Integration:** Careful integration with existing network security tools is crucial. This may involve configuring firewalls, intrusion detection systems, and SIEM tools to work in tandem with the new system.
    - **Data Privacy Compliance:** On-premises deployments must adhere to data protection regulations.
2. **Managed Security Service Provider (MSSP) Deployment:**
    - **Scalability:** MSSPs need to ensure the system can scale to accommodate multiple clients. This may involve implementing multi-tenancy features and load balancing mechanisms.
    - **Customization:** Each client may require specific customizations. MSSPs should develop a flexible framework that allows for client-specific rules and thresholds without compromising the core functionality.
    - **Service Level Agreements (SLAs):** Clear SLAs must be established, detailing response times, update frequencies, and support levels for the detection system.
3. **Cloud Service Provider Implementation:**

- **Integration with Existing Services:** Cloud providers must seamlessly integrate the detection system with their existing services without disrupting the user experience.
- **Real-time Analysis at Scale:** Given the massive volume of data, providers need to optimize the system for real-time analysis of large number of files simultaneously.

Each deployment scenario requires:

- Thorough testing in a staged environment before full implementation
- Regular model updates to adapt to evolving CPLS abuse techniques
- Continuous monitoring and tuning of system performance and accuracy
- Integration with existing security workflows and incident response procedures

1.7 Key Contributions

This thesis makes several novel and significant contributions to the field of cybersecurity, particularly in detecting CPLS abuse as C&C infrastructure:

1.7.1 Novel Taxonomies

These taxonomies significantly enhance our understanding of CPLS abuse techniques and C&C communication channels, providing researchers and security professionals with a structured insights to analyse and combat evolving threats.

1. **Taxonomy of CPLS Abuse Techniques:** We introduce a comprehensive classification of methods used by malicious actors to exploit CPLS. This taxonomy covers techniques such as steganography, encoding, cryptography, fraudulent

accounts, use of botmaster's credentials or hard-coded tokens, compromised victims' accounts, component object model (COM) hijacking, process injection, COMSPEC environment variable exploitation, multiple process exploitation, and AI-powered C&C, providing a structured understanding of emerging threat vectors.

2. **Taxonomy of C&C Communication Channels:** We present a systematic categorization of botnet communication strategies, covering traditional channels (IRC, HTTP, P2P) and emerging ones (DNS abuse, CPLS platforms).

These taxonomies fill a critical gap in existing literature and provide a foundation for future research in this rapidly evolving area of cybersecurity.

1.7.2 First Labelled Dataset for CPLS Abuse

We contribute the first dataset specifically addressing CPLS abuse, comprising 3,067 malicious and 3,067 benign Portable Executable (PE) files, which are binary executable files used in Windows operating systems. This dataset has been obtained from VirusTotal and spanning the years 2017 to 2021, addresses a critical gap in the field by enabling more accurate and relevant training of machine learning models for CPLS abuse detection. It includes malicious samples that exhibit communication with known CPLS-hosted domains and benign samples from trusted sources. Our dataset focuses on Portable Executable (PE) files due to their prevalence in Windows systems and high representation in malware samples. While other file types like PDF, DOCX, ZIP, and JavaScript can also be used maliciously, PE files are the primary executable format in enterprise environments and the most commonly submitted file type to VirusTotal, as illustrated in Figure 3.2.

1.7.3 Machine Learning ML-based CPLS Abuse Detection Systems

These high-performance detection systems represent a significant leap forward in identifying CPLS abuse, demonstrating the potential of machine learning techniques in tackling complex cybersecurity challenges.

1. **Static Analysis Approach:** we develop a ML CPLS abuse detection system achieving 98.26% detection accuracy, utilizing:

    - Tailored feature engineering: we tailor informative feature engineering to behaviours that indicate CPLS abuse, such as connecting to known CPLS domains and making potential C&C API calls.
    - Custom feature elimination (CFE) method for optimal feature selection: to identify the most relevant features, we introduced a custom feature elimination (CFE) method designed to determine the exact number of features needed for filter selection approaches. For wrapper-based feature selection, we utilise various techniques, including sequential feature selector forward (SFSF), sequential feature selector backward (SFSB), and recursive feature elimination (RFE).

2. **Dynamic Analysis Approach:** we develop a ML CPLS abuse detection system achieving 97.95% detection accuracy, utilizing:

    - Innovative Dual-Sandboxing technique: we introduce an innovative approach that combines Cuckoo and Triage sandboxes to extract a comprehensive set of features, including host-based features (API calls, file activities, system interactions) and network-based features (interactions with cloud services and the Internet).
    - Extraction of 99 features: we employ both filter-based (Information Gain and ReliefF) and wrapper-based (Random Forest and J48) feature selec-

tion strategies. However, these methods did not surpass the 98% accuracy achieved using the complete set of 99 features derived from our Dual-Sandbox approach, highlighting the importance of each feature in accurately differentiating between malicious and benign samples.

Our detection systems demonstrate low misclassification rates. For static analysis, we observe a false positive rate of 1.74% (falsely identifying benign samples as malicious) and a false negative rate of 2.57% (falsely identifying malicious samples as benign). For dynamic analysis, we observe a false positive rate of 2.05% and a false negative rate of 2.57%. These comprehensive results, showing both high accuracy and low misclassification rates in both directions, represent a significant advancement in identifying CPLS abuse.

1.7.4 Novel RMCP Adversarial Attack

The RMCP attack provides a crucial evaluation of the robustness of CPLS abuse detection systems. In the context of CPLS abuse, where attackers continuously evolve their techniques, understanding and improving the resilience of detection systems against potential evasion tactics is vital for maintaining long-term effectiveness. we propose the Replace Misclassified Parameter (RMCP) as a novel white-box adversarial attack to evaluate the robustness of our proposed abuse detection systems. Despite the adversarial attack, the approach retains a relatively high detection accuracy level. For the ML-based CPLS abuse detection system utilizing static malware analysis, the detection accuracy rate drops from 98% to 83% under the RMCP attack, outperforming the robustness rates of 69.03% and 54.37% achieved by Kumar et al. [126] and Raman et al. [145], respectively. On the other hand, for the ML-based CPLS abuse detection system utilizing dynamic malware analysis, the detection accuracy rate drops from 97.95% to 90.01%, comparing the robustness rates of 50.00% and 47.57% achieved by Sethi et al. [150] and Shijo et al. [152], respectively. This contribution has been published [72].

1.7.5 Comprehensive Analysis and Future Directions

We provide a thorough examination of existing studies, identifying key challenges and opportunities in CPLS abuse detection. This analysis lays the groundwork for future research, proposing directions such as real-time monitoring of CPLS platforms and development of collaborative detection frameworks. These contributions collectively advance the field of cybersecurity by:

- Providing structured frameworks for understanding and categorizing CPLS abuse
- Offering novel tools and methodologies for developing effective detection systems
- Demonstrating the potential and robustness of ML in addressing emerging cybersecurity challenges
- Setting the stage for future research and practical applications in CPLS abuse detection

1.8 Thesis Outline

This thesis is structured into seven chapters, each focusing on a specific aspect of the research:

- **Chapter 1 - Introduction:** This chapter provides an overview of the research problem, highlighting the growing threat of CPLS abuse as C&C infrastructure. It presents the motivation behind the study, the aims and objectives, and the significant contributions of the thesis to the field of cloud security.
- **Chapter 2 - Literature Review:** This chapter presents a comprehensive review of the literature related to the abuse of CPLS as C&C infrastructure. It analyses

and compares previous studies, identifying the strengths, limitations, and gaps in current detection approaches. The chapter also introduces novel taxonomies for attack techniques and C&C communication channels, providing a structured framework for understanding the threat landscape. This contribution has been published [73].

- **Chapter 3 - Dataset and Preprocessing:** This chapter describes the dataset used in the research, including the collection process, data sources, and the preprocessing techniques applied to prepare the data for analysis. It provides insights into the characteristics of the dataset and the steps taken to ensure its quality and suitability for the experiments. This contribution has been published [72].

- **Chapter 4 - Static Analysis Using Machine Learning for Detection of CPLS Abuse as C&C Infrastructure:** This chapter focusses on applying machine learning techniques to detect CPLS abuse using static analysis. It presents the feature engineering process, the selected machine learning algorithms, and the evaluation metrics used to assess the performance of the models. The chapter discusses the results obtained and provides insight into the effectiveness of static analysis in detecting CPLS abuse. This contribution has been published [72].

- **Chapter 5 - Dynamic Analysis Using Machine Learning for Detecting the CPLS Abuse as C&C Infrastructure:** This chapter explores machine learning techniques to detect CPLS abuse through dynamic analysis. It describes the dynamic analysis environment, the feature extraction process, and the machine learning models employed. The chapter presents the experimental results and compares the performance of dynamic analysis with static analysis in the context of CPLS abuse detection. This contribution has been submitted for publication.

- **Chapter 6 - Robustness of CPLS Abuse Detection System:** This section introduces the Replace Misclassified Parameter (RMCP) as a novel adversarial attack to evaluate the robustness of the proposed machine learning models. It compares the resilience of the models against related works and discusses the implications of the RMCP attack on the overall security of the detection system. This contribution has been published [72].

- **Chapter 7 - Conclusion:** The final chapter summarizes the essential findings and contributions of the thesis. It revisits the research objectives and discusses how they have been addressed throughout the study. The chapter also highlights the practical implications of the research for enhancing the security of cloud environments and provides recommendations for future work in the field of CPLS abuse detection.

1.9 Summary

This chapter introduces the research problem of malware abusing CPLS as C&C infrastructure and the motivations behind this thesis. It outlines the main objectives, including establishing a taxonomy of attack techniques, developing a threat model, and investigating AI techniques for malware analysis. The chapter also presents the significant contributions of the thesis, such as novel taxonomies, comprehensive analysis of existing studies, a novel labelled dataset, the development of machine learning-based CPLS abuse detection systems, and insights into their robustness. In conclusion, these contributions collectively represent significant progress in the field of CPLS abuse detection. By providing such contributions, this thesis not only enhances our current capabilities in detecting and mitigating CPLS abuse, but also lays a strong foundation for future research and practical applications. These developments are crucial in the ongoing effort to secure cloud environments and protect against increasingly sophisticated cyber threats.

Chapter 2

Literature review

2.1 Introduction

This chapter presents a literature review on the abuse of Cloud-based Public Legitimate Services (CPLS) as Command and Control (C&C) channels by botnets. Although numerous surveys have focused on botnet attacks and their detection mechanisms, only a few, such as the work of Radunovic et al. [144], specifically address botnets abusing CPLS as C&C channels. Our Systematic Literature Review (SLR) aims to provide a comprehensive overview of this research area, identifying and categorizing various types of abusive attacks. In contrast to the four abusive tactics proposed by Radunovic et al., our SLR identifies ten different types of abusive attacks. The chapter also includes a comparison of our SLR with the survey by Radunovic et al. [144], highlighting the unique methodology adopted in our SLR. This methodology incorporates a structured and comprehensive search strategy, criteria for inclusion and exclusion, an evaluation of study quality, and data synthesis to formulate a taxonomy of attack techniques employed by botmasters to exploit CPLS C&C channels. Furthermore, the chapter discusses other relevant surveys, such as those by Khattak et al. [123] and Latah [128], which provide valuable insights into the evolving threat landscape of botnets and their use of non-traditional C&C channels. Despite the differing focuses and methodologies between these related surveys and our SLR, they collectively emphasize the necessity for ongoing research and innovative solutions to tackle emerging cyber threats. The chapter is organized as follows: Section 2.2 presents related surveys on the topic, followed by a comparison of our proposed SLR to that of Radunovic et al. [144]. The subsequent sections detail the systematic literature review methodology, including the research strategy, research questions, search process, and study eligibility selection. The chapter concludes with a discussion on the challenges associated with detecting and preventing the botnet exploitation of these non-traditional C&C channels. This literature review aims to provide a comprehensive understanding of the current state of research on the abuse of CPLS as C&C channels, thereby paving the way for future research in this critical area of cybersecurity.

2.2 Related Surveys

Although there is a significant number of surveys focusing on botnet attacks and their detection mechanisms, such as those of Khattak et al. [123], Kuitert [125], Silva et al. [154], and Singh et al. [158], only Radunovic et al. [144] specifically address botnets abusing CPLS as C&C channels. In their survey, Radunovic et al. present a taxonomy of malicious social bots and discuss various attack types at different stages. They also propose four abusive tactics, including encoding communications within text-based social media posts, using steganography, exploiting free accounts, and implementing a username generation algorithm (UGA). Despite recognizing four abusive techniques in this survey, our proposed SLR identifies and categorizes ten different types of abusive attacks. Table 2.1 compares our SLR to the survey by Radunovic et al. [144]. Our SLR adopts a unique methodology that diverges from the approach utilized in the related work. The SLR we conduct incorporates a structured and comprehensive search strategy, criteria for inclusion and exclusion, an evaluation of study quality, and data synthesis to formulate a taxonomy of attack techniques employed by botmasters to exploit CPLS C&C channels. Conversely, the related work does not follow a systematic review methodology. It instead provides an overview of current research on botnet C&C channels through social media, investigating emerging trends, potential defense strategies, and areas warranting further research. Khattak et al. [123] offer a concise overview of an evasive strategy that abuses social networking websites like Facebook and Twitter for C&C functions and provides a detailed taxonomy of the botnet phenomenon. This review also anticipates a trend, wherein CPLS would either be used to create bots (i.e., botcloud) or to host the C&C on the cloud in the future. This insight aligns with our findings, which highlight the growing abuse of CPLS platforms as C&C channels. Latah [128] proposes a taxonomy of malicious social bots and characterization of various attack strategies at different stages (initiation, listening, and execution-stage attacks). The main focus is on a social bot, with the discussion being around one attack technique, which involves using steganography to abuse Facebook and Twitter as C&C mediums. This technique is also identified in our SLR as one of the abusive tactics employed by botmasters. It is important to note that the aforementioned reviews focus on social media C&C infrastructures, while our SLR explores the abuse of cloud-based C&C infrastructures. Despite this difference in focus, all the aforementioned reviews lack a systematic approach, thereby failing to provide a comprehensive overview of this research area. Despite the differing focuses and methodologies between the related surveys and our SLR, they offer significant insights into the evolving threat landscape of botnets and their use of non-traditional C&C channels. Our SLR, along with two other reviews [128, 144], discusses the challenges associated with detecting and preventing the botnet exploitation of these non-traditional C&C channels. These reviews emphasise the necessity for ongoing research and innovative solutions to tackle emerging cyber threats, which is consistent with the conclusions drawn from our SLR.

Table 2.1: Comparison of our proposed SLR to that of Radunovic et al. [144].

Aspect	**Our Proposed SLR**	**Radunovic et al. [144]**
Focus	Abuse of the CPLS platforms as C&C channels	Abuse of social media platforms as C&C channels
Specific attacks discussed	Covers 10 types of attack techniques that are employed to abuse the CPLS platforms as C&C infrastructure.	Focuses specifically on the use of social media platforms through means such as status updates, comments, direct messages, and the creation of fake accounts.
Taxonomy details	Provides a comprehensive taxonomy of attack techniques used by botmasters to abuse CPLS as C&C channels. These techniques include steganography, encoding, cryptography, fraudulent accounts, use of Botmaster's credentials or hard-coded tokens, compromised victims' accounts, component object model (COM) hijacking, process injection, COMSPEC environment variable exploitation, multiple process exploitation, and AI-powered C&C.	This review discusses the use of text-based social media (SM) posts, hidden communications through image and linguistic steganography, and the utilization of public cloud storage for the unobservable exchange of communications and uploading of stolen files. It also includes the use of domain generation algorithms and the conveying of C&C messages through comments on public SM posts.
Review methodology	Systematic Literature Review.	Not specified.
Time frame	2008–July 2023.	Not specified.
Number of studies	91	Not specified.

2.3 Systematic Literature Review Methodology

2.3.1 Research Strategy

Based on the review aims, the methodological framework by Kitchenham and Charters [124] was adopted in this SLR. The review process outlined in this framework summarizes the stages of an SLR into three main phases: planning, conducting, and reporting. The primary motivation for adhering to these stages is to identify, analyze, and interpret all available literature related to the abuse of CPLS as a C&C infrastructure. Adherence to a predefined protocol is crucial for reducing potential research bias in data selection and analysis. It also enhances reliability through the replicability of the process, enabling others to follow the same procedure. Our systematic review commenced with the selection of bibliographic databases for the search, along with the development of a set of inclusion and exclusion criteria and search strings. The search process and the inclusion and exclusion criteria used are presented below. The search strings were employed to query two primary sources: the academic and industrial literature. Abbreviations and synonyms of search terms were taken into consideration. To retrieve the most relevant literature, the search strings were restricted to titles, abstracts, and keywords.

2.3.2 Research Questions

Our SLR aimed to explore the state-of-the-art attack and detection techniques employed for abusing CPLS as C&C communication channels. To achieve this objective, we formulated the research categories and questions, which are outlined in Table 2.2.

Table 2.2: Research questions.

Category	Research Questions	Aim of Discussion
Abuse Technique	(1) What techniques are utilized to abuse the CPLS as C&C communication channels, and how frequently are these attack techniques employed?	• Investigate and understand the specific strategies or methods used by attackers to abuse CPLS for their C&C communication channels. • Assess the prevalence of these attack techniques, which may help to understand their popularity or effectiveness.
	(2) Which types of CPLS are targeted for abuse?	• Identify the specific CPLS that are most vulnerable or frequently targeted by these abusive techniques.
Abuse Detection	(3) What countermeasures have been proposed to detect the abusive use of CPLS as C&C infrastructure?	• Explore and evaluate the existing countermeasures or detection methods that have been proposed to identify and combat the abusive use of CPLS as C&C infrastructure.

2.3.3 Search Process

The SLR process followed is depicted in Figure 2.1. To extract all literature relevant to the defined research questions, we employed a search strategy using Boolean expressions ‘AND’ and ‘OR’ to combine the search terms. The main keywords used in the search for relevant articles were as follows:

- (C2 **OR** C&C **OR** “Command and Control”) **AND**.
- (cloud **OR** Legitimate **OR** platform **OR** Service **OR** public **OR** “public service” **OR** OSN **OR** “social network" **OR** blogging **OR** blog) **AND**.
- (bot **OR** botnet **OR** malware) **AND**.
- (abuse **OR** exploit).

These combined queries were then applied to a selection of academic databases that included the IEEE Xplore Digital Library, SpringerLink, ACM Digital Library, ScienceDirect, and Scopus.

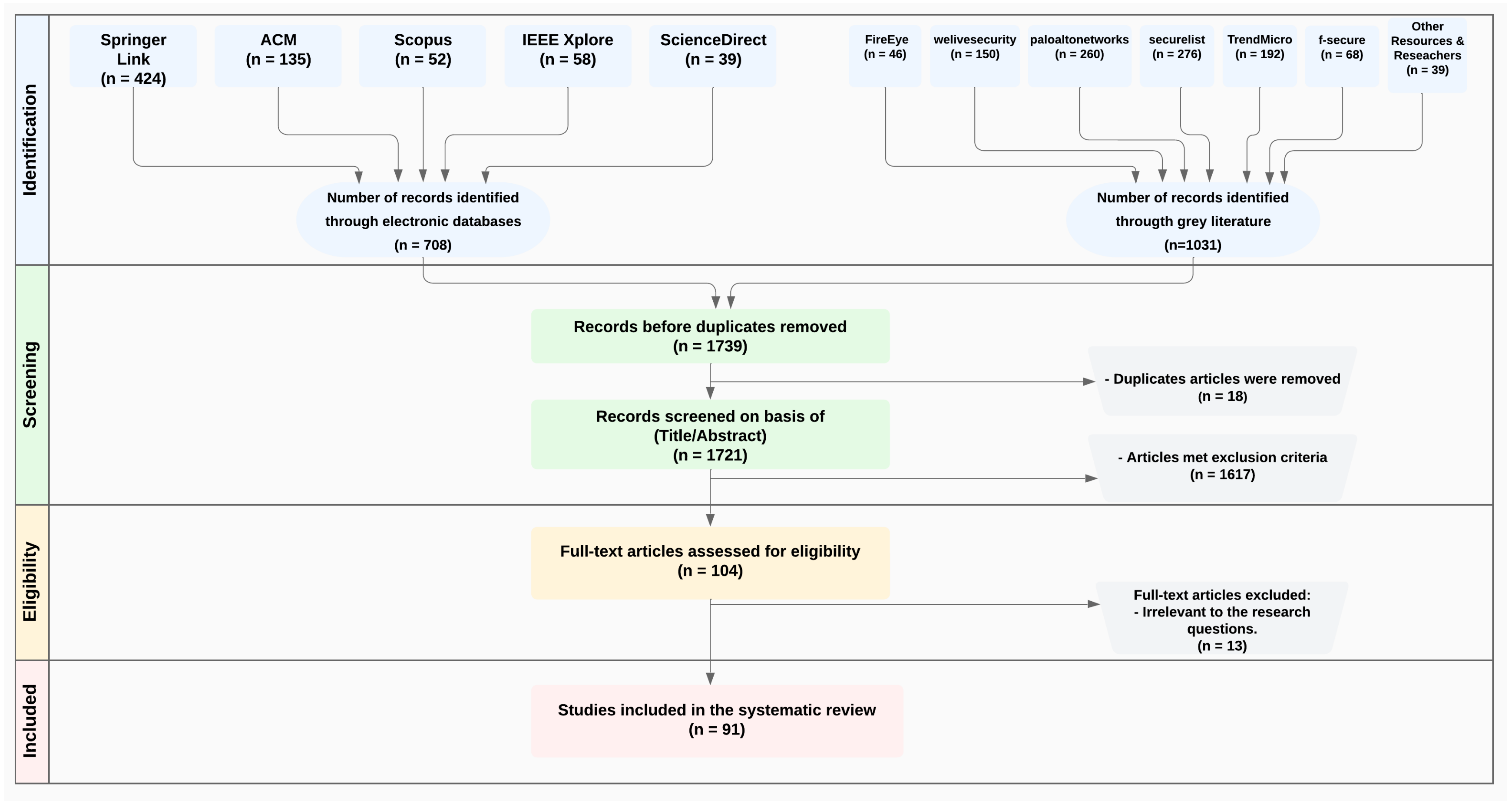

Identification
Springer Link (n = 424)
ACM (n = 135)
Scopus (n = 52)
IEEE Xplore (n = 58)
ScienceDirect (n = 39)
FireEye (n = 46)
welivesecurity (n = 150)
paloaltonetworks (n = 260)
securelist (n = 276)
TrendMicro (n = 192)
f-secure (n = 68)
Other Resources & Reseachers (n = 39)
Number of records identified through electronic databases (n = 708)
Number of records identified througth grey literature (n=1031)
Screening
Records before duplicates removed (n = 1739)
- Duplicates articles were removed (n = 18)
Records screened on basis of (Title/Abstract) (n = 1721)
- Articles met exclusion criteria (n = 1617)
Eligibility
Full-text articles assessed for eligibility (n = 104)
Full-text articles excluded: - Irrelevant to the research questions. (n = 13)
Included
Studies included in the systematic review (n = 91)

Figure 2.1: Process for extracting relevant articles.

Given that the majority of relevant incidents involving the abuse of CPLS as C&C channels were reported by the threat intelligence industry, we expanded our research to include these sources. Examples of these sources include FireEye, TrendMicro, WeLiveSecurity, F-Secure, Unit42 Palo Alto, and SecureList.

2.3.4 Study Eligibility Selection

The search is limited to the literature that focuses on the abuse of CPLS C&C communication channels. Given the large number of articles that could be retrieved using the previously mentioned research strategy, an assessment criterion was necessary to select those that best address our research questions. To retrieve the most relevant articles in the field, we applied the selection criteria outlined in Table 2.3.

Table 2.3: Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
• Studies published from 2008, when the first abuse incident was discovered, through July 2023. • Studies focus on the abuse of CPLS as C&C channels. • Studies discuss an approach for identifying and/or preventing the abuse of CPLS as C&C channels.	• Studies written in a language other than English. • Studies focus on the behavior of malicious or automated accounts, bots, using social media to amplify and spread misinformation, increase fake followers, and impersonate genuine (human) accounts. • Studies that do not provide an answer to the research questions.

2.3.5 Data Extraction and Synthesis

After applying the inclusion and exclusion criteria, we obtained a total of 91 studies. Figure 2.1 illustrates the process of extracting relevant articles, while Table 2.4 and Table 2.5 show the number of included academic and industrial publications, respectively. In the next phase, our aim is to review each of these studies to distill the following core information:

- The publication date of each study.
- The specific CPLS that was abused for C&C channels.
- The attack techniques that were utilized to abuse the CPLS as C&C channels.
- Any proposed detection mechanisms aiming to mitigate such abuses.

Subsequently, we embarked on the phase of data synthesis, wherein we aggregated all relevant data. This was a crucial step enabling us to comprehensively address our different research questions.

Table 2.4: Number of included academic publications.

Source	**Academic Publications**
IEEE	5
Springer Link	9
AMC	2
ScienceDirect	4
Scopus	8

Table 2.5: Number of included industrial publications.

Source	Industrial Publications
FireEye	4
TrendMicro	11
welivesecurity	15
f-secure	2
paloaltonetworks	4
securelist	4
Threat Intelligence	23

2.4 Research Findings

In this SLR, we explored the topic of CPLS abuses as C&C channels, focusing particularly on understanding the attack techniques employed and countermeasures proposed. This abuse is not limited to a particular type of service; instead, it is pervasive across a variety of platforms, including cloud storage, social media, business communication platforms, and more. Our findings indicate that botnets have evolved significantly over the years, adopting diverse methods to abuse the CPLS for C&C communication. For instance, we observed techniques such as steganography, cryptography, COM hijacking, process injection, COMSPEC environment variable exploitation, and AI-powered C&C. Further discussion and additional attack techniques are presented in Section 2.6. Many mainstream platforms were found to be popular targets of abuse: cloud storage sites such as Dropbox and Google Drive, social media sites like Twitter and Facebook, business communication platforms like Slack, developer repositories (GitHub), online clipboard sites (Pastebin), push services for iOS and Android notifications, video and photo sharing sites (YouTube and Instagram), email services (Outlook and Gmail), digital distribution platforms (Discord), and instant messaging software (Telegram). The countermeasure strategies proposed to date, both in computer and Android environments, involve detecting anomalies in user behavior, CAPTCHA verification, reputation score calculation, and causality measurement between user activity and network traffic. However, each of these strategies come with their own limitations as outlined in Section 2.4.3.

2.4.1 RQ1: What Techniques Are Utilised to Abuse the CPLS as C&C Communication Channels, and How Frequently Are These Attack Techniques Employed?

Investigating and comprehending the specific strategies or methods employed by attackers to abuse CPLS for their C&C communication channels is essential for understanding the nature and extent of these attacks. The taxonomy of attack techniques presented in Table 2.6 showcases the wide range of methods utilized by threat actors to exploit CPLS for C&C operations. These techniques align with several of the tactics outlined in the MITRE ATT&CK framework, a widely-used knowledge base of adversary tactics and techniques in the industry [17]. Our taxonomy identifies techniques such as steganography, encoding, cryptography, fraudulent accounts, use of botmaster's credentials or hard-coded tokens, compromised victims' accounts, component object model (COM) hijacking, process injection, and modifying system variables. These techniques correspond to specific techniques listed under the "Command and Control" tactic in the MITRE ATT&CK framework [4]. For instance, the use of steganography in our taxonomy aligns with the "Data Obfuscation: Steganography" technique (T1001.002) in the MITRE ATT&CK framework, which involves hiding C&C traffic within seemingly benign data or files, such as images or documents [6]. Similarly, the encoding technique in our taxonomy corresponds to the "Data Encoding" technique (T1132) in the MITRE framework [5], where adversaries encode data using standard or non-standard encoding schemes to make the content of C&C traffic more difficult to detect. The cryptography technique in our taxonomy relates to the "Encrypted Channel" technique (T1573) [10], which involves employing known encryption algorithms to conceal command and control traffic. This can include the use of symmetric cryptography (T1573.001) [9] or asymmetric cryptography (T1573.002) [8]. The use of fraudulent accounts in our taxonomy also relates to the "Account Manipulation: Additional Cloud Credentials" sub-technique (T1098.001) [1] under the "Persistence" and "Privilege Escalation" tactics in the MITRE ATT&CK framework. This sub-technique involves adversaries adding their own credentials to a cloud account to maintain persistent access to victim accounts and instances within the environment. The COM hijacking technique employed by malware such as Turla [106] and ComRAT [101] aligns with the "Event Triggered Execution: Component Object Model Hijacking" sub-technique (T1546.015) [11] under the "Persistence" tactic in the MITRE ATT&CK framework. This sub-technique involves adversaries establishing persistence by executing malicious content triggered by hijacked references to COM objects. Turla utilises the legitimate messaging application programming interface (MAPI) to interact with Outlook, granting complete control over the target mailbox, and utilising additional MAPI functionalities. To ensure persistence and concealment, the operators employ the COM system to modify the Windows Registry, enabling the malware to execute every time the user logs in. Most of the techniques in our taxonomy of attack techniques for abusing CPLS as C&C infrastructure (Table 2.6) align with the techniques listed under the "Command and Control" tactic in the MITRE ATT&CK framework [4]. This includes techniques such as steganography (T1001.002), data encoding (T1132), encrypted channel (T1573), web service (T1102), and more. However, one technique in our taxonomy that is not currently represented in the MITRE ATT&CK framework is the AI-powered C&C technique, as described in Section 2.6.8. This technique, proposed by Wang et al. in their DeepC2 framework [165], leverages AI to enable covert communication between the malware and the attacker on online social networks (OSNs). By aligning our taxonomy to the MITRE ATT&CK framework, we provide a more comprehensive understanding of the attack techniques employed by threat actors to abuse CPLS for C&C operations. This alignment helps bridge the gap between our research and the industry-standard knowledge base, facilitating the development of effective countermeasures and defense strategies. Based on the insights gained from investigating the techniques used to abuse CPLS as C&C communication channels, we extracted two key derived features in the static analysis chapter (Chapter 4): presence_of_CLS_domains and potential_C&C_api_calls. These features are specifically designed to capture the characteristics of CPLS abuse as C&C infrastructure, complementing the common PE features typically used in malware analysis. These derived features, presence_of_CLS_domains and potential_C&C_api_calls, play a crucial role in capturing the specific characteristics of CPLS abuse as C&C infrastructure. By incorporating these features alongside the common PE features, our static analysis approach aims to enhance the detection of malware that exploits CPLS for malicious purposes. As demonstrated in Table 4.6, the inclusion of these derived features, particularly potential_C&C_api_calls, significantly improves the detection accuracy of our machine learning models. In addition to the static analysis approach, we also introduced the innovative Dual-Sandboxing (DS) technique in the dynamic analysis chapter (Chapter 5). This technique is specifically designed to extract a comprehensive set of dynamic features that capture the characteristics of CPLS abuse as C&C infrastructure. The DS technique combines the strengths of two sandbox environments, Cuckoo and Triage, to gather a broad spectrum of features related to both system interactions and network activities. The features extracted using the DS technique include API call patterns, file interaction metrics, system interaction patterns, domain communication logs, and network traffic profiles. These features are selected to align with the attack techniques identified in the taxonomy to detect malware that abuses CPLS for C&C purposes. Further exploration and analysis of these attack techniques can be found in Section 2.6.

Table 2.6: Taxonomy of Abusive Techniques Targeting CPLS

Technique	Abused CPLS	Description	Reference	Occurrences
Steganography	Dropbox, Google Cloud Messaging (GCM), Discord, Facebook, Twitter, Imgur, ImgBB, Evernote	Hiding communication between bots and C&C servers within legitimate-looking files, such as images or videos, and then transmitting them through cloud storage services	[95, 136, 51, 41, 129, 91, 149, 55, 57]	9

Continued on next page

Table 2.6 – continued from previous page

Technique	Abused CPLS	Description	Reference	Occurrences
Encoding	weibo.com, Twitter, Facebbok, Google Docs, Instagram, YouTube, Yahoo, Quora, GitHub, outdrive, Dropbox, Google Drive, OneDrive, GCM, Microsoft TechNet, Pastebin, Mega Facebook Instance Messenger, Alibaba Cloud	Using encoding to make communications more difficult to detect, such as base64 encoding used to obfuscate C&C commands or data sent to the C&C server	[118, 51, 99, 83, 155, 157, 121, 90, 74, 140, 108, 96, 84, 167, 160, 67, 112, 116, 139, 94, 162, 120, 35, 132, 56, 91, 146, 114, 62, 104, 97, 39, 33, 109, 52, 127, 92, 70, 143, 168, 163, 138]	40

Continued on next page

Table 2.6 – continued from previous page

Technique	Abused CPLS	Description	Reference	Occurrences
Cryptography	Microsoft Outlook, Gmail, Dropbox, CloudMe, YouTube, Google Drive, OneDrive, Pastebin, Google Docs, Slack, Twitter, Facebbok, Weibo, GCM, pCloud, Yandex Disk, Github, Mega, Alibaba Cloud	Using encryption to secure communications between bots and C&C servers hosted on cloud-based services	[111, 149, 99, 41, 51, 63, 106, 101, 37, 32, 48, 66, 67, 91, 102, 54, 61, 134, 160, 49, 133, 70, 143, 168, 163, 40]	21

Continued on next page

Table 2.6 – continued from previous page

Technique	**Abused CPLS**	**Description**	**Reference**	**Occurrences**
Fraudulent account creation	Teams, OneNote, Outlook, Discord, Pastebin, Facebook, Twitter	Creating fraudulent accounts on cloud-based services to use as a disguise for C&C servers or to store botnet-related data	[149, 167, 55, 57, 52, 59]	6

Continued on next page

Table 2.6 – continued from previous page

Technique	Abused CPLS	Description	Reference	Occurrences
Botmaster's credentials or hard-coded tokens	Twitter, Telegram, Evernote, Slack, GitHub, Pastebin, Google+, CloudMe, GCM, Google Docs, Dropbox, OneDrive, Google Drive, Gmail, Microsoft Exchange Web Services, pCloud, Yandex Disk, Mega, Alibaba Cloud	Obtaining botmaster credentials or hard-coded tokens to access cloud-based services, which can then be used to host C&C servers or store botnet-related data	[155, 157, 80, 100, 47, 166, 35, 38, 147, 159, 142, 93, 49, 44, 50, 53, 32, 171, 91, 140, 108, 96, 84, 135, 34, 77, 64, 102, 54, 47, 166, 36, 89, 43, 35, 88, 58, 113, 116, 65, 134, 114, 62, 104, 144, 37, 45, 133, 143, 168, 163, 141, 115, 40]	51

Continued on next page

Table 2.6 – continued from previous page

Technique	Abused CPLS	Description	Reference	Occurrences
Compromised Accounts	Facebook Instance Messenger, Facebook, Twitter, Outlook, GCM, Dropbox, Microsoft Exchange Web Services Google Drive	Compromising legitimate user accounts on cloud-based services to use as a disguise for C&C servers or to store botnet-related data	[100, 161, 80, 106, 81, 97, 39, 171, 91, 144, 169, 168, 95]	13
COM hijacking	Outlook, Gmail, Dropbox	Hijacking COM components on an infected system to communicate with a C&C server hosted on a cloud-based service	[106, 101, 36, 89]	4

Continued on next page

Table 2.6 – continued from previous page

Technique	Abused CPLS	Description	Reference	Occurrences
AI-powered C&C	Twitter	Employing neural networks for dynamic addressing, identifies attacker accounts via avatars, and embeds command in tweets via hash collisions and data augmentation,	[165]	1
Process injection	Evernote	Injecting malicious code into legitimate processes to communicate with a C&C server hosted on a cloud-based service and evade detection	[159]	1
ComSpec environment variable	Dropbox	Modifying the ComSpec environment variable to point to a command shell on a cloud-based service to execute commands and communicate with a C&C server	[64]	1

Evaluating the frequency and prevalence of attack techniques used to exploit CPLS as C&C channels is crucial for understanding their popularity and effectiveness. Based on the information presented in Table 2.6, these attack techniques have been observed and documented across multiple references, indicating their use to varying degrees. However, obtaining precise quantitative data on the frequency of these techniques can be challenging due to the clandestine nature of cybercriminal operations and the ever-evolving threat landscape. Consequently, we acknowledge that these references may not encompass the entire spectrum of cyber threats associated with abusing CPLS. Despite this, the comprehensive list of references presented in Table 2.6 and the count of abuses by year for each technique illustrated in Figure 2.2 emphasize the importance of paying attention to these techniques.

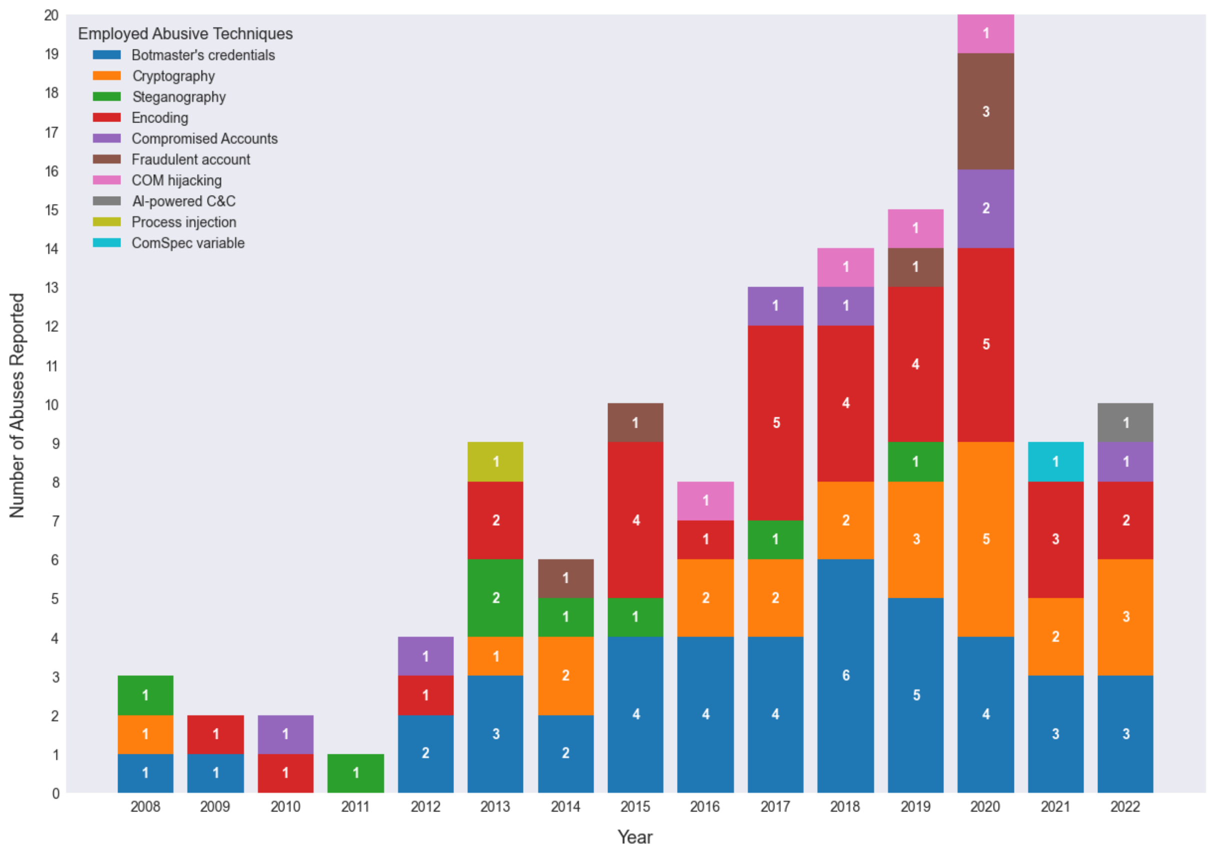


Figure 2.2: Number of abuses by year for each technique.

Prevalent Techniques

Encoding and the use of botmaster's credentials or hard-coded tokens are the most frequently employed techniques across platforms and over time. Their prevalence could stem from their relatively straightforward implementation and their ability to blend with normal network traffic, which likely make them a preferred choice for many cybercriminals.

Evolution and Trends

The complexity of techniques used by attackers seems to be increasing over the years. While earlier years primarily witnessed encoding and the use of botmaster credentials or hard-coded tokens, later years show a rise in more advanced techniques, such as encoding, cryptography, compromised accounts, and even AI-powered C&C.

Increasing Complexity

In recent years, attackers have increasingly employed a combination of techniques, and often abuse multiple CPLS platforms concurrently to increase the resilience of the C&C infrastructure and complicate tracking and takedown efforts. For instance, we have observed cases where cybercriminals combine encoding and cryptography techniques to enhance the concealment of C&C operations. This trend implies that attackers are continually advancing their tactics in order to evade detection. The combination of techniques increases the difficulty of detection, suggesting that multi-layered security strategies are essential for effective defense.

2020: A Year of Escalated CPLS Abuse for C&C

As depicted in Figure 2.2, the year 2020 witnessed a substantial increase in the abuse of CPLS platforms for C&C operations. This escalation could potentially be attributed to the global COVID-19 pandemic, which resulted in a significant increase in online activities. The expanded online presence created a fertile ground for cybercriminals, offering them a larger pool of potential targets.

In response to RQ1, our analysis reveals that attackers employ a diverse range of techniques to abuse CPLS as C&C channels. These techniques, as categorized in our taxonomy (Table 2.6), include steganography, encoding, cryptography, fraudulent accounts, use of botmaster's credentials or hard-coded tokens, compromised victims' accounts, COM hijacking, AI-powered C&C, process injection, and modifying system variables. Many of these techniques align with tactics outlined in the MITRE ATT&CK framework, underscoring their relevance to real-world threats. Significantly, the AI-powered C&C technique proposed in the DeepC2 framework is a novel addition not yet represented in MITRE ATT&CK, highlighting the continual evolution of attacker strategies. In terms of frequency, encoding and the use of botmaster's credentials or hard-coded tokens emerge as the most prevalent techniques across platforms and over time, likely due to their ease of implementation and ability to blend with legitimate traffic. However, the complexity of techniques appears to be increasing, with recent years witnessing more advanced and combined techniques like encoding, cryptography, compromised accounts, and AI-powered C&C. The year 2020, in particular, saw a significant escalation in CPLS abuse for C&C, potentially linked to the increased online activity during the COVID-19 pandemic. This surge underscores the adaptive nature of cybercriminals in exploiting global events. In conclusion, attackers leverage a wide array of techniques, from simple encoding to sophisticated AI, to abuse CPLS as C&C channels. The frequency and complexity of these techniques are on the rise, with attackers often combining multiple methods to enhance stealth and resilience. This trend emphasizes the critical need for robust, multi-layered defense strategies to counter the evolving threat landscape of CPLS abuse.

2.4.2 RQ2: Which Types of CPLS Are Targeted for Abuse?

Correlation between Platform User Base and Abuse Occurrence

A positive correlation is observed between the number of users on a platform and the number of malware instances targeting that platform. Platforms with a larger user base, such as Google Docs, Dropbox, Twitter, Google Docs, Google Drive, YouTube, and Facebook, tend to report more abuse instances. The logic behind this correlation is intuitive; a larger user base provides a wider pool of potential victims for cyber attacks. Based on the conducted statistical analysis, there appears to be a positive association between the number of users on a platform and the number of abuse occurrences. Specifically, the slope of the regression line was calculated to be approximately 1.647—see Figure 2.3. This suggests that for each increase of one unit (one billion users) in a platform's user count, the occurrence of abuse tends to increase by 1.647 units on average.

Platform Specific Trends and Corporate Usage

Table 2.7 presents a summary of malware occurrences grouped by CPLS. An occurrence is either an in-the-wild reported malware incident or a proof-of-concept (PoC) malware developed by security researchers. The data showcase the diverse range of CPLS exploited as C&C communication channels. CPLS like Dropbox, Google Docs, Google Drive, Outlook, and OneDrive, widely utilized in corporate environments, are particularly appealing to threat actors due to their integral roles in data storage, collaboration, and communication. Our analysis reveals that these platforms have been exploited using a variety of techniques over the years, including botmaster credentials, encoding, cryptography, and compromised accounts. The popularity of these platforms in both the public sphere and corporate settings, coupled with their common usage for file storage and sharing, makes them attractive targets for abuse as C&C infrastructure. Threat actors can leverage the trust and ubiquity associated with these platforms

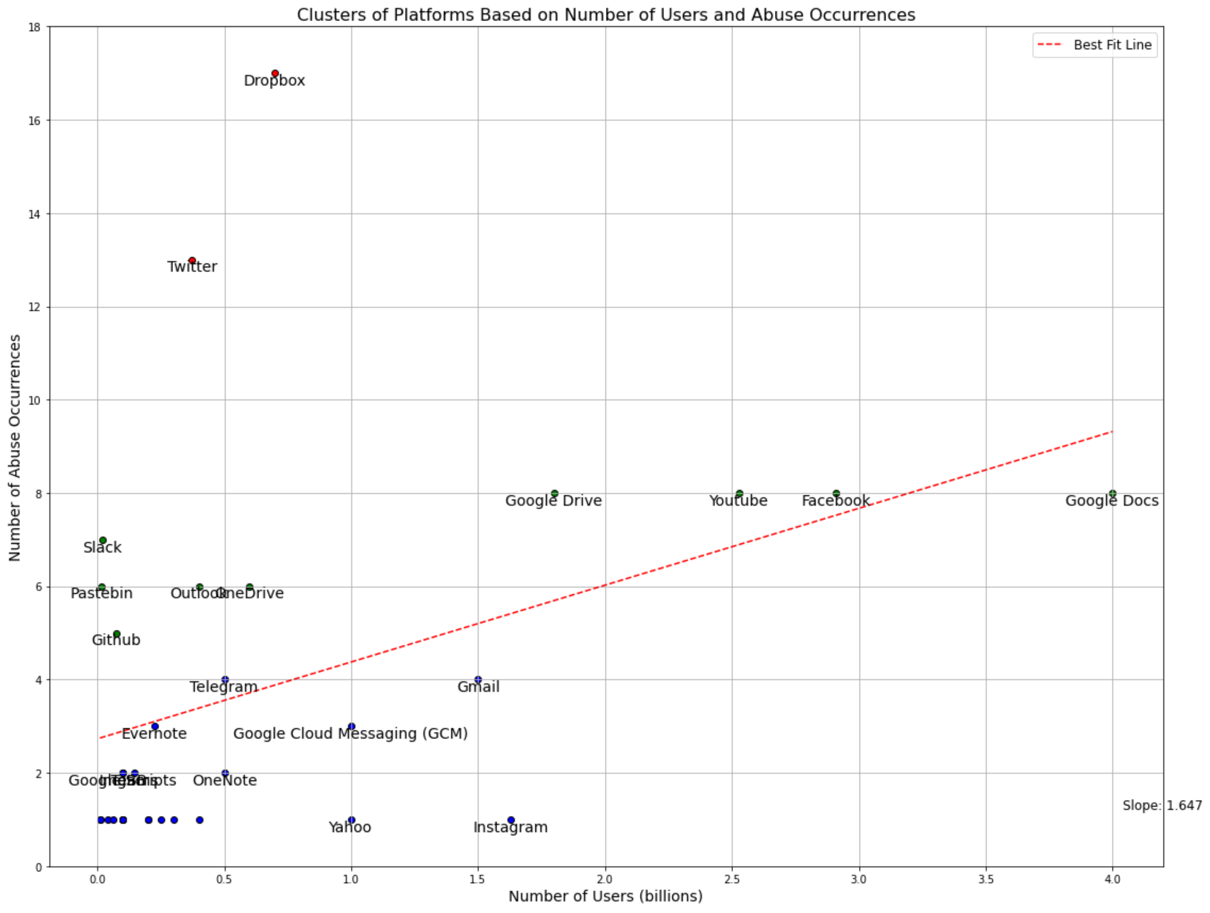


Figure 2.3: Correlation between number of users and abuse occurrences.

to blend their malicious activities with legitimate traffic, making detection more challenging. Moreover, the use of these platforms for C&C communication allows threat actors to maintain a persistent and stealthy presence within compromised networks. By abusing the built-in features and APIs of these platforms, malware can retrieve commands, exfiltrate data, and maintain a bidirectional communication channel with the attacker, all while evading traditional security measures. The prevalence of these platforms in corporate environments further amplifies the risk of abuse. As employees rely on these platforms for their daily work, malware that abuses them for C&C communication can more easily infiltrate and propagate within the organization.

Table 2.7: Malware occurrences grouped by CPLS: a reported abuse is for in-the-wild malware incident, whereas a PoC (proof of concept) is malware created by security researchers.

		Incident Categories	
CPLS	**Occurrences**	**Reported Abuse**	**PoC**
Dropbox	17	[51, 34, 135, 77, 64, 102, 54, 166, 143]	[91, 88, 58, 169, 36, 40, 89, 43]
Twitter	13	[41, 51, 138, 155, 167, 47]	[121, 149, 111, 99, 157, 35, 165]
Google Docs	8	[140, 56, 96, 84, 167, 108, 160, 67]	—
Google Drive	8	[146, 114, 61, 62, 104, 168, 167]	[91]
Youtube	8	[48, 139, 112, 116, 94, 66, 67, 63]	—
Facebook	8	[161, 115, 90, 63, 92]	[95, 136, 111]
Slack	7	[141, 142]	[49, 44, 50, 53, 42]
Outlook	6	[106, 83, 81, 97, 39]	[59]
OneDrive	6	[116, 113, 65, 134, 143]	[91]
Pastebin	6	[140, 109, 52, 108, 127, 66]	—
Github	6	[41, 120, 141, 35, 132]	—
Gmail	4	[101]	[45, 37, 158]
Telegram	4	[38, 93, 147, 115]	—
Google Cloud Messaging (GCM)	3	—	[129, 171, 91]

Continued on next page

Table 2.7 – continued from previous page

		Incident Categories	
CPLS	**Occurrences**	**Reported Abuse**	**PoC**
Evernote	2	[51, 104, 159]	—
Google Scripts	2	[140, 108]	—
Discord	2	—	[55, 57]
ImgBB	2	[51, 167]	—
Microsoft TechNet	1	[33]	—
CloudMe	1	[32]	—
Imgur	1	[51]	—
Google+	1	[109]	—
Facebook Instance Messenger	1	—	[92]
File.io	1	[141]	—
Yahoo	1	[162]	—
Quora	1	[162]	—
Microsoft Teams	1	—	[59]
Microsoft OneNote	1	—	[59]
Google Sites	1	[63]	—
Instagram	1	[83]	—
pCloud	1	[133]	—
Yandex Disk	1	[133]	—
Alibaba Cloud	1	[163]	—

Continued on next page

Table 2.7 – continued from previous page

		Incident Categories	
CPLS	**Occurrences**	**Reported Abuse**	**PoC**
Mega	1	[143]	—
Exchange Web Services (EWS)	1	[144]	—

Anomaly Analysis

There are notable anomalies, wherein some platforms report low abuse instances despite maintaining large user bases. For example, Instagram, with over 1.6 billion users, recorded merely a single instance of abuse as indicated in Figure 2.3. Conversely, platforms such as Dropbox, Twitter, Slack, Pastebin, and Telegram are outliers with a smaller user base but a high number of abuse occurrences. This can be attributed to the following factors:

- Platform usage patterns: Different platforms cater to different user behaviors and usage patterns. For instance, platforms like Dropbox and Slack, despite having smaller user bases, often attract business and professional users. This can make them more appealing targets for abusers seeking access to sensitive information.
- Security measures: The level of security measures implemented by the platforms plays a crucial role in the number of abuse occurrences. Platforms with robust security features and stringent user verification processes might experience fewer instances of abuse, even with a large user base.
- Platform features and accessibility: platforms that offer a broader range of functionality and ease of use tend to attract more abusers, which provides more opportunities and tools for abusers to exploit.

- Anonymity: platforms that offer a certain level of anonymity seem to attract abusers. This could be because anonymity can make it easier for abusers to avoid identification.

Emerging Threats

The advent of AI-powered C&C attacks in 2022 highlights the continually increasing sophistication of cybercriminal tactics. While currently only observed on Twitter, it is conceivable that threat actors may leverage AI in abusing more platforms like Dropbox, Google Drive, Outlook, and OneDrive in the future. COM hijacking, though less prevalent currently, has been observed in several instances and poses a significant risk due to its ability to persist in a system undetected. This anticipatory threat underscores the need for continuous research and proactive defense strategies in cybersecurity.

In response to RQ2, our analysis reveals that a wide range of CPLS are being targeted for abuse as C&C infrastructure by malware. Platforms with large user bases, such as Google Docs, Dropbox, Twitter, Google Drive, YouTube, and Facebook, tend to experience more abuse occurrences. This is likely because a larger user base provides attackers with more potential victims. However, some anomalies exist where platforms with smaller user bases like Dropbox, Twitter, Slack, Pastebin and Telegram report a disproportionately high number of abuse incidents. This may be attributed to factors such as the types of users on the platform (e.g. business users being targeted), varying security measures across platforms, the range of features and ease of exploit on each platform, and the level of anonymity provided. Particularly, platforms commonly used in corporate environments like Google Docs, Dropbox, Google Drive, Outlook and OneDrive are major targets. Attackers abuse these trusted, ubiquitous platforms to hide malicious traffic, persist stealthily in networks, and infiltrate organizations as employees use them in daily work. A concerning emerging threat is the use of AI in attacks, currently seen on Twitter but likely to spread to other platforms. In summary, attackers are targeting a diverse set of popular CPLS platforms to abuse as C&C infrastructure, with corporate platforms being prime targets. Continuous research and proactive defenses are needed as attack sophistication grows.

2.4.3 RQ3: What Countermeasures Have Been Proposed to Detect the Abusive Use of CPLS as C&C Infrastructure?

Several techniques have been proposed for detecting abuse of cloud-native platforms as C&C communication channels. Six of these detection strategies have been implemented for use in a computer environment, while only one has been specifically implemented for use on the Android OS. These techniques primarily focus on three approaches: rule-based, behavior tree-based, and ML-based detection methods.

Rule-based Detection

Kartaltepe et al. [121] introduced a dual-level abuse detection system, comprising client-side and server-side mechanisms. On the client side, they defined three features to detect botnets: self-concealment, unusual network traffic, and questionable provenance. They posited that connections to social media platforms might be considered suspicious unless driven by human actions. To differentiate between legitimate users and bots, they employed behavioral biometrics, user input reactions, and graphical user interface (GUI) interactions as detection metrics. On the server side, they operated under the assumption that any communication with social media platforms that involves textually encoded messages or posts is suspicious. They utilized the J48 decision tree algorithm to categorize input messages, differentiating between Base64 or Hexadecimal-encoded text and regular language content. However, these detection methods come with certain constraints: (i) they don't offer real-time detection since the tests were conducted in a post-analysis lab environment, and (ii) crafty adversaries could potentially bypass detection by using image-steganography techniques to embed malicious commands within posts.

Vo et al. [164] developed the API Verifier, a tool that uses CAPTCHA verification to authenticate social media account access based on MAC addresses. This tool determines whether an API call is made by a human or a bot, adding a protective layer against automated bot activities. However, the API Verifier proposed by Vo et al. has several drawbacks. First, the CAPTCHA verification system can be vulnerable to relay attacks, potentially enabling botnets to circumvent the verification. Second, depending solely on MAC addresses for user identification might fall short in situations where users switch between multiple devices or when MAC addresses are easily spoofed.

Ghanadi et al. [107] delve into the study of stego-botnets that utilize steganographic images on online social networks for C&C operations. They proposed a system named SocialClymene, designed to detect covert botnets in social networks using stego-images. The system has a negative reputation subsystem that analyzes images shared by social network users and calculates a reputation score for each user based on their history of participating in suspicious activities. The goal is to recognize botnets by analyzing the behavior of the users and their history of involvement in suspicious activities. Nevertheless, these detection approaches have certain limitations: (i) the system might not detect new botnets lacking a history of suspicious behavior, and (ii) it can be challenging to accurately identify a user's reputation, especially in dynamic online settings where reputations can shift swiftly.

Behavior Tree-based Detection

Yuede et al. [118] introduced a behavior tree-based detection framework designed to identify social bots through host activity monitoring. This framework is composed of three main components: a host behavior monitor, a host behavior analyzer, and a detection methodology. To construct and analyze a suspicious host behavior tree, they crafted a social botnet called wbbot. Their design incorporated sample collections from two distinct sources: real-world social bots [161, 115, 155, 157] and social bot malware samples curated by researchers [136]. After running and evaluating this collection of social bots over a specified duration, they created a template library. This library was subsequently used to determine the highest similarity value when compared to the suspicious behavior tree. Upon the behavior tree's completion, the tree edit distance method was utilized to to calculate its similarity to the template, resulting in the final detection result. Nevertheless, this detection methodology presents a notable limitation: a substantial false positive rate of 29.6%. Additionally, this system's could potentially bypass if attackers deploy a multi-process strategy or spread malicious behaviors over varied time intervals.

Burghouwt et al. [85] proposed a causality detection mechanism designed to pinpoint Twitter-based C&C channel communication. This is achieved by measuring the correlation between user activity and network traffic. The authors operate under the assumption that any network traffic directed to the OSN, which isn't a result of human actions like specific keystrokes or mouse movements, should be considered suspicious. The causality detection approach utilizes a time frame that begins immediately after a user event. This helps differentiate network activities triggered by genuine user actions from those initiated by bots. However, this detection approach has certain limitations. First, legitimate API calls, which are used for routine automated polling, might be mistakenly identified as suspicious. Second, the primary metrics used to determine the time gap between user activity and network requests might not be universally accurate. This is because different machines and operating systems can have varied delay times and performance attributes. Lastly, sophisticated bots could potentially circumvent this detection by observing user activities and executing commands in response to user-initiated events.

ML-based Detection

Ji *et al.* [117] undertook a detailed assessment of several previously studied abusive social bots. The authors incorporate spatial and temporal correlations to identify patterns of the social bots. They gathered source code, builders, and execution patterns from established social botnets, including Twitterbot (Singh [156]), Twebot (Burghouwt *et al.* [86]), Yazanbot (Boshmaf *et al.* [82]), Nazbot (Kartaltepe *et al.* [121]), wbbot (Ji *et al.* [119]), and fbbot. Their objective was to scrutinize the techniques these bots employ to bypass current detection systems. Drawing from their analysis, they proposed a detection strategy using 18 features. This strategy uses spatial correlations to recognize patterns across child processes, like multiple bots on one IP, and temporal correlations to study event sequences for behavior patterns. However, their focus on just six bots could limit the study's applicability to other bots.

Ahmadi *et al.* [71] introduced a method designed to detect Android applications that abuse Google Cloud Messaging (GCM) for C&C communication. By adopting the Flowdroid tool [78] to extract GCM flows and identify GCM callbacks, they trained an ML model using features like GCM registration ID, sender ID, and the type of GCM message. Their findings indicate that GCM flow features can effectively distinguish malicious applications. Nonetheless, the method might be vulnerable to evasion tactics like obfuscation, which adversaries might use to mask GCM flows. Additionally, its applicability is constrained, as it only works for Android OS and is not applicable in Windows OS environments.

In response to RQ3, which asks about the countermeasures proposed to detect the abusive use of CPLS as C&C infrastructure, our analysis reveals a diverse set of approaches spanning rule-based, behaviour tree-based, and ML-based detection methods. These countermeasures, implemented across both computer and Android platforms, employ various techniques such as host and server-side detection, GCM flow analysis, API verification with CAPTCHA, negative reputation scores, and causality detection. However, it is crucial to acknowledge the limitations of these countermeasures, as detailed in Table 2.8. Common challenges include the potential for adversarial evasion, the difficulty in accurately identifying reputation in dynamic online settings, and the constraints in applicability across different platforms. Continuous research and improvement are necessary to enhance the effectiveness and accuracy of these detection mechanisms, ultimately strengthening the security of CPLS and preventing their abuse as C&C infrastructure.

Table 2.8: Comparative Overview of CPLS Abuse Detection Mechanisms in Related Works: categorized proposed detection mechanisms by whether they are host-based or server-based, and identifies each as either proactive (P) or active (A) in their approach to anomaly detection. Proactive approaches are those that aim to prevent abuse before it occurs, while active approaches are those that respond to abuse as it happens

Reference	Host-Based	Server-Based	Detection Mechanism	Limitation
Yuede et al. [118]	P	-	Constructs a behavior tree based on host activity and calculates its similarity to a designated template using the tree edit distance	Evade detection by performing random time delays between different behaviors of social bots.
Kartaltepe et al. [121]	P	P	Host-based: Uses behavioral biometrics (keyboard, mouse input) to identify suspicious connections. Server-based: Classifies incoming messages using the J48 decision tree algorithm	Lack of empirical data and evaluation and relies heavily on case studies.
Ahmadi et al. [71]	P	-	Modifies the Flowdroid tool for flow analysis to extract Google Cloud Messaging (GCM) flows, which are then used as features in a machine learning model to identify malicious Android apps	Limited to Android OS, challenging to identify if obfuscation or hiding is implemented on GCM flows

Continued on next page

Table 2.8 continued from previous page

Reference	Host-Based	Server-Based	Detection Mechanism	Limitation
Vo et al. [164]	-	A	Adopts a CAPTCHA verification technique to authenticate social media accounts by utilizing the MAC address, distinguishing between API calls from human users or automated bots	Can be bypassed using relay attacks, MAC addresses can be easily spoofed or changed by attackers
Ghanadi et al. [107]	-	A	Proposes SocialClymene, a system that uses stego-images to detect covert botnets in social networks by analyzing user behavior and calculating a reputation score based on suspicious activities	May not detect new botnets with no previous history of suspicious behavior, challenging to accurately assess user reputation in dynamic online environments
Burghouwt et al. [85]	P	-	Measures the causal relationship between network traffic and human activity to distinguish between user-triggered and bot-originated network events	Can be evaded by mimicking human activity such as mouse clicks or keyboard strokes, may produce false positives from legitimate APIs used for automate
Ji et al. [117]	P	-	Incorporates spatial and temporal correlations to identify patterns of behavior that may be indicative of social bot activity	Focuses on only six social malicious bots and may not generalize to other botnets

2.5 Challenges and Directions for Future Research

While some progress has been made in detecting CPLS abuses as C&C channels, several challenges remain, and there are directions for future research that can further enhance the detection mechanisms.

2.5.1 Quantity and Quality of Datasets

One of the challenges is the lack of dedicated datasets specifically designed for detecting malware that abuses CPLS platforms as C&C channels. Future research should focus on developing comprehensive and representative datasets that cover a wide range of CPLS platforms and abuse techniques, thereby enabling researchers to effectively evaluate and enhance detection mechanisms.

2.5.2 Quantity and Quality of Datasets

One of the challenges is the lack of dedicated datasets specifically designed for detecting malware that abuses CPLS platforms as C&C channels. Future research should focus on developing comprehensive and representative datasets that cover a wide range of CPLS platforms and abuse techniques, thereby enabling researchers to effectively evaluate and enhance detection mechanisms.

It is important to note, however, that obtaining precise quantitative data on the frequency of these techniques can be challenging due to the secretive nature of cybercriminal operations and the constantly evolving threat landscape.

2.5.3 Emergence of AI-Powered C&C

As discussed in Section 2.6.8, AI-powered C&C emerged in 2022, indicating that attackers are utilizing AI to abuse Twitter as a C&C infrastructure. Although its current observation is limited to Twitter, it is conceivable that threat actors may employ AI in their abuses across various platforms. This highlights the importance of future research focusing on developing defenses against AI-powered C&C.

2.5.4 Deep Packet Inspection (DPI)

DPI plays a crucial role in identification the of malicious activities within cloud environments. Its effectiveness, however, is often hindered by tactics employed by attackers such as payload encryption, protocol encapsulation, and obfuscation. These tactics can mask the presence of malicious commands and communications, including those that involve CPLS platforms used as C&C infrastructure. Furthermore, the complexity of CPLS traffic, combined with the large volume of encrypted data and the legitimate use of these platforms, adds another layer of challenge to the application of DPI. This complexity requires more sophisticated analysis and detection techniques, which can discern subtle patterns of abuse amidst the large volume of normal traffic. In light of these challenges, future research should explore innovative approaches to overcome these obstacles and distinguish between legitimate and malicious uses of CPLS platforms without violating user privacy.

2.5.5 Evasion Tactics

Attackers continuously evolve their tactics to evade detection mechanisms. Future research should investigate advanced evasion techniques, such as obfuscation, polymorphism, and steganography. Robust countermeasures need to be developed to effectively detect and mitigate these tactics.

2.5.6 Cross-Platform Abuse Detection

Cybercriminals often abuse multiple platforms simultaneously and launch coordinated attacks. Research could focus on the development of cross-platform detection techniques that can identify and correlate malicious activities across different CPLS platforms.

2.5.7 Collaboration and Information Sharing

CPLS abuses often transcend individuals and require collaboration and information sharing among different stakeholders, including cloud providers, security researchers, and law enforcement agencies. Future research should explore ways to facilitate collaboration and information sharing to enhance the detection and mitigation of CPLS abuses.

2.6 Abuse Attack Techniques

2.6.1 Steganography

This section examines the advancements in abusing CPLS platforms through image and text steganography. Steganography as a covert channel has garnered substantial attention from academia as demonstrated by proofs of concept, and from threat actors as observed in actual hacking incidents. Our review uncovered multiple instances in which CPLS were abused through the application of steganography techniques as C&C.

CPLS as a Primary C&C Communication Channel

ELISA (Elusive Social Army) [95] is an OSN-based botnet that abuses Facebook as a covert C&C channel, disseminating its messages through victims' social media accounts. ELISA establishes a covert channel using a Unicode steganography technique, inserting non-printable characters and invisible glyphs into user-generated messages posted on OSNs. These hidden elements are not displayed during the rendering process, making them hard to detect.

Stegobot [136], another OSN-based botnet, abuses Facebook as its main C&C communication channel. To establish this covert communication path within the social network, Stegobot employs a JPEG image steganography scheme known as YASS [56]. By doing so, it effectively embeds hidden information within digital images, thereby augmenting its stealth and evasion capabilities.

Punobot [129], a mobile botnet targeting Android systems, cleverly utilizes Google Cloud Messaging (GCM) for its C&C operations. Punobot employs a steganography technique that transforms the original command messages into different ones to evade detection by both users and the push notification service (PNS) server.

HAMMERTOSS [41] is a backdoor developed by the APT29 threat-active community. HAMMERTOSS was well crafted to cover the tracks of APT29 using several techniques, including building an algorithm that produces regular Twitter handles and the steganographic embedding of images with malicious commands. HAMMERTOSS also uses a domain generation algorithm (DGA) to create new Twitter handles. Whenever the malware creates a new handle, the corresponding Twitter page is fetched, and the page is searched for a particular pattern, which represents the encrypted C&C URL. HAMMERTOSS utilizes GitHub and cloud storage services as the primary C&C communication channels to transmit commands and relay stolen data from compromised networks. To obtain the malicious commands, HAMMERTOSS implements steganography techniques through images that contain encrypted malicious commands. Once

HAMMERTOSS obtains the GitHub URL from its regular Twitter account, it visits the page, retrieves the steganographic images containing encrypted data, and upon successful connection and downloading, begins the decryption process to extract the actual command and perform the intended malicious operation.

RegDuke [51] malware employs steganography and cryptography techniques to hide data in PNG images. The developer of RegDuke misuses Dropbox by hosting steganography images containing an encrypted malicious command for covert C&C operations. The backdoor lists the Dropbox directory corresponding to a clientId (the compromised machine) and downloads the embedded PNG files. When images from the Dropbox directory are downloaded, the RegDuke code scans through all the image pixels and extracts data from them. The hidden data are specifically extracted from the implant, and the content is decrypted using an advanced encryption standard (AES) key that is hard-coded in the payload, which can be one of the following weapons of attack: Windows executable, Windows DLL, or PowerShell script.

This backdoor only resides in memory and relies on steganography to hide data in images using a technique called “least significant bit”, which stores and combines 8 bits of data into a total of 24 bits of data per pixel: 8 for red, 8 for green, and 8 for blue. RegDuke consists of a loader and a payload, both components written in .NET. RegDuke persists by using a WMI consumer named MicrosoftOfficeUpdates. The threat technical report [51] identifies four different primary variants of the RegDuke loader between August 2017 and June 2019. The first version was not obfuscated and had the encryption key hard-coded in the code. Later versions directly read the encryption key from the Windows registry and employed various obfuscation techniques, such as flattening the control flow or using .NET Reactor, a commercial obfuscator.

In 2012, Shuai et al. [153] proposed a malware called SUbot that leverages blog websites for the creation of covert channel communications to evade detection. The author of SUbot implemented steganography and cryptography strategies, using RC4 to hide a secret message and appending the ciphertext message to the end of a JPG file. The modified JPG file containing an executable command was then uploaded to a blog. When the infected mobile device visits the URL of the blog site, it downloads and retrieves the plaintext commands from the JPG image.

CPLS as a Redirector to C&C Domain

Twitter has been abused by the HAMMERTOSS malware as a mapper for the malicious URL. This is achieved by searching for a tweet with a URL and a hashtag; the URL points to the location of the C&C website with one or more images, while the hashtag allows HAMMERTOSS to extract encrypted instructions from an image file.

PolyglotDuke [51] is malware that is used by APT29 cyber espionage as a downloader for the MiniDuke backdoor. It uses various public websites, such as Twitter, Imgur, ImgBB, or Evernote public notes, to retrieve and decode the C&C URLs. It moreover relies on image steganography for its C&C communication channel.

Insights

Our review of steganography in the context of abusing the CPLS as C&C channels presents several important insights:

- Steganography has become a popular method among threat actors for maintaining a covert communication channel, often serving as the primary C&C channel. Cases like ELISA, Stegobot, Punobot, HAMMERTOSS, and RegDuke demonstrate the use of different steganographic techniques—from Unicode and image steganography to the least significant bit technique—effectively hiding malicious commands and evading detection.
- Online social networks (OSNs), along with other online platforms, have become common targets for misuse. As seen with ELISA and Stegobot using Facebook and HAMMERTOSS exploiting GitHub, these platforms provide a vast, noisy

environment in which malicious activities can blend in, thus enhancing the effectiveness of the steganography.

- The sophistication and complexity of steganographic techniques have been increasing. Advanced encryption standard (AES) keys, domain generation algorithms (DGAs), and obfuscation techniques like .NET Reactor have been used in conjunction with steganography to further hide and protect the malicious payloads.
- Mobile platforms are not immune to these types of attacks. As shown by Punobot, steganography can also be employed in attacks targeting mobile devices, in this case using Google Cloud Messaging as a C&C channel.
- Steganography is not only used for direct C&C communications but also serves as a means of redirecting to C&C domains. HAMMERTOSS and PolyglotDuke represent cases where steganography was used to decode the C&C URLs.

2.6.2 Encoding

Encoding, typically employed to ensure data integrity and confidentiality, is paradoxically being weaponized to aid C&C operations. This technique obscures malicious communications, making them harder to detect and interpret. Our study reveals that all identified malware variants using encoding are designed to evade detection and maintain the secrecy of their operations. The following example illustrates how encoding is used to support confidentiality in the context of CPLS abuse.

CPLS as a Primary C&C Communication Channel

Nazario [138, 155] identified a Twitter account being maliciously used as a C&C operations coordinator. The botmaster set up a Twitter account named “upd4t3” to conceal and disseminate malicious C&C messages, demonstrating how encoding enhances the confidentiality of malicious communications:

1. The botmaster broadcasts Base64-encoded commands via tweets.
2. Bots retrieve these encoded messages through an RSS feed.
3. Bots then decode the messages to execute the hidden commands.

This encoding process supports confidentiality in several ways:

- It obfuscates the true nature of the commands, making them incomprehensible to casual observers and many automated detection systems.
- The encoded messages blend with the normal encoded web content, making malicious traffic less distinguishable.
- The decoding step adds a layer of protection, as the true content cannot be understood without first decoding the message.

This encoding technique significantly enhances the confidentiality of C&C communications. It makes the malicious messages more difficult to identify and interpret without prior knowledge of the specific encoding scheme used, thereby supporting the overall goal of maintaining secrecy in botnet operations.

CPLS as Redirector to C&C Domain

ESET researchers discovered a Korplug variant [56] used by the Advanced Persistent Threat group, the Winnti group. This variant exploited publicly shared Google Docs files to extract its C&C address from a seemingly legitimate block of text using the well-known DZKS and DZJS delimiter strings.

An APT group named Turla used a trojan backdoor housed in a Firefox extension to retrieve the C&C URL. Turla APT is a cyberespionage group with over a decade of activity. In the analyzed sample of this threat report [83], the C&C domain was obscured using an encoding technique in a well-known celebrity's Instagram comment. More specifically, the Firefox extension parsed the photo comments on the official Instagram account and calculated a custom hash value. If the comment hash value equaled 183, a regular expression `(?:\\u200d(?:#|@)(\\w)` was then executed against the matched comment to obtain the C&C shortened URL path, which subsequently led to the actual C&C URL.

Stantinko [112] is a remotely configured cryptomining module that utilizes most of the compromised machine's resources. Stantinko malware interacts with its mining pool indirectly, through an IP address collected from YouTube video descriptions. These C&C IP addresses are concealed in a hexadecimal format in the video description string.

Janicab malware has exploited YouTube [116] to obtain the C&C IP. To retrieve the actual C&C IP address, the malware navigates through comments on specific YouTube videos. If a format such as *"our 50380702789658th psy anniversary"* is matched, the obscured number from the matched comments is extracted and converted to the actual C&C IP address.

Palo Alto [162] documented two similar malware variations, CONFUCIUS_A and CONFUCIUS_B, which abused legitimate websites to retrieve C&C server IP addresses instead of using DNS lookups. To illustrate how the IP for the C&C domain is decoded, CONFUCIUS_A and CONFUCIUS_B use Yahoo and Quora to circumvent traditional mechanisms, parsing for keywords between specific phrases previously posted by the threat actor. The malware then decodes the interim phrase by substituting words for components of an IP address. A basic lookup table is used to decode and derive the C&C IP address.

Palo Alto also analyzed a new variant of the SunOrcal malware family [120], which exploited the GitHub service by employing a de-obfuscation process of Base64 and XOR decoding to extract a C&C server. Strategically, this sample was configured to connect to a specific file hosted on the GitHub repository to extract the data leading to the real C&C server. Text between two particular strings within this target file is parsed and encoded to derive the C&C URL.

The Scote backdoor is malware, discovered by Palo Alto Networks [109], that misuses legitimate third-party web platforms like Pastebin and Google+ as covert C&C communication channels. The Scote payload establishes a connection to these legitimate platforms' URLs to retrieve data and parse for specific commands to be executed by the compromised machine.

Insights

Our study of encoding in the context of abusing the CPLS as C&C channels presents several important insights:

- Encoding serves a dual purpose in these attacks. It not only provides a method of concealing C&C communications within legitimate service traffic, but also, it allows threat actors to hide the actual location of their C&C servers.
- Many of the observed malware variants, including the Korplug variant, Janicab, and CONFUCIUS_A and CONFUCIUS_B, demonstrate creative use of regular expressions and encoding techniques to retrieve C&C server addresses.
- Threat actors exhibit adaptability in their encoding methodologies, leveraging the specific characteristics of each platform to embed encoded data. For instance, using Instagram photo comments or YouTube video descriptions to conceal encoded commands or C&C server addresses.

2.6.3 Cryptography

CPLS as a Primary C&C Communication Channel

Shuai et al. [153] introduce SUbot, a botnet specifically designed for mobile platforms. SUbot leverages micro-blogging for evasion purposes and implements cryptography using RC4 to conceal a secret message. This message, which contains a malicious command, is appended to the end of a JPG file. Following this, the author of SUbot uploads the modified JPG file, which now contains an executable command, to a blog. When an infected mobile device visits the URL of the site, it downloads the image and retrieves the encrypted commands from the JPG file.

Sebastian et al. [149] designed a botnet to conceal malicious commands within a tweet. In their proposed experiment, the Botmaster covertly injects encrypted malicious commands into tweets to evade security measures. The format of the malware-infected tweet follows the `#keyword command` pattern, where the value of the `command` is encrypted. The bot then retrieves the tweet from the Botmaster's fake Twitter accounts and extracts the provided encrypted command. This command is then decrypted and interpreted, launching an attack on the infected machine.

He et al. [111] constructed prototypes of web test automation rootkit (WTAR) bots: Fbbot, Twbot, and Wbbot. These bots abuse Facebook, Twitter, and Weibo, respectively, as C&C infrastructures. The web test automation (WTA) technique, initially developed for automating browsers and testing websites, can perform actions such as filling in forms, reading data from web pages, and clicking elements on web pages. The WTAR-based technique has been leveraged to mimic typical user behaviors on an OSN. To better conceal the communication, thus making detection more difficult, both the botmaster and the bot use the data encryption standard (DES) to encrypt the commands and their corresponding execution results using a predefined key.

The Turla backdoor is a type of malware that exploits a victim's Outlook mailbox, using it as a transport layer for its C&C operations, receiving commands, and exfiltrating data. To better conceal the malicious commands and execution results, the designer of the Turla backdoor takes advantage of a previously opened session of the victim to gain access to the default mailbox profile [106]. In addition, the backdoor creator employs the MISTY1 symmetric encryption method. This method is used to create specially crafted PDF documents that contain either encrypted malicious instructions or confidential information, which are then attached to the compromised Outlook account's inbox.

ESET researchers [101] investigated a backdoor variant called ComRAT, which abuses Gmail as a covert C&C channel to receive commands and exfiltrate data. The ComRAT botnet authorizes the Gmail account using credentials embedded in the malware payload. ComRAT then connects to the Gmail HTML web interface to search for an email with a specific subject. Once matched, the email attachment is downloaded and decrypted via the AES-256 algorithm to extract the malicious command for execution. The executed command result is then encrypted using RSA-2048 and emailed to the threat actor, often hosted on GMX or VFEmail. To ensure persistence, ComRAT developers rely on a technique known as COM hijacking to tamper with the Windows registry, causing the ComRAT botnet to execute every time the user logs in.

RegDuke malware [51] utilizes cryptographic techniques to secure the data transmitted between the botmaster and the bots. Once an image is downloaded from Dropbox, the bot loops over the image pixels to extract data. The bot then decrypts the information with an AES key hard-coded in the payload to retrieve malicious commands.

The CloudMe platform was exploited by CloudAtlas malware [32] as a covert communication channel. To make the transmitted messages between the attacker and the victim undetectable, the designer of the CloudAtlas malware employs cryptography with AES and data compression with LZMA techniques. The malware is configured to include encrypted contents such as the C&C's CloudMe URL, a username and password, and two folders on the CloudMe server for storing malicious commands and uploading the victim's data. The malware downloads the encrypted malicious commands uploaded by the threat actor, decrypts them, and then interprets them. It then uses the same mechanism to upload the result back to the server.

CPLS as Redirector to C&C Domain

Dong et al. [99] proposed a botnet design that combines QR codes, Twitter search, domain generation algorithms (DGAs), cryptography (AES and RSA), and Tor. In their design, Twitter is utilized as a mapper to the C&C web server. The Twitter search engine is queried for a particular keyword to locate the botmaster's post, which includes a QR code image. Upon scanning the QR code, the encoded combination of three components—the C&C web server address, the hard-coded token, and the RSA public key is retrieved. The recovered RSA public key is then used to encrypt the data to be sent to the C&C web server.

The Casbaneiro botnet [48] utilizes YouTube, a legitimate website, to store its C&C server domains. The malware operator of Casbaneiro embeds an encrypted C&C web server address in a false Facebook or Instagram URL within the description of a specific YouTube video. This serves to redirect compromised machines to the threat actor's C&C infrastructure.

Insights

The use of cryptography techniques plays a crucial role in malware and botnet operations. It is primarily used for two purposes: protecting C&C communications and hiding malicious commands or data within legitimate-looking content. Our review of cryptography in the context of abusing the CPLS as C&C channels presents several important insights:

- The combined use of cryptography along with the abuse of popular CPLS platforms for C&C channels illustrates the increasing sophistication of botnet strategies.

These platforms provide an additional layer of security and reliability, making it more difficult to detect botnet activities due to the reputable nature of these services.

- Botmasters frequently embed encrypted malicious commands into digital content, such as images or social media posts. These commands are later extracted and decrypted by bots for execution as demonstrated in cases like SUbot, ComRAT, and RegDuke.
- Cryptography is often combined with other techniques to enhance security and evasion capabilities. For instance, the ComRAT botnet merges cryptography with COM hijacking to ensure persistent execution.

2.6.4 Botmaster Login Credentials or Hard-Coded Token

CPLS as Primary C&C Communication Channel

Nazario [138, 155] identified a Twitter account, named “upd4t3”, that was being maliciously utilized as a C&C operations hub. The botmaster used this account to distribute Base64-encoded commands via tweets. The bot then fetched these commands using the RSS (really simple syndication) feed, decoded them, and executed them, thus establishing a covert C&C communication route.

Singh et al. [157] developed an OSN-based botnet named SocialNetworkingBot that leverages Twitter for its C&C communications. The malware author used authenticated official Twitter accounts to post tweets containing disguised commands to be interpreted by the bots. The botmaster posted tweets using approximately 300 pre-defined keywords to facilitate fetching the tweets. Bots queried the Twitter search engine for specific keywords to retrieve these tweets from the botmaster account, which served as a rendezvous point. The bots then fetched and executed the malicious commands embedded in these tweets.

Telecrypt [38, 115] is a type of ransomware that abuses Telegram's instant messaging API for its C&C infrastructure. Initially, the botmaster sets up a Telegram bot, which utilizes TeleCrypt ransomware to enable interaction between the threat actors and the compromised machine.

Upon infecting a machine, the ransomware employs Telegram's 'sendMessage' method. This method enables the bot to transmit messages to a chat thread associated with a specific number. This number is hard-coded into the ransomware's body, and is used to deliver a "successful infection" report back to the attackers. This is performed using the following request:

```
https://api.telegram.org/bot<token>/sendmessage?chat_id=<chat>
&text=<computer_name>_<infection_id>_<key_seed>
```

The request includes the following parameters:

- `<chat>`: Represents the chat number with the cybercriminal.
- `<computer_name>`: The name of the infected computer.
- `<infection_id>`: A unique identifier for the infection.
- `<key_seed>`: A number used as the basis for generating the file encryption key.

Following the transmission of this information, the trojan scans the infected machine's hard drives, looking for files with specific extensions. Once such files are identified, they are encrypted bytewise. This encryption is achieved by adding each file byte to the corresponding key bytes, employing a simple encryption algorithm.

ESET researchers have uncovered a unique malicious toolkit, TeleBot.AA [93], developed by the TeleBots APT group. This toolkit was specifically designed to abuse Telegram for C&C operations, utilizing the Telegram Bot API. Each version of the backdoor contains individual hard-coded credentials, indicating the presence of a Telegram Messenger account for each sample. The attacker communicates with the compromised systems through private Telegram chats, enabling the exchange of commands and retrieval of results.

Palo Alto Networks has detailed in an article [147] a malicious Android trojan known as “TeleRAT”. This trojan abuses the Bot API of Telegram to carry out C&C activities. The TeleRAT spyware infiltrates the victim’s Telegram app by masquerading as a legitimate application that promises to provide a count of profile visitors. The APKs of the malicious app contain hard-coded Telegram bot API keys, enabling them to periodically send beaconing signals at precise intervals of 4.6 s and await specific commands from the attacker.

TrendMicro [104] recently discovered a malware variant named BKDR_VERNOT.A that employs a clever technique to avoid detection. This malware utilizes legitimate services like Evernote, a web-based note-taking application, as a proxy server to establish communication with the botmaster. Upon successfully infecting the victim’s machine, BKDR_VERNOT.A drops a .DLL file that injects itself into a genuine process, generating legitimate network traffic in order to evade detection by security solutions. The payload of BKDR_VERNOT.A leverages hard-coded official Evernote account credentials to connect to saved notes, enabling the backdoor to retrieve malicious commands and upload stolen data to a designated drop-off zone.

TrendMicro researchers discovered and analyzed a backdoor called “SLUB”, which abuses three legitimate platforms—Slack, GitHub, and File.io—to establish its C&C infrastructure [141]. The threat actor sets up a Slack workspace and a GitHub account to facilitate SLUB backdoor C&C operations. To interact with the Slack API, the designer of SLUB embeds two hard-coded authentication tokens. The operator of SLUB uploads malicious commands to GitHub snippets, which the backdoor then retrieves and executes. The results of these commands are subsequently uploaded to both Slack and File.io.

TrendMicro researchers discovered a variant of the SLUB backdoor four months after the first version was identified [142]. This evolved version discontinued using GitHub for its C&C operations, instead opting to fully integrate Slack workspaces as the covert C&C communication channel between the malware and its controller. The updated SLUB variant employs the same authentication approach as the previous version between the backdoor and its controller. Once a victim machine is infected by the SLUB backdoor and attempts to join a Slack workspace, a new channel titled `<use_name>-<pc_name>` is created. If the SLUB threat actor wants to execute a malicious command, they post the message to a victim-specific channel in Slack. The SLUB backdoor on the victim's machine then responds by parsing and executing the requested command.

The CloudAtlas malware exploited the CloudMe platform as a covert communication channel [32]. To secure the messages transmitted between the attacker and the victim and to make them undetectable, the designer of CloudAtlas employed a combination of AES cryptography and LZMA data compression techniques. The malware was configured to contain encrypted contents including the C&C's CloudMe URL, a username and password, and two folders on the CloudMe server designated for storing malicious commands and uploading the victim's data. Specifically, the botmaster uploads the encrypted malicious commands to the account, which the malware then downloads, decrypts, and interprets. The malware employs the same mechanism to upload the results back to the server.

Zhao et al. [171] introduced C2DM, an Android botnet architecture that abuses Google's Cloud to Device Messaging (C2DM) service for the dissemination of C&C commands. C2DM, a cloud-based push notification service for Android developers, was exploited in this botnet to eliminate the need for a direct connection between the botmaster and the bots. By blending the malicious bot traffic with the legitimate C2DM traffic from other Android apps, this botnet can covertly transmit its traffic. Furthermore, Zhao et al. [171] highlighted that many existing botnet detection strategies struggle to detect this type of push-like mobile botnet, as both malicious bots and legitimate applications use official push servers to receive updates.

Chen et al. [91] developed CloudBot, an enhanced version of a push-styled botnet [91]. CloudBot is a hybrid structured smartphone botnet (combining hierarchical and P2P structures) that abuses ten cloud-based push services (GCM, JPush, XGPush, ZYPush, GeXinPush, and Airbop) as a C&C downstream channel. Meanwhile, prominent cloud services like Dropbox, OneDrive, and Google Drive are used as a C&C upstream channel. CloudBot's design allows botmasters to send commands to bots in the form of legitimate push traffic via cloud-based push services. The bot then uploads the extracted data using cloud-based storage services. A significant advantage of using push services in a mobile botnet is the ability to avoid direct communication with C&C servers for command retrieval. CloudBot accesses the cloud storage services by embedding the account information and access token into a push message, which is then delivered to bots using the push services. Cloud-based push services can only support text messages. To satisfy this requirement and evade detection during command transmission, CloudBot incorporates three levels of obfuscation: encryption, encoding, and high-order mimic functions.

CPLS as Redirector to C&C Domain

Chen et al. [91] designed and implemented an Android-based push-styled botnet that abuses Google's message push service, GCM. This service is used as a mapper to direct users to the C&C URL domain to carry out malicious activities. For a more detailed discussion, refer to Section 2.6.4.

Insights

Our review of botmaster login credentials or hard-coded tokens in the context of abusing the CPLS as C&C channels presents the following insight:

- Many types of malware, like Telecrypt, TeleBot.AA, TeleRAT, BKDR_VERNOT.A, and CloudAtlas, use hard-coded credentials or tokens to authenticate themselves to these online services. These credentials or tokens are often embedded directly into the malware, enabling it to automatically and seamlessly connect to the service.

2.6.5 Compromised Accounts

CPLS as a Primary C&C Communication Channel

ELISA [95] is an OSN-based botnet that enables the botmaster to communicate with their bots by leveraging unaware user interactions and concealing its messages within victims' posts. ELISA constructs an overlay network, which interacts with typical users to deliver messages across the entire botnet. To ensure confidentiality, the C&C communication is safeguarded by using encryption and signatures between the botmaster and their bots.

As outlined in Sections 2.6.3, the Turla malware [106] leverages the victim's already opened session to access the default mailbox profile. Consequently, communication between the compromised Outlook email and the botmaster's email takes place through either an encrypted malicious instruction or encrypted PDF attachments.

CPLS as Redirector to C&C Domain

The Koobface botnet [161] [115] is a social botnet that primarily relies on popular social networking sites like Facebook and Twitter for propagation. To achieve this, the botnet spams legitimate social network users and takes them through multiple layers of URL redirection to evade blocklist detection. The obfuscation process involves using blogs, RSS feeds, and shortened URLs to resolve and connect to the C&C URL.

2.6.6 Fraudulent Account

CPLS as a Primary C&C Communication Channel

Sebastian et al. [149] described a method that abuses Twitter as the main C&C communication medium by concealing the malicious commands within tweets. As discussed in Section 2.6.3, the bot retrieves the tweet from the botmaster's fake Twitter accounts, decrypts the embedded command, and executes it.

2.6.7 Component Object Model (COM) Hijacking

CPLS as a Primary C&C Communication Channel

Researchers at ESET conducted a thorough analysis of the Turla backdoor malware [106]. Turla exploits the victim's Outlook mailbox for C&C communication, receiving instructions, and exfiltrating data. After infecting the host, Turla utilizes the legitimate messaging application programming interface (MAPI) to interact with Outlook, granting complete control over the target mailbox and utilizing additional MAPI functionalities. To ensure persistence and concealment, the operators employ the COM to modify the Windows registry. This Microsoft technology enables developers to manipulate objects in various applications. The communication between the botmaster and the infected bot takes place through email, utilizing specially encrypted PDF attachments to transmit operational commands and exfiltrated information between compromised Outlook emails and the botmaster's email.

2.6.8 Artificial Intelligence (AI)-Powered C&C

CPLS as a Primary C&C Communication Channel

Wang et al. proposed DeepC2 [165], an innovative AI-powered C&C framework designed to address the challenges of covert communication on OSNs. Their approach leverages a neural network model for dynamic addressing, allowing the malware to identify the attacker's accounts through the extraction of feature vectors from avatars. By utilizing hash collision and easy data augmentation techniques, the attacker embeds commands within normal-looking tweets, ensuring covert communication while avoiding detection by OSN platforms. The framework leverages Twitter Trends as a rendezvous point.

2.6.9 Process Injection

CPLS as a Primary C&C Communication Channel

As discussed in Section 2.6.4, the BKDR_VERNOT.A malware [104] abuses the Evernote platform for its malicious operations. The threat actor's Evernote account credentials, hard-coded into the malware binary, enable the bot to fetch malicious codes stored in the notes on the service. Upon execution, BKDR_VERNOT.A drops a specifically formatted .DLL file, or component, into the computer's temporary directory and injects itself into the legitimate process of Windows Explorer. This .DLL file initiates the backdoor's actual operations.

2.6.10 COMSPEC Environment Variable

CPLS as a Primary C&C Communication Channel

The BoxCaon backdoor, discovered by researchers at Checkpoint [64], abuses Dropbox as its C&C infrastructure. The backdoor employs the COMSPEC environment variable, which typically points to the command-line interpreter (cmd.exe), to execute malicious commands. The procedure works as follows: the attacker uploads files or commands to the Dropbox folder. The malware then fetches this folder and downloads all its contents to a working directory. If the file 'c.txt', which contains the attacker's command, is found in this working directory, the backdoor executes the command using the COMSPEC environment variable. The results of the command execution are then uploaded back to Dropbox, and the command is deleted.

2.6.11 Exploit Multiple Processes

CPLS as a Primary C&C Communication Channel

Yuede et al. [117] developed Wbbot, a social bot that exploits Twitter for C&C operations. The bot is designed to divide malicious behaviors into multiple processes, aiming to evade behavior detection mechanisms. Each process is dedicated to a specific malicious behavior, allowing them to exhibit benign behavior. Such a decentralized approach can make it challenging for traditional detection methods to accurately identify the overall malicious behavior.

2.7 Research Gap Analysis

Despite the range of defense mechanisms proposed in the literature, this chapter analysis revealed a deficiency in detection systems for effectively identifying the abuse of the CPLS as C&C infrastructure. This research gap was identified through this SLR. Our work contributes to the research in this area by introducing a refined taxonomy of abusive strategies. This taxonomy categorizes the methods used by threat actors to abuse CPLS as C&C infrastructures. In addition, this chapter elaborates on existing detection methods, examines their effectiveness, and highlights their limitations. The identified research gap underscores the need for further research and development of more robust and comprehensive detection systems.

2.8 Summary

In this chapter, we have conducted a comprehensive literature review on the abuse of Cloud-based Public Legitimate Services (CPLS) as Command and Control (C&C) channels by botnets. Our Systematic Literature Review (SLR) has identified and categorized ten different types of abusive attacks, providing a more granular understanding of the threat landscape compared to previous surveys. Through our research, we have discovered that the abuse of CPLS as C&C channels is not limited to a particular type of service; instead, it is pervasive across a variety of platforms, including cloud storage, social media, business communication platforms, and more. Botnets have evolved significantly over the years, adopting diverse methods to exploit CPLS for C&C communication, such as steganography, cryptography, COM hijacking, process injection, and COMSPEC environment variable exploitation. Our findings also highlight the increasing sophistication and complexity of attack techniques employed by threat actors. The combination of multiple techniques, such as encoding and cryptography, and the concurrent abuse of multiple CPLS platforms, demonstrate the adaptability and resilience of modern botnets. Moreover, the emergence of AI-powered C&C attacks underscores the continual evolution of cybercriminal tactics. The countermeasure strategies proposed to date, both in computer and Android environments, involve detecting anomalies in user behavior, CAPTCHA verification, reputation score calculation, and causality measurement between user activity and network traffic. However, each of these strategies comes with its own limitations, emphasizing the need for continued research and development of more robust and effective detection and prevention mechanisms. Our SLR also highlights the correlation between platform user base and abuse occurrence, with platforms like Google Docs, Dropbox, Twitter, Google Drive, YouTube, and Facebook reporting more abuse instances due to their larger user bases. However, anomalies exist, with some platforms like Instagram reporting low abuse instances despite their substantial user base, while others like Dropbox, Twitter, Slack, Pastebin, and Telegram experience high abuse occurrences despite a smaller user base. In conclusion, this literature review has provided a comprehensive understanding of the current state of research on the abuse of CPLS as C&C channels. It has identified the diverse range of attack techniques employed by threat actors, the increasing sophistication and complexity of these techniques, and the limitations of existing countermeasures. The insights gained from this review underscore the critical need for ongoing research and innovative solutions to effectively detect, prevent, and mitigate the abuse of CPLS as C&C channels by botnets.

Chapter 3

CPLS Malware and Benignware Dataset: Collection and Preprocessing

3.1 Introduction

In this chapter, we provide background information on the dataset and preprocessing steps utilised in our research on detecting the abuse of Cloud and Public Legitimate Services (CPLS) for malicious command and control (C&C) infrastructure. Our dataset comprises malware files in the Portable Executable (PE) format, which is widely used in the Windows operating system and is a common vehicle for malware delivery.

First, we discuss the rationale behind our decision to concentrate on the PE file format, highlighting the dominance of Windows in the desktop operating system market and the prevalence of PE files among submissions to VirusTotal, a leading malware analysis platform. Next, we describe the process of identifying the exact CPLS domains that have been exploited by malicious actors for C&C communications, as well as the trends in CPLS domain abuse over the years, which underscores the growing threat posed by this type of attack. Finally, we outline the steps involved in collecting and curating our dataset, including the use of VirusTotal Intelligence Agent, Cuckoo sandbox, and various sources for both malicious and benign samples. We also explain the measures taken to ensure a balanced dataset and maintain data quality throughout the preprocessing phase.

3.2 Background

The dataset in this work include only the Portable Executable (PE) file format. The PE file is a standard format for executables, object files, and dynamic-link library (DLL) in the Windows operating system [20]. It contains information about the structure and layout of an executable file, including the entry point, the code and data sections, and the dependencies of the program.

The focus on the PE file format is driven by two primary factors. First, its widespread use in the Windows environment is highlighted in Figure 3.1, showing that Windows has maintained a dominant market share in desktop operating systems (OS) from 2013 to 2023. Second, Figure 3.2 indicates that the PE file format is the most commonly submitted among all file types on VirusTotal, which is a widely used repository that allows users to submit suspicious files and URLs for scanning by multiple antivirus engines and other security tools.

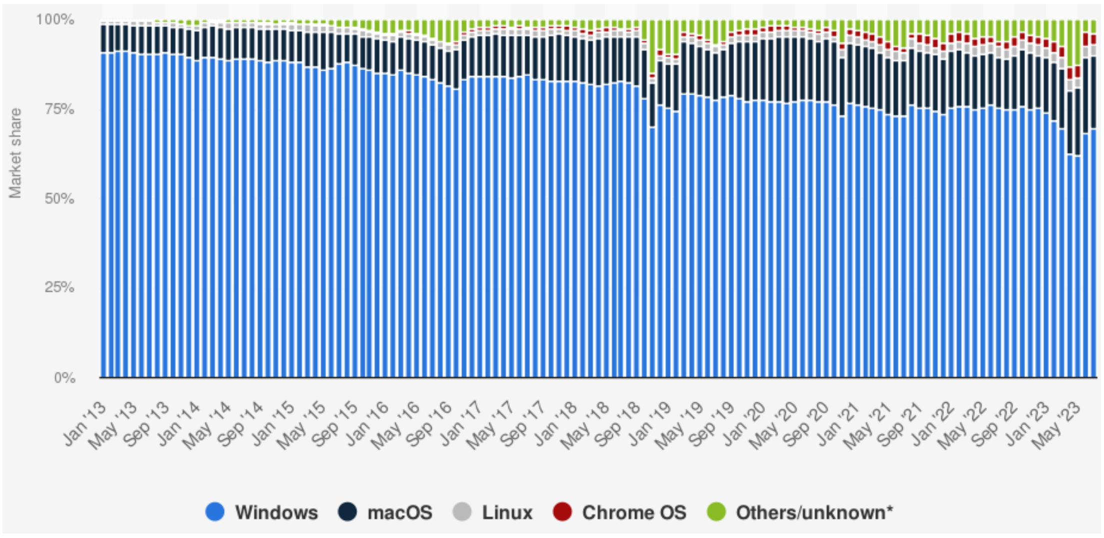


Figure 3.1: Global distribution of market share among different OS used in desktop PCs [7].

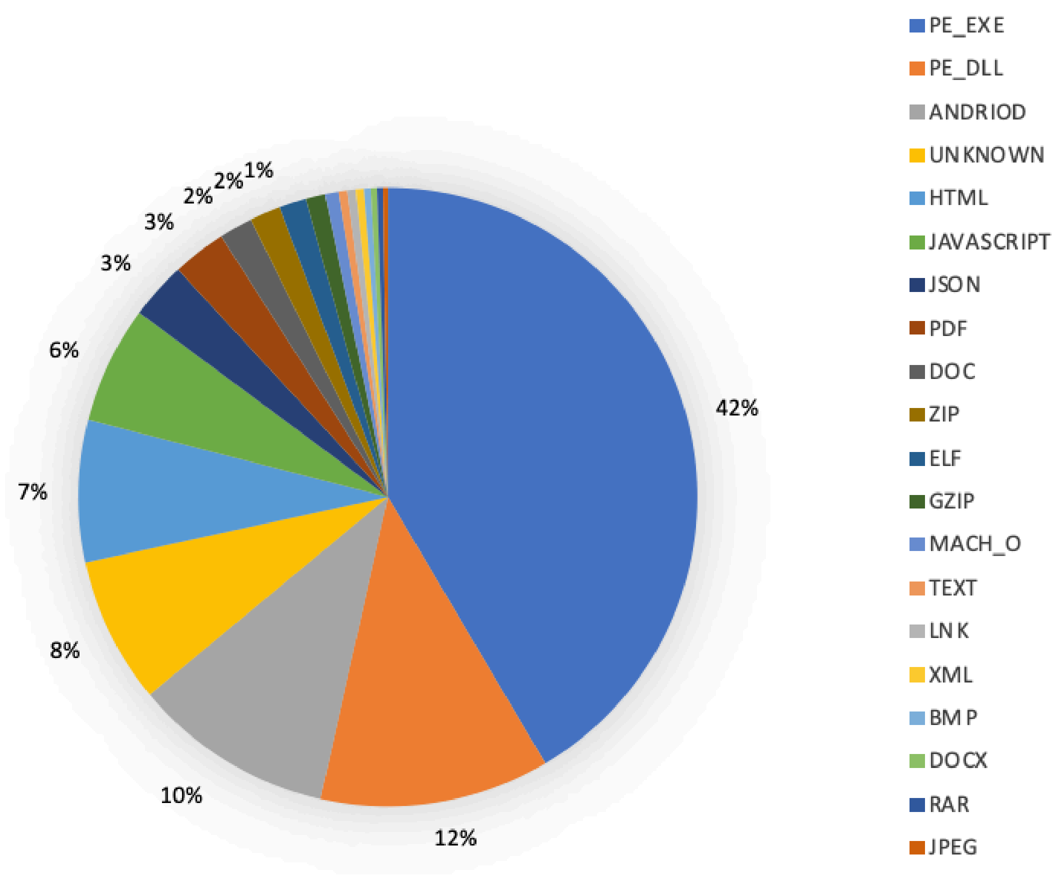


Figure 3.2: VirusTotal submission by file format [26].

In Section 3.2.1, we summarise the PE file format [20], including details on the structure and content of PE files, which are commonly used to run programmes on Windows systems.

3.2.1 PE File Format

The PE file format is one of the most prevalent types of executable files used in malware. PE files have a certain structure and contain various fields that provide information about the file [16]. In the context of malware CPLS abuse detection, specific fields within the PE file format can be crucial for distinguishing between malicious and benign files. The entry point of a PE file is a crucial component that specifies the starting address where the program begins execution. It's defined in the Optional Header of the PE file structure and is represented by the AddressOfEntryPoint field. When the operating system loads the executable, it uses this address to determine where to begin executing the program's code. In the context of malware analysis, the entry point is significant as it's often the first part of the code that malware authors modify to insert their malicious routines. As depicted in Figure 3.3, the PE file format comprises several fields: COFF Header, Optional Header, Import Table, Export Table, Resource Directory, Relocation Table, Debug Information, and Section Table. Instead of offering an exhaustive description of every field, we focus on the ones most pertinent to abuse detection, as outlined in Table 4.2. For instance, the number of sections and the characteristics of the DLL have been shown to be useful in differentiating between malware and benign files. The optional header provides information such as the linker version and the sizes of code and data, which can also be valuable for classification purposes. Moreover, the section table contains crucial data regarding the file's sections, including code, initialized data, imports, exports, and resources.

While our analysis considers various properties of PE files, several specific attributes are particularly valuable for malware detection and could be candidates for additional analysis:

- Entry Point (AddressOfEntryPoint): Often modified by malware to insert malicious code at the start of execution.
- Import Address Table (IAT): Lists external functions used by the executable, which can indicate suspicious API calls.
- Section characteristics: Unusual section names or permissions can be indicative of malware.
- File alignment and section alignment: Abnormal values might suggest attempts to hide malicious code.

- Timestamp: Can be used to identify compile time, which might reveal anachronisms in malware.
- Digital signatures: Absence or invalidation of signatures can be suspicious.
- Resources: Unusual or hidden resources might contain malicious payloads.
- DLL characteristics: Certain flags can indicate potentially malicious behavior.

These properties are particularly interesting because they are often manipulated by malware authors to evade detection or to facilitate malicious activities. Future research could delve deeper into analyzing these specific attributes, potentially uncovering new patterns or indicators of CPLS abuse.

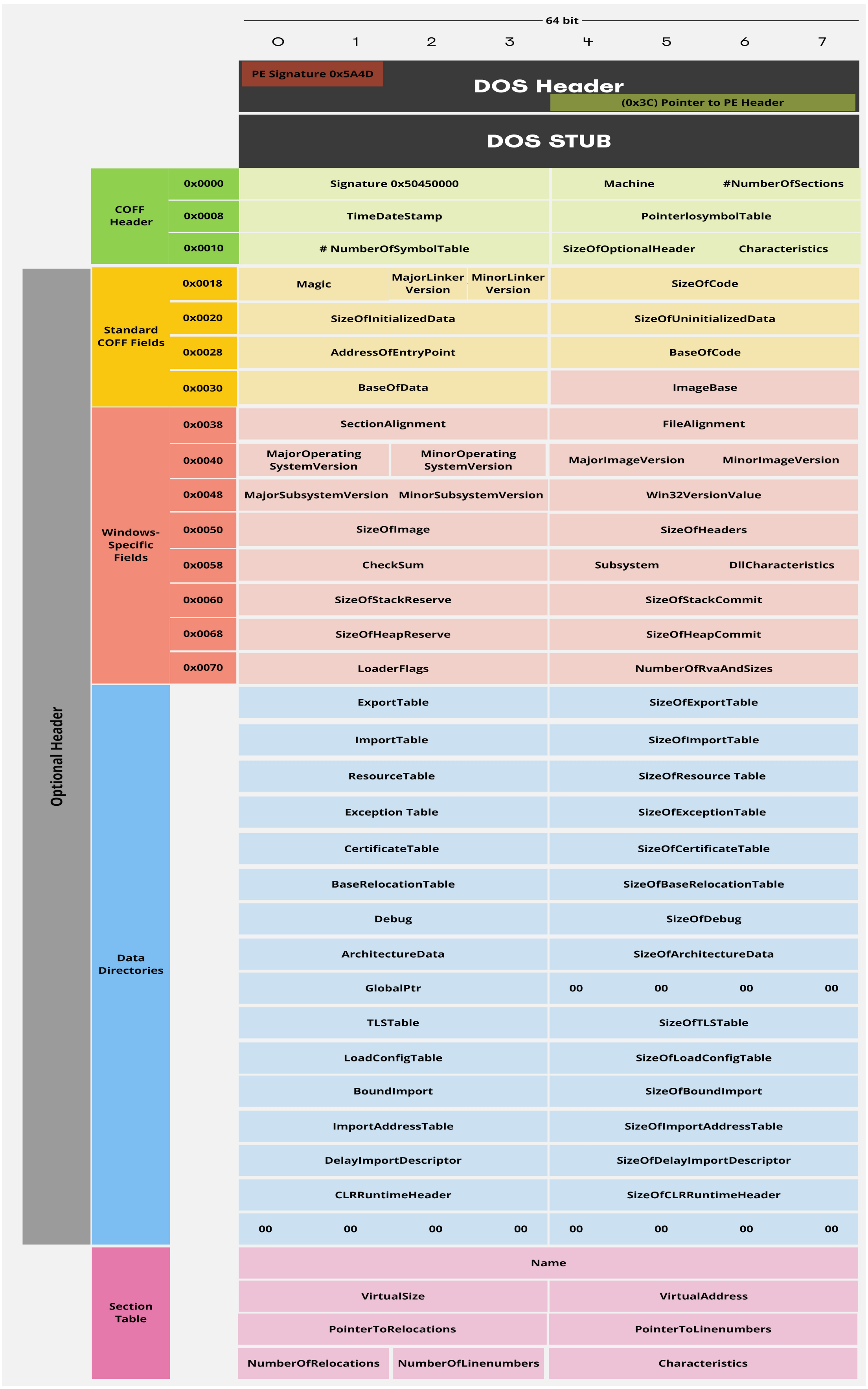


Figure 3.3: Detailed diagram of the structure of a Portable Executable (PE) file [16, 21].

3.3 Dataset

3.3.1 CPLS Domain Identification

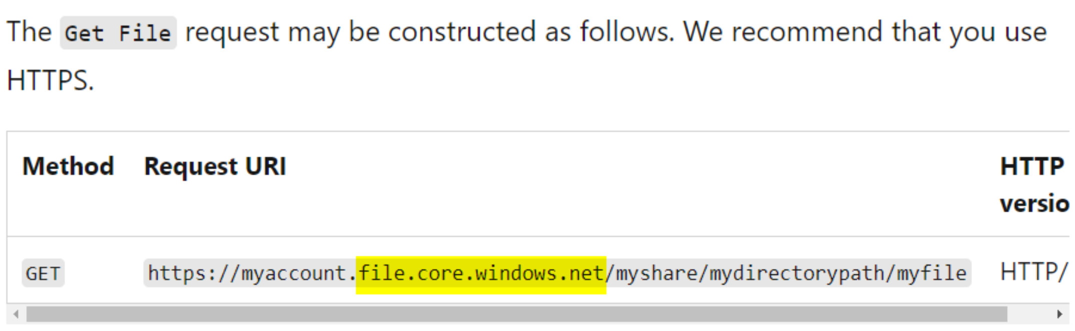
Identifying the precise CPLS domains that have been abused for C&C communication is crucial for two reasons. First, having a comprehensive list of abused CPLS domains enables us to create a more representative dataset for our analysis. By including malware samples that specifically communicate with these domains, we can ensure that our dataset captures the characteristics and behaviors of malware that abuse CPLS for C&C purposes. Second, it allows us to focus our research on the specific domains that are most commonly exploited by malicious actors. By targeting these domains, we can develop more effective detection tailored to the current threat landscape.

To identify the exact domains that malware connects to, we first conducted a thorough review of the abused CPLS services identified in the literature review chapter, particularly in Table 2.6 and Table 2.7. These tables showcase the diverse range of CPLS platforms abused as C&C infrastructure. Based on the CPLS services identified in these tables, we searched for the corresponding domains and included them in Table 3.1. This process involved reviewing API documentation and executing actual connections to these services to determine the precise domains utilized by each CPLS platform. Importantly, we identified these CPLS domains before collecting the malware samples for our dataset. By having a predefined list of abused CPLS domains, we were able to filter and retain only the malware samples that specifically communicated with one of these domains during data collection process, see Section 3.3.4 . This targeted approach ensured that our dataset was highly relevant to our research focus.

Table 3.1: CPLS domains abused as C&C infrastructure.

Names of the CPLS Domains		
api-content.dropbox.com	api.twitter.com	docs.google.com
mail.google.com	chat.google.com	classroom.googleapis.com
sheets.googleapis.com	slides.googleapis.com	storage.googleapis.com
mail.google.com	smtp.gmail.com	onedrive.com
dropbox.com	twitter.com	github.com
pastebin.com	raw.githubusercontent.com	api.twitter.com
dev.twitter.com	publish.twitter.com	apps.twitter.com
status.twitter.com	youtube.com	twitter.com
docs.google.com	script.google.com	translate.google.com
storage.googleapis.com	spreadsheets.google.com	api.slack.com
app.slack.com	slack.com	gmail.com
hotmail.com	outlook.com	amazonaws.com
azure.com	portal.office.com	discord.com
telegram.com	instagram.com	OneNote.com
teams.microsoft.com	Evernote.com	publish.twitter.com
apis.google.com	imap.gmail.com	m.youtube.com
aws.amazon.com	file.core.windows.net	blob.core.windows.net
onedrive.live.com	cf-my.sharepoint.com	graph.microsoft.com
outlook.office365.com	alibabacloud.com	yandex.com/disk/
imgbb.com	cloudme.com	imgur.com
file.io	pcloud.com	mega.io

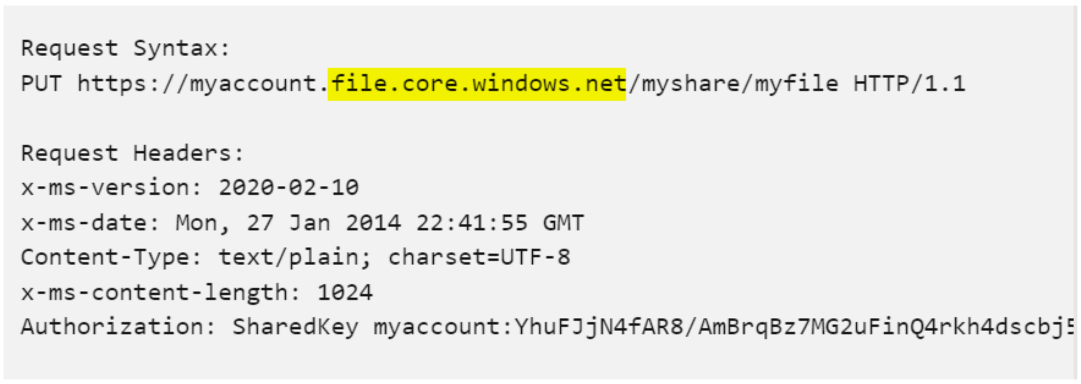
For example, Figures 3.4 and 3.5 indicate the domains of the Azure service for download as a GET request and for upload as a PUT request, respectively.



The `Get File` request may be constructed as follows. We recommend that you use HTTPS.

Method	Request URI	HTTP versio
`GET`	`https://myaccount.file.core.windows.net/myshare/mydirectorypath/myfile`	HTTP/

Figure 3.4: Azure service domain for downloading files using a GET request.

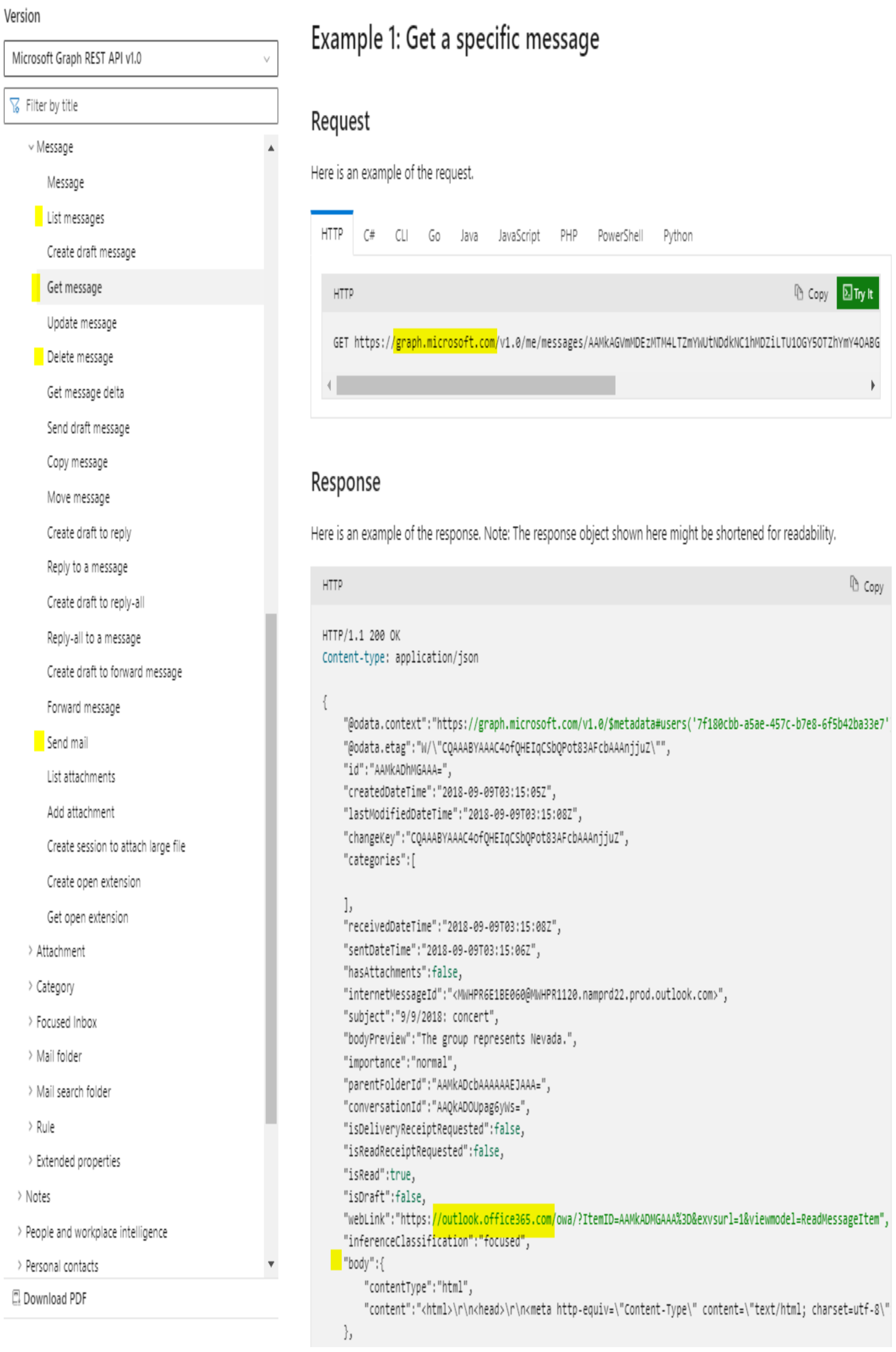


```
Request Syntax:
PUT https://myaccount.file.core.windows.net/myshare/myfile HTTP/1.1

Request Headers:
x-ms-version: 2020-02-10
x-ms-date: Mon, 27 Jan 2014 22:41:55 GMT
Content-Type: text/plain; charset=UTF-8
x-ms-content-length: 1024
Authorization: SharedKey myaccount:YhuFJjN4fAR8/AmBrqBz7MG2uFinQ4rkh4dscbj5
```

Figure 3.5: Azure service domain for uploading files using a PUT request.

Furthermore, as illustrated in Figures 3.6 and 3.7, Outlook uses the Microsoft Graph REST API [13] for various operations: retrieving a message via a GET request and creating a message with an attachment through a POST request. These operations are fundamental for services that manage email interactions programmatically, such as automating email notifications or processing incoming messages for data extraction purposes.



Example 1: Get a specific message

Request

Here is an example of the request.

```
HTTP
GET https://graph.microsoft.com/v1.0/me/messages/AAMkAGVmMDEzMTM4LTZmYWUtNDdkNC1hMDZiLTU1OGY5OTZhYmY4OABG
```

Response

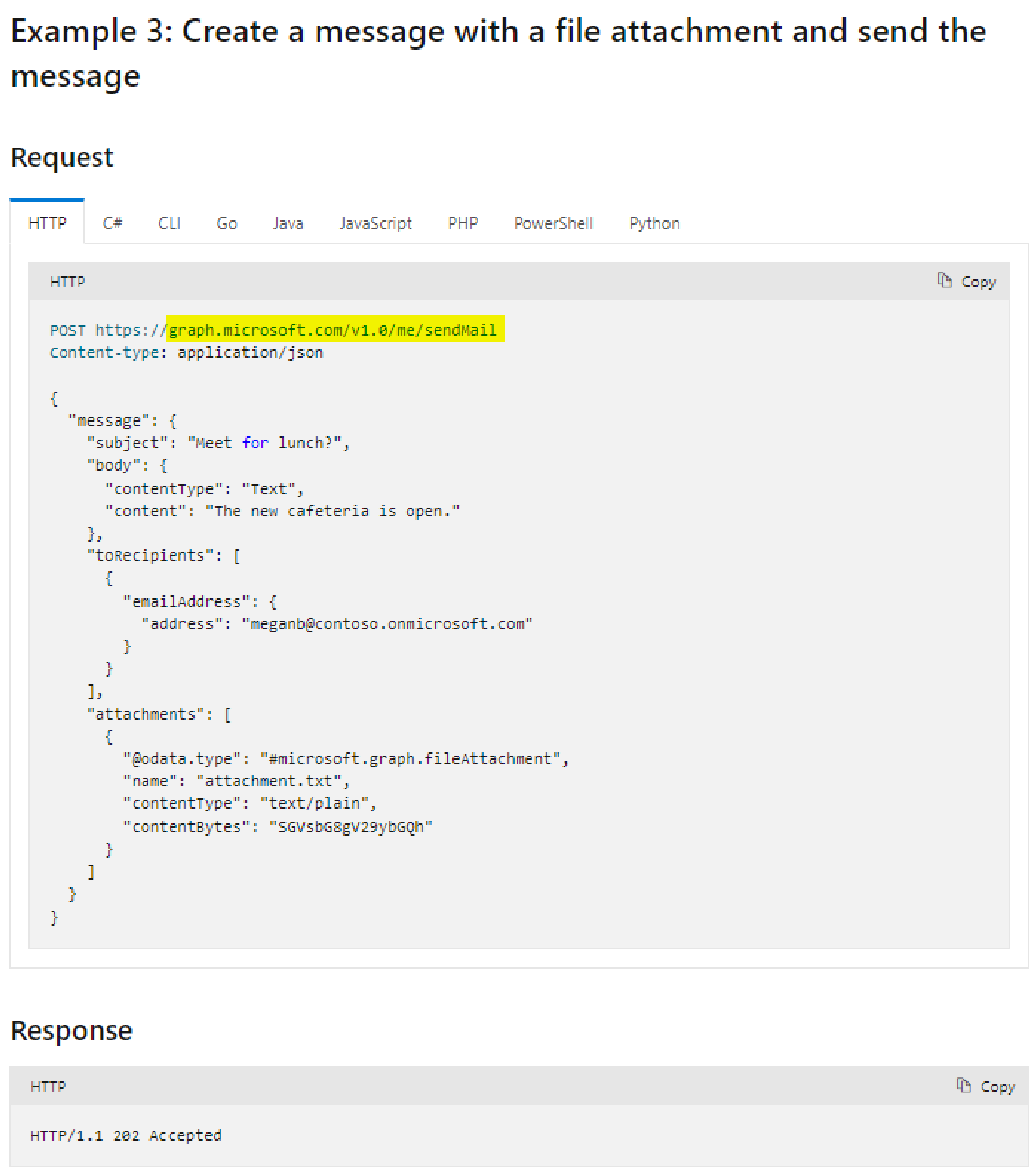
Here is an example of the response. Note: The response object shown here might be shortened for readability.

```
HTTP
HTTP/1.1 200 OK
Content-type: application/json

{
   "@odata.context":"https://graph.microsoft.com/v1.0/$metadata#users('7f180cbb-a5ae-457c-b7e8-6f5b42ba33e7'
   "@odata.etag":"W/\"CQAAABYAAAC4ofQHEIqCSbQPot83AFcbAAAnjjuZ\"",
   "id":"AAMkADhMGAAA=",
   "createdDateTime":"2018-09-09T03:15:05Z",
   "lastModifiedDateTime":"2018-09-09T03:15:08Z",
   "changeKey":"CQAAABYAAAC4ofQHEIqCSbQPot83AFcbAAAnjjuZ",
   "categories":[

   ],
   "receivedDateTime":"2018-09-09T03:15:08Z",
   "sentDateTime":"2018-09-09T03:15:06Z",
   "hasAttachments":false,
   "internetMessageId":"<MWHPR6E1BE060@MWHPR1120.namprd22.prod.outlook.com>",
   "subject":"9/9/2018: concert",
   "bodyPreview":"The group represents Nevada.",
   "importance":"normal",
   "parentFolderId":"AAMkADcbAAAAAAEJAAA=",
   "conversationId":"AAQkADOUpag6yWs=",
   "isDeliveryReceiptRequested":false,
   "isReadReceiptRequested":false,
   "isRead":true,
   "isDraft":false,
   "webLink":"https://outlook.office365.com/owa/?ItemID=AAMkADMGAAA%3D&exvsurl=1&viewmodel=ReadMessageItem",
   "inferenceClassification":"focused",
   "body":{
      "contentType":"html",
      "content":"<html>\r\n<head>\r\n<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\"
   },
```

Figure 3.6: Example of a GET request to retrieve a specific message using Microsoft Graph REST API. [13].


Example 3: Create a message with a file attachment and send the message

Request

HTTP C# CLI Go Java JavaScript PHP PowerShell Python

HTTP Copy

```
POST https://graph.microsoft.com/v1.0/me/sendMail
Content-type: application/json

{
  "message": {
    "subject": "Meet for lunch?",
    "body": {
      "contentType": "Text",
      "content": "The new cafeteria is open."
    },
    "toRecipients": [
      {
        "emailAddress": {
          "address": "meganb@contoso.onmicrosoft.com"
        }
      }
    ],
    "attachments": [
      {
        "@odata.type": "#microsoft.graph.fileAttachment",
        "name": "attachment.txt",
        "contentType": "text/plain",
        "contentBytes": "SGVsbG8gV29ybGQh"
      }
    ]
  }
}
```

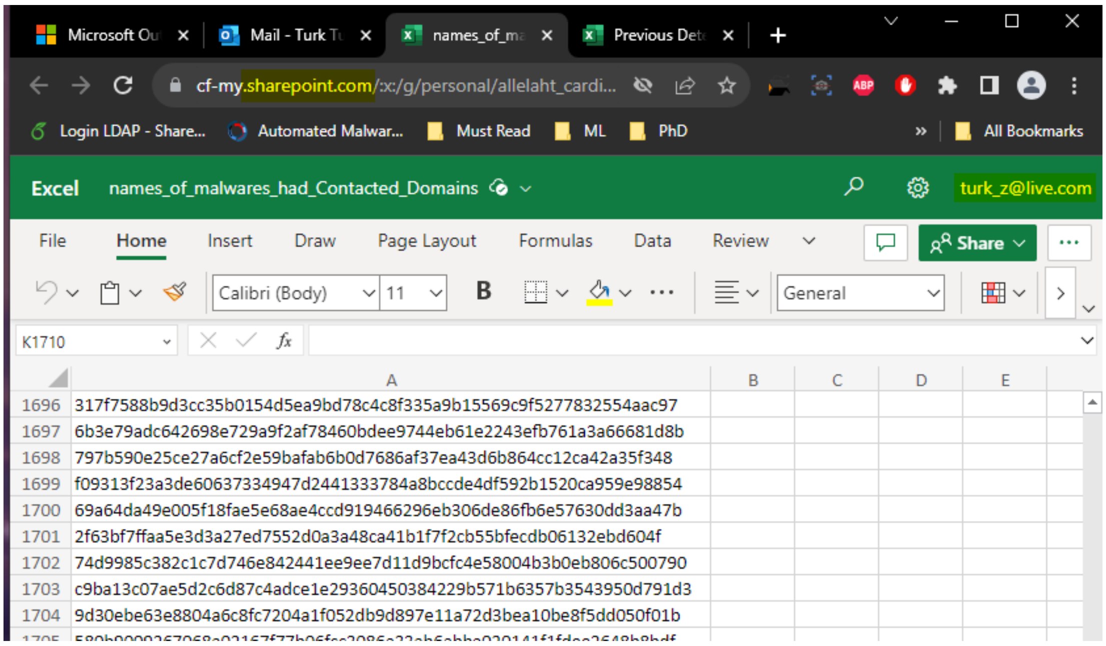
Response

HTTP Copy

```
HTTP/1.1 202 Accepted
```

Figure 3.7: Example of a POST request to create a message with a file attachment using Microsoft Graph REST API. [25].

For OneDrive, regardless of whether a file is shared via the OneDrive website or through the OneDrive app, the shared URL will be hosted on https://cf-my.sharepoint.com. As demonstrated by the examples where files were shared using both methods, as presented in Figures 3.8 and 3.9



cf-my.sharepoint.com/:x:/g/personal/allelaht_cardi...

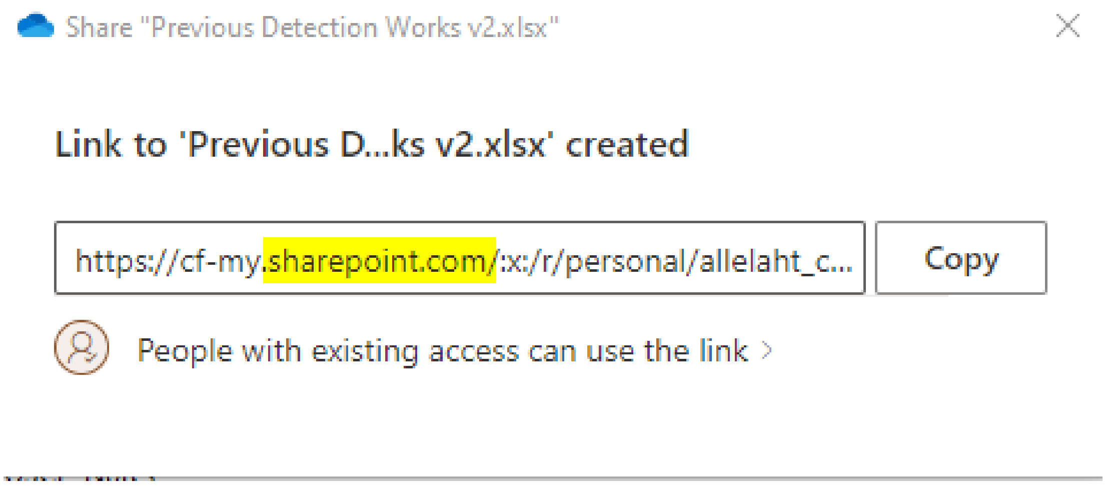
Excel names_of_malwares_had_Contacted_Domains

turk_z@live.com

K1710

	A
1696	317f7588b9d3cc35b0154d5ea9bd78c4c8f335a9b15569c9f5277832554aac97
1697	6b3e79adc642698e729a9f2af78460bdee9744eb61e2243efb761a3a66681d8b
1698	797b590e25ce27a6cf2e59bafab6b0d7686af37ea43d6b864cc12ca42a35f348
1699	f09313f23a3de60637334947d2441333784a8bccde4df592b1520ca959e98854
1700	69a64da49e005f18fae5e68ae4ccd919466296eb306de86fb6e57630dd3aa47b
1701	2f63bf7ffaa5e3d3a27ed7552d0a3a48ca41b1f7f2cb55bfecdb06132ebd604f
1702	74d9985c382c1c7d746e842441ee9ee7d11d9bcfc4e58004b3b0eb806c500790
1703	c9ba13c07ae5d2c6d87c4adce1e29360450384229b571b6357b3543950d791d3
1704	9d30ebe63e8804a6c8fc7204a1f052db9d897e11a72d3bea10be8f5dd050f01b

Figure 3.8: Shared file URL on OneDrive via the website.



Share "Previous Detection Works v2.xlsx"

Link to 'Previous D...ks v2.xlsx' created

https://cf-my.sharepoint.com/:x:/r/personal/allelaht_c...

Figure 3.9: Shared file URL on OneDrive via the app.

3.3.2 Categorisation of Abused Cloud Services

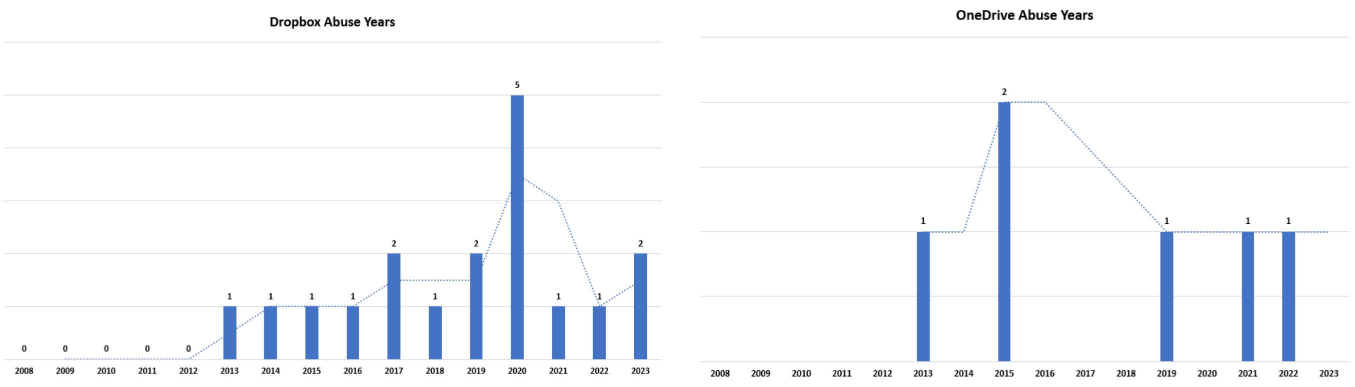
Building upon our identification of CPLS domains, we can categorise the CPLS that are commonly abused for C&C infrastructure. This categorisation provides a comprehensive view of the CPLS abuse landscape observed in our dataset:

1. **File Storage and Sharing Services:**
   - Examples: Dropbox, Google Drive, OneDrive
   - Abuse methods: Storing malicious payloads, exfiltrating data, hosting C&C instructions
2. **Social Media Platforms:**
   - Examples: Twitter, Facebook, Instagram
   - Abuse methods: Disseminating encoded commands, coordinating botnet activities
3. **Business Communication Platforms:**
   - Examples: Slack, Microsoft Teams
   - Abuse methods: Covert C&C communication, data exfiltration
4. **Code Repository Services:**
   - Examples: GitHub, GitLab
   - Abuse methods: Storing malicious scripts, hosting C&C instructions
5. **Email Services:**
   - Examples: Gmail, Outlook
   - Abuse methods: Sending C&C instructions, exfiltrating data
6. **Messaging Services:**
   - Examples: Telegram, Discord
   - Abuse methods: Real-time C&C communication, coordinating attacks

Each category of CPLS offers unique features that malicious actors exploit for their C&C operations. For instance, file storage services are often abused to store and retrieve malicious payloads, while social media platforms are exploited for their ability to broadcast encoded messages to large numbers of compromised devices simultaneously.

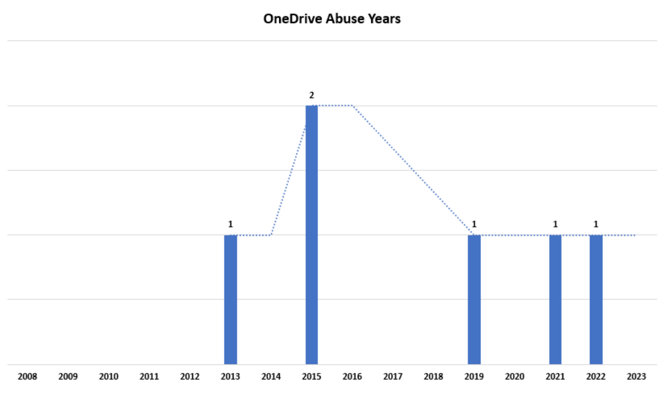
3.3.3 CPLS domain Abuse Trends

As part of our investigation into the abuse of CPLS listed in Table 3.1 for C&C purposes, we analysed the frequency of such abuses across different CSPs over the years. Figures 3.10 3.11, 3.12 , 3.13 and 3.14 illustrate the number of incidents per year where popular CSPs such as OneDrive, Outlook, GitHub, Pastebin, Slack, Google Drive, Facebook, Google Docs, Twitter, and Dropbox were exploited for malicious activities. This temporal abuse patterns, which show the increasing exploitation of these CPLS domains by malicious actors in recent years, with each data point representing the number of abuse occurrences detected per year.

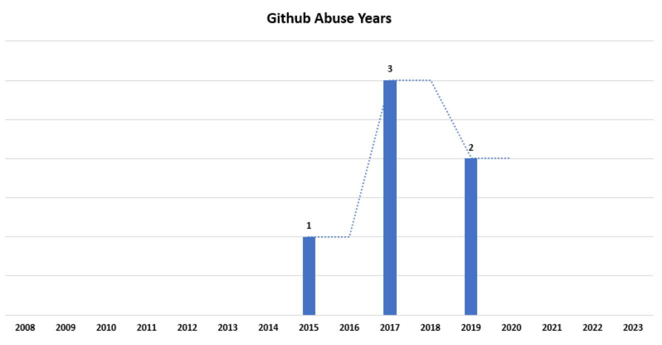


(a) Dropbox (b) OneDrive

Figure 3.10: Yearly abuse occurrences of the (a) Dropbox domain and (b) OneDrive domain.

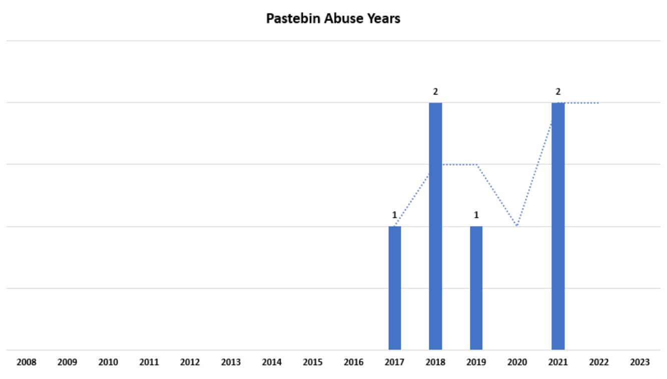


(a) OneDrive

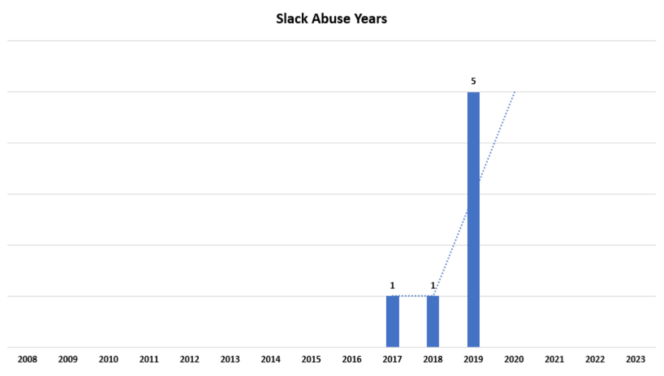


(b) Github

Figure 3.11: Yearly abuse occurrences of (a) the Outlook domain and (b) the GitHub domain.

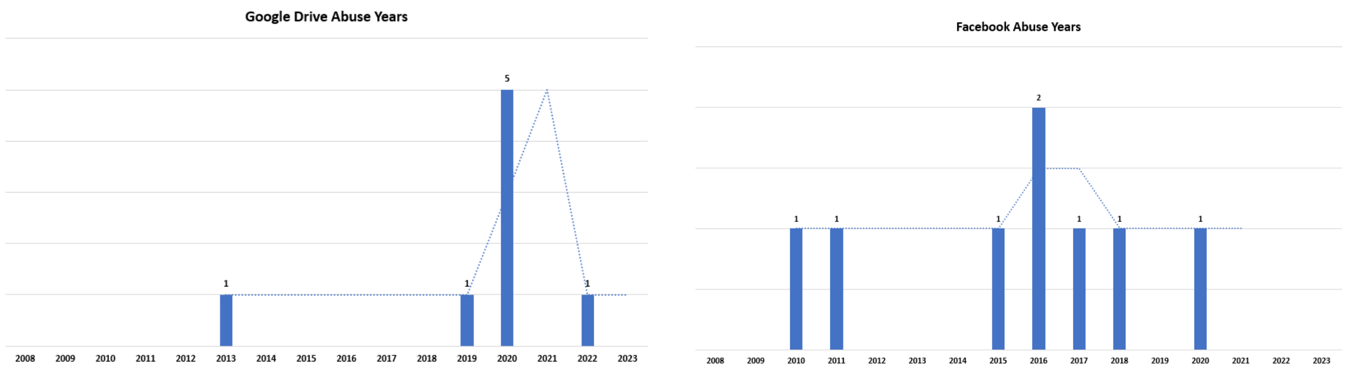


(a) Pastebin



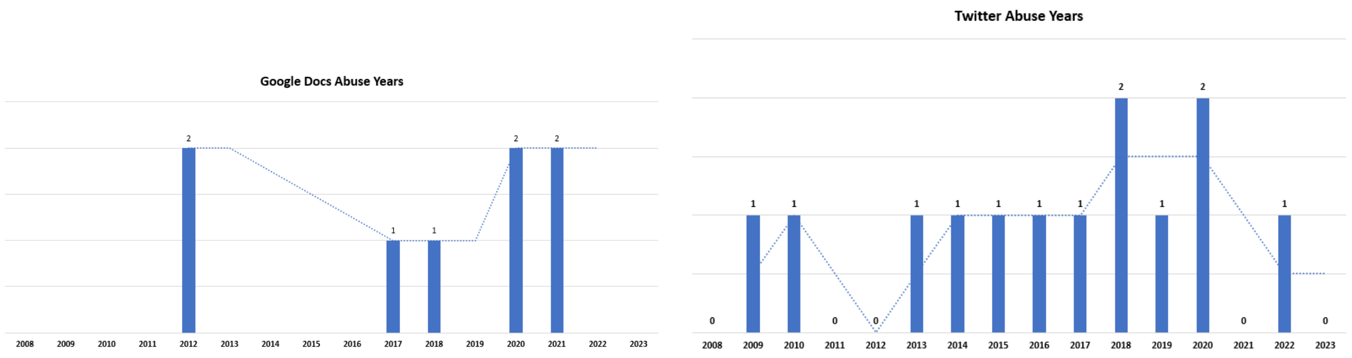
(b) Slack

Figure 3.12: Yearly abuse occurrences of (a) the Pastebin domain and (b) the Slack domain.



(a) Google Drive (b) Facebook

Figure 3.13: Yearly abuse occurrences of the (a) Google Drive and (b) Facebook domains.



(a) Google Docs (b) Twitter

Figure 3.14: Yearly abuse occurrences of (a) the Google Docs domain and (b) the Twitter domain.

3.3.4 Dataset Collection and Preprocessing

Malicious Samples Inclusion

In this research, we utilized a comprehensive dataset obtained from VirusTotal [30], a prominent online platform for malware analysis, spanning the period from 2017 to 2021. The dataset encompassed various malware formats, but our focus was specifically on Portable Executable (PE) files that abuse CPLS as C&C infrastructure. The initial dataset contained 324,841 samples with a total size of 409.7GB.

To collect relevant malware samples, we employed a two-step approach. First, we utilised the VirusTotal (VT) Intelligence [69] in conjunction with a custom Python script. This script was designed to extract samples that exhibited communication with known CPLS-hosted domains, as listed in Table 3.1. By leveraging the capabilities of the VT Intelligence and our tailored script, we were able to identify and retrieve a targeted subset of 621 malware samples that specifically interacted with the CPLS domains of interest.

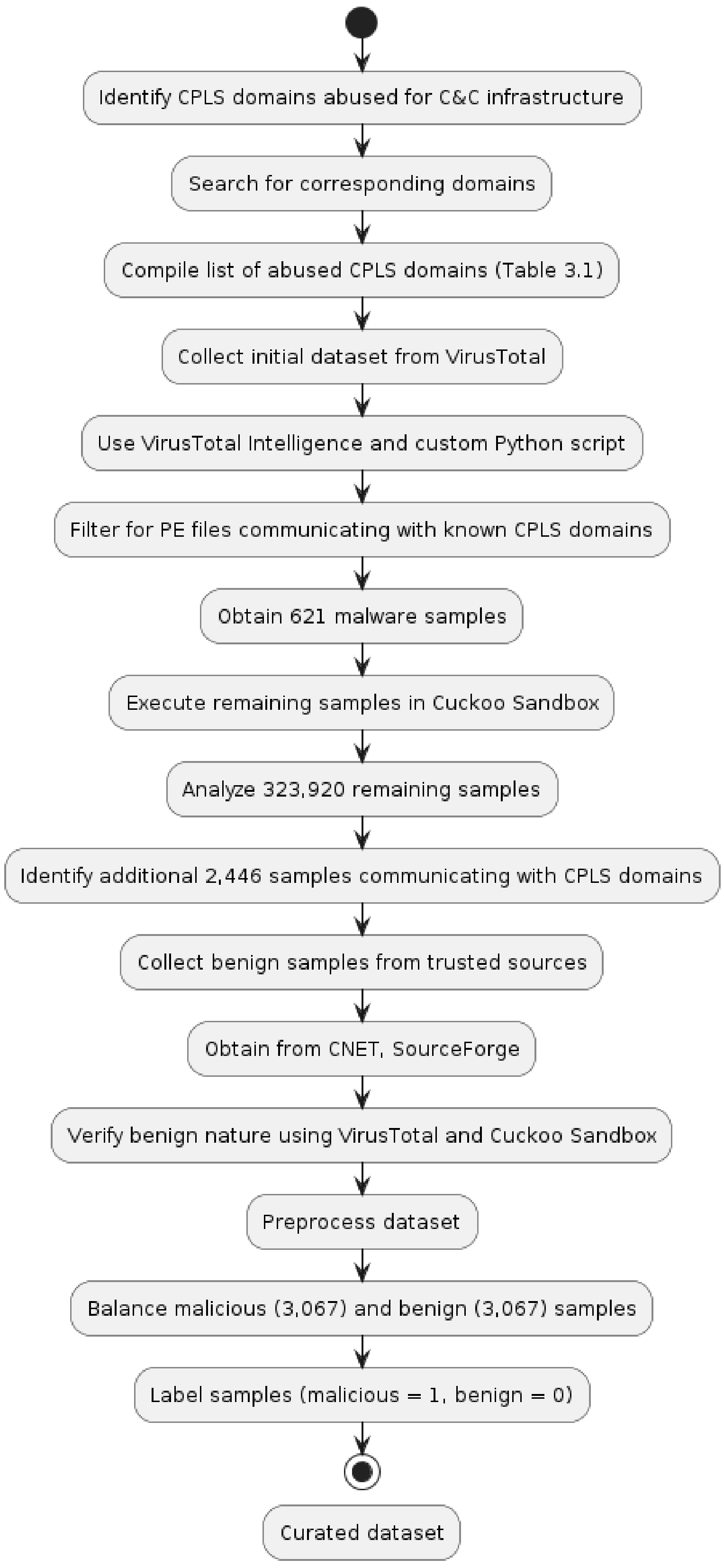
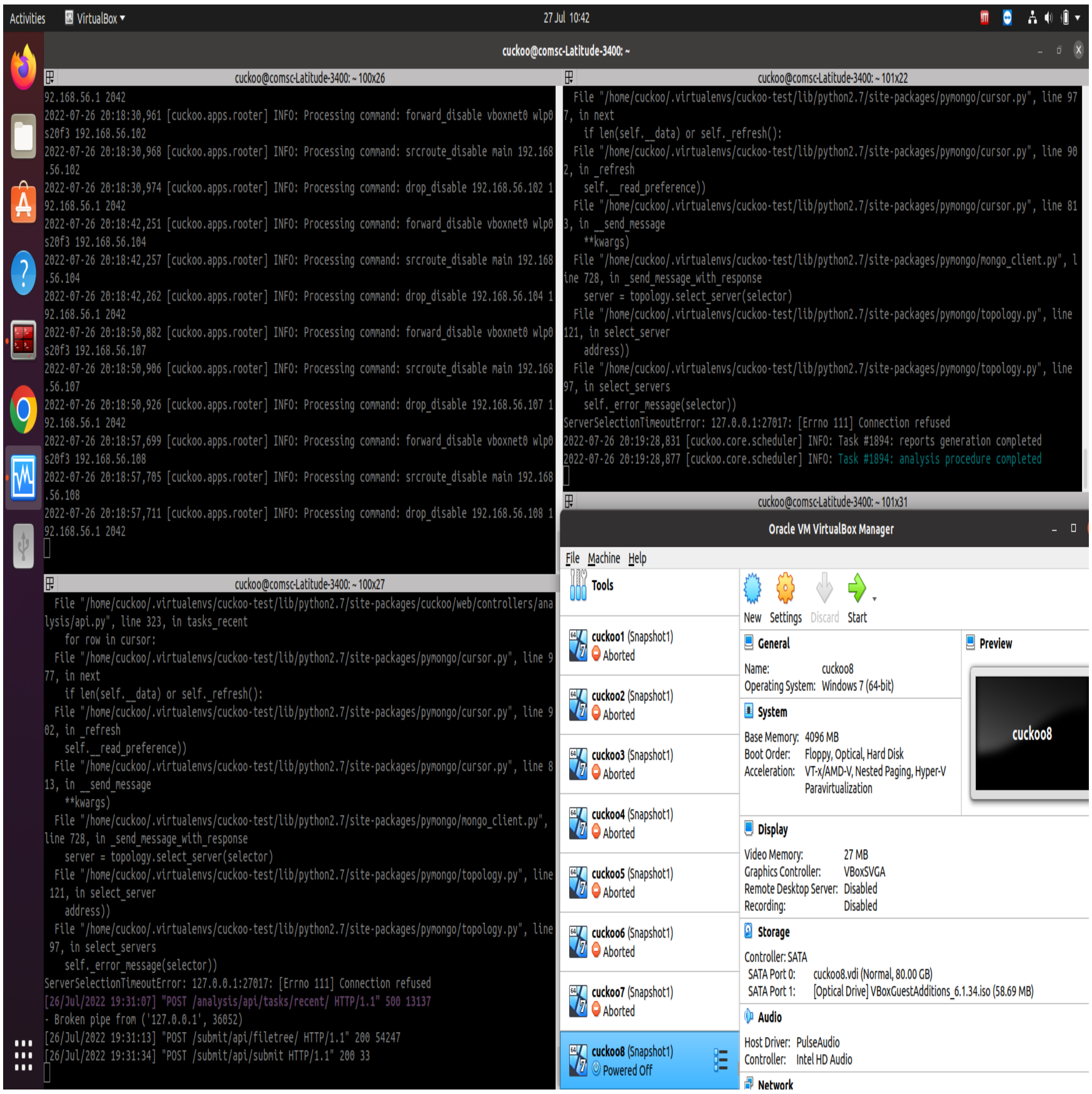

Identify CPLS domains abused for C&C infrastructure
Search for corresponding domains
Compile list of abused CPLS domains (Table 3.1)
Collect initial dataset from VirusTotal
Use VirusTotal Intelligence and custom Python script
Filter for PE files communicating with known CPLS domains
Obtain 621 malware samples
Execute remaining samples in Cuckoo Sandbox
Analyze 323,920 remaining samples
Identify additional 2,446 samples communicating with CPLS domains
Collect benign samples from trusted sources
Obtain from CNET, SourceForge
Verify benign nature using VirusTotal and Cuckoo Sandbox
Preprocess dataset
Balance malicious (3,067) and benign (3,067) samples
Label samples (malicious = 1, benign = 0)
Curated dataset

Figure 3.15: Workflow for CPLS Malware and Benignware Dataset Collection and Preprocessing.

Next, to further expand our dataset and ensure comprehensive coverage, we executed the remaining 323,920 samples (excluding the 621 samples already extracted) in a controlled Cuckoo sandbox environment [31]. Cuckoo sandbox is an open-source automated malware analysis system that allows for the dynamic analysis of malware in an isolated environment. Figure 3.16 shows a screenshot of the Cuckoo sandbox during the analysis of a malware sample. The sandbox generates detailed analysis reports, including information about the sample's interactions with the system and network communication.


Activities
VirtualBox
27 Jul 10:42
cuckoo@comsc-Latitude-3400: ~
cuckoo@comsc-Latitude-3400: ~ 100x26
cuckoo@comsc-Latitude-3400: ~ 101x22
cuckoo@comsc-Latitude-3400: ~ 100x27
cuckoo@comsc-Latitude-3400: ~ 101x31
Oracle VM VirtualBox Manager
File Machine Help
Tools
New Settings Discard Start
Preview
cuckoo8

```
92.168.56.1 2042
2022-07-26 20:18:30,961 [cuckoo.apps.rooter] INFO: Processing command: forward_disable vboxnet0 wlp0
s20f3 192.168.56.102
2022-07-26 20:18:30,968 [cuckoo.apps.rooter] INFO: Processing command: srcroute_disable main 192.168
.56.102
2022-07-26 20:18:30,974 [cuckoo.apps.rooter] INFO: Processing command: drop_disable 192.168.56.102 1
92.168.56.1 2042
2022-07-26 20:18:42,251 [cuckoo.apps.rooter] INFO: Processing command: forward_disable vboxnet0 wlp0
s20f3 192.168.56.104
2022-07-26 20:18:42,257 [cuckoo.apps.rooter] INFO: Processing command: srcroute_disable main 192.168
.56.104
2022-07-26 20:18:42,262 [cuckoo.apps.rooter] INFO: Processing command: drop_disable 192.168.56.104 1
92.168.56.1 2042
2022-07-26 20:18:50,882 [cuckoo.apps.rooter] INFO: Processing command: forward_disable vboxnet0 wlp0
s20f3 192.168.56.107
2022-07-26 20:18:50,906 [cuckoo.apps.rooter] INFO: Processing command: srcroute_disable main 192.168
.56.107
2022-07-26 20:18:50,926 [cuckoo.apps.rooter] INFO: Processing command: drop_disable 192.168.56.107 1
92.168.56.1 2042
2022-07-26 20:18:57,699 [cuckoo.apps.rooter] INFO: Processing command: forward_disable vboxnet0 wlp0
s20f3 192.168.56.108
2022-07-26 20:18:57,705 [cuckoo.apps.rooter] INFO: Processing command: srcroute_disable main 192.168
.56.108
2022-07-26 20:18:57,711 [cuckoo.apps.rooter] INFO: Processing command: drop_disable 192.168.56.108 1
92.168.56.1 2042
```

```
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/cursor.py", line 97
7, in next
    if len(self.__data) or self._refresh():
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/cursor.py", line 90
2, in _refresh
    self.__read_preference))
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/cursor.py", line 81
3, in __send_message
    **kwargs)
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/mongo_client.py", l
ine 728, in _send_message_with_response
    server = topology.select_server(selector)
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/topology.py", line
121, in select_server
    address))
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/topology.py", line
97, in select_servers
    self._error_message(selector))
ServerSelectionTimeoutError: 127.0.0.1:27017: [Errno 111] Connection refused
2022-07-26 20:19:28,831 [cuckoo.core.scheduler] INFO: Task #1894: reports generation completed
2022-07-26 20:19:28,877 [cuckoo.core.scheduler] INFO: Task #1894: analysis procedure completed
```

```
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/cuckoo/web/controllers/ana
lysis/api.py", line 323, in tasks_recent
    for row in cursor:
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/cursor.py", line 9
77, in next
    if len(self.__data) or self._refresh():
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/cursor.py", line 9
02, in _refresh
    self.__read_preference))
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/cursor.py", line 8
13, in __send_message
    **kwargs)
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/mongo_client.py",
line 728, in _send_message_with_response
    server = topology.select_server(selector)
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/topology.py", line
 121, in select_server
    address))
  File "/home/cuckoo/.virtualenvs/cuckoo-test/lib/python2.7/site-packages/pymongo/topology.py", line
 97, in select_servers
    self._error_message(selector))
ServerSelectionTimeoutError: 127.0.0.1:27017: [Errno 111] Connection refused
[26/Jul/2022 19:31:07] "POST /analysis/api/tasks/recent/ HTTP/1.1" 500 13137
- Broken pipe from ('127.0.0.1', 36052)
[26/Jul/2022 19:31:13] "POST /submit/api/filetree/ HTTP/1.1" 200 54247
[26/Jul/2022 19:31:34] "POST /submit/api/submit HTTP/1.1" 200 33
```

cuckoo1 (Snapshot1) Aborted
cuckoo2 (Snapshot1) Aborted
cuckoo3 (Snapshot1) Aborted
cuckoo4 (Snapshot1) Aborted
cuckoo5 (Snapshot1) Aborted
cuckoo6 (Snapshot1) Aborted
cuckoo7 (Snapshot1) Aborted
cuckoo8 (Snapshot1) Powered Off

General
Name: cuckoo8
Operating System: Windows 7 (64-bit)

System
Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, Hyper-V Paravirtualization

Display
Video Memory: 27 MB
Graphics Controller: VBoxSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: SATA
SATA Port 0: cuckoo8.vdi (Normal, 80.00 GB)
SATA Port 1: [Optical Drive] VBoxGuestAdditions_6.1.34.iso (58.69 MB)

Audio
Host Driver: PulseAudio
Controller: Intel HD Audio

Network

Figure 3.16: Screenshot of Cuckoo sandbox during malware analysis.

The malware samples are submitted to the central management component of Cuckoo, which then distributes them to multiple virtual machines (VMs) for analysis. Once the analysis is completed, Cuckoo generates detailed reports in JavaScript Object Notation (JSON) file format. By leveraging the distributed architecture and comprehensive analysis capabilities of Cuckoo, we were able to process a large number of malware samples efficiently. Figure 3.17 illustrates the architecture of the Cuckoo sandbox, highlighting the interaction between the malware sample repositories, virtual machines, and analysis reports.

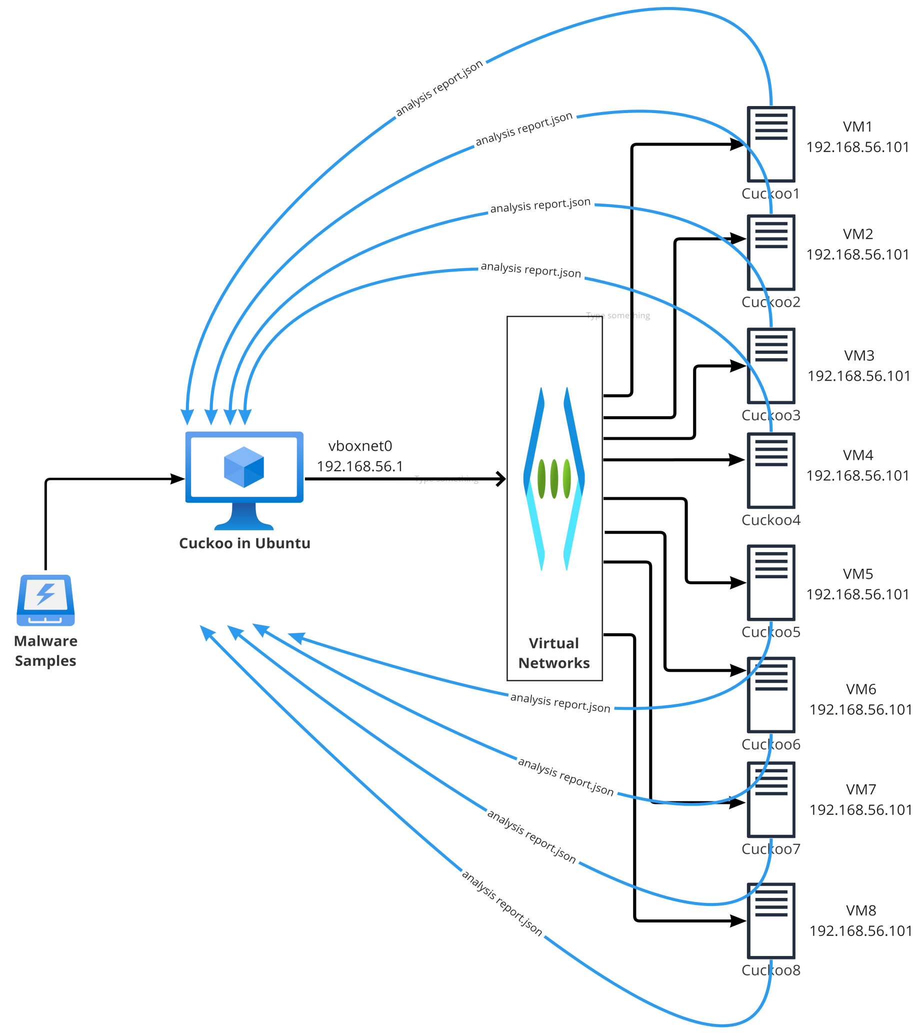


Figure 3.17: Architecture of Cuckoo sandbox for malware analysis.

In our Cuckoo sandbox configuration, we set a 5-minute interval for monitoring malware behavior. This choice is based on both existing literature and our empirical observations. Although there is no best practice for malware execution time due to the varying behaviours of different samples, several studies have successfully used a 5-

minute execution period for malware analysis [75, 76, 87, 105, 122]. In our experimental setup, we observed that some known CPLS abusing malware samples did not exhibit any suspicious behavior in the first few minutes of execution. However, at the 5-minute mark, we began to observe the first connections to CPLS domains. While longer durations could potentially capture more behaviors, we found that the 5-minute interval struck an optimal balance between detection effectiveness and computational resource management, especially considering the large-scale nature of our dataset.

The detailed workflow of our dataset collection and preprocessing process is depicted in Figure 3.15. Through this dynamic analysis, we identified an additional 2,446 samples that demonstrated actual connections to the CPLS domains of interest. By combining the samples extracted using the VT Intelligence, 621 samples, and those identified through the Cuckoo sandbox analysis, 2,446 samples, we obtained a total of 3,067 malware samples that specifically abused CPLS for C&C communication. This targeted dataset allowed us to focus our research on the most relevant samples, ensuring the quality and relevance of our findings. The detailed workflow of our dataset collection process is depicted in Figure 3.18.

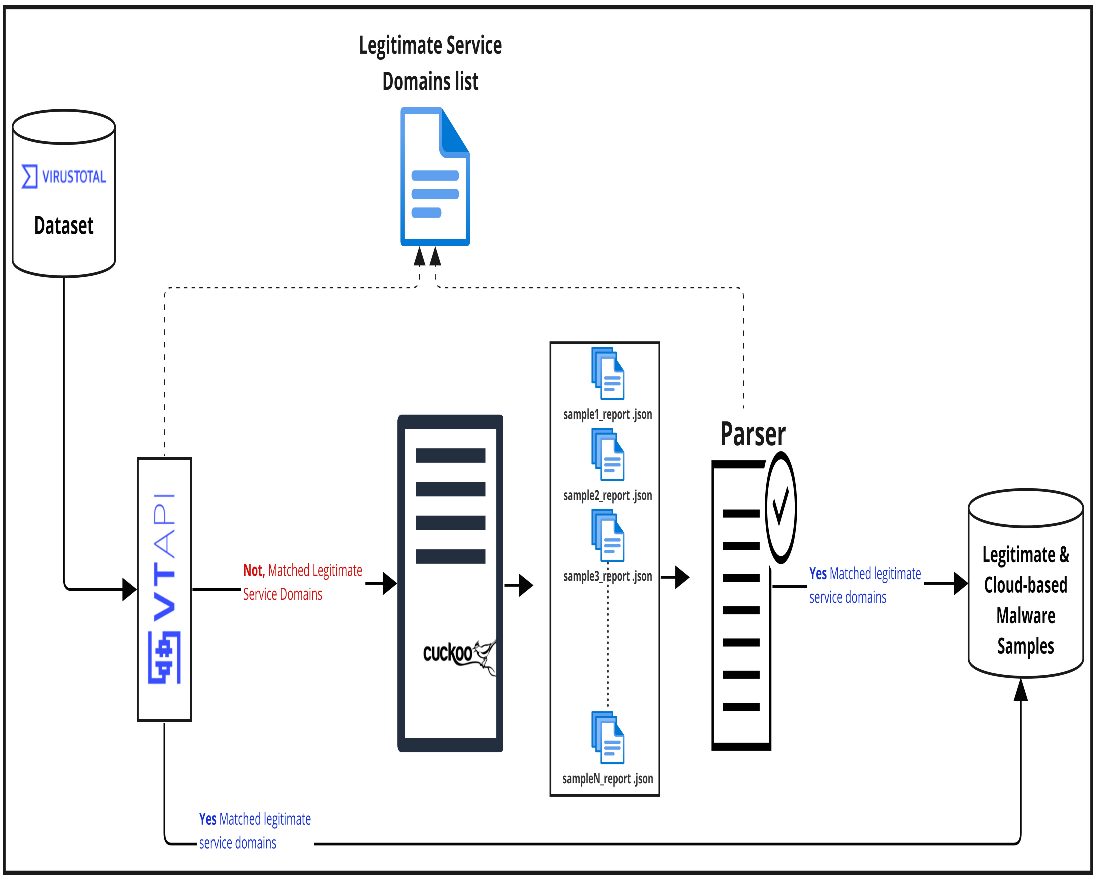

Legitimate Service Domains list
VIRUSTOTAL
Dataset
VT API
Not, Matched Legitimate Service Domains
cuckoo
sample1_report.json
sample2_report.json
sample3_report.json
sampleN_report.json
Parser
Yes Matched legitimate service domains
Legitimate & Cloud-based Malware Samples
Yes Matched legitimate service domains

Figure 3.18: Detailed workflow for extracting a sub-dataset from the VirusTotal datasets.

Table 3.2 presents a detailed breakdown of the dataset, including the size, total number of samples, and the number of samples connecting to CPLS domains for each month and year.

Table 3.2: Overview of the dataset

Year	**Month**	**Size**	**Malware Samples**	**Malware Samples Connected to CPLS**
2017	Oct	24.7 GB	18,293	218
	Nov	3.5 GB	6,881	11
2018	Mar	91.5 GB	75,066	376
	Jun	84.8 GB	93,061	187
2019	May	47.3 GB	34,112	782
	Nov	9.3 GB	23,573	29
2020	May	40.4 GB	38,445	489
	Nov	29.4 GB	12,289	382
2021	Apr	42.0 GB	10,490	287
	Nov	36.8 GB	12,631	306
Total		**409.7 GB**	**324,841**	**3,067**

Benign Samples Inclusion

In addition to the malicious samples, we also included a set of benign samples in our dataset to facilitate a comprehensive analysis and evaluation of our detection techniques. These benign samples were obtained from trusted sources such as CNET [28] and SourceForge [29], which are well-known repositories for legitimate software. To validate the benign nature of these samples, we submitted them to VirusTotal for anti-virus scanning. Only the samples that received a detection score of zero, indicating no flagging by any of the participating anti-virus engines, were considered for inclusion in our dataset. Furthermore, to ensure the network connectivity of the benign samples, we executed them in Cuckoo [31] sandbox environment similar to the one used for malware analysis. The purpose of confirming that the benign samples establish internet connections is twofold. First, it ensures that the benign samples are not corrupted and fully functional, and exhibit their intended behaviour, which may include network communication for legitimate purposes. Second, by verifying that the benign samples establish internet connections, we can assess the effectiveness of our detection techniques in distinguishing between malicious and benign network activity. By applying these stringent selection criteria, we curated a reliable and representative set of benign samples to complement the malicious ones in our dataset.

Dataset Labelling

The dataset labelling process is a crucial step in preparing the data for training and evaluating machine learning models. We employed a binary labelling scheme, assigning a label of 1 to malicious samples and 0 to benign samples. The labelled samples were then utilized for feature extraction and training machine learning models, as discussed in Chapters 4 and 5.

Dataset Balancing

Initially, the collected dataset exhibited an imbalance in the number of malicious and benign samples, with 3,067 malicious samples and 3,652 benign samples. Imbalanced datasets can potentially introduce bias in machine learning classifiers, leading to skewed results and suboptimal performance. To mitigate this issue and ensure a balanced representation, we removed the excess benign samples. The dataset balancing process was carried out to maintain the dataset's quality and integrity. Since the collection of benign samples was based solely on their ability to establish connections to the internet and their VirusTotal detection score of zero, removing the excess benign samples did not alter the core properties of the dataset.

Experimental Setup

The entire experimental setup, including the extraction of the sub-dataset from VirusTotal, the parsing of PE files, and the execution of malware and benign samples in the Cuckoo sandbox [31], was conducted on a dedicated machine equipped with an Intel Xeon (Skylake IBRS) CPU running at 2.2 GHz, 64GB of RAM, and the Ubuntu 20.04.1 LTS amd64 operating system. This robust hardware and software configuration ensured efficient processing and analysis of the dataset, enabling us to handle the computational demands of the research effectively.

Dataset Uniqueness and Value

The resulting dataset, comprising 3,067 malware samples and 3,067 benign samples, with a total of 6,134 samples, is unique and valuable to the field of cybersecurity due to its specific focus on malware that abuses CPLS as a C&C infrastructure. Unlike many existing datasets that primarily contain generic malware samples, our dataset provides a targeted representation of this emerging threat, which has been relatively underrepresented in prior research. By curating a dataset that specifically addresses CPLS abuse, we aim to contribute to the development of more effective detection and mitigation strategies tailored to this particular type of malicious activity. Moreover, the comprehensive preprocessing steps applied to the dataset, including the careful selection of malware samples, the inclusion of benign samples, and the balancing of the dataset, enhance its quality and reliability. These preprocessing measures help ensure that the insights derived from the dataset are accurate, meaningful, and applicable to real-world scenarios.

3.3.5 Adversarial Assumption

In our research, particularly as detailed in Chapter 6, we adopt a white-box attack model to evaluate the robustness of our detection system. This model assumes the following.

- **Full Knowledge of the Model:** The attacker has complete knowledge of our detection system, including its architecture, parameters, and the features used.

- **Access to Feature Information:** The adversary understands the features we extract from PE files and how they are used in the detection process.
- **Ability to Modify Samples:** The attacker can modify malicious samples to attempt to evade detection, within the constraints of maintaining the malicious functionality of the file.
- **Goal of Evasion:** The primary objective of the attacker is to modify malicious samples in a way that causes our model to misclassify them as benign.

This white-box model represents a worst-case scenario, where the adversary has maximum knowledge about our system. By adopting this stringent model, we aim to evaluate and improve the robustness of our detection approach against highly sophisticated attacks. The details of how we test our system against this adversarial model, including the specific Replace Misclassified Parameter (RMCP) attack, are elaborated in Chapter 6.

3.4 Summary

In this chapter, we have presented a detailed exploration of the dataset and preprocessing steps that form the foundation of our research on detecting the abuse of CPLS for malicious C&C infrastructure. We emphasized the significance of focusing on the PE file format. By examining the global market share of desktop operating systems and the prevalence of PE files in VirusTotal submissions, we justified our decision to concentrate on this specific file format. Moreover, we provided insights into the process of identifying the exact CPLS domains that have been exploited by malicious actors for C&C communications. By reviewing API documentation and executing actual connections to these services, we were able to compile a comprehensive list of abused CPLS domains. The analysis of CPLS domain abuse trends over the years revealed the alarming growth of this threat, underscoring the urgent need for effective detection and mitigation strategies. The dataset collection and preprocess was meticulously described, highlighting the use of the VirusTotal Intelligence Agent, custom Python scripts, and the Cuckoo sandbox environment. We detailed the steps involved in extracting relevant malware samples that communicate with known CPLS domains and the filtering process that ensured the inclusion of only the most relevant samples in our final dataset. Additionally, we discussed the incorporation of benign samples from trusted sources and the verification measures employed to validate their legitimacy and functionality. To address the challenges posed by imbalanced datasets, we emphasized the importance of balancing the number of malicious and benign samples. By carefully removing excess benign samples while preserving the dataset's quality and integrity, we aimed to prevent classifier bias and ensure accurate evaluation of our detection techniques. Furthermore, we highlighted the unique characteristics and value of our dataset in the field of cybersecurity. By specifically focusing on malware that abuses CPLS as a C&C channel, our dataset fills a critical gap in existing research and contributes to the development of targeted detection and mitigation strategies. Throughout the chapter, we stressed the importance of maintaining data quality and integrity during the preprocessing phase. The meticulous attention to detail in dataset collection, sample verification, and balancing demonstrates our commitment to ensuring the reliability and representativeness of the dataset. Overall, this chapter lays the foundation for our research by providing a solid understanding of the dataset and the preprocessing steps involved, while also highlighting the uniqueness and value of our dataset in addressing the emerging threat of CPLS abuse for malicious C&C infrastructure.

Chapter 4

Detecting CPLS abuse as C&C infrastructure with Static features

4.1 Introduction

In this thesis, we propose two main approaches for detecting the abuse of CPLS as C&C infrastructure: static analysis and dynamic analysis. This chapter focusses on the static analysis approach, which involves the use of machine learning techniques applied to static features extracted from Portable Executable (PE) files. The dynamic analysis approach is discussed in detail in Chapter 5. PE files are a common format for executables on Windows operating systems and contain a wealth of information that can be used for malware detection. The methodology we employ consists of several stages, each of which contributes to the overall detection process. These stages are data collection, feature extraction, feature selection, classification, and evaluation. We analyze the PE files in our dataset to extract static features. These features, which are characteristics of the PE files that can be determined without execution, provide the raw data that our machine learning techniques operate on. The feature selection stage involves identifying the most informative features from the ones we extracted. Not all features are equally useful for distinguishing between benign and malicious files, so this stage is crucial to improve the efficiency and effectiveness of the subsequent classification stage. Finally, in the classification stage, we apply machine learning techniques to classify each PE file in our dataset as either benign or malicious. The output of this stage is a model that can be used to classify new, unseen PE files. Each of these stages is detailed in the following sections, where we discuss the specific techniques and approaches we used, as well as the rationale behind our choices. Through this comprehensive approach, our aim is to provide a robust and effective means to detect the abuse of CPLS as C&C infrastructure in PE files.

4.2 Methodology

Our methodology incorporates several novel elements that distinguish it from previous approaches in malware detection, particularly in the context of CPLS abuse. The approach is structured into five key subsections: Data Collection, Feature Extraction, Feature Selection, Classification, and Evaluation.

4.2.1 Novel Aspects of Our Approach

The key innovations in our methodology include:

- **Custom Feature Elimination (CFE) Method:** We introduce a novel CFE technique for optimal feature selection, designed to determine the exact number of features needed for filter selection approaches.
- **Tailored Feature Engineering:** Our approach includes derived features specifically designed to detect CPLS abuse, such as "presence_of_CLS_domains" and "potential_C&C_api_calls".
- **CPLS Abuse Detection:** Unlike general malware detection systems, our approach is specifically tailored to identify the abuse of Cloud and Public Legitimate Services as C&C infrastructure.

In Data Collection, we detail the acquisition of a dataset, focusing on PE files. Feature Extraction discusses the identification of key features from PE file headers, crucial for differentiating between benign and malicious files. Feature Selection elaborates on the application of filter-based and wrapper-based methods, including our novel CFE technique, to refine the feature set for enhanced malware detection accuracy. The Classification and Evaluation stages complete our methodological framework.

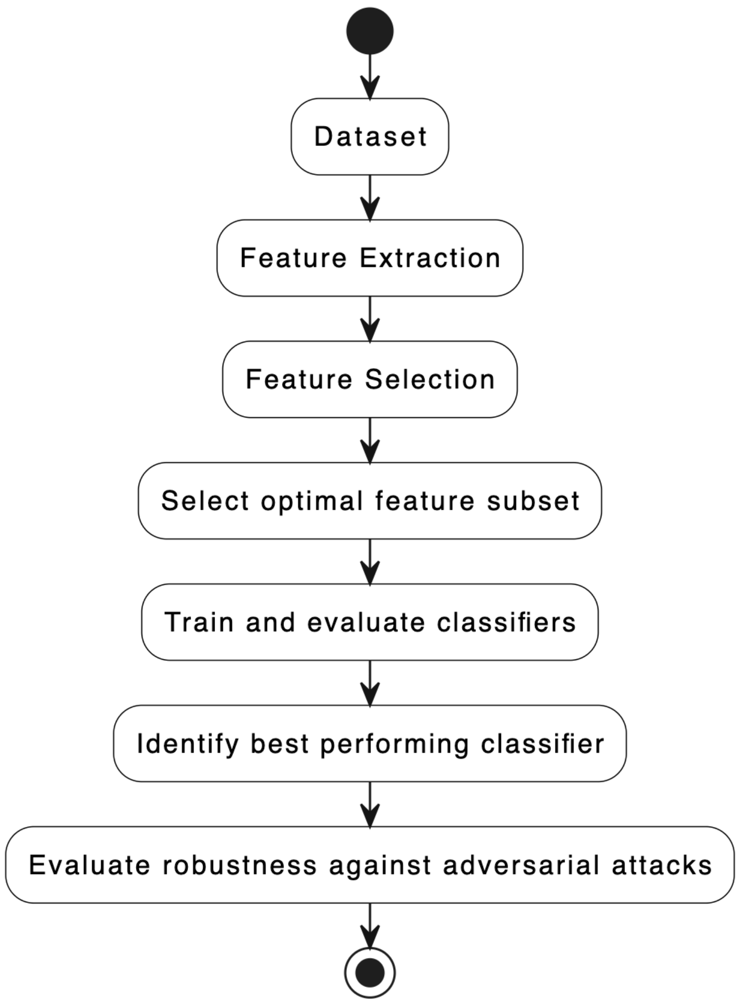

Dataset
Feature Extraction
Feature Selection
Select optimal feature subset
Train and evaluate classifiers
Identify best performing classifier
Evaluate robustness against adversarial attacks

Figure 4.1: Workflow of the proposed static analysis approach for detecting CPLS abuse as C&C infrastructure.

Figure 4.1 illustrates the workflow of our proposed static analysis approach for detecting the abuse of CPLS as C&C infrastructure. The workflow outlines the key stages involved, including data collection, feature extraction, feature selection, classification, and evaluation.

4.2.2 Data Collection

For a comprehensive description of the dataset collection process, refer to Chapter 3. In summary, we used a dataset obtained from VirusTotal [30] between 2017 and 2021, which included various malware formats. Our research specifically focused on PE files that abuse CPLS as C&C infrastructure. The dataset included malicious samples that showed communication with known CPLS-hosted domains and benign samples obtained from trusted sources. The initial dataset was imbalanced, but we ensured a balanced dataset by removing extra benign samples while retaining the same characteristics as the remaining ones.

4.2.3 Feature Extraction

Before delving into the specific features we extract from PE files, it's essential to understand the concept of static analysis in malware detection. Static analysis involves examining the binary code of a program without executing it. This approach is fundamental in identifying potential malware by analyzing its code structure, content, and other attributes that remain unchanged unless the file is modified.

In our static analysis approach, we primarily utilize raw features extracted directly from the PE file structure, complemented by two derived features. This combination allows us to capture both the structural characteristics of the executable and specific indicators of potential CPLS abuse. The raw features include various attributes from the COFF Header, Optional Header, and Section Table, providing a comprehensive view of the file's structure and potential behavior without execution.

Our focus on static analysis and PE header features is driven by the rich information these headers provide about the executable, without the need to execute potentially harmful code. The PE format, being a standard for Windows executables, offers various headers and sections that describe the binary's structure, behavior, and dependencies. We analyzed the header and sections of each file in our sample and identified a total of 38 relevant features, comprising 36 raw features and two derivative features (Table 4.2). Raw features can be directly extracted from the PE file without further processing. These features include Characteristics, DllCharacteristics, SizeOfImage, AddressOfEntryPoint, and ResourceSize.

Derived Features

In addition to the raw features extracted directly from the PE file, we also derive two additional features: presence_of_CLS_domains and potential_C&C_api_calls. These derived features provide valuable insights into the behaviour and characteristics of the PE files in relation to the potential abuse of CPLS for C&C communication. while many of the raw features extracted from PE files are commonly used in malware analysis, our derived features, particularly "presence_of_CLS_domains" and "potential_C&C_api_calls", are specifically tailored to detect CPLS abuse. These features are novel in the context of CPLS abuse detection and provide crucial indicators of potential C&C activity.

Presence of CPLS Domains The presence_of_CLS_domains feature is generated by examining each section of the PE file to check for the presence of any CPLS domains. If a CPLS domain is found in any section, the feature value is set to 1; otherwise, it is set to 0. This feature helps identify PE files that may be communicating with known CPLS domains, indicating potential malicious behavior.

Identifying Potential C&C API Calls After identifying the CPLS domains of interest that potentially communicate with malware, we further analyzed the collected samples to identify specific API calls commonly employed for C&C operations. We focused on a set of well-known networking-related API functions that malware frequently uses for establishing connections, sending and receiving data, and performing other C&C-related tasks. It is important to note that while these API calls are commonly associated with C&C communication, they are not exclusive to botnets or HTTP traffic. Many of these calls can be used for legitimate purposes or in non-HTTP protocols. Our approach considers these calls as potential indicators when combined with other features, rather than definitive proof of malicious activity. A limitation of focussing on these calls is the potential to miss C&C communication using non-HTTP protocols or custom communication methods. Table 4.1 presents the list of these potential C&C API calls and their descriptions.

Table 4.1: Potential API calls for abusing CPLS as C&C infrastructure.

API Call	Description	Potential Abuse Case
`InternetOpenA`	Open an Internet session	Establish a connection to CPLS
`InternetConnectA`	Connect to a remote server	Connect to CPLS servers
`HttpOpenRequestA`	Open an HTTP request handle	Open HTTP requests to CPLS
`InternetReadFile`	Read data from an Internet file	Read data from a file on CPLS
`InternetWriteFile`	Write data to an Internet file	Write data to a file on CPLS
`InternetCloseHandle`	Close an Internet handle	Close connections to CPLS
`WinHttpOpen`	Open an HTTP session	Open HTTP sessions with CPLS
`WinHttpConnect`	Connect to a remote server	Connect to CPLS servers
`WinHttpOpenRequest`	Open an HTTP request handle	Open HTTP requests to CPLS
`WinHttpSendRequest`	Send an HTTP request	Send HTTP requests to CPLS
`WinHttpReceiveResponse`	Receive an HTTP response	Receive HTTP responses from CPLS
`WinHttpReadData`	Read data from an HTTP request	Read data from an HTTP request to CPLS
`WinHttpWriteData`	Write data to an HTTP request	Write data to an HTTP request to CPLS
`WinHttpCloseHandle`	Close an HTTP handle	Close HTTP connections to CPLS
`connect`	Connect to a remote server	Connect to CPLS servers
`gethostbyname`	Resolve a host name	Resolve CPLS domain names
`recv`	Receive data from a socket	Receive data from CPLS
`send`	Send data to a socket	Send data to CPLS
`socket`	Create a socket	Create sockets for CPLS communication
`URLDownloadToFileA`	Download a file from the Internet	Download files from CPLS
`HttpSendRequestA`	Send an HTTP request	Send HTTP requests to CPLS
`InternetCrackUrlA`	Split an URL into its components	Parse CPLS URLs
`InternetOpenUrlA`	Open an HTTP or FTP session	Open HTTP or FTP sessions with CPLS
`InternetQueryDataAvailable`	Check for available data	Check for data from CPLS
`InternetReadFileExA`	Read data from an Internet file	Read data from a file on CPLS
`InternetWriteFileExA`	Write data to an Internet file	Write data to a file on CPLS
`InternetSetOptionA`	Set an Internet option	Set options for CPLS communication
`InternetQueryOptionA`	Query an Internet option	Query options for CPLS communication
`InternetGetLastResponseInfoA`	Retrieve error/status information	Get information from CPLS communication
`InternetGetConnectedStateExA`	Check the connection state	Check connection state with CPLS
`InternetCheckConnectionA`	Check if a resource is reachable	Check if CPLS is reachable
`InternetDialA`	Dial an Internet connection	Dial a connection to CPLS
`InternetGetAutoProxyUrlA`	Retrieve auto-config script URL	Retrieve CPLS auto-config script URL
`InternetGetCookieA`	Retrieve a cookie	Retrieve cookies from CPLS
`InternetGetCookieExA`	Retrieve a cookie (extended)	Retrieve cookies from CPLS (extended)
`InternetSetCookieA`	Set a cookie	Set cookies for CPLS
`InternetSetCookieExA`	Set a cookie (extended)	Set cookies for CPLS (extended)
`InternetGetDomainNameA`	Retrieve Internet domain name	Retrieve CPLS domain name

The rationale behind extracting these API calls is that they provide functionalities that can be exploited to facilitate communication between malware and CPLS servers. For instance, API calls like `InternetOpenA`, `InternetConnectA`, and `HttpOpenRequestA` can establish connections and open HTTP requests to CPLS. Similarly, functions such as `InternetReadFile`, `InternetWriteFile`, `WinHttpReadData`, and `WinHttpWriteData` enable reading from and writing to files on CPLS, which can facilitate data exfiltration or retrieval of additional malicious payloads.

It is important to note that while these API calls are commonly associated with C&C communication, they are not exclusive to botnets or HTTP traffic. Many of these calls can be used for legitimate purposes or in non-HTTP protocols. Our approach considers these calls as potential indicators when combined with other features, rather than definitive proof of malicious activity. A limitation of focussing on these calls is the potential to miss C&C communication using custom communication protocol methods.

By examining the Import Address Table (IAT) of a PE file for the presence of these API calls, we can identify executables capable of interacting with CPLS in ways commonly associated with C&C communication. While the presence of these APIs alone does not necessarily indicate malicious behavior, when combined with other features and analyzed using machine learning techniques, they can contribute to accurate detection of PE files abusing CPLS for C&C purposes.

The potential_C&C_api_calls feature is generated by analyzing the Import Address Table (IAT) of the PE file. The IAT contains information about the imported functions used by the executable. We examine the IAT for the presence of specific API function calls that are commonly used for C&C communication, such as those listed in Table 4.1. If any of these potential C&C API calls are found in the IAT, the feature value is set to 1; otherwise, it is set to 0. This feature helps identify PE files that may be utilizing APIs associated with C&C activities.

Algorithm 4.1 Feature Extraction Process

```
1:  function ExtractFeatures(list_of_samples, output_file)
2:      with open(output_file, 'w+') as csv_file:
3:          write_header(csv_file)
4:      for each sample in list_of_samples do
5:          pe ← pefile.PE(sample)
6:          write_raw_features(csv_file, pe)
7:          presence_of_CPLS_domains ← check_CPLS_domains(pe)
8:          potential_C2_api_calls ← check_api_calls(pe)
9:          write_derived_features(csv_file, presence_of_CPLS_domains,
10:                 potential_C2_api_calls)
11:     end for
```

Table 4.2: PE file fields and derived features for abuse classification.

Field	Description
COFF Header	• Machine: Type of machine that the object file is intended to run on. • NumberOfSections: Number of sections in the object file. • TimeDateStamp: Time and date that the object file was created.

Continued on next page

Table 4.2 – continued from previous page

Field	Description
Optional Header	• Linker version: Version of the linker that created the object file. • Code and data sizes: Sizes of the code and initialised and uninitialised data in the object file. • Entry point address: Address of the entry point for the object file. • ImageBase: Address of the executable in memory. • CheckSum: Value used to validate the integrity of the image. • DllCharacteristics: DLL characteristics of the executable. • Import Table: List of DLLs and functions imported by the executable that can provide information about the functionality of the executable and indicate potential malicious behaviour. • Export Table: List of functions exported by the executable that can provide information about the functionality of the executable and indicate potential malicious behaviour. • Resource Directory: Resources used by the executable, such as icons, cursors, and bitmaps, that can provide information about the appearance and behaviour of the executable and indicate potential malicious behaviour. • Relocation Table: Information used by the linker to adjust addresses in the code when the executable is loaded into memory that can provide information about how the executable is organised and indicate potential malicious behaviour. • Debug Information: Information used by debuggers to help debug the executable that can provide information about the internal structure of the executable and indicate potential malicious behavior.

Continued on next page

Table 4.2 – continued from previous page

Field	**Description**
Section Table	• Name: Name of the section. • VirtualSize: Size of the section in memory. • VirtualAddress: Address of the section in memory. • SizeOfRawData: Size of the section in the object file. • PointerToRawData: Location of the section in the object file.
Derived Features	• presence_of_CLS_domains: If any of the CLS domains appear in sections of the PE file, the value is 1; otherwise, it is 0. • potential_C&C_api_calls: If any of the potential C&C API calls appear as an import function in the PE file, the value is 1; otherwise, it is 0.

4.2.4 Feature Selection

To identify the most relevant and informative features from the extracted set, we employed a combination of filter-based and wrapper-based feature selection techniques. The filter-based methods included information gain (InfoGain), chi-squared, and ReliefF, while the wrapper-based methods utilized random forest (RF) and decision tree (DT) as foundational models, paired with sequential feature selector forward (SFSF), sequential feature selector backward (SFSB), and recursive feature elimination (RFE) strategies.

Filter-based Methods

For the filter-based methods, we chose information gain (InfoGain), chi-squared, and ReliefF due to their proven effectiveness in the literature. InfoGain has been widely used and has been shown to be efficient in malware detection studies [148, 151, 79]. Similarly, ReliefF has been successfully employed in the context of malware analysis [170]. These methods are well-established and have shown promising results in identifying informative features for malware detection tasks. Filter-based methods rank features according to their relevance to the label class but do not specify the exact number of features to use. To address this, we developed and implemented a Custom Feature Elimination (CFE) technique 4.2.4 on each of the filter-based methods to determine the optimal number of features to use.

Custom Feature Elimination (CFE) Technique We developed and implemented a CFE technique to determine the optimal number of features for the filter-based methods. The CFE technique leverages the feature importance rankings provided by InfoGain, chi-squared, and ReliefF, and iteratively evaluates the impact of adding features on the classification accuracy using a random forest classifier and 10-fold stratified cross-validation.

The CFE technique follows these steps:

1. Initialize an empty set of selected features.
2. Obtain the feature importance rankings from a filter-based method (e.g., InfoGain, chi-squared, or ReliefF).
3. Iterate through the ranked features:
    a. Append the current feature to the temporary set of selected features.
    b. Train a random forest classifier using the temporary feature set.
    c. Evaluate the classifier's accuracy using 10-fold stratified cross-validation.
    d. If the accuracy improves compared to the previous iteration:
        - Add the current feature to the set of selected features.
        - Record the accuracy progress.

e. If there is no improvement in accuracy:

- Terminate the process.
- Return the set of selected features.

4. The final set of selected features represents the optimal subset that maximizes classification accuracy.

Accuracy Progress and Selected Features Figures 4.2, 4.3, and 4.4 illustrate the accuracy progress and the subset of features selected by the CFE technique using InfoGain, chi-squared, and ReliefF, respectively.

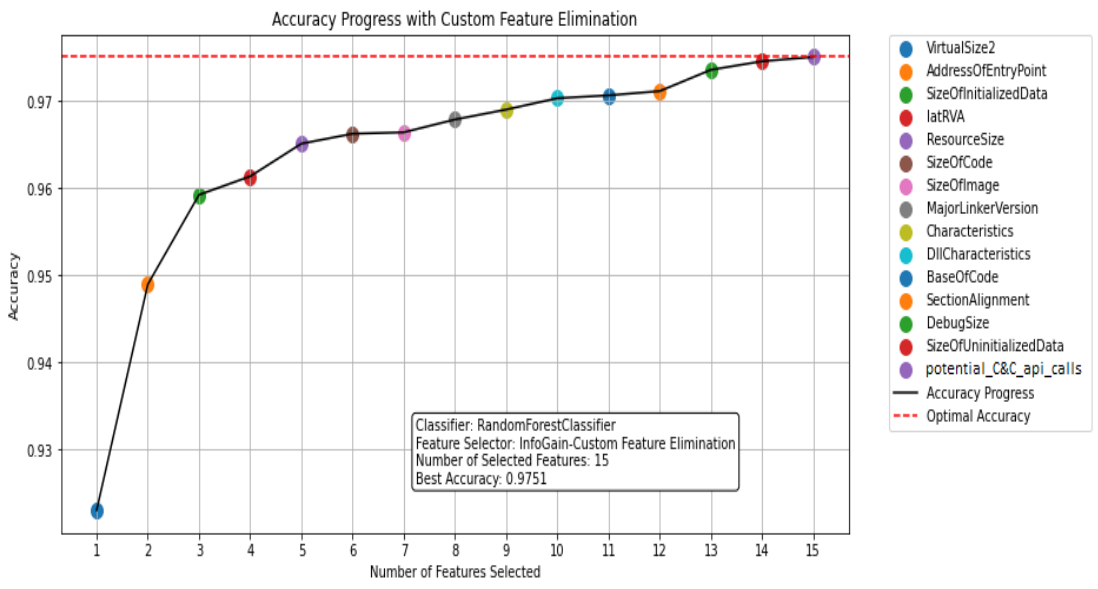


Figure 4.2: Accuracy and subset of features using InfoGain.

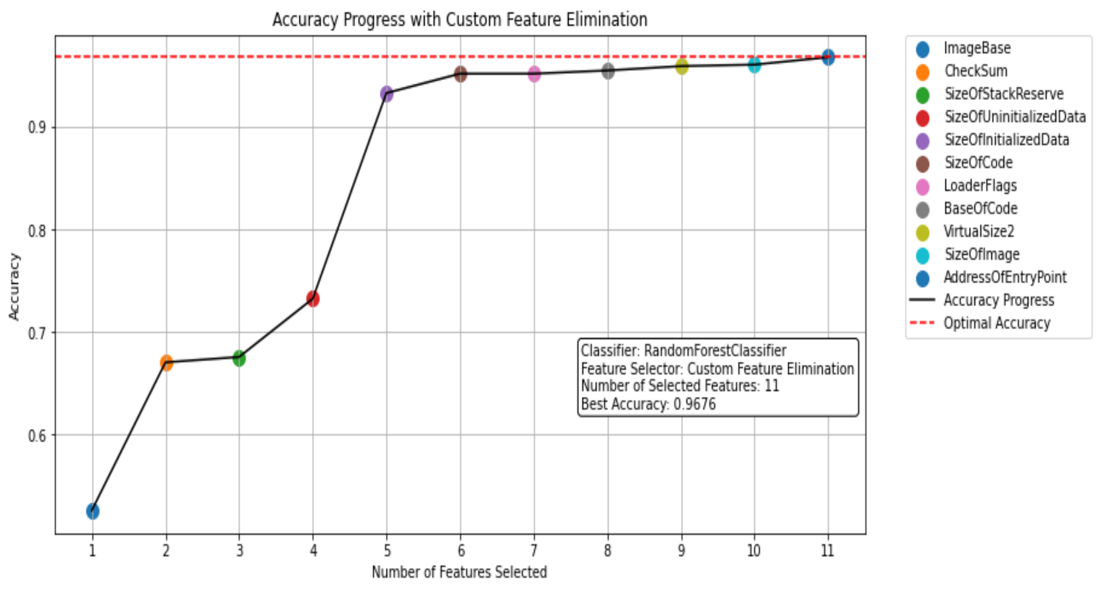


Figure 4.3: Accuracy and subset of features using chi-squared.

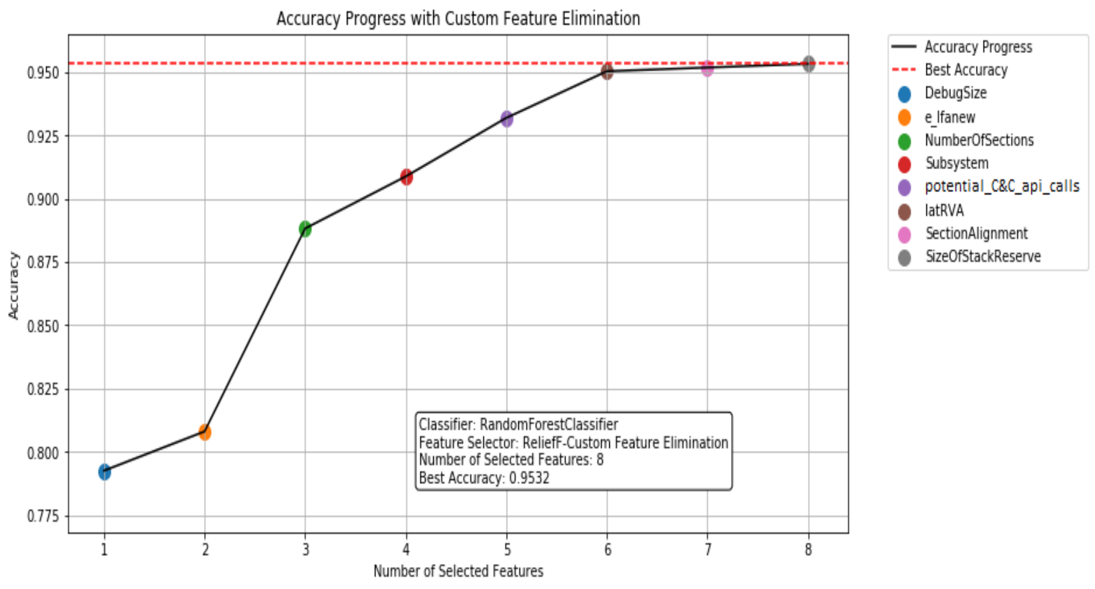


Figure 4.4: Accuracy and subset of features using ReliefF.

Wrapper-based Methods

For the wrapper-based methods, we evaluated each combination of the foundational model (RF or DT) and feature selection strategy (SFSF, SFSB, or RFE) to find the optimal subset of features that maximizes the model's performance. The three feature selection strategies are described below:

1. Sequential Forward Selection (SFSF)

- SFSF starts with an empty set of features and iteratively adds features one by one.
- In each iteration, SFSF evaluates the performance of the model with each candidate feature added to the current subset.
- The feature that yields the best improvement in the model's performance is selected and added to the subset.
- The process continues until adding more features does not improve the model's performance or until a desired number of features is reached.

2. Sequential Backward Selection (SFSB)

- SFSB starts with the full set of features and iteratively removes features one by one.
- In each iteration, SFSB evaluates the performance of the model with each candidate feature removed from the current subset.
- The feature whose removal yields the least degradation in the model's performance is selected and removed from the subset.
- The process continues until removing more features does not improve the model's performance or until a desired number of features is reached.

3. Recursive Feature Elimination (RFE)

- RFE recursively removes less important features based on a feature ranking criterion.
- It starts with the full set of features and trains the model.

- The features are ranked based on their importance (e.g., using feature coefficients or feature importances).
- The least important features are removed, and the model is retrained with the remaining features.
- The process is repeated until a desired number of features is reached.

Optimal Feature Subset Selection The RF-SFSF approach outperformed the other five wrapper-based approaches, achieving the highest accuracy rate of 98.26% with 22 out of 38 features. This finding suggests that the sequential forward selection strategy, combined with the random forest model, was able to identify the most informative subset of features that maximized the model performance. While other feature selection methods, such as InfoGain, identified smaller feature sets (15 features), they did not achieve the same level of accuracy. Our comprehensive evaluation of different feature selection methods demonstrates that 22 features, as selected by RF-SFSF, provide the optimal balance between model complexity and detection accuracy.

Figures 4.5 and 4.6 present a comparative analysis of the wrapper-based feature selection methods using random forest and decision tree as the foundational models, respectively. The optimal feature count for maximum accuracy is highlighted by dotted lines in each figure.

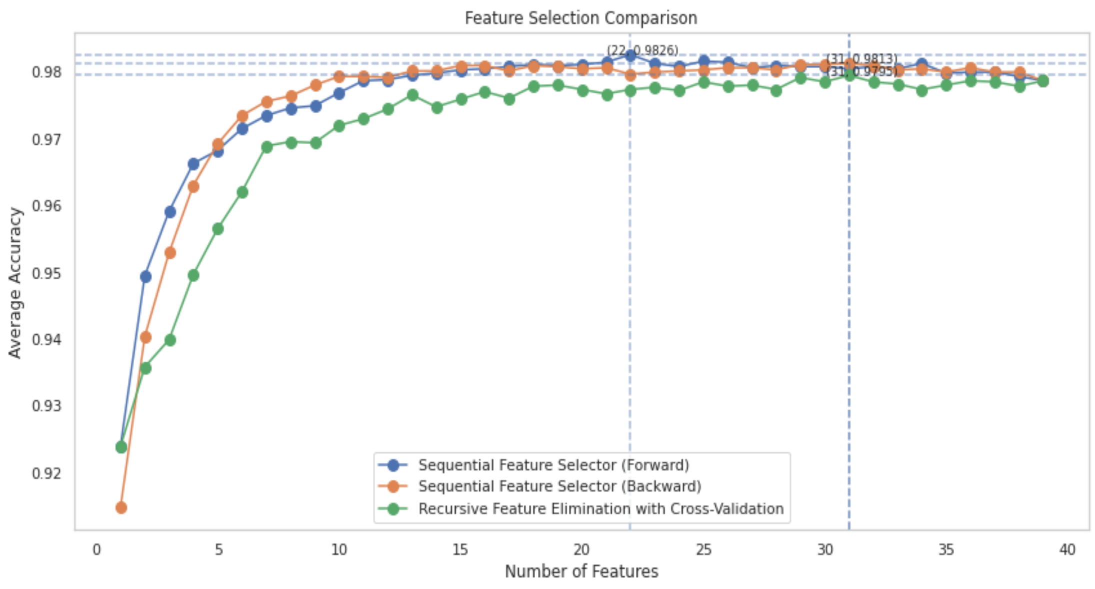


Figure 4.5: Comparative analysis of wrapper feature selection: RF-SFSF vs. RF-SFSB vs. RF-RFE, highlighting the optimal feature count for maximum accuracy as indicated by dotted lines.

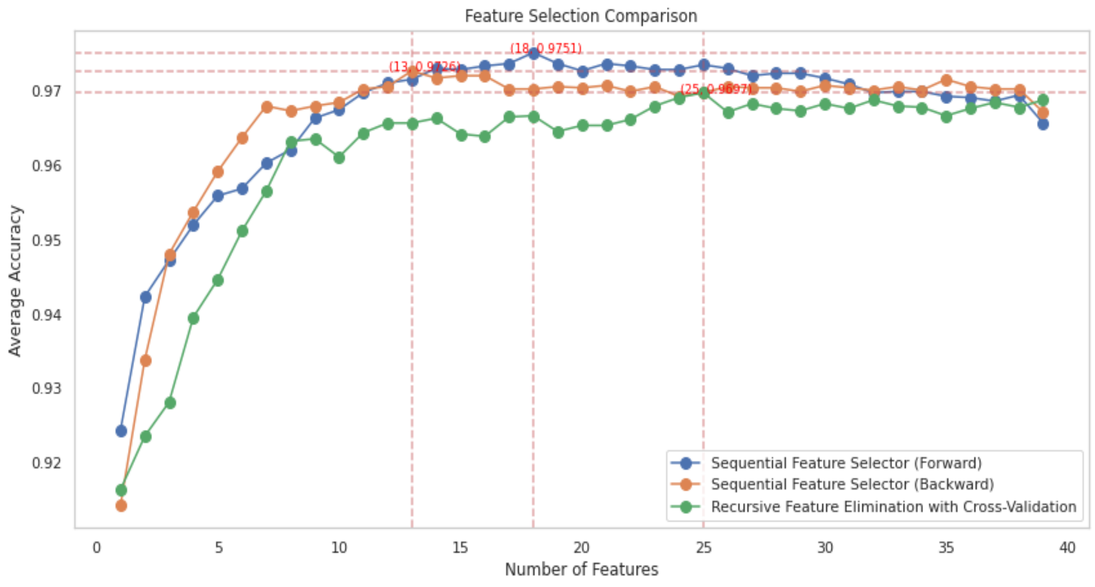


Figure 4.6: Comparative analysis of wrapper feature selection: DT-SFSF vs. DT-SFSB vs. DT-RFE, highlighting the optimal feature count for maximum accuracy as indicated by dotted lines.

By employing both filter-based and wrapper-based feature selection methods, we were able to identify the most informative and relevant features for malware detection. The custom feature elimination (CFE) technique for filter-based methods and the RF-SFSF approach for wrapper-based methods demonstrated their effectiveness in selecting optimal feature subsets that maximize classification accuracy.

4.2.5 Classification

Our evaluation applies five ML-based classifiers, which are implemented in Scikit-learn [22]: namely decision tree (J48), random forest (RF), naïve Bayes (NB), k-nearest neighbors (K-NN), and support vector machine (SVM). These classifiers were selected for their diverse underlying algorithms and their widespread usage in malware detection literature [148, 137, 170]. By employing a range of classifiers, we aim to comprehensively assess the performance of various ML classifiers on our dataset, which is detailed in Section 5.3.1.

To measure the accuracy of these classifiers, we utilized two distinct evaluation techniques: 10-fold cross-validation and a training-to-testing split ratio of 70:30. The evaluations were conducted on an ASUS computer equipped with an Intel i7-9700K processor with a clock speed of 3.60 GHz, supported by 32 GB of RAM, and running the Windows 10 operating system.

4.3 Experiments

4.3.1 Overview of the Detection System

Figure 4.7 offers an illustrative overview to provide a clear and comprehensive understanding of the proposed detection system. This visualisation encapsulates the process from data collection through feature extraction, leading up to the application of machine learning techniques to identify CPLS abuse as C&C infrastructure. The figure illustrates our systematic approach, highlighting key components and their interactions within the detection system.

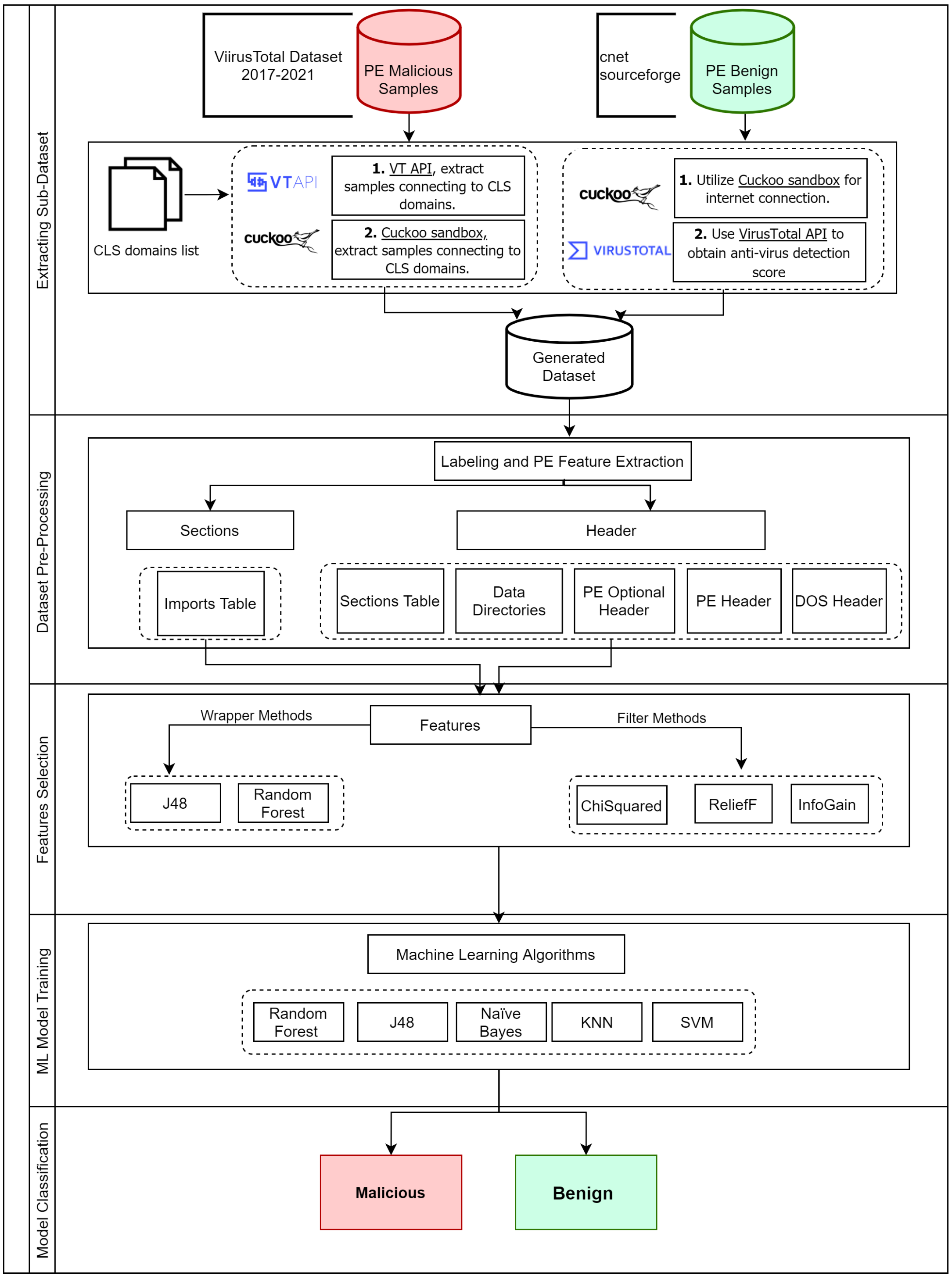


Figure 4.7: Illustrative overview of the proposed detection system.

4.3.2 Classification

Our evaluation applies five ML-based classifiers, which are implemented in Scikit-learn [22]: namely decision tree (J48), random forest (RF), naïve Bayes (NB), k-nearest neighbors (K-NN), and support vector machine (SVM). These classifiers were selected for their diverse underlying algorithms and their widespread usage in malware detection literature [148, 137, 170]. By employing a range of classifiers, we aim to comprehensively assess the performance of various ML classifiers on our dataset, which is detailed in Section 5.3.1.

To measure the accuracy of these classifiers, we utilised two distinct evaluation techniques: 10-fold cross-validation and a training-to-testing split ratio of 70:30. The evaluations were conducted on an ASUS computer equipped with an Intel i7-9700K processor with a clock speed of 3.60 GHz, supported by 32 GB of RAM, and running the Windows 10 operating system.

4.3.3 Evaluation

This stage includes the evaluation of the classification results, comparison with related works, and robustness evaluation against adversarial attacks.

Experimental Results of All Features

The classifiers were analysed and compared with regard to detection accuracy. We present the detection accuracy of five classifiers, all utilising the extracted features for classification, as demonstrated in Tables 4.3 and 4.4.

The results show that the random forest classifier outperforms the other classifiers, achieving a high detection accuracy of 97.77% in the 70:30 split scenario and 97.80% in the 10-fold cross-validation scenario. The J48 and K-NN classifiers also demonstrate high detection accuracy rates, with detection rates of 96.41% and 93.12%, respectively, in the 70:30 split scenario, and 96.84% and 93.80%, respectively, in the 10-fold cross-validation scenario.

However, the NB and SVM classifiers show significantly lower accuracy compared to the other classifiers, indicating that they may not be suitable for this dataset. The NB classifier has detection accuracies of 65.13% and 63.76% in the 70:30 split and 10-fold cross-validation techniques, respectively. The SVM classifier has even lower detection accuracies of 52.47% and 52.54% in the 70:30 split and 10-fold cross-validation techniques, respectively.

Ultimately, when considering all extracted features, the results demonstrate that the RF classifier, combined with the selected set of features, is suitable for accurately detecting the abuse of CPLS for C&C infrastructure in our dataset.

Table 4.3: Comparative analysis of abuse detection accuracy: all features vs. selected features (Part 1). Here, (70:30) denotes a split of 70% training data and 30% testing data, while (10-fold) stands for 10-fold cross-validation.

Classifier	**All Features**		**Feature Selector**			
			Wrapper Methods			
			RF as Feature Selector		**J48 as Feature Selector**	
	70:30	**10-fold**	**70:30**	**10-fold**	**70:30**	**10-fold**
J48	96.41%	96.84%	96.80%	96.77%	96.20%	96.90%
RF	97.77%	97.80%	98.15%	**98.26%**	97.45%	97.90%
NB	65.13%	63.76%	63.50%	63.06%	54.16%	52.87%
KNN	93.21%	93.80%	92.67%	93.63%	92.07%	92.70%
SVM	52.47%	52.54%	51.11%	52.48%	51.11%	52.53%
Selected Features	AddressOfEntryPoint, SizeOfCode, SizeOfInitializedData, SizeOfUninitializedData, BaseOfCode, MajorLinkerVersion, MajorImageVersion, MajorOperatingSystemVersion, DllCharacteristics, SizeOfStackReserve, NumberOfSections, ImageBase, SectionAlignment, FileAlignment, MinorOperatingSystemVersion, MinorImageVersion, MajorSubsystemVersion, MinorSubsystemVersion, SizeOfImage, SizeOfHeaders, CheckSum, Subsystem, SizeOfStackCommit, SizeOfHeapReserve, SizeOfHeapCommit, LoaderFlags, NumberOfRvaAndSizes, SizeOfOptionalHeader, Characteristics, Machine, e_lfanew, DebugSize, ExportSize, VirtualSize2, ResourceSize, IatRVA, presence_of_CLS_domains, potential_C&C_api_calls		AddressOfEntryPoint, MajorLinkerVersion, MajorImageVersion, MajorOperatingSystemVersion, DllCharacteristics, SizeOfStackReserve, NumberOfSections, ImageBase, SectionAlignment, FileAlignment, MinorOperatingSystemVersion, MajorSubsystemVersion, SizeOfImage, CheckSum, Subsystem, SizeOfHeapCommit, NumberOfRvaAndSizes, Machine, DebugSize, VirtualSize2, IatRVA, potential_C&C_api_calls		AddressOfEntryPoint, SizeOfUninitializedData, MajorLinkerVersion, MajorOperatingSystemVersion, DllCharacteristics, SizeOfStackReserve, NumberOfSections, ImageBase, SectionAlignment, FileAlignment, MinorOperatingSystemVersion, MinorImageVersion, MajorSubsystemVersion, MinorSubsystemVersion, SizeOfImage, CheckSum, SizeOfStackCommit, SizeOfHeapCommit, NumberOfRvaAndSizes, SizeOfOptionalHeader, DebugSize, VirtualSize2, IatRVA, presence_of_CLS_domains	

Table 4.4: Comparative analysis of abuse detection accuracy: all features vs. selected features (Part 2). Here, (70:30) denotes a split of 70% training data and 30% testing data, while (10-fold) stands for 10-fold cross-validation.

Classifier	Feature Selector					
	Filter Methods					
	InfoGain		ChiSquared		ReliefF	
	70:30	10-fold	70:30	10-fold	70:30	10-fold
J48	95.87%	95.83%	94.19%	95.04%	94.51%	94.93%
RF	97.12%	97.47%	96.47%	96.59%	95.93%	95.32%
NB	52.74%	52.38%	65.02%	65.57%	59.32%	64.18%
KNN	91.85%	92.96%	93.16%	93.77%	93.75%	93.81%
SVM	53.72%	53.52%	52.47%	52.54%	61.11%	60.68%
Selected Features	VirtualSize2, AddressOfEntryPoint, SizeOfInitializedData, IatRVA, ResourceSize, SizeOfCode, SizeOfImage, MajorLinkerVersion, Characteristics, DllCharacteristics, BaseOfCode, SectionAlignment, DebugSize, SizeOfUninitializedData, potential_C&C_api_calls		ImageBase, CheckSum, SizeOfStackReserve, SizeOfUninitializedData, SizeOfInitializedData, SizeOfCode, LoaderFlags, BaseOfCode, VirtualSize2, SizeOfImage, AddressOfEntryPoint		DebugSize, e_lfanew, NumberOfSections, Subsystem, potential_C&C_api_calls, IatRVA, SectionAlignment, SizeOfStackReserve	

Experimental Results on Selected Features

It is important to note that using all extracted features may not always be optimal, and feature selection techniques may be necessary to improve classification accuracy. Therefore, this section discusses the result of an accuracy detection rate achieved with selected features. Tables 4.3 and 4.4 compare the detection accuracy of various classifiers after feature selection, employing wrapper and filter techniques.

Regarding detection accuracy, the RF wrapper technique consistently outperforms other feature selection methods, whether the data is split into a 70:30 training-to-testing ratio or subjected to 10-fold cross-validation. For instance, the detection accuracy with RF as a feature selector is 98.15% for the 70:30 split and 98.04% for 10-fold cross-validation, whereas the detection accuracy with J48 as a feature selector is 96.80% for the 70:30 split and 96.77% for 10-fold cross-validation. Among the filter methods, InfoGain and ReliefF perform better than chi-squared. For example, the detection accuracy with InfoGain as a feature selector is 97.12% for the 70:30 split and 97.47% for 10-fold cross-validation, while the accuracy using chi-squared as a feature selector is 96.47% for the 70:30 split and 96.59% for 10-fold cross-validation.

Figures 4.5 and 4.6 show the mean accuracy rate obtained by each combination using a different number of selected features ranging from 1 to the maximum number of 38 features in the dataset. The RF-SFSF approach outperformed the other five wrapper-based approaches, achieving the highest accuracy rate of 98. 26% with 22 of 38 characteristics.

The superior performance of the Random Forest classifier can be attributed to several factors:

- **Ensemble learning**: Random Forest is an ensemble method that combines multiple decision trees, which helps to reduce overfitting and improve generalisation.
- **Feature importance**: Random Forest can effectively handle high-dimensional data and identify the most important features for classification, which is particularly useful given our comprehensive feature set.
- **Reduced variance**: By averaging the predictions of multiple trees, Random Forest reduces the variance in the predictions, leading to more stable and accurate results.

While other ensemble methods like AdaBoost or Gradient Boosting could potentially perform well for this task, Random Forest's specific characteristics made it particularly suitable for our CPLS abuse detection system. Future research could explore the performance of other ensemble methods in this context.

These characteristics make Random Forest particularly suitable for the complex task of detecting CPLS abuse as C&C infrastructure.

Our analysis revealed that these characteristics were particularly beneficial for CPLS abuse detection. The ensemble learning approach proved especially effective in handling the complex patterns of CPLS abuse, as evidenced by the consistently high accuracy across both validation techniques (98.15% for 70:30 split and 98.26% for 10-fold cross-validation). The model's ability to identify important features was reflected in its successful prioritization of key indicators like potential_C&C_api_calls, which emerged as one of the most influential features in our analysis.

Upon comparing the results of the filter-based and wrapper-based methods, we concluded that the RF-SFSF approach is the most compelling feature selection method for this dataset. The final subset of features employed in our study is detailed in Table 4.3.

Comparison with Deep Learning Approaches

Deep Learning (DL) has gained significant popularity in various domains, including malware detection, due to its ability to automatically learn hierarchical representations from raw data. However, in our study, we opted to use traditional machine learning techniques, particularly the Random Forest (RF) classifier, instead of DL for the following reasons:

- Performance: Our experiments demonstrated that the RF classifier, in conjunction with wrapper-based selected features, achieved a high detection rate of 98.26%. This performance surpassed the results obtained using DL approaches. As shown in Figure 4.8, the DL model achieved a test accuracy of 95.19% and validation accuracy of 95.93%, both of which are lower than the 98.26% achieved by the RF classifier.

- Feature engineering: Our approach involved careful feature engineering, including the extraction of both raw and derived features from PE files. The RF classifier, combined with effective feature selection techniques, was able to leverage these handcrafted features to achieve high detection accuracy. DL, on the other hand, typically relies on automatically learned representations from raw data, which may not always capture the domain-specific knowledge incorporated through manual feature engineering.

- Computational resources: DL models often require significant computational resources, including powerful GPUs, for training and inference. In contrast, the RF classifier can be trained and deployed efficiently on standard hardware, making it more practical for real-world deployment.

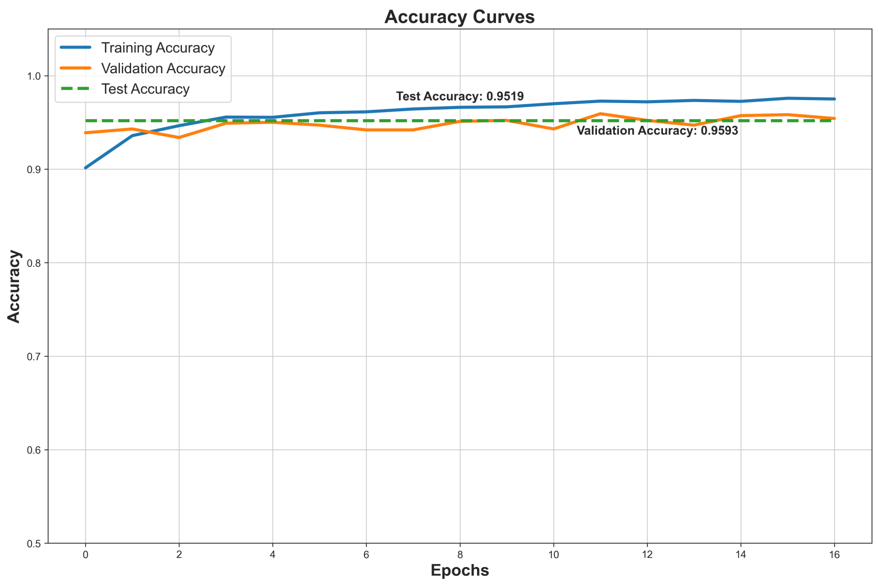


Figure 4.8: Accuracy curves for the DL model, showing the validation and testing accuracy over epochs.

Comparison to Related Works

Detection Accuracy Comparison Given the limited research on using ML techniques to detect abuse of the CLS as a C&C infrastructure, we conduct a comparative analysis with existing studies that identify general malware using PE file properties as ML features, similar to the studies by Kumar et al. [126] and Raman et al. [145]. It is essential to highlight that the evaluations for both our work and the other two studies were based on our unique dataset.

Table 4.5 presents the results of the comparative analysis. With a detection accuracy of 98%, our proposed work outperforms the other two studies, which posted rates of 94% and 95%, respectively. This improved performance is attributed to our unique approach of leveraging both raw and derived features from PE files. Notably, the derived features, presence_of_CLS_domains and potential_C&C_api_calls, played a significant role in enhancing the detection efficacy.

Table 4.5: Comparison of abuse detection accuracy between proposed and existing works.

Reference	J48				Random Forest			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
Kumar et al. [126]	94.46	94.90	93.63	94.26	94.68	94.93	94.08	94.50
Raman et al. [145]	94.75	93.83	95.08	94.45	95.65	96.36	94.64	95.49
Proposed work	96.80	96.04	97.43	96.73	98.26	98.75	97.43	98.09

Importance of Derived Features

In our study, we evaluated a total of 38 pertinent features, of which 36 were raw PE features and two were derived features. Utilising various feature selectors, as discussed in Section 4.2.4, our results identified a combination of 21 raw features and the derived feature potential_C&C_api_calls to yield the highest classification accuracy. Detailed results and comparisons before and after inclusion of the derived features can be perused in Table 4.6, and Tables 4.3 and 4.4. Our final model, which delivered the highest accuracy, employed the RF classifier in conjunction with the aforementioned feature selector. We also charted the importance of each feature. This plot offers a visual representation of the relative significance of each feature as depicted in Figure 4.9.

Table 4.6: Comparison of detection accuracy: evaluating the impact of including derived features.

Features Used	Validation Approach	
	70:30 Split	**10-Fold CV**
Optimal feature using RF feature selector (excluded potential C&C calls)	0.977186	0.981416
+ presence_of_CLS_domains	0.977729	0.980600
+ potential_C&C_api_calls	0.981532	0.982557
Included derived features	0.978273	0.986601

Although the presence_of_CLS_domains feature may not have prominently boosted the classification accuracy, as indicated in Table 4.6, its relevance stems from our data collection approach, detailed in Section 5.3.1. In particular, the dynamic analysis phase of our data collection was designed to extract a subset from the VirusTotal dataset that predominantly connects CLS domains.

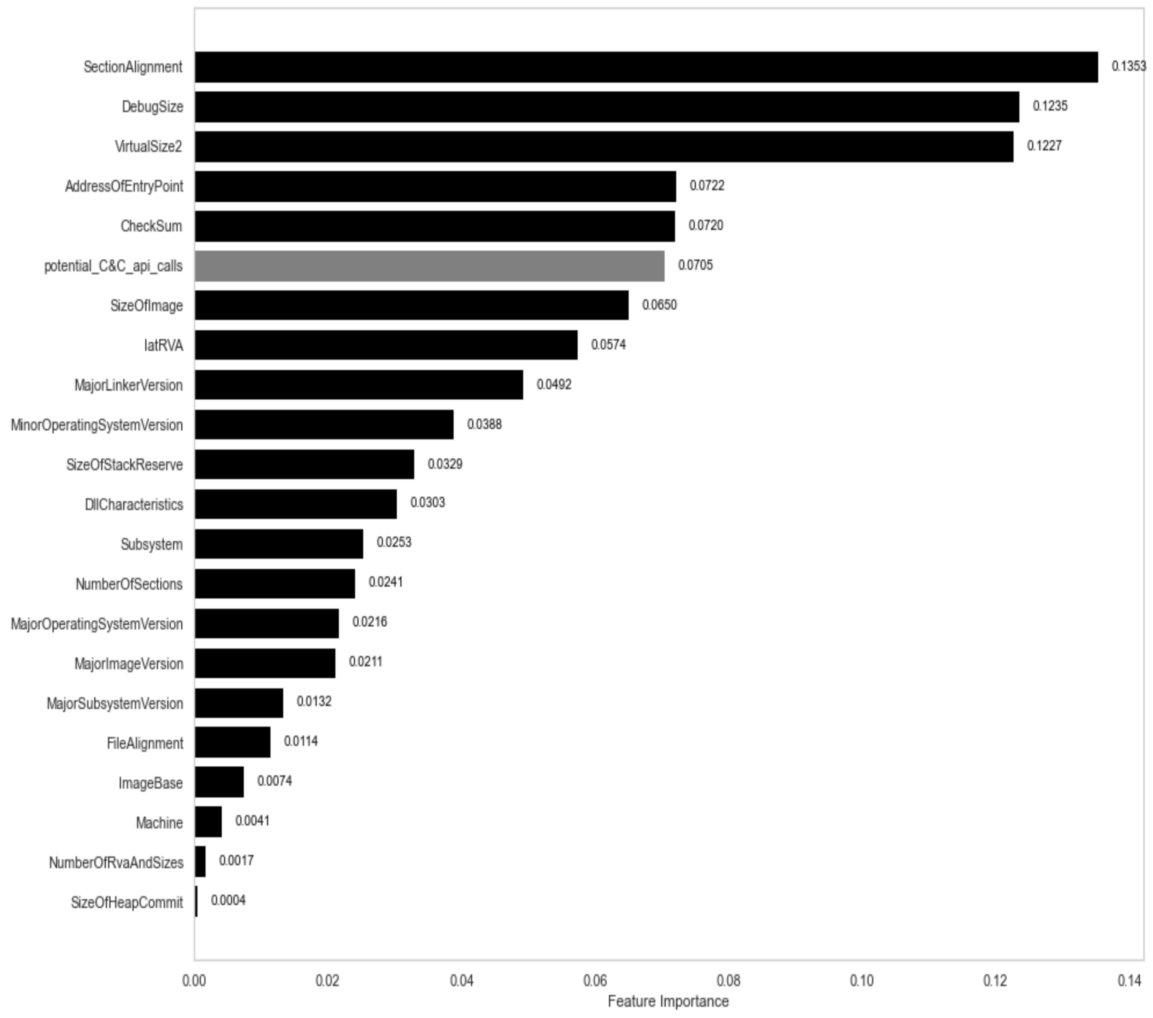


Figure 4.9: Feature importance highlighting the prominence of "potential_C&C_api_calls" among the top influential features in the model.

Robustness Evaluation For a detailed analysis of our model's robustness against adversarial attacks, please refer to Chapter 6, titled "Robustness of CPLS Abuse Detection System." In summary, to compare the robustness of our ML models with related works, we propose the Replace Misclassified Parameter (RMCP) as a novel white-box adversarial attack. Our proposed model demonstrates a robustness rate of 83.54% against RMCP, outperforming the 69.03% and 54.37% achieved by Kumar et al. [126] and Raman et al. [145], respectively. These findings highlight the resilience of our work against adversarial attacks, offering a more robust and reliable solution for abuse detection.

4.4 Summary

In this chapter, we presented a novel approach that utilised machine learning techniques on static features extracted from PE files to detect the abuse of CPLS as a C&C infrastructure.

Through our experiments, the RF classifier combined with wrapper-based selected features achieved the highest detection rate of 98.26%, outperforming related works such as Kumar et al. [126] and Raman et al. [145], who reported detection rates of 94% and 95%, respectively. This demonstrates the effectiveness of our proposed approach in comparison to existing words. Our work highlights the potential of machine learning-based techniques to detect the abuse of CPLS as a C&C infrastructure effectively. By delivering a robust detection system, we contribute to the ongoing efforts to combat such sophisticated cyber threats.

4.5 Conclusion and Limitations

In this chapter, we addressed the critical issue of detecting the abuse of CPLS as a C&C infrastructure using ML techniques applied to static features extracted from PE files. Our primary objective was to develop an effective and robust detection system to combat this sophisticated cyber threat. Through extensive experiments, we demonstrated that the RF classifier, in conjunction with wrapper-based selected features, achieved the highest detection rate of 98.26%. This finding underscores the efficacy of our proposed approach in identifying CPLS abuse. The implications of our research are significant for the cybersecurity domain. By providing a reliable and accurate detection system, our work contributes to the ongoing efforts to mitigate the risks associated with the abuse of CPLS as a C&C infrastructure. However, it is essential to acknowledge the limitations of our research. The encryption of a PE can pose significant challenges for our technique, especially when attempting to extract pivotal features like presence_of_CLS_domains and potential_C&C_api_calls. To address these limitations and further enhance the detection of CPLS abuse, we explore a dynamic analysis approach combined with machine learning techniques in Chapter 5. Dynamic analysis focuses on observing a program's real-time execution behavior, yielding in-depth insights into its functionality and interactions with other systems. By applying machine learning algorithms to the features extracted through dynamic analysis, we aim to uncover patterns and behaviors that static analysis alone might miss.

In conclusion, our research demonstrates the potential of ML-based techniques in detecting the abuse of CPLS as a C&C infrastructure.

Chapter 5

Detecting CPLS abuse as C&C infrastructure with Dynamic features

5.1 Introduction

This chapter presents our comprehensive approach to detecting the abuse of CPLS as C&C infrastructure using dynamic analysis and ML. The focus of our research is on the application of machine learning techniques to dynamic features extracted from Portable Executable (PE) files. Our methodology is a multi-stage process that includes data collection, feature extraction, machine learning model development, and evaluation stages. In the data collection stage, we assemble a dataset sourced from VirusTotal, focusing specifically on PE files that misuse the CPLS. This dataset comprises both malicious samples that exhibit communication with known CPLS-hosted domains and benign samples obtained from trusted sources. The feature extraction stage introduces an innovative technique called "Dual-Sandboxing" for advanced feature extraction, which employs the Cuckoo and Triage tools to enhance feature extraction. This dual-sandboxing approach gathers a broad spectrum of features related to both system interactions and network activities, providing a rich dataset for machine learning applications. In the ML model stage, we evaluate various algorithms including Decision Tree (J48), Naïve Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Random Forest. We apply both a 10-fold cross-validation and a 70/30 split to our dataset, ensuring the robustness of our predictive models. Finally, in the evaluation stage, we assess the performance of our proposed approach for detecting CPLS abuse as C&C infrastructure. We conduct experiments involving the application of the Dual-Sandboxing technique for feature extraction and evaluate the performance of various machine learning algorithms on the extracted features. Each of these stages is detailed in the following sections, providing a thorough overview of our approach to detecting CPLS abuse as C&C infrastructure. Our findings demonstrate the effectiveness of our proposed approach and highlight the significance of the comprehensive feature set obtained through the Dual-Sandboxing technique.

5.2 Dynamic Malware Analysis Fundamentals

Before detailing our innovative Dual-Sandboxing technique, it is crucial to understand the principles of dynamic malware analysis that underpin our approach. Dynamic malware analysis involves executing malware samples in a controlled environment to observe their behavior and interactions with the system and network resources. This analysis method contrasts with static analysis, as it allows for the collection of runtime data, including system calls, registry changes, network traffic, and other real-time interactions that malware engages in during execution.

Dynamic analysis offers several advantages for detecting sophisticated malware, especially those utilizing CPLS for C&C purposes:

- **Behavioral Insights:** By observing the malware in action, we can identify malicious patterns and behaviors that may not be apparent through static analysis alone.
- **Evasion Detection:** Many advanced malware samples employ techniques such as encryption, polymorphism, or other obfuscation methods to evade static analysis. These techniques can make it challenging for static analysis to identify malicious behaviours or patterns. Dynamic analysis, on the other hand, can potentially uncover these evasion techniques by analysing the malware's runtime behaviour, as the malware must decrypt or deobfuscate itself during execution, revealing its true intentions.
- **Network Communication:** Dynamic analysis is particularly effective in studying malware that uses network communication for C&C activities. It allows for the monitoring of network traffic to identify suspicious connections and data transfers to CPLS-hosted domains.

Despite its advantages, dynamic analysis also faces challenges, including evasion tactics specifically designed to detect and alter its malicious behavior accordingly in analysis environments. Our Dual-Sandboxing technique addresses some of these challenges by combining the strengths of two sandbox environments to capture a comprehensive set of dynamic features. By leveraging the capabilities of both Cuckoo Sandbox and Triage Sandbox, our approach enhances the resilience against evasion attempts and provides a more robust analysis of malware behavior.

5.3 Methodology

The methodology section is divided into three main components. The first, "Data Collection," details the process of assembling the dataset. The second part introduces a new technique called "Dual-Sandboxing for Advanced Feature Extraction," which employs the Cuckoo and Triage tools to enhance feature extraction. This dual-sandboxing approach gathers a broad spectrum of features related to both system interactions and network activities. The third component, "Feature Selection," assesses the necessity of including the entire set of features or only a subset for our machine learning model development. This involves the use of filter-based methods like Information Gain and ReliefF, as well as wrapper-based strategies such as Random Forest and J48.

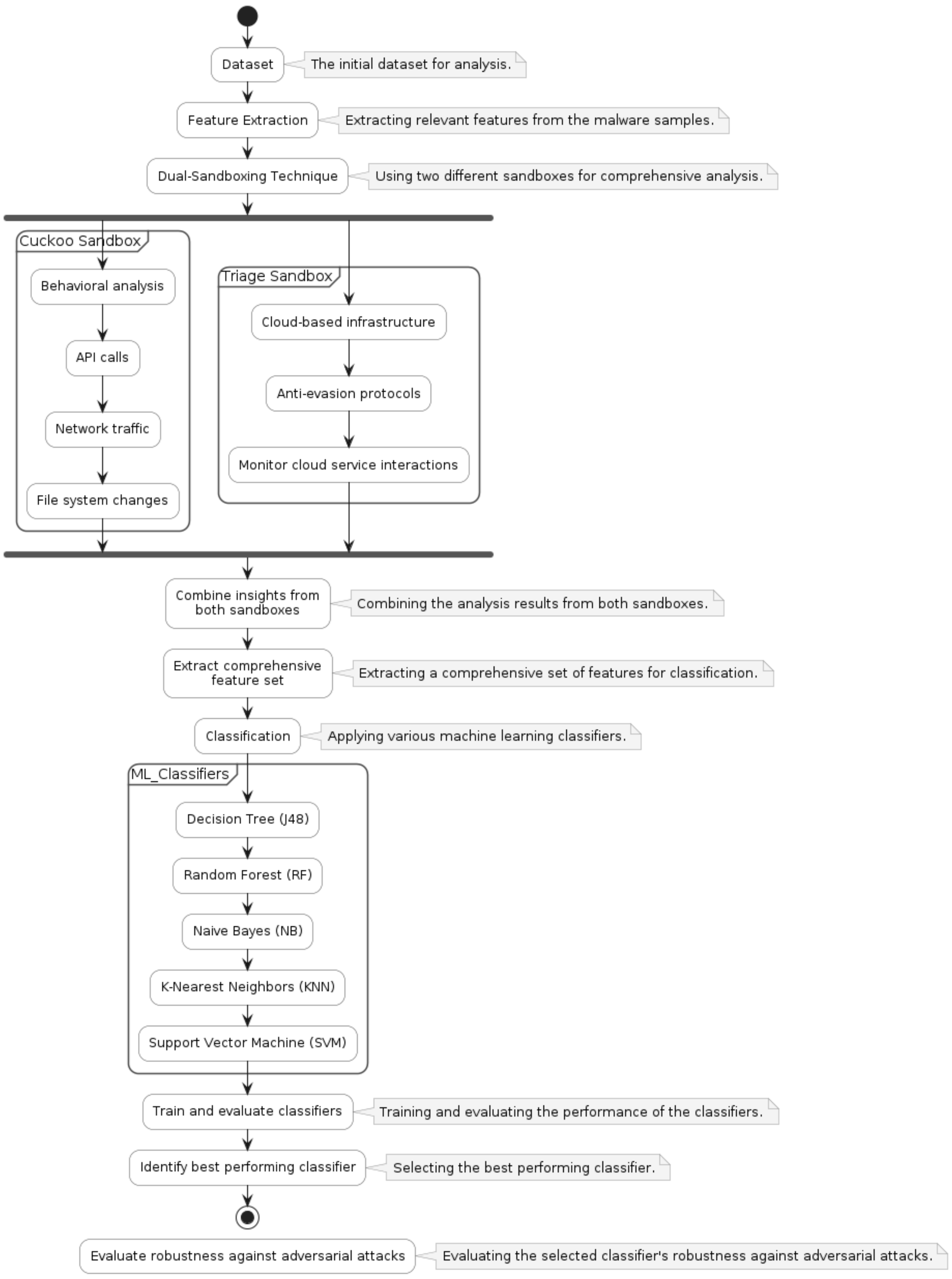

Dataset
The initial dataset for analysis.
Feature Extraction
Extracting relevant features from the malware samples.
Dual-Sandboxing Technique
Using two different sandboxes for comprehensive analysis.
Cuckoo Sandbox
Behavioral analysis
API calls
Network traffic
File system changes
Triage Sandbox
Cloud-based infrastructure
Anti-evasion protocols
Monitor cloud service interactions
Combine insights from both sandboxes
Combining the analysis results from both sandboxes.
Extract comprehensive feature set
Extracting a comprehensive set of features for classification.
Classification
Applying various machine learning classifiers.
ML_Classifiers
Decision Tree (J48)
Random Forest (RF)
Naive Bayes (NB)
K-Nearest Neighbors (KNN)
Support Vector Machine (SVM)
Train and evaluate classifiers
Training and evaluating the performance of the classifiers.
Identify best performing classifier
Selecting the best performing classifier.
Evaluate robustness against adversarial attacks
Evaluating the selected classifier's robustness against adversarial attacks.

Figure 5.1: Workflow of the proposed dynamic analysis approach for detecting CPLS abuse as C&C infrastructure.

Figure 5.1 illustrates the workflow of our proposed dynamic analysis approach for detecting the abuse of CPLS as C&C infrastructure. The workflow outlines the key stages involved, including data collection, feature extraction using the Dual-Sandboxing technique, classification, and identification of the best performing classifier.

5.3.1 Data Collection

The dataset collection process is comprehensively detailed in Chapter 3. Briefly, we utilized a dataset sourced from VirusTotal [30] spanning the years 2017 to 2021, encompassing various malware formats. Our research focused specifically on Portable Executable (PE) files that misused Cloud and Public Legitimate Services (CPLS) for C&C infrastructure. The dataset comprised malicious samples exhibiting communication with known CPLS-hosted domains, as well as benign samples obtained from trusted sources. While the initial dataset was imbalanced, we ensured a balanced dataset by removing excess benign samples while preserving the characteristics of the retained ones.

5.3.2 Dual-Sandboxing for Advanced Feature Extraction

In our dataset, our objective is to extract features for ML applications. To this end, we introduce an innovative Dual-Sandboxing (DS) technique that combines the strengths of sandbox environments: Cuckoo and Triage. This method is tailored to capture a broad spectrum of features:

Cuckoo Sandbox Environment

The Cuckoo Sandbox [31] is a widely adopted open-source automated malware analysis system used for the dynamic analysis of malware samples in our research. It provides a controlled environment for executing malware samples in isolation, preventing any potential harm to the host system. Cuckoo Sandbox's extensive features and capabilities make it particularly suitable for our research on detecting the abuse of Cloud and Legitimate Services (CLS) for malicious command and control (C&C) infrastructure. One of the key advantages of Cuckoo Sandbox is its ability to perform comprehensive behavioral analysis. It monitors and logs the behavior of malware during execution, capturing detailed information such as API calls, network traffic, and file system changes. This behavioral data is crucial for identifying patterns and indicators of CLS abuse, as it allows us to observe the actual communication between the malware and CLS domains. Cuckoo Sandbox also generates detailed analysis reports, providing valuable insights into the behavior and characteristics of the analyzed malware samples. These reports facilitate the extraction of discriminative features that can be used for training machine learning models to detect CLS abuse effectively. In terms of scalability, Cuckoo Sandbox can handle a large number of malware samples, making it suitable for analyzing the diverse dataset collected from VirusTotal. Its efficient resource utilization and distributed architecture allow for parallel processing of samples, reducing the overall analysis time. Compared to other popular malware analysis sandboxes, such as Joe Sandbox or VMRay, Cuckoo Sandbox stands out for its open-source nature, extensive community support, and flexibility in customization. These factors, along with its robust features and integration capabilities, make Cuckoo Sandbox an ideal choice for our research on detecting CLS abuse as C&C infrastructure.

Triage Sandbox Environment

In addition to the Cuckoo Sandbox, we also employ the Triage Sandbox [46] as part of our Dual-Sandboxing (DS) technique. Triage Sandbox operates within a cloud-based infrastructure and is equipped with sophisticated anti-evasion protocols. Its primary role is to capture the malware's external communication vectors, focusing on monitoring the malware's engagements with cloud services and the internet, and identifying potential cloud platform abuse attempts. By leveraging Triage Sandbox's cloud-based infrastructure and anti-evasion capabilities, we can effectively analyze malware samples that may employ evasion techniques to evade detection or analysis within traditional sandboxes. Triage Sandbox's sophisticated protocols help mitigate these evasion tactics, ensuring that we can capture the malware's true behavior and communication patterns with cloud services. The integration of Triage Sandbox with Cuckoo Sandbox in our DS technique allows us to gain a comprehensive understanding of the malware's behavior, both within the host environment (captured by Cuckoo Sandbox) and its interactions with external entities, including cloud services (captured by Triage Sandbox).

This multi-dimensional analysis is crucial for identifying instances of cloud service abuse by malware, as it provides insights into the malware's communication patterns, network traffic profiles, and potential abuse attempts.

Dual-Sandboxing Technique

By integrating the insights from both Cuckoo and Triage sandboxes, our Dual-Sandboxing (DS) technique facilitates a multi-dimensional analysis of malware, yielding a rich dataset encompassing both internal operations and external communications. This dataset includes information on API call patterns, file interaction metrics, system interaction patterns, domain communication logs, and network traffic profiles. Figure 5.2 illustrates the operational framework of the DS technique, highlighting the integration between Cuckoo and Triage sandboxes in capturing diverse malware features.

The hosting of the sandbox environments used in our Dual-Sandboxing technique is a crucial consideration that impacts the effectiveness, scalability, and security of the CPLS abuse detection system. In our research, we utilised two primary hosting options:

- Local Hosting: The Cuckoo sandbox was hosted locally on dedicated hardware within our research environment. This approach provided full control over the sandbox configuration and allowed for in-depth analysis of malware behavior.
- Cloud-based Hosting: Triage sandbox was used specifically for sample submission and behaviour report extraction through its cloud API (tria.ge/api/v0/) This approach leveraged the sandbox's scalability and advanced anti-evasion capabilities.

For a comprehensive discussion of the system architecture, deployment scenarios, and practical implementation considerations, please refer to Sections 1.4, 1.5, and 1.6.

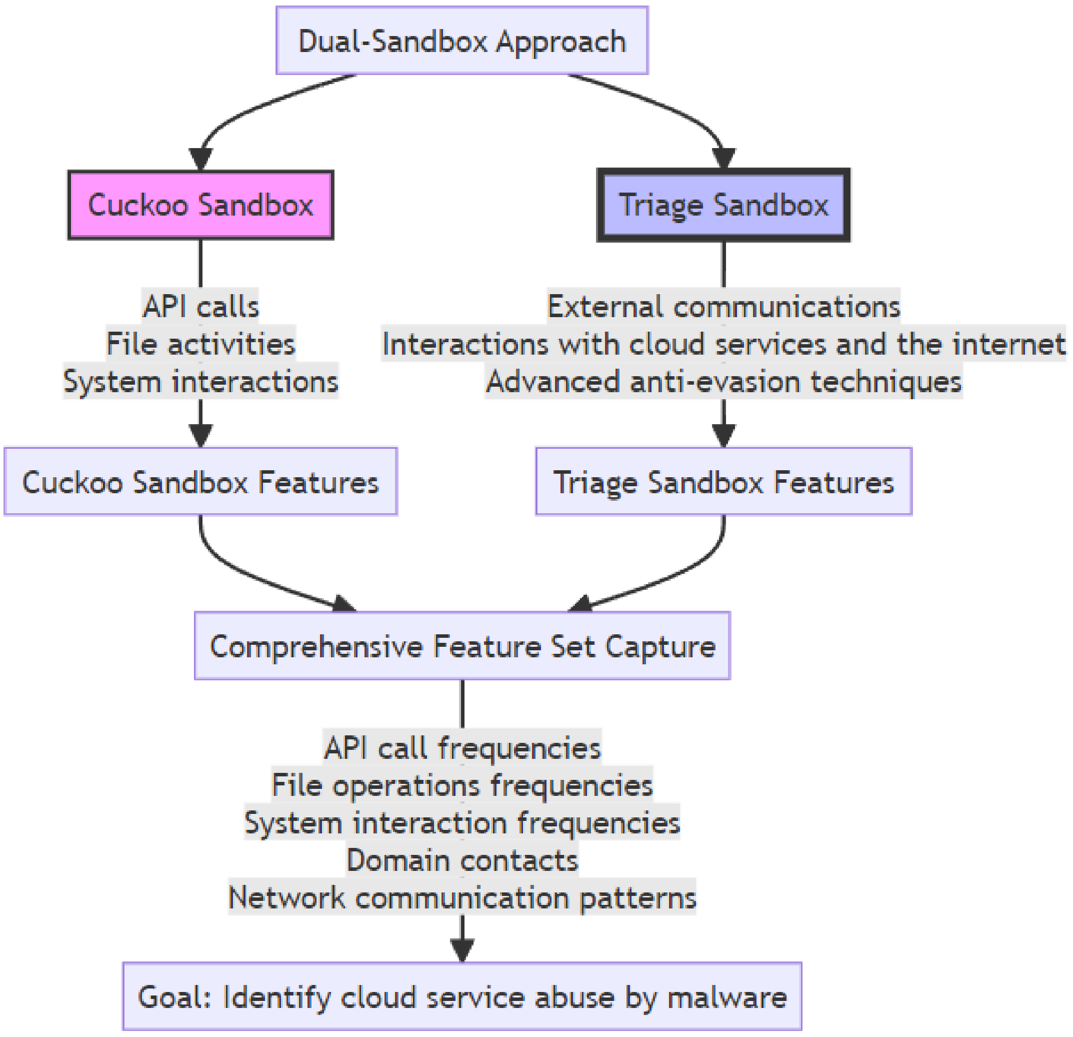

Dual-Sandbox Approach
Cuckoo Sandbox
Triage Sandbox
API calls
File activities
System interactions
External communications
Interactions with cloud services and the internet
Advanced anti-evasion techniques
Cuckoo Sandbox Features
Triage Sandbox Features
Comprehensive Feature Set Capture
API call frequencies
File operations frequencies
System interaction frequencies
Domain contacts
Network communication patterns
Goal: Identify cloud service abuse by malware

Figure 5.2: Illustration of Dual-Approach Sandboxing Technique for Enhanced Feature Extraction.

Technical Challenges in Dual-Sandboxing Integration

While the Dual-Sandboxing technique provides comprehensive malware analysis capabilities, its implementation presents several technical challenges:

1. Synchronization of Analysis: Running the same malware sample simultaneously in both Cuckoo and Triage environments requires careful synchronization. We must ensure that the features extracted from both environments reflect the behavior of the same malware

sample at comparable execution stages, preventing any cross-contamination of behavioral data between different samples.

2. Log Processing and Feature Extraction: Although both Cuckoo and Triage generate JSON-based reports, handling and processing these large log files presents significant challenges. Each sandbox generates extensive behavioral data, and extracting relevant features from these voluminous logs while maintaining the correlation between the environments requires efficient data processing strategies.

3. Time Synchronization: Ensuring consistent analysis duration across both environments is crucial for comparable results. We standardised the execution time to five minutes in both sandboxes to maintain temporal consistency in behavioural analysis.

This comprehensive feature set is essential for identifying cloud service abuse by malware, marking a significant advancement in efforts to protect cloud-based services.

Evasion Techniques and Our Countermeasures

Malware authors employ various evasion techniques to avoid detection during dynamic analysis. Some common evasion methods include:

- Environment Detection: Malware may check for signs of a sandbox environment, such as specific registry keys, processes, or hardware configurations.
- Time-based Evasion: Some malware incorporates delays or checks system uptime to evade short-duration analysis.
- User Interaction Dependency: Malware may require specific user actions before exhibiting malicious behavior.
- Anti-debugging Techniques: Advanced malware can detect and respond to the presence of debugging tools.

Our Dual-Sandboxing technique addresses these evasion methods in several ways:

1. Diverse Environments: By using both Cuckoo (local) and Triage (cloud-based) sandboxes, we create diverse analysis environments, making it harder for malware to reliably detect a sandbox.

2. Extended Analysis Time: Our approach allows for longer analysis durations, countering time-based evasion techniques.

3. Behavior Simulation: Triage sandbox incorporates user behavior simulation, triggering behaviors that depend on user interactions.

4. Multi-layer Analysis: Our feature extraction process captures behaviors at various levels (API calls, system interactions, network activities), allowing us to detect malicious activities even if some aspects are obfuscated.

By combining these strategies, our approach significantly enhances our ability to detect and analyze malware that employs evasion techniques, particularly in the context of CPLS abuse for C&C purposes.

Feature Engineering and Novel Aspects of Our Approach

While many of the features we extract are commonly used in malware analysis, our approach introduces several novel elements specifically tailored for detecting CPLS abuse as C&C infrastructure:

1. CPLS-specific features: We introduce features that are directly related to CPLS abuse, such as domain contact features for specific cloud services (e.g., domain_contact_onedrive.live.com, domain_contact_docs.google). These features are unique to our approach and specifically designed to capture interactions with CPLS platforms.

2. Comprehensive API call monitoring: Our feature set includes a wide range of API calls commonly associated with C&C communication (e.g., api_call_InternetOpenA, api_call_WinHttpSendRequest). While some of these API calls are used in general malware analysis, our comprehensive set is tailored to capture the specific behaviours of CPLS abuse.

3. Dual-Sandboxing synergy: By combining features from both Cuckoo and Triage sandboxes, we create a unique feature set that captures both system interactions and network behaviours. This synergy allows us to detect subtle patterns that might be missed by single-sandbox approaches.

4. Network activity profiling: We incorporate features that profile network activities (e.g., total_http_requests, num_domains), which are crucial for identifying C&C communication patterns specific to CPLS abuse.

While some individual features may be used in other contexts, the combination and specific focus on CPLS abuse make our feature set unique. The Dual-Sandboxing technique allows us to extract this comprehensive set of features, providing a more holistic view of malware behavior in the context of CPLS abuse than traditional approaches.

Extracted Features from Dual-Sandboxing Technique

The Dual-Sandboxing Technique enables the extraction of a comprehensive set of features that capture various aspects of malware behaviour. Our dynamic analysis approach focusses on extracting features from both network behaviour and system interactions observed during the execution of samples in our sandbox environments. This approach aligns with our goal of detecting CPLS abuse for C&C purposes. The extracted features can be categorised into four main groups:

1. Domain Contact: These features track connections to specific CPLS domains, allowing us to identify malware that communicates with known cloud services. This category includes connections to domains associated with Microsoft Azure, Google services, Dropbox, GitHub.

2. API calls: We monitor the usage of specific API functions commonly associated with network communication and file operations. These features help identify malware that utilises certain functions for C&C communication.

3. System Interactions: These features capture various system-level activities, including file operations, registry modifications, process creation, and other system changes. They provide context on the malware's behaviour on the infected system.

4. Network Activities: This category focusses on overall network behavior, including HTTP/HTTPS requests, DNS queries, TCP/UDP connections, and the number of domains and IPs contacted.

By focussing on these diverse feature sets, we aim to capture not only the network communication patterns indicative of CPLS abuse for C&C purposes but also the associated system behaviours. The complete list of features extracted using this technique is provided in the Appendix Table 7.5.

The extracted features encompass a wide range of malware characteristics, including interactions with various cloud services (e.g., Microsoft Azure, Google Cloud, Amazon Web Services), usage of specific API calls commonly associated with C&C communication, system-level interactions such as file and registry operations, and network activities like HTTP requests, DNS queries, and TCP/UDP connections. By leveraging the DS, we can capture a rich set of features that provide valuable insights into the behavior of malware samples, particularly those that abuse CPLS for C&C purposes. The comprehensive nature of these features enables the development of effective machine learning models for detecting CPLS abuse, as demonstrated by the high accuracy achieved in our experiments (see Section 5.5.2). The inclusion of domain contact features specific to various cloud services allows our approach to identify malware that communicates with CPLS-hosted domains. Additionally, the API call features help in detecting malware that utilizes specific functions commonly used for C&C communication. The system interaction and network activity features further contribute to the identification of malicious behavior patterns indicative of CPLS abuse. In summary, the DS technique empowers our detection approach by extracting a diverse and comprehensive set of features that capture the key characteristics of malware abusing CPLS for C&C purposes. These features, as listed in Appendix Table 7.5, form the foundation for building robust machine learning models capable of accurately detecting such threats.

5.3.3 Feature Selection

To evaluate whether all features should be included or only a subset, we employed both filter-based (Information Gain and ReliefF) and wrapper-based (Random Forest and J48) feature selection strategies. Figure 5.3 illustrates the accuracy progression with the custom feature elimination technique introduced and described in detail in Section 4.2.4, using Information Gain. Figure 5.4 presents the accuracy progression using ReliefF for feature selection.

Figures 5.5 and 5.6 depict the accuracy progression with wrapper-based feature selection strategies (Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), and Recursive Feature Elimination (RFE)) using Random Forest and Decision Tree as wrapper-based feature selectors, respectively. These approaches have been described in detail in Section 4.2.4.

Despite these efforts, these methods were unable to surpass the 98% accuracy rate achieved with the comprehensive suite of 99 features derived from our Dual-Sandbox approach. This outcome highlights the collective importance of each feature in accurately distinguishing between malicious and benign samples. Considering the complexity and moderate size of our feature set, our analysis suggests that reducing the number of features is not necessary to achieve optimal detection performance.

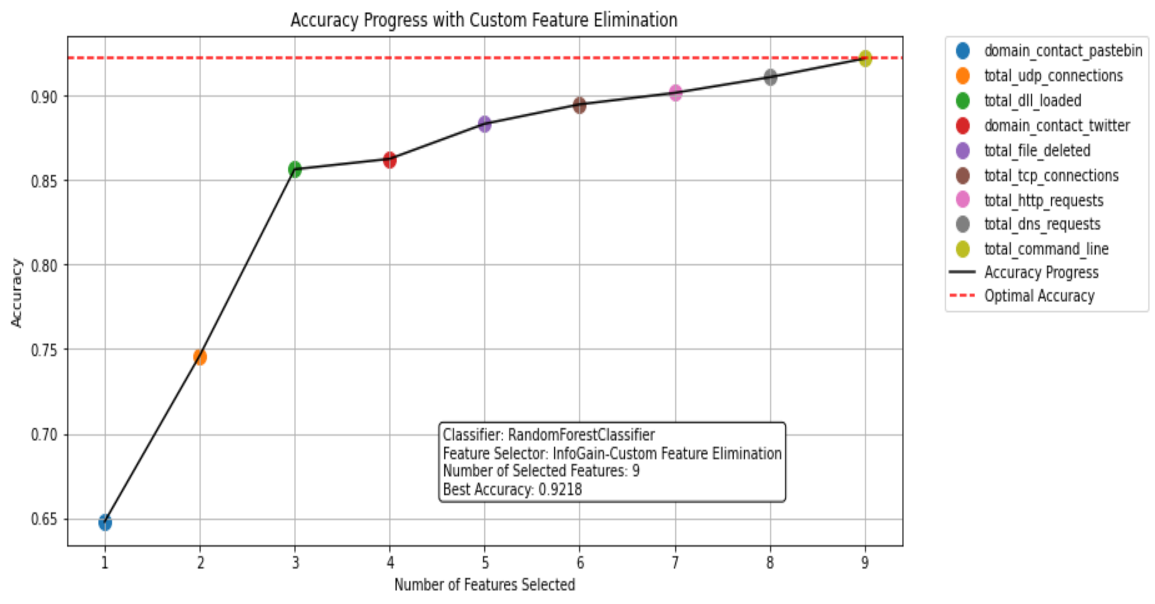


Figure 5.3: Accuracy Progress with Custom Feature Elimination using Information Gain.

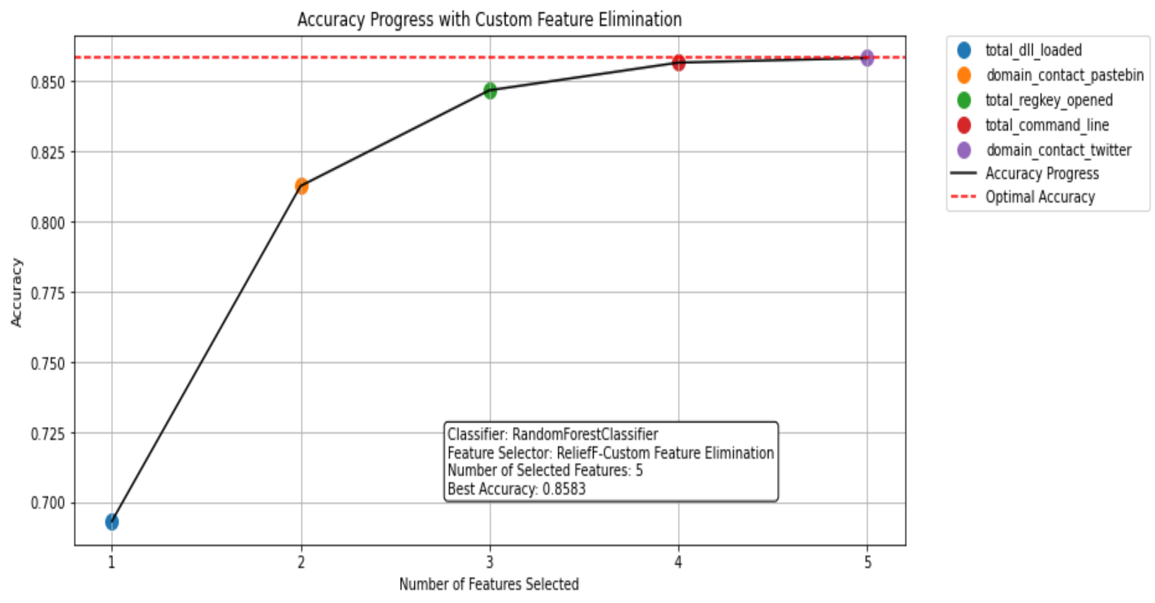


Figure 5.4: Accuracy Progress with Custom Feature Elimination using ReliefF.

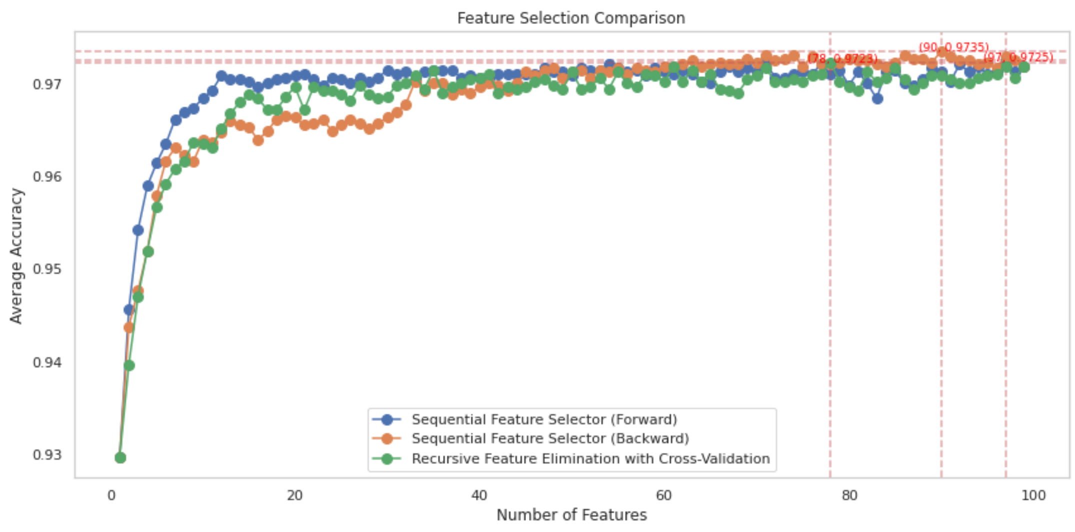


Figure 5.5: Accuracy Progression with Feature Selection Strategies (SFS, SBS, and RFE) using Random Forest as Wrapper-based Feature Selector.

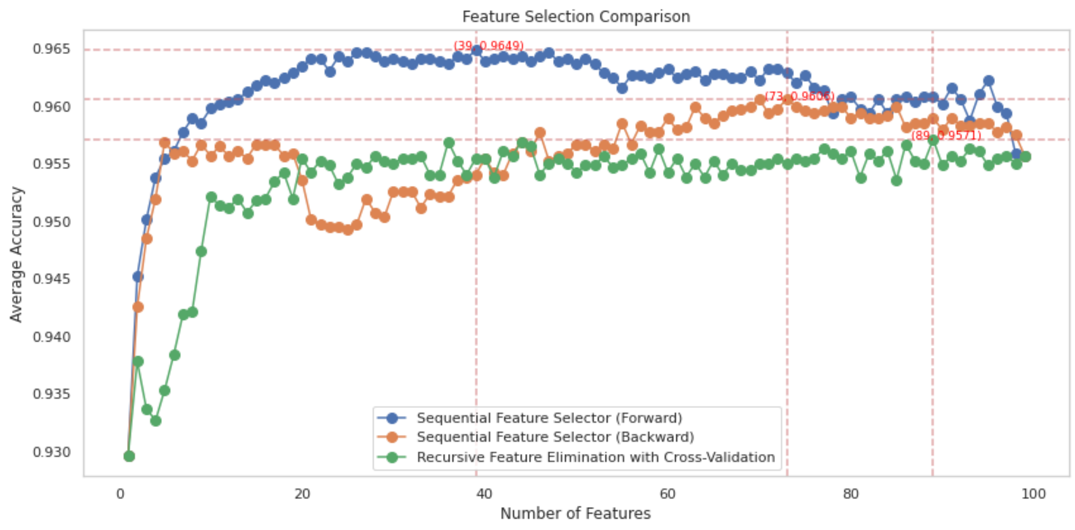


Figure 5.6: Accuracy Progression with Feature Selection Strategies (SFS, SBS, and RFE) using Decision Tree as Wrapper-based Feature Selector.

5.3.4 Machine Learning Model

We rigorously evaluated various algorithms including Decision Tree (J48), Naïve Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Random Forest, applying both a 10-fold cross-validation and a 70/30 split to our dataset. This thorough examination across multiple validation techniques ensured the robustness of our predictive models.

5.4 Experiments

We conducted experiments to evaluate the performance of our proposed approach for detecting CPLS abuse as C&C infrastructure. The experiments involved applying the Dual-Sandboxing technique for feature extraction and evaluating the performance of various machine learning algorithms on the extracted features. Both 10-fold cross-validation and a 70/30 split were used to validate the models' performance.

5.5 Evaluation

5.5.1 Experimental Setup

To evaluate the performance of our proposed approach for detecting CPLS abuse as C&C infrastructure, we conducted experiments using the features extracted by the Dual-Sandboxing technique. We evaluated various machine learning algorithms, including Decision Tree (J48), Naïve Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Random Forest, applying both a 10-fold cross-validation and a 70/30 split to our dataset. This thorough examination across multiple validation techniques ensured the robustness of our predictive models.

5.5.2 Detection Accuracy

We conducted a meticulous evaluation of various algorithms. The results are shown in Table 5.1. In comparing the two validation techniques, the Random Forest algorithm consistently outperformed other models, achieving the highest accuracy of 97.95% (70/30 split) and 97.31% (10-fold cross-validation). This performance suggests that Random Forest effectively captures complex relationships within the dataset, thereby demonstrating its suitability for the task of detecting CPLS abuse by malware. On the other hand, the J48 decision tree model also performed well, achieving an accuracy of 96.51% (70/30 split) and 95.96% (10-fold cross-validation). Although slightly lower than Random Forest, J48 still showed reliable performance.

Table 5.1: Detection Accuracy of ML Algorithms

Algorithm	70/30 Split (%)	10-fold CV (%)
Random Forest	**97.95**	97.31
J48	96.51	95.96
NB	71.32	71.81
K-NN	96.10	95.98
SVM	93.50	92.98

5.5.3 Comparison to Related Work

While there is an abundance of literature on malware detection systems that utilise dynamic analysis, there appears to be a gap in research specifically aimed at detecting the abuse of cloud services as C&C infrastructure. Therefore, we compare our work with general dynamic analysis-based malware detection systems, Sethi et al.[150] and Shijo et al.[152]. The comparison serves a dual purpose: it not only emphasizes the enhancements and contributions our work introduces to the field through advanced feature engineering but also measures our system's performance against the established benchmarks of dynamic malware analysis.

Detection Accuracy Comparison

This comparative analysis, elaborated in Table 5.2, utilises a diverse array of classifiers, including J48, RF, NB, KNN, and SVM. To ensure the reliability and generalisability of our findings, these classifiers were evaluated using two distinct validation methods: a 70:30 split and 10-fold cross-validation.

Table 5.2: Comparison of Abuse Detection Accuracy with Related Works

Reference	J48		RF		NB		KNN		SVM	
	70:30	10-fold	70:30	10-fold	70:30	10-fold	70:30	10-fold	70:30	10-fold
Sethi et al.[150]	74.89	75.94	76.95	76.44	68.39	67.63	76.12	76.15	73.87	74.55
Shijo et al.[152]	71.90	72.04	72.70	73.08	67.90	69.80	71.83	72.70	71.30	73.26
Proposed work	96.51	95.96	**97.95**	97.31	71.32	71.81	96.10	95.98	93.50	92.98

Our findings reveal a significant improvement in detection accuracy with the proposed set of features. For instance, using the RF classifier, our system achieves an accuracy of 97.94% with a 70:30 split and 97.31% with 10-fold cross-validation. This is substantially higher compared to the accuracy reported in the works of Sethi et al. [150] and Shijo et al. [152], which underscores the advanced feature engineering using the DS. Furthermore, the results from the other classifiers corroborate the superiority of our proposed system. With the J48 classifier, our system consistently outperforms the related works with a 96.50% accuracy on a 70:30 split and a 95.95% with 10-fold cross-validation. Similar trends are observed with KNN and SVM classifiers, in which the proposed system exhibits robust performance.

5.5.4 Model Performance

Across both validation techniques, the Random Forest algorithm consistently outperformed other models with the highest accuracy of 97.95% (70/30 split) and 97.31% (10-fold cross-validation). This suggests that Random Forest effectively captures complex relationships within the dataset, demonstrating its suitability for the task of detecting cloud service abuse.

5.6 Summary

In this chapter, we presented our approach to detecting the abuse of Cloud and Public Legitimate Services (CPLS) as command and control (C&C) infrastructure using machine learning techniques on dynamic features extracted from Portable Executable (PE) files. Our methodology included data collection from VirusTotal, a novel Dual-Sandboxing technique for advanced feature extraction, and the evaluation of various machine learning algorithms. The Random Forest algorithm consistently outperformed other models, achieving the highest accuracy of 97.95% (70/30 split) and 97.31% (10-fold cross-validation). Our findings demonstrate the effectiveness of our proposed approach in detecting CPLS abuse as C&C infrastructure and highlight the significance of the comprehensive feature set obtained through the Dual-Sandboxing technique. This research contributes to the ongoing efforts in developing robust detection systems to combat the evolving landscape of cybersecurity threats in the era of digital transformation and widespread adoption of cloud services.

Chapter 6

Robustness of CPLS Abuse Detection System

6.1 Introduction

In the realm of cybersecurity, the robustness of a detection system against adversarial attacks is a critical aspect to consider. Adversarial attacks are designed to deceive machine learning models by making subtle modifications to the input data, causing the model to misclassify samples. These attacks pose a significant threat to the integrity and reliability of machine learning-based detection systems, potentially allowing malicious samples to evade detection.

In this chapter, we introduce the Replace Misclassified Parameter (RMCP) attack, a novel white-box adversarial attack specifically designed to evaluate the robustness of our proposed abuse detection system. The RMCP attack leverages complete knowledge of the target machine learning model, including its architecture, parameters, and feature set, to manipulate the feature values of malicious samples. The goal is to make these samples appear as benign to the model, while preserving their malicious functionality. This sophisticated attack method mimics the tactics of real-world adversaries, who continually evolve their strategies to bypass security measures.

We present a comprehensive analysis of the RMCP attack, detailing its methodology, implementation, and impact on our proposed detection system. We evaluate the robustness of our machine learning models against this attack for both static and dynamic features, and compare the results with related works to assess the effectiveness of our approach in maintaining high accuracy under adversarial conditions. This comparative analysis provides valuable insights into the robustness of the proposed and different detection systems.

6.2 Methodology

6.2.1 Replace Misclassified Parameter (RMCP) Attack

The RMCP attack is a white-box adversarial attack that assumes complete knowledge of the target ML model, including its architecture, parameters, and feature set. The attack aims to manipulate the feature values of malicious samples, making them appear as benign to the model, while preserving their malicious functionality.

The RMCP attack was specifically developed for this research for two main reasons:

- Domain-specific approach: RMCP is specifically designed for malware detection, enabling it to address unique challenges in analysing malicious software, unlike general adversarial attacks.
- Preserving malware functionality: RMCP focusses on manipulating features that can be altered without compromising the functionality of the PE malware. This is crucial for simulating real-world evasion attempts, as actual attackers would need their malware to remain operational while evading detection.

The RMCP attack consists of two main stages:

Identifying Modifiable Features

In this stage, we analyze the feature set to determine which features can be modified without corrupting the executable file. We systematically alter each feature value in the malicious samples and observe whether the file remains executable or becomes corrupted. This step mimics the approach of real-world adversaries who aim to maintain the malicious capabilities of their software while evading detection.

To automate this process, we developed a Python script that follows the workflow depicted in Figure 6.1. The script reads unique feature values from a CSV file and iteratively modifies the PE file with each value. It then checks the executability of the modified file and determines whether the modification caused corruption or not. This approach allows researchers to efficiently identify modifiable features without manual intervention.

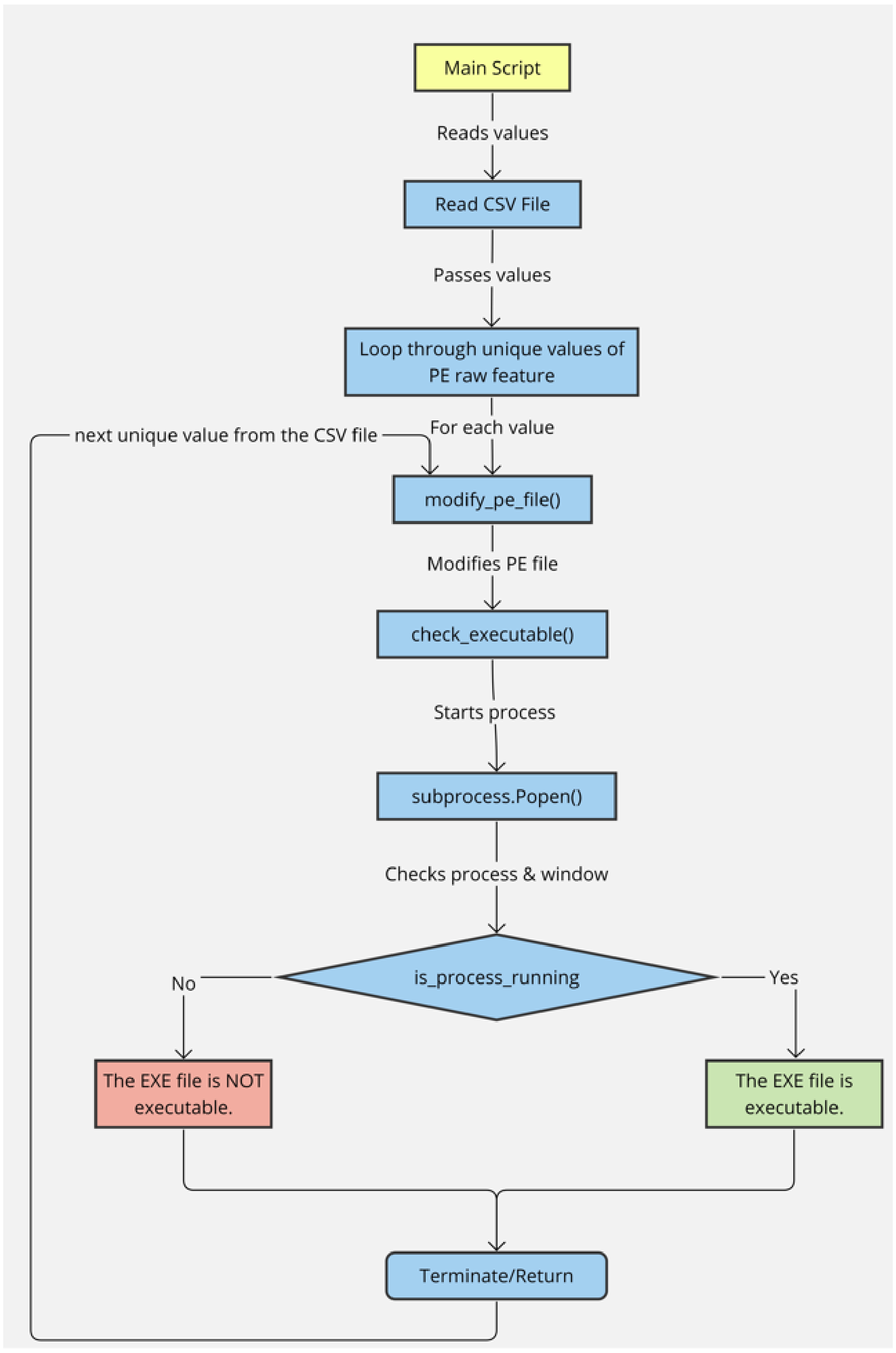
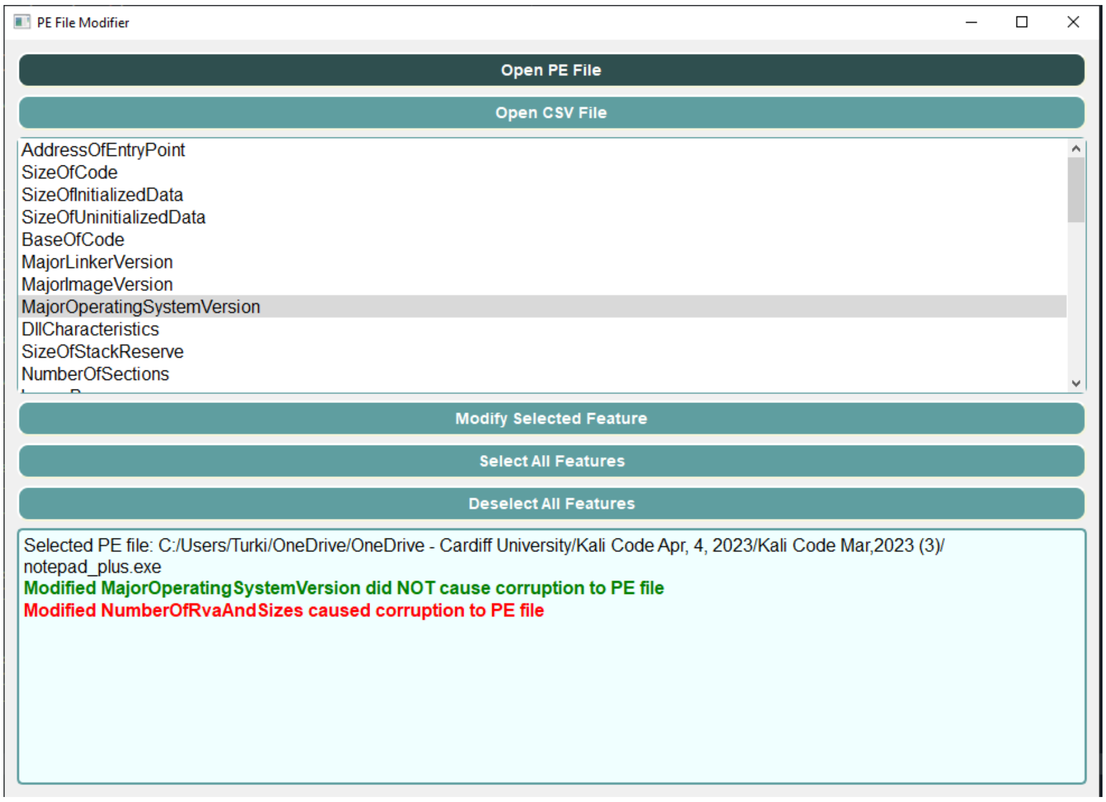


Figure 6.1: Workflow of the Python script for identifying modifiable features

Figure 6.2 showcases the PE File Modifier tool, which provides a user-friendly interface for researchers to select and modify specific features of a PE file. The tool displays the list of available features and allows users to open both PE and CSV files containing feature values. It also provides feedback on whether the modifications caused corruption to the PE file, enabling researchers to easily identify modifiable features.



```
Selected PE file: C:/Users/Turki/OneDrive/OneDrive - Cardiff University/Kali Code Apr, 4, 2023/Kali Code Mar,2023 (3)/
notepad_plus.exe
Modified MajorOperatingSystemVersion did NOT cause corruption to PE file
Modified NumberOfRvaAndSizes caused corruption to PE file
```

Figure 6.2: PE File Modifier tool for identifying modifiable features

As Table 6.1 shows, for 9 out of 38 features, modifying their value in a malicious file results in file corruption, rendering the malware non-executable. For instance, replacing the 'Number-OfSections' value of a malicious sample with other corresponding values from a benign one resulted in a corrupted executable file.

Table 6.1: Features whose value modification corrupts PE and causes of corruption.

Features	Reasons for Potential Corruption
AddressOfEntryPoint	Starts execution from an incorrect location.
NumberOfSections	OS misinterpreting the structure of the PE file.
ImageBase	New base address conflicts with other programs or system components.
SectionAlignment	OS may not properly load the sections into memory.
Subsystem	Program being run in an inappropriate environment.
Machine	OS attempting to run the code on an incompatible architecture.
VirtualSize2	Leads to incorrect memory allocation.
SizeOfImage	Leads to incorrect memory allocation.
IatRVA	Breaks the linking of imported functions.

Identifying Feature Values for Maximum Misclassification

Once the modifiable features are identified, we perturb the significant features of malicious samples with values from benign samples, aiming to make our proposed model misclassify the malicious samples as benign. We exploit the modifiable features identified earlier to deceive the model. For each of these features, we identify the value that maximizes the number of false negatives when applied to malicious samples. To guide the model towards such misclassification, we used the RMCP adversarial attack technique, detailed in Algorithm 6.1.

Algorithm 6.1 RMCP Adversarial Attack

Input Dataset X containing records with features selected based on their modifiability without corrupting the executable file (Section 6.2.1). And the 22 most significant features selected by the RF-SFSF feature selector (initially achieving an accuracy of 98.15%)

Output Benign value per feature causing the worst accuracy and confusion matrix

```
 1: procedure ADVERSARIAL ATTACK
 2:     Split dataset $X, y$ into $X_{train}$, $X_{test}$, $y_{train}$, $y_{test}$
 3:     Train the model on $X_{train}, y_{train}$
 4:     Evaluate model on $X_{test}, y_{test}$
 5:     Record accuracy and confusion matrix as baseline
 6:     for each feature $f$ in $X$ do
 7:         Get unique benign values for $f$ as $B$
 8:         for each benign value $b$ in $B$ do
 9:             Create modified $X_{test}$ by substituting malicious $X_{test}[f]$ with $b$
10:             Predict on modified $X_{test}$
11:             Compare the new accuracy and confusion matrix to the baseline
12:             if new accuracy is worse than the current worst accuracy then
13:                 Update the worst benign value for feature $f$
14:                 Update the worst accuracy and confusion matrix
15:             end if
16:         end for
17:     end for
18:     return Worst benign value per feature, worst accuracy, worst confusion matrix
19: end procedure
```

6.3 Static Feature-Based Under Adversarial Attack

In this section, we evaluate the robustness of our proposed static feature-based abuse detection system, which was introduced in Chapter 4, against the RMCP adversarial attack. We assess the impact of the attack on the model's performance and compare the results with related works to demonstrate the effectiveness of our approach in maintaining high accuracy under adversarial conditions.

6.3.1 Experimental Setup

Our experiments involve evaluating the robustness of our proposed abuse detection system against the RMCP attack for static features. We begin by establishing a baseline performance by assessing the model's accuracy and generating the confusion matrix on the original, unperturbed dataset. We then proceed to apply the RMCP attack, systematically replacing feature values in the malicious samples within the test set. For our experiments, we target the 22 most significant features selected by the RF-SFSF feature selector, which initially achieved an accuracy of 98.15% on the unmodified dataset. After each feature replacement, we re-evaluate the model's performance on the modified test set without retraining, recording the new accuracy and confusion matrix.

6.3.2 Evaluation Metrics

To assess the impact of the RMCP attack on the model's performance, we employ the confusion matrix as our primary evaluation metric. The confusion matrix provides a detailed breakdown of the model's predictions, categorizing them into true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Our focus lies particularly on the change in false negatives, as an increase in this metric indicates the success of the adversarial attack in causing the model to misclassify malicious samples as benign. To elucidate the impact of adversarial attacks on model accuracy, we have detailed the components of the confusion matrix as follows, which are also presented in Table 6.2.

Table 6.2: Confusion matrix: delineating Predicted vs. Actual outcomes for Benign and Malware classifications.

		Predicted	
		Benign	**Malware**
Actual	**Benign**	True positive (TP)	False positive (FP)
	Malware	False negative (FN)	True negative (TN)

- True positive (TP): The number of instances where the model correctly predicted benign software (0). This means the model identified a sample as benign and it was indeed benign.
- False positive (FP): The number of instances in which the model incorrectly predicted malware (1) when the sample was actually benign (0), which means the model misclassified benign software as malware.
- False negative (FN): The number of instances where the model incorrectly predicted benign software (0) when the software was actually malware (1), which means the model was misclassifying the malware as benign software.
- True negative (TN): The number of instances where the model correctly predicted malware (1). This means that the model identified the software as malware and it was indeed malware.

6.3.3 Experimental Results

The experimental results, summarized in Table 6.6, reveal the impact of the RMCP attack on our proposed abuse detection system for static features. While most features demonstrate robustness against the attack, maintaining high true positive and true negative rates, certain features exhibit vulnerability. In particular, the 'DebugSize' feature stands out, with a notable decrease in accuracy to 83.54% from 98.15% and a significant increase in false negatives from 23 to 292 after the RMCP attack.

Table 6.3: Comparison of detection accuracy before and after the RMCP adversarial attack, highlighting changes in FN values in the confusion matrix (CM). The original detection accuracy was 0.981532 and the original confusion matrix was TP:935, FP:11, FN:23, TN:872.

Features	Benign Parameter	Post-RMCP Detection Accuracy	Post-RMCP CM	
MajorLinkerVersion	8	0.977729	TP:935	FP:11
			FN:30	TN:865
MajorImageVersion	10	0.975557	TP:935	FP:11
			FN:34	TN:861
MajorOperatingSystemVersion	1	0.976099	TP:935	FP:11
			FN:33	TN:862
DllCharacteristics	1024	0.971211	TP:935	FP:11
			FN:42	TN:853
SizeOfStackReserve	65536	0.978815	TP:935	FP:11
			FN:28	TN:867
FileAlignment	4096	0.976099	TP:935	FP:11
			FN:33	TN:862
MinorOperatingSystemVersion	0	0.980445	TP:935	FP:11
			FN:25	TN:870
MajorSubsystemVersion	4	0.979902	TP:935	FP:11
			FN:26	TN:869
CheckSum	32467821	0.953829	TP:935	FP:11
			FN:74	TN:821
SizeOfHeapCommit	4096	0.981532	TP:935	FP:11
			FN:23	TN:872
NumberOfRvaAndSizes	16	0.980989	TP:935	FP:11
			FN:24	TN:871
DebugSize	28	0.835416	TP:935	FP:11
			FN:292	TN:603
potential_C&C_api_calls	0	0.965779	TP:935	FP:11
			FN:52	TN:843

6.3.4 Robustness Comparison

To gauge the effectiveness of our proposed approach, we compare its robustness against the RMCP attack with two related works: Kumar et al. [126] and Raman et al. [145].

As presented in Table 6.4, the model proposed by Kumar et al. [126] exhibits significant vulnerability to the RMCP attack. The replacement of certain feature values, such as the 'Characteristics' feature, leads to a substantial drop in accuracy from 94.67% to 69% and a dramatic increase in false negatives from 53 to 525.

Table 6.4: Comparison of detection accuracy before and after the RMCP adversarial attack, highlighting changes in FN values in the confusion matrix (CM): Kumar et al. [126]. The original detection accuracy was 0.946768 and the original confusion matrix was TP:901, FP:45, FN:53, TN:842.

Features	Benign Parameter	Post-RMCP Detection Accuracy	Post-RMCP CM	
MajorOperatingSystemVersion	1	0.923411	TP:901	FP:45
			FN:96	TN:799
DllCharacteristics	34112	0.813688	TP:901	FP:45
			FN:298	TN:597
SizeOfStackReserve	16777216	0.811515	TP:901	FP:45
			FN:302	TN:593
MajorSubsystemVersion	6	0.937534	TP:901	FP:45
			FN:70	TN:825
MinorSubsystemVersion	0	0.941879	TP:901	FP:45
			FN:62	TN:833
Characteristics	263	0.690386	TP:901	FP:45
			FN:525	TN:370
e_lfanew	304	0.697990	TP:901	FP:45
			FN:511	TN:384

Similarly, Table 6.5 demonstrates the impact of the RMCP attack on the model proposed by Raman et al. [145]. The attack results in a considerable reduction in model accuracy, with the 'DebugSize' feature value substitution causing the accuracy to plummet from 95.84% to 54.37% and false negatives to surge from 48 to 808.

Table 6.5: Comparison of detection accuracy before and after the RMCP adversarial attack, highlighting changes in FN values in the confusion matrix (CM): Raman et al. [145]. The original detection accuracy was 0.956545 and the original confusion matrix was TP:914, FP:32, FN:48, TN:847.

Features	Benign Parameter	Post-RMCP Detection Accuracy	Post-RMCP CM	
MajorImageVersion	10	0.697447	TP:914	FP:32
			FN:525	TN:370
DebugSize	84	0.543726	TP:914	FP:32
			FN:808	TN:87
ExportSize	393,079	0.951113	TP:914	FP:32
			FN:58	TN:837
ResourceSize	5296	0.762629	TP:914	FP:32
			FN:405	TN:490

In contrast, our proposed abuse detection system exhibits greater resilience against the RMCP attack. Despite the attack, our model maintains an accuracy of 83.54%, outperforming the robustness rates of 69.03% and 54.37% achieved by Kumar et al. [126] and Raman et al. [145], respectively.

These comparative results highlight the superior robustness of our approach in the face of adversarial attacks. While the initial detection accuracy rates were comparable across the models, the RMCP attack exposes the vulnerabilities of the related works, leading to significant performance degradation. Our model, on the other hand, demonstrates resilience, maintaining a higher level of accuracy and minimizing the impact of the adversarial attack.

6.4 Dynamic Feature-Based Under Adversarial Attack

This section focuses on evaluating the robustness of our dynamic feature-based abuse detection system, introduced in Chapter 5, when subjected to the RMCP adversarial attack. We analyze the impact of the attack on the model's performance and compare the results with related works to demonstrate the enhanced resilience of our approach against adversarial manipulations.

6.4.1 Experimental Setup

In addition to evaluating the robustness of our detection system against adversarial attacks on static features, we also assess its resilience when dynamic features are targeted. We apply the RMCP attack to the dynamic features used in our ML-based detection model and compare the results with related works by Sethi et al. [150] and Shijo et al. [152].

6.4.2 Experimental Results

The results of adversarial attacks on the detection accuracy of our proposed model, as well as related works by Sethi et al. [150] and Shijo et al. [152], are presented in Tables 6.6, 6.7, and 6.8 respectively. The discussion focuses on the changes in the number of false negatives (FN) in the confusion matrix (CM), which reflect the model's ability to correctly identify malicious samples under FPT attack.

For our proposed work, the feature 'num_ips' caused the most significant drop in detection accuracy after feature perturbation testing (FPT), from 97.95% to 90.01%. This is indicative of the feature's importance in the detection algorithm and its vulnerability to adversarial manipulation. Other features also caused declines in detection accuracy, but to a lesser extent, suggesting a varied impact on the model's robustness.

Table 6.6: Top 10 Impacted Features of Adversarial Attack on Detection Accuracy, with a Focus on Changes in FN Values in the Confusion Matrix (CM). The original detection accuracy was 0.9795 and the original CM TP:688 FP:10 FN:20 TN:743.

Features	Benign Parameter	Post-RMCP Accuracy	Post-RMCP CM	
num_ips	2	0.9001	TP:688	FP:10
			FN:135	TN:628
total_dns_requests	43	0.9521	TP:688	FP:10
			FN:59	TN:704
total_udp_connections	101	0.9542	TP:688	FP:10
			FN:56	TN:707
num_domains	15	0.9603	TP:688	FP:10
			FN:47	TN:716
total_file_failed	11	0.9617	TP:688	FP:10
			FN:45	TN:718
total_tcp_connections	3	0.9624	TP:688	FP:10
			FN:44	TN:719
total_directory_enumerated	41	0.9631	TP:688	FP:10
			FN:43	TN:720
total_regkey_read	94	0.9637	TP:688	FP:10
			FN:42	TN:721
total_http_requests	7	0.9637	TP:688	FP:10
			FN:42	TN:721
unique_dns_requests	30	0.9637	TP:688	FP:10
			FN:42	TN:721

6.4.3 Robustness Comparison

To assess the efficacy of our proposed method, we compare its robustness against the RMCP attack with two related works: Sethi et al. [150] and Shijo et al. [152].

The model by Sethi et al. [150] showed a significant impact on the detection accuracy when subjected to the RMCP attack. The accuracy dropped to around 50% for the top 46 features out of 1022 features, with some features exhibiting a substantial increase in the FN rate. For example, the feature 'f_RegOpenKeyExW' saw a dramatic rise in FN from the original value of 10 to 404 post-RMCP attack, indicating that the model's ability to correctly classify malicious samples was severely compromised for these specific features. However, for other features, such as 'f_FindFirstFileExW' and 'rc_LdrGetDllHandle_0', the post-RMCP confusion matrices show a relatively smaller increase in FN (from 10 to 12) and a high TP rate (354). This suggests that the model's robustness is not uniformly affected by the RMCP attack across all features.

The varying impact on the FN rates and the corresponding changes in accuracy indicate that certain features are more susceptible to adversarial attack than others. While some features, when subjected to the RMCP attack, can significantly degrade the model's ability to correctly classify malicious samples, others have a less severe impact on the model's performance. Table 6.7 presents the top 10 impacted features and their corresponding changes in the confusion matrix. The features 'f_RegOpenKeyExW', 'f_GetFileType', 'rc_NtOpenKey_0', and 'f_GetSystemMetrics' show a substantial increase in FN rates, with the accuracy dropping to around 50.00 - 50.21%. On the other hand, features like 'f_FindFirstFileExW' and 'rc_LdrGetDllHandle_0' exhibit a smaller increase in FN rates, with the accuracy remaining around 76.81%.

In contrast, our proposed model demonstrates superior robustness by maintaining a relatively high accuracy level of around 90% under the RMCP attack. This demonstrates significantly better resilience compared to related works, where accuracies dropped to around 50%. Specifically, Sethi et al.'s model accuracy fell to 50.21% and Shijo et al.'s to 47.57% under attack, while our model maintained 90.01% accuracy. This substantial difference in performance highlights the enhanced robustness of our approach.

Table 6.7: Top 10 Impacted Features of Adversarial Attack on Detection Accuracy, with a Focus on Changes in FN Values in the Confusion Matrix (CM): Sethi et al.[150]. The original detection accuracy was 0.7695 and the CM TP:354 FP:327 FN:10 TN:771.

Features	Benign Parameter	Post-RMCP Detection Accuracy	Post-RMCP CM
f_SetUnhandledExceptionFilter	1	0.5000	TP:354 FP:327 FN:404 TN:377
f_GetFileType	4	0.5000	TP:354 FP:327 FN:404 TN:377
rc_NtOpenKey_0	3	0.5007	TP:354 FP:327 FN:403 TN:378
f_GetSystemMetrics	465	0.5007	TP:354 FP:327 FN:403 TN:378
s_WSAStartup	3	0.5014	TP:354 FP:327 FN:402 TN:379
s_NtAllocateVirtualMemory	11	0.5014	TP:354 FP:327 FN:402 TN:379
⋮			
rc_RegOpenKeyExA_2	3	0.5021	TP:354 FP:327 FN:401 TN:380
s_GetSystemInfo	77	0.5021	TP:354 FP:327 FN:401 TN:380
f_LdrLoadDll	0	0.7681	TP:354 FP:327 FN:12 TN:769
s_NtProtectVirtualMemory	0	0.7681	TP:354 FP:327 FN:12 TN:769

Similarly, Shijo et al.'s [152] model showed a similar trend, with the detection accuracy dropping to a range of approximately 47.57% to 47.84% across the first 155 features. The increase in FN for features such as 'LdrGetDllHandle_NtTerminateProcess
_NtTerminateProcess' was substantial, demonstrating that the model is not robust against the RMCP attack.

Table 6.8 represents the top impacted features of the RMCP attack.

Table 6.8: Top Impacted Features of Adversarial Attack on Detection Accuracy, with a Focus on Changes in FN Values in the Confusion Matrix (CM): Shijo et al. [152]. The original detection accuracy was 0.7332 and the original confusion matrix TP:340 FP:399 FN:2 TN:762.

Features	Benign Parameter	Post-RMCP Detection Accuracy	Post-RMCP CM	
LdrGetDllHandle _NtTerminateProcess _NtTerminateProcess	1	0.4757	TP:340 FN:389	FP:399 TN:375
NtClose _NtOpenKey _RegOpenKeyExW	5	0.4770	TP:340 FN:387	FP:399 TN:377
NtOpenKey _NtQueryValueKey _NtClose	3	0.4777	TP:340 FN:386	FP:399 TN:378
⋮				
RegQueryValueExW _RegCloseKey _RegOpenKeyExW	21	0.4784	TP:340 FN:385	FP:399 TN:379
NtUnmapViewOfSection _NtClose _NtClose	506	0.4784	TP:340 FN:385	FP:399 TN:379
RegCloseKey _RegOpenKeyExA _RegQueryValueExA	2	0.4784	TP:340 FN:385	FP:399 TN:379

These comparative results highlight the superior robustness of our approach in the face of adversarial attacks. While the initial detection accuracy rates were comparable across the models, the RMCP attack exposes the vulnerabilities of the related works, leading to significant performance degradation. Our model, on the other hand, demonstrates resilience, maintaining a higher level of accuracy and minimizing the impact of the adversarial attack.

6.5 Summary

In this chapter, we introduced the Replace Misclassified Parameter (RMCP) attack, a novel white-box adversarial attack designed to evaluate the robustness of our proposed abuse detection system. By systematically manipulating feature values of malicious samples, the RMCP attack aims to deceive the machine learning model into misclassifying malware as benign.

We conducted extensive experiments to assess the impact of the RMCP attack on our model's performance for both static and dynamic features. While most features exhibited robustness, certain features, such as 'DebugSize' for static features and 'num_ips' for dynamic features, demonstrated vulnerability to the attack.

To contextualize our findings, we compared the resilience of our approach against the RMCP attack with related works for both static and dynamic features. The results reveal that our model maintains a higher level of accuracy and robustness compared to the models proposed by Kumar et al. [126] and Raman et al. [145] for static features, and Sethi et al. [150] and Shijo et al. [152] for dynamic features.

The insights gained from this study underscore the importance of considering adversarial attacks when designing and evaluating abuse detection systems. By demonstrating the superior robustness of our approach for both static and dynamic features, we highlight its potential to provide more reliable and effective protection against malware that abuses cloud and public legitimate services for command and control purposes.

Chapter 7

Conclusion

This thesis addressed the critical issue of malware abusing Cloud and Public Legitimate Services (CPLS) as command and control (C&C) infrastructure. The research aimed to develop effective and robust detection systems to combat this sophisticated cyber threat. The main objectives included conducting a comprehensive literature review to establish a taxonomy of attack techniques, identifying gaps in existing defense mechanisms, investigating the use of static and dynamic malware analysis methods through artificial intelligence (AI) techniques, and evaluating the robustness of the proposed detection systems against adversarial attacks. The thesis employed a multi-stage methodology, encompassing data collection, feature extraction, machine learning model development, and evaluation. The dataset, sourced from VirusTotal, consisted of both malicious samples communicating with known CPLS-hosted domains and benign samples obtained from trusted sources. The research introduced novel techniques, such as the Dual-Sandboxing approach for dynamic feature extraction and custom feature selection methods for static analysis.

These techniques led to the identification of a unique set of features. When combined with ML algorithms, they enable our approach to outperform other methods in the literature. Moreover, the proposed detection systems have demonstrated enhanced robustness against adversarial attack, highlighting the effectiveness and reliability of our methodology.

7.1 Addressing the Research Questions

The research findings successfully addressed the posed research questions:

- **RQ1:** The literature review identified various techniques utilized to abuse CPLS as C&C communication channels, including steganography, encoding, cryptography, and AI-powered C&C.
- **RQ2:** The targeted CPLS platforms for abuse included popular services such as Google Drive, OneDrive, and Dropbox.
- **RQ3:** What countermeasures have been proposed to detect the abusive use of CPLS as C&C infrastructure?
  The analysis of existing countermeasures revealed several gaps in effectively detecting CPLS abuse. These gaps include the lack of the following: comprehensive taxonomies for attack techniques, labelled datasets specifically focused on CPLS abuse, and advanced detection approaches that are capable of detecting the abuse of CPLS as C&C infrastructure.
- **RQ4:** How can we develop and validate a comprehensive dataset and labelling strategy for Portable Executable (PE) files to support the detection of CPLS abuses?
  A comprehensive dataset of 3,067 malicious and 3,067 benign PE files was developed and labelled to support the detection of CPLS abuse.
- **RQ5:** How can statically extracted features from Portable Executable (PE) files be leveraged through artificial intelligence or machine learning methodologies to enhance the detection and mitigation of CPLS abuses?
  Static and derived features extracted from PE files, combined with machine learning techniques, achieved a high detection accuracy of 98.26%.
- **RQ6:** In what ways can dynamic feature extraction from Portable Executable (PE) files, integrated with machine learning models, improve the detection of CPLS abuse as C&C infrastructure?
  Dynamic features extracted using the innovative Dual-Sandboxing technique, integrated with machine learning models, demonstrated a detection accuracy of 97.95%.
- **RQ7:** How can the robustness of detection models against adversarial attacks be enhanced for both statically and dynamically extracted features from Portable Executable (PE) files in CPLS abuse?

While our proposed detection systems exhibited enhanced robustness against the novel RMCP adversarial attack compared to related works, further enhancements are possible:

- For the static feature-based detection, our proposed system maintained an accuracy of 83.54% against the RMCP attack, a decrease from the initial 98.15%, with FN increasing from 23 to 292. In comparison, Kumar et al.'s system saw a drop in accuracy from 94.67% to 69% with FN increasing from 53 to 525, while Raman et al.'s system's accuracy plummeted from 95.84% to 54.37% with FN increasing from 48 to 808.
- For the dynamic feature-based detection, our proposed system demonstrated superior robustness by maintaining a relatively high accuracy level of around 90.01% under the RMCP attack, compared to the original accuracy of 97.95%, with FN increasing from 20 to 135. In contrast, Sethi et al.'s model, with an original accuracy of 76.95%, showed a significant impact with the accuracy dropping to around 50% with FN increasing from 10 to 404, while Shijo et al.'s model, with an original accuracy of 73.32%, saw its accuracy drop to 47.57% with FN increasing from 2 to 389 under the RMCP attack.

Further enhancement of the model robustness against adversarial attacks remains an important area for future research.

7.2 Key Outcomes

This thesis has produced several significant outcomes that address the critical issue of CPLS abuse as C&C infrastructure:

7.2.1 Novel Taxonomies

Our taxonomies of CPLS abuse techniques and C&C communication channels have:

- Identified and categorised ten different types of abusive attacks (steganography, encoding, cryptography, fraudulent accounts, use of botmaster's credentials/hard-coded tokens, compromised victims' accounts, COM hijacking, process injection, COMSPEC environment variable exploitation, and AI-powered C&C), significantly expanding on previous work that only identified four.

- Provided a structured framework for understanding and categorising emerging threats, including DeepC2, an AI-powered C&C technique that uses neural networks for dynamic addressing and embeds commands in tweets through hash collisions, not currently represented in industry frameworks like MITRE ATTCK.
- Facilitated more effective analysis and combat strategies against evolving threats in cloud environments.

7.2.2 First Labelled Dataset for CPLS Abuse

The creation of our novel dataset has:

- Provided 3,067 malicious and 3,067 benign samples specifically focused on CPLS abuse.
- Enable more accurate and relevant training of machine learning models for CPLS abuse detection.
- Addressed a critical gap in the field, providing a valuable resource for future research and development of defence mechanisms.

7.2.3 Machine Learning-based CPLS Abuse Detection Systems

Our ML-based detection systems have demonstrated:

- High accuracy rates of 98.26% for static analysis and 97.95% for dynamic analysis, outperforming existing methods in the literature.
- The effectiveness of our tailored feature engineering and custom feature elimination methods in static analysis.

- The potential of our innovative Dual-Sandboxing technique in extracting comprehensive feature sets for dynamic analysis.

7.2.4 Novel RMCP Adversarial Attack

The introduction of the RMCP attack has:

- Provided a crucial evaluation of the robustness of CPLS abuse detection systems.
- Demonstrated the resilience of our detection systems, maintaining accuracy levels of 83.54% for static analysis and 90.01% for dynamic analysis under attack, significantly outperforming related works.
- Highlighted the importance of considering adversarial scenarios in cybersecurity research and real-world deployment of detection systems.

These outcomes represent significant advancements in the field of CPLS abuse detection, enhancing our capabilities to detect and mitigate such threats, and providing a strong foundation for future research and practical applications in securing cloud environments.

7.3 Limitations

Despite the significant contributions, the research faced certain limitations:

- The dataset could be expanded to include a wider variety of file formats, since the focus was primarily on PE file type.
- The research relied solely on a malware dataset obtained from VirusTotal. To overcome this limitation, the implementation of a honeypot could be considered.

These limitations provide opportunities for future research to further enhance the generalisability and scalability of the proposed detection systems.

7.4 Future Research Directions

Based on the findings and limitations, several promising avenues for future research are identified:

- Expanding the dataset to include a more diverse range of malware samples and CPLS platforms, enhancing the generalisability of the detection systems.
- Investigating the integration of static and dynamic analysis techniques to leverage the strengths of both approaches and improve detection performance.
- Exploring the application of deep learning or other advanced AI techniques for the analysis of CPLS abuse.
- Exploring other ensemble learning methods like AdaBoost and Gradient Boosting for CPLS abuse detection, comparing their performance against our Random Forest-based approach.
- Implementation of a honeypot, a system designed to attract and analyse malicious activities. By deploying a honeypot, it becomes possible to collect malware samples that demonstrate signs of abuse related to CPLS as C&C infrastructure. This approach could potentially provide a more diverse and realistic set of malware samples for analysis, enhancing the robustness and generalisability of the proposed detection systems.
- Enhancing adversarial attack robustness, while our detection systems demonstrated improved robustness compared to existing approaches, future research could explore advanced techniques to further enhance resilience against adversarial attacks.

7.5 Conclusion

This thesis aimed to combat the critical threat of malware abusing Cloud and Public Legitimate Services (CPLS) as command and control (C&C) infrastructure by developing effective and robust detection systems. The research employed a multi-stage methodology, encompassing data collection, feature extraction, machine learning model development, and evaluation. The main contributions of this thesis are summarized as follows:

- Conducted a comprehensive literature review to establish novel taxonomies of attack techniques and C&C communication channels, enabling a better understanding of the threat landscape and facilitating the development of targeted defense mechanisms.
- Identified gaps in existing defence mechanisms and the need for comprehensive taxonomies, labelled datasets, and advanced detection approaches for CPLS abuse as C&C infrastructure.
- Created a novel labelled dataset comprising 3,067 malicious and 3,067 benign samples, serving as a valuable resource for training and evaluating machine learning classifiers.
- Proposed ML-based CPLS abuse detection systems utilising static and dynamic malware analysis, achieving remarkable detection accuracies of 98.26% and 97.95%, respectively, outperforming existing works.
- Introduced the RMCP adversarial attack to evaluate the robustness of detection systems, with the proposed models demonstrating enhanced resilience compared to related works.

The research addressed the posed research questions by identifying techniques utilized for CPLS abuse, targeted platforms, gaps in existing countermeasures, and proposing novel static and dynamic analysis methods integrated with machine learning models. The detection systems exhibited enhanced robustness against the RMCP adversarial attack compared to related works. Despite the significant contributions, the research faced certain limitations, including the focus on the PE file format and reliance on a VirusTotal dataset. Future research could expand the dataset to include a more diverse range of malware samples and CPLS platforms, investigate the integration of static and dynamic analysis techniques, explore deep learning or advanced AI techniques, and implement honeypots for collecting CPLS abuse-related malware. The research findings have significant practical implications for enhancing cloud security. Organizations and security practitioners can leverage the proposed detection systems, novel taxonomies, and insights gained from the comprehensive analysis to identify and mitigate the abuse of CPLS as C&C infrastructure effectively.

Bibliography

[1] Account manipulation: Additional cloud credentials, sub-technique t1098.001 - enterprise | mitre att&ckő. https://attack.mitre.org/techniques/T1098/001/. (Accessed on 04/01/2024).

[2] Big airline heist | group-ib blog. https://www.group-ib.com/blog/colunmtk-apt41/. (Accessed on 03/30/2024).

[3] Botnet taxonomy white paper final. https://sites.cs.ucsb.edu/~kemm/courses/cs595G/TM06.pdf. (Accessed on 03/20/2020).

[4] Command and control, tactic ta0011 - enterprise | mitre att&ckő. https://attack.mitre.org/tactics/TA0011/. (Accessed on 04/01/2024).

[5] Data encoding, technique t1132 - enterprise | mitre att&ckő. https://attack.mitre.org/techniques/T1132/. (Accessed on 04/01/2024).

[6] Data obfuscation: Steganography, sub-technique t1001.002 - enterprise | mitre att&ckő. https://attack.mitre.org/techniques/T1001/002/. (Accessed on 04/01/2024).

[7] Desktop operating system market share 2013-2023 | statista. https://www.statista.com/statistics/218089/global-market-share-of-windows-7/. (Accessed on 08/31/2023).

[8] Encrypted channel: Asymmetric cryptography, sub-technique t1573.002 - enterprise | mitre att&ckő. https://attack.mitre.org/techniques/T1573/002/. (Accessed on 04/01/2024).

[9] Encrypted channel: Symmetric cryptography, sub-technique t1573.001 - enterprise | mitre att&ckő. https://attack.mitre.org/techniques/T1573/001/. (Accessed on 04/01/2024).

[10] Encrypted channel, technique t1573 - enterprise | mitre att&ckő. `https://attack.mitre.org/techniques/T1573/`. (Accessed on 04/01/2024).

[11] Event triggered execution: Component object model hijacking, sub-technique t1546.015 - enterprise | mitre att&ckő. `https://attack.mitre.org/techniques/T1546/015/`. (Accessed on 04/01/2024).

[12] Exposing polonium activity and infrastructure targeting israeli organizations | microsoft security blog. `https://www.microsoft.com/en-us/security/blog/2022/06/02/exposing-polonium-activity-and-infrastructure-targeting-israeli-organizations/`. (Accessed on 03/30/2024).

[13] Get message - microsoft graph v1.0 | microsoft learn. `https://learn.microsoft.com/en-us/graph/api/message-get?view=graph-rest-1.0&tabs=http#request-headers`. (Accessed on 04/07/2024).

[14] Gold dragon widens olympics malware attacks, gains permanent presence on victims' systems. `https://www.mcafee.com/blogs/other-blogs/mcafee-labs/gold-dragon-widens-olympics-malware-attacks-gains-permanent-presence-on-victims-systems/`. (Accessed on 03/30/2024).

[15] In-depth analysis of a new variant of .net malware agenttesla. `https://www.fortinet.com/blog/threat-research/in-depth-analysis-of-net-malware-javaupdtr`. (Accessed on 03/30/2024).

[16] Inside windows: An in-depth look into the win32 portable executable file format, part 2 | microsoft learn. `https://learn.microsoft.com/en-us/archive/msdn-magazine/2002/march/inside-windows-an-in-depth-look-into-the-win32-portable-executable-file-format-part-2`. (Accessed on 03/07/2023).

[17] Mitre att&ckő. `https://attack.mitre.org/`. (Accessed on 03/30/2024).

[18] New wekby attacks use dns requests as command and control mechanism. `https://unit42.paloaltonetworks.com/unit42-new-wekby-attacks-use-dns-requests-as-command-and-control-mechanism/`. (Accessed on 03/30/2024).

[19] North korean advanced persistent threat focus: Kimsuky | cisa. `https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-301a`. (Accessed on 03/30/2024).

[20] Pe format - win32 apps | microsoft learn. https://learn.microsoft.com/en-us/windows/win32/debug/pe-format.

[21] Portable executable - wikipedia. https://en.wikipedia.org/wiki/Portable_Executable. (Accessed on 09/04/2023).

[22] scikit-learn: machine learning in python scikit-learn 1.2.2 documentation. https://scikit-learn.org/stable/.

[23] Securelist | the "kimsuky" operation: A north korean apt? | securelist. https://securelist.com/the-kimsuky-operation-a-north-korean-apt/57915/. (Accessed on 03/30/2024).

[24] Threat spotlight: Amadey bot targets non-russian users. https://blogs.blackberry.com/en/2020/01/threat-spotlight-amadey-bot. (Accessed on 03/30/2024).

[25] user: sendmail - microsoft graph v1.0 | microsoft learn. https://learn.microsoft.com/en-us/graph/api/user-sendmail?view=graph-rest-1.0&tabs=http. (Accessed on 04/07/2024).

[26] Virustotal - stats. https://www.virustotal.com/gui/stats. (Accessed on 08/31/2023).

[27] Web service: Bidirectional communication, sub-technique t1102.002 - enterprise | mitre att&ckő. https://attack.mitre.org/techniques/T1102/002/. (Accessed on 03/30/2024).

[28] Free software downloads and reviews for windows, android, mac, and ios cnet download, 1996.

[29] SourceForge.Net. https://sourceforge.net/projects/sourceforge/, 1999.

[30] Virustotal - home. https://www.virustotal.com/gui/home/upload, 2004.

[31] Cuckoo sandbox - automated malware analysis. https://cuckoosandbox.org/, February 2011. (Accessed on 05/18/2023).

[32] Cloud atlas: Redoctober apt is back in style | securelist. https://securelist.com/cloud-atlas-redoctober-apt-is-back-in-style/68083/, December 2014.

[33] Apt17: Hiding in plain sight - fireeye and microsoft expose obfuscation tactic | fireeye. https://www.fireeye.com/current-threats/apt-groups/rpt-apt17.html, May 2015.

[34] China-based cyber threat group uses dropbox for malware communications and targets hong kong media outlets | mandiant. `https://www.mandiant.com/resources/china-based-threat`, December 2015. 14.

[35] Github - paulsec/twittor: A fully featured backdoor that uses twitter as a c&c server. `https://github.com/PaulSec/twittor`, September 2015.

[36] Github - arno0x/dbc2: Dbc2 (dropboxc2) is a modular post-exploitation tool, composed of an agent running on the victim's machine, a controler, running on any machine, powershell modules, and dropbox servers as a means of communication. `https://github.com/Arno0x/DBC2`, 2016.

[37] Github - maldevel/gdog: A fully featured windows backdoor that uses gmail as a c&c server. `https://github.com/maldevel/gdog`, May 2016.

[38] Telecrypt - the ransomware abusing telegram api - defeated! | malwarebytes labs. `https://blog.malwarebytes.com/threat-analysis/2016/11/telecrypt-the-ransomware-abusing-telegram-api-defeated/`, November 2016.

[39] Apt32, sealotus, oceanlotus, apt-c-00, group g0050 | mitre att&ckő. `https://attack.mitre.org/groups/G0050/`, December 2017.

[40] Command and control dropbox penetration testing lab. `https://pentestlab.blog/2017/08/29/command-and-control-dropbox/`, August 2017.

[41] Hammertoss: Stealthy tactics define a russian cyber threat group | fireeye. `https://www.fireeye.com/current-threats/apt-groups/rpt-apt29.html`, July 2017.

[42] How new chat platforms can be abused by cybercriminals - noticias de seguridad - trend micro es. `https://www.trendmicro.com/vinfo/es/security/news/cybercrime-and-digital-threats/how-new-chat-platforms-abused-by-cybercriminals`, June 2017.

[43] Github - 0x09al/dropboxc2c: Dropboxc2c is a post-exploitation agent which uses dropbox infrastructure for command and control operations. `https://github.com/0x09AL/DropboxC2C`, October 2018.

[44] Github - bkup/slackshell: Powershell to slack c2. `https://github.com/bkup/SlackShell`, May 2018.

[45] Github - byt3bl33d3r/gcat: A poc backdoor that uses gmail as a c&c server. `https://github.com/byt3bl33d3r/gcat`, November 2018.

[46] Login | triage. `https://tria.ge/`, Auguast 2018. (Accessed on 02/25/2022).

[47] Threat analysis: Rokrat malware - vmware security blog - vmware. `https://blogs.vmware.com/security/2018/02/threat-analysis-rokrat-malware.html`, February 2018.

[48] Casbaneiro: Dangerous cooking with a secret ingredient | welivesecurity. `https://www.welivesecurity.com/2019/10/03/casbaneiro-trojan-dangerous-cooking/`, October 2019.

[49] Github - coalfire-research/slackor: A golang implant that uses slack as a command and control server. `https://github.com/Coalfire-Research/Slackor`, October 2019.

[50] Github - praetorian-inc/slack-c2bot: Slack c2bot that executes commands and returns the output. `https://github.com/praetorian-inc/slack-c2bot`, April 2019.

[51] Operation ghost: The dukes arent back they never left | welivesecurity. `https://www.welivesecurity.com/2019/10/17/operation-ghost-dukes-never-left/`, October 2019.

[52] Rocke evolves its arsenal with a new malware family written in golang | anomali labs. `https://www.anomali.com/blog/rocke-evolves-its-arsenal-with-a-new-malware-family-written-in-golang`, March 2019.

[53] Using slack web services as a c2 channel (att&ck t1102) - praetorian. `https://www.praetorian.com/blog/using-slack-as-c2-channel-mitre-attack-web-service-t1102/`, April 2019.

[54] Apt-31 leverages covid-19 vaccine theme | zscaler blog. `https://www.zscaler.com/blogs/security-research/apt-31-leverages-covid-19-vaccine-theme-and-abuses-legitimate-online`, October 2020.

[55] Daac2 - using discord as a c2 | crawl3r. `https://crawl3r.github.io/2020-01-25/DaaC2`, January 2020.

[56] Eset_threat_report_q22020.pdf. `https://www.welivesecurity.com/wp-content/uploads/2020/07/ESET_Threat_Report_Q22020.pdf`, July 2020.

[57] Github - crawl3r/daac2: Discord as a c2. `https://github.com/crawl3r/DaaC2`, January 2020.

[58] Github - fsecurelabs/c3: Custom command and control (c3). a framework for rapid prototyping of custom c2 channels, while still providing integration with existing offensive toolkits. https://github.com/FSecureLABS/C3, August 2020.

[59] Introduction to callidus. https://3xpl01tc0d3r.blogspot.com/2020/03/introduction-to-callidus.html, March 2020.

[60] Pawn storms lack of sophistication as a strategy. https://www.trendmicro.com/en_us/research/20/l/pawn-storm-lack-of-sophistication-as-a-strategy.html, December 2020. (Accessed on 08/31/2023).

[61] Raccoon stealers abuse of google cloud services and multiple delivery techniques - trendlabs security intelligence blog. https://blog.trendmicro.com/trendlabs-security-intelligence/raccoon-stealers-abuse-of-google-cloud-services-and-multiple-delivery-techniques/, March 2020.

[62] Targeted attacks using fake flash against tibetans | volexity. https://www.volexity.com/blog/2020/03/31/storm-cloud-unleashed-tibetan-community-focus-of-highly-targeted-fake-flash-campaign/, March 2020.

[63] The tetrade: Brazilian banking malware goes global | securelist. https://securelist.com/the-tetrade-brazilian-banking-malware/97779/, July 2020.

[64] Indigozebra apt continues to attack central asia with evolving tools - check point research. https://research.checkpoint.com/2021/indigozebra-apt-continues-to-attack-central-asia-with-evolving-tools/, July 2021.

[65] North korean apt inkysquid infects victims using browser exploits | volexity. https://www.volexity.com/blog/2021/08/17/north-korean-apt-inkysquid-infects-victims-using-browser-exploits/, August 2021.

[66] Numando: Count once, code twice | welivesecurity. https://www.welivesecurity.com/2021/09/17/numando-latam-banking-trojan/, September 2021.

[67] Ousaban: Private photo collection hidden in a cabinet | welivesecurity. https://www.welivesecurity.com/2021/05/05/ousaban-private-photo-collection-hidden-cabinet/, May 2021.

[68] Apt41, a dual espionage and cyber crime operation | mandiant. https://www.mandiant.com/sites/default/files/2022-02/rt-apt41-dual-operation.pdf, February 2022. (Accessed on 03/30/2024).

[69] Virustotal - intelligence overview. https://www.virustotal.com/gui/intelligence-overview, 2022. (Accessed on 02/25/2022).

[70] Information on attacks involving 3cx desktop app. https://www.trendmicro.com/en_us/research/23/c/information-on-attacks-involving-3cx-desktop-app.html, Unknown 2023.

[71] Mansour Ahmadi, Battista Biggio, Steven Arzt, Davide Ariu, and Giorgio Giacinto. Detecting misuse of google cloud messaging in android badware. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 103--112, 2016.

[72] Turki Al lelah, George Theodorakopoulos, Amir Javed, and Eirini Anthi. Machine learning detection of cloud services abuse as c&c infrastructure. *Journal of Cybersecurity and Privacy*, 3(4):858--881, 2023.

[73] Turki Al lelah, George Theodorakopoulos, Philipp Reinecke, Amir Javed, and Eirini Anthi. Abuse of cloud-based and public legitimate services as command-and-control (c&c) infrastructure: a systematic literature review. *Journal of Cybersecurity and Privacy*, 3(3):558--590, 2023.

[74] Norah Alanazi, Esam Khan, and Adnan Gutub. Inclusion of unicode standard seamless characters to expand arabic text steganography for secure individual uses. *Journal of King Saud University-Computer and Information Sciences*, 2020.

[75] Blake Anderson and David McGrew. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*, pages 1723--1732, 2017.

[76] Blake Anderson, Daniel Quist, Joshua Neil, Curtis Storlie, and Terran Lane. Graph-based malware detection using dynamic analysis. *Journal in computer Virology*, 7:247--258, 2011.

[77] Liviu ARSENE. Iranian chafer apt targeted air transportation and government in kuwait and saudi arabia. https://www.bitdefender.com/blog/labs/iranian-chafer-apt-targeted-air-transportation-and-government-in-kuwait-and-saudi-arabia/, May 2020.

[78] Steven Arzt. Static data flow analysis for android applications. 2017.

[79] Usukhbayar Baldangombo, Nyamjav Jambaljav, and Shi-Jinn Horng. A static malware detection system using data mining methods. *arXiv preprint arXiv:1308.2831*, 2013.

[80] RF Jonell Baltazar, Joey Costoya, and R Flores. The heart of koobface: C&c and social network propagation. *Trend Micro Threat Research*, 2009.

[81] Joe Hannon Ben Koehl. Microsoft securitydetecting empires in the cloud - microsoft security blog. `https://www.microsoft.com/security/blog/2020/09/24/gadolinium-detecting-empires-cloud/`, September 2020.

[82] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. Design and analysis of a social botnet. *Computer Networks*, 57(2):556--578, 2013.

[83] Jean-Ian Boutin. Turlas watering hole campaign: An updated firefox extension abusing instagram | welivesecurity. `https://www.welivesecurity.com/2017/06/06/turlas-watering-hole-campaign-updated-firefox-extension-abusing-instagram/`, June 2017.

[84] Chris Brook. Windows 8 malware using google docs to target brazilians | threatpost. `https://threatpost.com/windows-8-malware-using-google-docs-target-brazilians-111912/77227/`, November 2012.

[85] Pieter Burghouwt, Marcel Spruit, and Henk Sips. Towards detection of botnet communication through social media by monitoring user activity. In *International Conference on Information Systems Security*, pages 131--143. Springer, 2011.

[86] Pieter Burghouwt, Marcel Spruit, and Henk Sips. Detection of covert botnet command and control channels by causal analysis of traffic flows. In *International Symposium on Cyberspace Safety and Security*, pages 117--131. Springer, 2013.

[87] Pete Burnap, Richard French, Frederick Turner, and Kevin Jones. Malware classification using self organising feature maps and machine activity data. *computers & security*, 73:399--410, 2018.

[88] Alfie Champion. Attack detection fundamentals: C2 and exfiltration - lab #3. `https://labs.f-secure.com/blog/attack-detection-fundamentals-c2-and-exfiltration-lab-3/`, July 2020.

[89] Raj Chandel. Command and control with dropboxc2. `https://www.hackingarticles.in/command-and-control-with-dropboxc2/`, April 2019.

[90] Joey Chen. Blackgear cyberespionage campaign resurfaces. `https://www.trendmicro.com/en_us/research/18/g/blackgear-cyberespionage-campaign-resurfaces-abuses-social-media-for-cc-communication.html`, July 2018.

[91] Wei Chen, Peihua Gong, Le Yu, and Geng Yang. An adaptive push-styled command and control mechanism in mobile botnets. *Wuhan University Journal of Natural Sciences*, 18(5):427--434, 2013.

[92] Wei Chen, Xiapu Luo, Chengyu Yin, Bin Xiao, Man Ho Au, and Yajuan Tang. Cloudbot: Advanced mobile botnets using ubiquitous cloud technologies. *Pervasive and Mobile Computing*, 41:270--285, 2017.

[93] Anton Cherepanov. The rise of telebots: Analyzing disruptive killdisk attacks | welivesecurity. `https://www.welivesecurity.com/2016/12/13/rise-telebots-analyzing-disruptive-killdisk-attacks/`, December 2016.

[94] Catalin Cimpanu. Astaroth malware hides command servers in youtube channel descriptions | zdnet. `https://www.zdnet.com/article/astaroth-malware-hides-command-servers-in-youtube-channel-descriptions/`, May 2020.

[95] Alberto Compagno, Mauro Conti, Daniele Lain, Giulio Lovisotto, and Luigi Vincenzo Mancini. Boten elisa: A novel approach for botnet c&c in online social networks. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 74--82. IEEE, 2015.

[96] Lucian Constantin. Malware uses google docs as proxy to command and control server. `https://www.pcworld.com/article/455736/malware-uses-google-docs-as-proxy-to-command-and-control-server.html`, November 2012.

[97] Assaf Dahan. Operation cobalt kitty: A large-scale apt in asia carried out by the oceanlotus group. `https://www.cybereason.com/blog/operation-cobalt-kitty-apt`, May 2017.

[98] Christian J Dietrich, Christian Rossow, Felix C Freiling, Herbert Bos, Maarten Van Steen, and Norbert Pohlmann. On botnets that use dns for command and control. In *2011 seventh european conference on computer network defense*, pages 9--16. IEEE, 2011.

[99] Yulong Dong, Jun Dai, and Xiaoyan Sun. A mobile botnet that meets up at twitter. In *International Conference on Security and Privacy in Communication Systems*, pages 3--21. Springer, 2018.

[100] Mohammad Reza Faghani and Uyen Trang Nguyen. Socellbot: A new botnet design to infect smartphones via online social networking. In *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1--5. IEEE, 2012.

[101] Matthieu Faou. From agent.btz to comratăv4: A tenyear journey | welivesecurity. `https://www.welivesecurity.com/2020/05/26/agentbtz-comratv4-ten-year-journey/`, May 2020.

[102] Matthieu Faou. Turla crutch: Keeping the back door open | welivesecurity. `https://www.welivesecurity.com/2020/12/02/turla-crutch-keeping-back-door-open/`, December 2020.

[103] Gregory Fedynyshyn, Mooi Choo Chuah, and Gang Tan. Detection and classification of different botnet c&c channels. In *Autonomic and Trusted Computing: 8th International Conference, ATC 2011, Banff, Canada, September 2-4, 2011. Proceedings 8*, pages 228--242. Springer, 2011.

[104] Lord Alfred Remorin Feike Hacquebord. Pawn storms lack of sophistication as a strategy. `https://www.trendmicro.com/en_us/research/20/l/pawn-storm-lack-of-sophistication-as-a-strategy.html`, December 2020.

[105] Dan Fleck, Arnur Tokhtabayev, Alex Alarif, Angelos Stavrou, and Tomas Nykodym. Pytrigger: A system to trigger & extract user-activated malware behavior. In *2013 International Conference on Availability, Reliability and Security*, pages 92--101. IEEE, 2013.

[106] Tomá Foltýn. Turla: In and out of its unique outlook backdoor | welivesecurity. `https://www.welivesecurity.com/2018/08/22/turla-unique-outlook-backdoor/`, August 2018.

[107] Mansoureh Ghanadi and Mahdi Abadi. Socialclymene: A negative reputation system for covert botnet detection in social networks. In *7'th International Symposium on Telecommunications (IST'2014)*, pages 954--960. IEEE, 2014.

[108] Nicholas Griffin. Carbanak group uses google for malware command-and-control | forcepoint. `https://www.forcepoint.com/blog/x-labs/carbanak-group-uses-google-malware-command-and-control`, January 2017.

[109] Josh Grunzweig. The tophat campaign: Attacks within the middle east region using popular third-party services. `https://unit42.paloaltonetworks.com/unit42-the-tophat-campaign-attacks-within-the-middle-east-region-using-popular-third-party-services/`, January 2018.

[110] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. 2008.

[111] Yukun He, Guangyan Zhang, Jie Wu, and Qiang Li. Understanding a prospective approach to designing malicious social bots. *Security and Communication Networks*, 9(13):2157--2172, 2016.

[112] Vladislav Hrka. Stantinko botnet adds cryptomining to its pool ofăcriminal activities | welivesecurity. `https://www.welivesecurity.com/2019/11/26/stantinko-botnet-adds-cryptomining-criminal-activities/`, November 2019.

[113] Noora Hyvärinen. The dukes: 7 years of russian cyber-espionage - f-secure blog. `https://blog.f-secure.com/the-dukes-7-years-of-russian-cyber-espionage/`, September 2015.

[114] PIERRE DELCHER IVAN KWIATKOWSKI, FÉLIX AIME. Holy water: ongoing targeted water-holing attack in asia | securelist. `https://securelist.com/holy-water-ongoing-targeted-water-holing-attack-in-asia/96311/`, March 2020.

[115] ANTON IVANOV and FEDOR SINITSYN. The first cryptor to exploit telegram | securelist. `https://securelist.com/the-first-cryptor-to-exploit-telegram/76558/`, November 2016.

[116] Karmina Jarkko. News from the lab archive : January 2004 to september 2015. `https://archive.f-secure.com/weblog/archives/00002803.html`, April 2015.

[117] Yuede Ji, Yukun He, Xinyang Jiang, Jian Cao, and Qiang Li. Combating the evasion mechanisms of social bots. *computers & security*, 58:230--249, 2016.

[118] Yuede Ji, Yukun He, Xinyang Jiang, and Qiang Li. Towards social botnet behavior detecting in the end host. In *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 320--327. IEEE, 2014.

[119] Yuede Ji, Yukun He, Dewei Zhu, Qiang Li, and Dong Guo. A mulitiprocess mechanism of evading behavior-based bot detection approaches. In *International conference on information security practice and experience*, pages 75--89. Springer, 2014.

[120] Jen Miller-Osborn Josh Grunzweig. Sunorcal adds github and steganography to its repertoire, expands to vietnam and myanmar. `https://unit42.paloaltonetworks.com/unit42-sunorcal-adds-github-steganography-repertoire-expands-vietnam-myanmar/`, November 2017.

[121] Erhan J Kartaltepe, Jose Andre Morales, Shouhuai Xu, and Ravi Sandhu. Social network-based botnet command-and-control: emerging threats and countermeasures. In *International conference on applied cryptography and network security*, pages 511--528. Springer, 2010.

[122] Yuhei Kawakoya, Eitaro Shioji, Makoto Iwamura, and Jun Miyoshi. Api chaser: Taint-assisted sandbox for evasive malware analysis. *Journal of Information Processing*, 27:297--314, 2019.

[123] Sheharbano Khattak, Naurin Rasheed Ramay, Kamran Riaz Khan, Affan A. Syed, and Syed Ali Khayam. A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys Tutorials*, 16(2):898--924, 2014.

[124] Barbara Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. 2007.

[125] SM Kuitert. War on botnets. *International Journal for Information Technology and Engineering Research*, 2009.

[126] Ajit Kumar, KS Kuppusamy, and Gnanasekaran Aghila. A learning model to detect maliciousness of portable executable using integrated feature set. *Journal of King Saud University-Computer and Information Sciences*, 31(2):252--265, 2019.

[127] TONY LAMBERT. Threat hunting in linux for rocke cryptocurrency mining malware. `https://redcanary.com/blog/rocke-cryptominer/`, April 2021.

[128] Majd Latah. Detection of malicious social bots: A survey and a refined taxonomy. *Expert Systems with Applications*, 151:113383, 2020.

[129] Hayoung Lee, Taeho Kang, Sangho Lee, Jong Kim, and Yoonho Kim. Punobot: Mobile botnet using push notification service in android. In *International workshop on information security applications*, pages 124--137. Springer, 2013.

[130] Ritthichai Limarunothai, Mohd Munlin, et al. Trends and challenges of botnet architectures and detection techniques. *Journal of Information Science & Technology*, 5(1), 2015.

[131] Lei Liu, Songqing Chen, Guanhua Yan, and Zhao Zhang. Bottracer: Execution-based bot-like malware detection. In *Information Security: 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008. Proceedings 11*, pages 97--113. Springer, 2008.

[132] Daniel Lunghi, Jaromir Horejsi, and Cedric Pernet. Untangling the patchwork cyberespionage group. `https://www.trendmicro.com/en_gb/research/17/l/untangling-the-patchwork-cyberespionage-group.html`, December 2017.

[133] Marc-Etienne M. Léveillé. I see what you did there: A look at the cloudmensis macos spyware. `https://www.welivesecurity.com/2022/07/19/i-see-what-you-did-there-look-cloudmensis-macos-spyware/`, July 2022.

[134] Romain Dumont Matthieu Faou. A dive into turla powershell usage | welivesecurity. `https://www.welivesecurity.com/2019/05/29/turla-powershell-usage/`, May 2019.

[135] Maersk Menrige. Plugx rat with time bomb abuses dropbox for command-and-control settings - trendlabs security intelligence blog. `https://blog.trendmicro.com/trendlabs-security-intelligence/plugx-rat-with-time-bomb-abuses-dropbox-for-command-and-control-settings/`, June 2014.

[136] Shishir Nagaraja, Amir Houmansadr, Pratch Piyawongwisal, Vijit Singh, Pragya Agarwal, and Nikita Borisov. Stegobot: a covert social network botnet. In *International Workshop on Information Hiding*, pages 299--313. Springer, 2011.

[137] Saima Naz and Dushyant Kumar Singh. Review of machine learning methods for windows malware detection. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1--6. IEEE, 2019.

[138] Jose Nazario. Twitter based botnet command and control (2009), 2015.

[139] Edmund Brumaghin Nick Biasini and Nick Lister. Cisco talos intelligence group - comprehensive threat intelligence: Threat spotlight: Astaroth maze of obfuscation and evasion reveals dark stealer. `https://blog.talosintelligence.com/2020/05/astaroth-analysis.html`, May 2020.

[140] STEVE MILLER BARRY VENGERIK NICK CARR, KIMBERLY GOODY. On the hunt for fin7: Pursuing an enigmatic and evasive global criminal operation | mandiant. `https://www.mandiant.com/resources/fin7-pursuing-an-enigmatic-and-evasive-global-criminal-operation`, August 2018.

[141] Cedric Pernet, Elliot Cao, Jaromir Horejsi, Joseph C. Chen, and William Gamazo Sanchez. New SLUB backdoor uses GitHub, communicates via Slack. `https://www.trendmicro.com/en_gb/research/19/c/new-slub-backdoor-uses-github-communicates-via-slack.html`, Mar 2019.

[142] Cedric Pernet, Elliot Cao, Jaromir Horejsi, Joseph C. Chen, and William Gamazo Sanchez. Slub gets rid of github, intensifies slack use - trendlabs security intelligence blog. https://blog.trendmicro.com/trendlabs-security-intelligence/slub-gets-rid-of-github-intensifies-slack-use/, July 2019.

[143] Matías Porolli. Polonium targets israel with creepy malware. https://www.welivesecurity.com/2022/10/11/polonium-targets-israel-creepy-malware/, October 2022.

[144] Vladimir Radunović and Mladen Veinović. Malware Command and Control Over Social Media : Towards the Server-less Infrastructure. 17(3):357--375, 2020.

[145] Karthik Raman et al. Selecting features to classify malware. *InfoSec Southwest*, 2012:1--5, 2012.

[146] Bryan Lee Robert Falcone. Darkhydrus delivers new trojan that can use google drive for c2 communications. https://unit42.paloaltonetworks.com/darkhydrus-delivers-new-trojan-that-can-use-google-drive-for-c2-communications/, January 2019.

[147] Kyle Wilhoit Ruchna Nigam. Telerat: Another android trojan leveraging telegrams bot api to target iranian users. https://unit42.paloaltonetworks.com/unit42-telerat-another-android-trojan-leveraging-telegrams-bot-api-to-target-iranian-users/, March 2018.

[148] Igor Santos, Jaime Devesa, Felix Brezo, Javier Nieves, and Pablo Garcia Bringas. Opem: A static-dynamic approach for machine-learning-based malware detection. In *International joint conference CISIS12-ICEUTE´ 12-SOCO´ 12 special sessions*, pages 271--280. Springer, 2013.

[149] Silpa Sebastian, Sonal Ayyappan, and P Vinod. Framework for design of graybot in social network. In *2014 international conference on advances in computing, communications and informatics (ICACCI)*, pages 2331--2336. IEEE, 2014.

[150] Kamalakanta Sethi, Rahul Kumar, Lingaraj Sethi, Padmalochan Bera, and Prashanta Kumar Patra. A novel machine learning based malware detection and classification framework, 2019.

[151] Andrii Shalaginov, Sergii Banin, Ali Dehghantanha, and Katrin Franke. Machine learning aided static malware analysis: A survey and tutorial. *Cyber threat intelligence*, pages 7--45, 2018.

[152] PV Shijo and AJPCS Salim. Integrated static and dynamic analysis for malware detection. *Procedia Computer Science*, 46:804--811, 2015.

[153] Wang Shuai, Cui Xiang, Liao Peng, and Li Dan. S-url flux: A novel c&c protocol for mobile botnets. In *International Conference on Trustworthy Computing and Services*, pages 412--419. Springer, 2012.

[154] Sérgio SC Silva, Rodrigo MP Silva, Raquel CG Pinto, and Ronaldo M Salles. Botnets: A survey. *Computer Networks*, 57(2):378--403, 2013.

[155] Ryan Singel. Hackers use twitter to control botnet | wired. `https://www.wired.com/2009/08/botnet-tweets/`, August 2009.

[156] Ashutosh Singh. Social networking for botnet command and control. 2012.

[157] Ashutosh Singh, Annie H. Toderici, Kevin Ross, and Mark Stamp. Social Networking for Botnet Command and Control. *International Journal of Computer Network and Information Security*, 5, 2013.

[158] Kapil Singh, Abhinav Srivastava, Jonathon Giffin, and Wenke Lee. Evaluating emails feasibility for botnet command and control. In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pages 376--385, 2008.

[159] Nikko Tamana. Backdoor uses evernote as command-and-control server - trend-labs security intelligence blog. `https://blog.trendmicro.com/trendlabs-security-intelligence/backdoor-uses-evernote-as-command-and-control-server/`, March 2013.

[160] Mathieu Tartare Thibaut Passilly. The sidewalk may be as dangerous as the crosswalk | welivesecurity. `https://www.welivesecurity.com/2021/08/24/sidewalk-may-be-as-dangerous-as-crosswalk/`, August 2021.

[161] Kurt Thomas and David M Nicol. The koobface botnet and the rise of social malware. In *2010 5th International Conference on Malicious and Unwanted Software*, pages 63--70. IEEE, 2010.

[162] Micah Yates Tom Lancaster. Confucius says...malware families get further by abusing legitimate websites. `https://unit42.paloaltonetworks.com/unit42-confucius-says-malware-families-get-further-by-abusing-legitimate-websites/`, September 2016.

[163] TrendMicro. Biopass rat new malware sniffs victims via live streaming. `https://www.trendmicro.com/en_us/research/21/g/biopass-rat-new-malware-sniffs-victims-via-live-streaming.html`, Unknown 2021.

[164] Nguyen H Vo and Josef Pieprzyk. Protecting web 2.0 services from botnet exploitations. In *2010 Second Cybercrime and Trustworthy Computing Workshop*, pages 18--28. IEEE, 2010.

[165] Zhi Wang, Chaoge Liu, Xiang Cui, Jie Yin, Jiaxi Liu, Di Wu, and Qixu Liu. Deepc2: Ai-powered covert command and control on osns. In *International Conference on Information and Communications Security*, pages 394--414. Springer, 2022.

[166] Jungsoo An Warren Mercer, Paul Rascagneres. Cisco talos intelligence group - comprehensive threat intelligence: Rokrat reloaded. `https://blog.talosintelligence.com/2017/11/ROKRAT-Reloaded.html`, November 2017.

[167] Vitor Ventura Eric Kuhla. Warren Mercer, Paul Rascagneres. Cisco talos intelligence group - comprehensive threat intelligence: Jhonerat: Cloud based python rat targeting middle eastern countries. `https://blog.talosintelligence.com/2020/01/jhonerat.html`, January 2020.

[168] WeLiveSecurity. Whos swimming in south korean waters? meet scarcrufts dolphin. `https://www.welivesecurity.com/2022/11/30/whos-swimming-south-korean-waters-meet-scarcrufts-dolphin/`, November 2022. (Accessed on 07/04/2023).

[169] Jake Williams. Dropsmack: How cloud synchronization services render your corporate firewall worthless. `https://docs.huihoo.com/blackhat/europe-2013/bh-eu-13-dropsmack-jwilliams-wp.pdf`, 2013.

[170] Guanhua Yan, Nathan Brown, and Deguang Kong. Exploring discriminatory features for automated malware classification. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 10th International Conference, DIMVA 2013, Berlin, Germany, July 18-19, 2013. Proceedings 10*, pages 41--61. Springer, 2013.

[171] Shuang Zhao, Patrick PC Lee, John CS Lui, Xiaohong Guan, Xiaobo Ma, and Jing Tao. Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 119--128, 2012.

Appendix

Appendix A: Extracted Features from Dual-Sandboxing Technique

Table 7.1: Features extracted using the Dual-Sandboxing Technique

Feature Category	Features
Domain Contact	domain_contact_file.core.windows.net, domain_contact_blob.core.windows.net, domain_contact_onedrive.live.com, domain_contact_sharepoint.com, domain_contact_graph.microsoft.com, domain_contact_twitter, domain_contact_docs.google, domain_contact_mail.google, domain_contact_chat.google, domain_contact_classroom.googleapis, domain_contact_sheets.googleapis, domain_contact_slides.googleapis, domain_contact_storage.googleapis, domain_contact_gmail, domain_contact_onedrive, domain_contact_dropbox, domain_contact_github, domain_contact_raw.githubusercontent, domain_contact_pastebin, domain_contact_youtube,

Table 7.1 – continued from previous page

Feature Category	Features
	domain_contact_script.google, domain_contact_translate.google, domain_contact_spreadsheets.google, domain_contact_slack, domain_contact_hotmail, domain_contact_outlook, domain_contact_amazonaws, domain_contact_azure, domain_contact_office, domain_contact_discord, domain_contact_telegram, domain_contact_instagram, domain_contact_OneNote, domain_contact_Teams, domain_contact_Evernote, domain_contact_aws.amazon, domain_contact_CloudMe, domain_contact_Imgur, domain_contact_pCloud, domain_contact_disk.yandex, domain_contact_alibabacloud, domain_contact_Mega
API Calls	api_call_InternetOpenA, api_call_InternetConnectA, api_call_HttpOpenRequestA, api_call_InternetReadFile, api_call_InternetWriteFile, api_call_WinHttpOpen, api_call_WinHttpConnect, api_call_WinHttpOpenRequest, api_call_WinHttpSendRequest, api_call_WinHttpReceiveResponse, api_call_WinHttpReadData, api_call_WinHttpWriteData,

Table 7.1 – continued from previous page

Feature Category	Features
	api_call_URLDownloadToFileA, api_call_HttpSendRequestA, api_call_InternetOpenUrlA, api_call_InternetReadFileExA, api_call_InternetWriteFileExA
System Interactions	total_processes, total_file_created, total_file_recreated, total_dll_loaded, total_file_opened, total_file_copied, total_regkey_opened, total_file_written, total_file_exists, total_command_line, total_file_read, total_regkey_read, total_file_deleted, total_registry_key_opened, total_registry_key_written, total_mutex_created, total_directory_created, total_regkey_deleted, total_mutex, total_file_failed, total_guid, total_directory_enumerated, total_regkey_written, total_resolves_host, total_file_moved, total_connects_host, total_connects_ip, total_wmi_query, total_downloads_file,

Table 7.1 – continued from previous page

Feature Category	**Features**
	total_fetches_url, total_directory_removed
Network Activities	total_http_requests, total_https_requests, total_dns_requests, unique_dns_requests, total_tcp_connections, total_udp_connections, num_domains, num_ips, num_urls