Imperial College London
Department of Computing

Network Intrusion Detection Against Advanced Persistent Threats

Almuthanna A. Alageel

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London
November 30, 2023

Abstract

The high complexity and low volume of Advanced Persistent Threats (APTs) attacks have led to limited insight into their behaviour and to a scarcity of data, hindering research on effective detection techniques. In this thesis, we explore the current defences used against APTs, and then we focus on network intrusion detection (NIDS). To understand the requirements of a novel NIDS against APTs, We select 33 APT campaigns based on the fair distribution over the past 22 years to observe the evolution of APTs over time We focus on their evasion techniques and how they stay undetected for months or years. We found that APTs cannot continue their operations without C&C servers, which are mostly addressed by Domain Name System (DNS). The next step for APT operators is to start communicating with a victim. One of the most popular protocols to deploy evasion techniques is using HTTP(S) with 81% of APT campaigns. HTTP(S) can evade firewall filtering and pose as legitimate web-based traffic. DNS protocol is also widely used by 45% of APTs for DNS resolution and tunnelling.

Therefore, we present a comprehensive study of the usage of domains in the context of C&C infrastructure of APTs, covering the available data of all reported APTs. Based on the available and historical data from industry reports, live DNS and WHOIS files, and historical data from SecurityTrail, we are able to collect data from 63 APT campaigns spanning the last 13 years. For those beyond these years, live data or historical ones are not available. We discuss and propose an evidence-based APT threat model, focusing in particular on evasion techniques, and collect an extensive dataset for studying APT C&C domains. Based on the gained insight, we propose a number of novel features to detect APTs, leveraging both the semantic properties of the domains themselves and the structural properties of their DNS infrastructure. We build HawkEye, a system to classify domain names extracted from PCAP files, and use it to evaluate the performance of the various features we studied and compare them to malicious domain detection features from the literature. The classification performance of HawkEye reaches an accuracy of 98.53%, a macro average F1-score of 90.38%, and a low FPR of 0.48% against unseen APT campaigns. In comparison, the baseline achieves lower performance, with accuracy, F1-score, and FPR values of 96.95%, 76.81%, and 0.68%, respectively.

For the malicious traffic, we present an evidence-based analysis of various Tactics, Techniques, and Procedures (TTPs) that use HTTP(S) protocols. These TTPs are known to be used by several campaigns to evade NIDS, according to MITRE ATT&CK [1]. The threat model

highlights the importance of the context around the malicious connections, and suggests traffic attributes which help APT detection. We also introduce measurement studies on various APT malware over popular and novel features to capture TTPs usage.

Next, We present EarlyCrow, an approach to detect APT malware Command and Control connections using *contextual summaries*. The design of EarlyCrow is informed by the threat model focused on TTPs present in traffic generated by malware recently used as part of APT campaigns. EarlyCrow defines a novel multipurpose network flow format called PairFlow, which is leveraged to build the contextual summary of a PCAP capture, representing key behavioural, statistical and protocol information relevant to APT TTPs. EarlyCrow shows a high classification performance against unseen APTs with a headline macro average F1-score of 93.72% and an accuracy of 98.11% with an FPR of 0.74%. In comparison, the state of the art stands at 60.29% and with no false positive rates.

However, other APT operators may use different protocols, such as Raw TCP, for the whole malicious operation. In addition, accessing the header of the application layer, such as HTTP(S) header, may not be allowed for some enterprises with stricter privacy requirements for their users. Therefore, we introduce NightVision, which can only access the packets length, timestamp, TCP flag and port number. NightVision leveraged statistical digital signal processing techniques, including Wavelet and Fourier transforms, to extract key information from the time-series data. In addition, we build hierarchical clustering to help in profiling the traffic volume based on an enterprise's network. NightVision also tracks TCP flags using a Finite State Machine (FSM) to identify scanning activities. Our system performance attains an average F1-score of 80.09% and an accuracy rate of 97.71%. with also a low FPR of 0.25%. In comparison, the state of the art baseline performs at 67.61%, 95.82%, and 1.61%, respectively.

Our tools presented in this thesis are designed to work together. The DNS requests are investigated first by HawkEye before a connection is established. The successful connections are parsed, and information is collected contextually via PairFlow. Based on the protocol type used for the traffic, the connection is investigated and classified using EarlyCrow and NightVision. Our approach is to enhance the security defences. Therefore, current Host Intrusion Detection Systems (HIDS) are also crucial to defend against APT. We recommend using our approach in parallel with the work published in the field of HIDS.

Acknowledgements

I have been fortunate to receive the support of many brilliant people throughout my PhD. I sincerely thank my supervisor, Dr Sergio Maffeis, for his unlimited support and guidance. I would also like thanks to my examiners, Dr Naranker Dulay (Imperial College London) and Dr Enrico Mariconti (University College London), for their invaluable feedback.

To Saleh, Musab, Adi, Layla, Arwa, and Omamah, thank you for your unconditional support throughout my life. To my wife Linah and my daughters Jolene and Judy, for all their love and encouragement over the years, without which this thesis would not have been possible.

Dedication

To my mother, Latifah, who raised me and imparted invaluable life lessons, including teaching me how to read and do math. I hope she feels a sense of pride in this work, and I deeply wish she could be here to experience this moment.

My father, Abdulaziz, who consistently took me to King Abdulaziz Library, inspired me to read as many books as possible.

My brother, Osamah, who never hesitated to provide any form of support, including financial support, has been a crucial lifeline throughout my journey.

To my brother, Asem, who has been by my side from the very beginning, providing encouragement, guidance, and financial support.

To Omar, my close brother, who has always been there when I needed him.

Their sacrifices have played a pivotal role in shaping the person I am today. Therefore, I dedicate this work to all of them.

Statement of Originality

I certify that the work presented in this thesis is my own. Where I have drawn on the work of others, it is appropriately referenced.

"By avoiding reliance, voyaging the world, enduring with patience unconsciously, and being as early as a crow."
Amer bin Sharaheel Al Shaabi, A.C. 721, on his quest to become a scholar

Frequent Abbreviations

AES - Advanced Encryption Standard

AGD - Algorithmically Generate Domain

AHC - Agglomerative Hierarchical Clustering

APT - Advanced Persistent Threats

ASN - Autonomous System Number

C&C - Command and Control

ccTLD - Country Code TLD

CTU - Concatenated Technical Unigram

DAG - Directed Acyclic Graph

DDE - Dynamic Data Exchange

DDNS - Dynamic DNS

DFT - Discrete Fourier Transform

DGA - Domain Generation Algorithm

DoH - DNS over HTTPS

DSP - Digital Signal Processing

DWT - Discrete Wavelet Transform

ELD - Enterprise Level Domain

FFT - Fast Fourier Transform

FSM - Finite State Machine

FIR - Finite Impulse Response

FQDN - Fully Qualified Domain Names

gTLD - Generic TLD

HPF - High-Pass Filter

HSG - High-Level Scenario Graph

HT-NN - Hamming Threshold Nearest Neighbour

HIDS - Host Intrusion Detection Systems

IIR - Infinite Impulse Response

IoC - Indicator of Compromise

LPF - Low-Pass Filter

MFTU - Most Frequent Technical Unigram

NCD - Normalised Compression Distance

NIDS - Network Intrusion Detection System

NLP - Natural Language Processing

NS - Name Server

NXDomain- Non-Existent Domain

OWA - Outlook Web Access

RAT - Remote Access Trojan

RC4 - Rivest Cipher 4

ROT13 - ROTate by 13 places

RR - Resource Records for the DNS

SEV - String ELD Vector

SLD - Secondary-Level Domain

SMA - Simple Moving Average

SMB - Server Message Block

SOC - Security Operations Center

SOCKS - Socket Secure

SWC - Strategic Web Compromise

TLD - Top-Level Domain

TOR - The Onion Router

TTPs - Tactics, Techniques, and Procedures

WLU - Weighted Longest Unigram

WLTU - Weighted Longest Technical Unigram

Mathmatical Symbols

$\vec{a}$: Vector of ASCII unigrams

α: The minimum threshold of the distance between two strings

b_k: Reconfigured scaling coefficients

$\vec{c}$: Vector of common unigrams

c_k Scaling coefficients

$c(.)$: Centroid function

$C(S_1, S_2)$: Compressor function to return the bytes of the compressed strings

$\vec{d}$: Vector of dictionary unigrams

$d_m(t)$: Signal details at scale m

$NCD_e(S_1, S_2)$: Entropy normalised compression distance between two strings

$\phi_{m,n}(t)$: Scaling or dilation function for wavelet transform

$\vec{p}$: Vector of popular unigrams

$\rho(X, Y)$: Pearson Correlation of X and Y

$S_m in$: The approximation coefficients

$S_{m,n}(t)$: Approximation coefficients at scale m and location n

$\vec{t}$: Vector of technical unigrams

$T_{m,n}(t)$: Wavelet coefficients at scale m and location n

$x[t]$: Signal in time domain

$X_R[k]$: The real part of Fourier transform basis functions

$X_J[k]$: The imaginary part of Fourier transform basis functions

$X_M[k]$: The magnitude part of Fourier transform in polar notation

$X_P[k]$: The phase part of Fourier transform in polar notation

$\vec{w}$: Vector of worldwide unigrams

$\Psi(t)$: Wavelet transform

$\Psi_{m,n}(t)$: Wavelet transform at scale m and location n

$argmin_x$: finds the global minimum of x subject to the constraints given

$P(Y|X)$: Probability of Y given X

$X \to Y$: X is a direct cause of Y

$\perp\!\!\!\perp$: B is independent of X

$\not\perp\!\!\!\perp$: B is dependent on X

Contents

List of Tables

List of Figures

Chapter 1

Introduction

APTs cost the UK government and worldwide nearly GBP 27 billion annually and USD 1 trillion [12], respectively. According to the National Institute of Standards and Technology (NIST) special publication (SP) 800-61 [13], an APT can be defined as a highly skilled adversary who targets organisations with specified objectives with multiple attack vectors over a long period. The majority of objectives of APTs are espionage, data exfiltration and damaging critical infrastructure. These campaigns' well-trained cyber security personnel come from military and governmental organisations, academia and R&D entities [3]. In this chapter, we discuss the motivation of this thesis, which helps us to identify the main objectives. We briefly describe each chapter and then summarise the thesis contributions.

1.1 Motivation

Malware families that rely on command and control for communication are well studied and are targeted by several NIDSs. For instance, conventional botnet behaviour includes launching DDoS, propagating quickly to other hosts, internal scanning, or sending spam [14]. In the past, anomaly-based and NetFlow feature-based were able to detect such behaviour and filter it out, detecting the abnormality of a connection over a relatively short period. APT malware is designed to keep a low profile and facilitate incoming targeted and stealthy attacks over a

long period. Other tools and TTPs are deployed to behave as legitimate-looking HTTP(S) connections [1]. Therefore, relying on the traffic frequency, beaconing behaviour, malicious domain [15], or NXDOMAIN generated by DGA in many botnets [16] is insufficient to detect APT communications. Also, the existing defences that adopt machine learning [17, 18, 19, 20, 21, 22] to detect non-targeted attacks, are less effective for APT detection, as we will discuss in Sections 5.1, 5.4 and 5.6.

TTPs can also be adopted to evade conventional rule-based NIDS or unexplainable ones using Deep Learning-based approaches [23]. One of the main reasons is that APT data is scarce, and it is difficult to collect enough data representing the adopted TTPs. Also, it is common for APT operators to deploy encrypted malicious traffic, and many NIDS struggle to detect such behaviour [24, 25]. Another recent concern is the privacy-preserving requirement, whether restricting the extraction of the information details or length of retainment in storage.

For the above reasons, which are further discussed in Chapter 2, a candidate system may adopt feature-based machine learning approaches. While machine learning models help in achieving generalisation, features can be proposed to improve the performance and the explainability according to the problem space, i.e. APT detection in this thesis. Raw data, such as PCAP, WHOIS or DNS zone files, can provide the raw fields for each packet or domain, respectively. These fields can be used as features. However, in this thesis, we refer to basic features as fields. These fields are required to be parsed, preprocessed, grouped or further action according to a proposed algorithm. We refer to features for those directly input to a supervised machine learning model for classification. These features are generated by algorithms that may use Natural Language Processing (NLP), probability models, rule-based, unsupervised machine learning, or Digital Signal Processing (DSP) techniques.

1.2 Objectives

The starting point for designing an effective APT detection is to study the TTPs reported by the industry. These TTPs are challenging to be classified as malicious or legitimate, as presented in Chapter 3. Therefore, a candidate approach should consider the context of the attack for a specific domain. In the APT domain used by C&C (Chapter 4), an adversary may choose their host in the same country where the enterprise is located, select the related brand name, and frequently change the IP address to escape from tracking by SOC analysts. Therefore, the character, lexical and infrastructure-based features of APTs should be considered holistically and be relevant to the targeted enterprise. In APT traffic (Chapters 5 and 6), TTPs cause significant overlap in statistical features between APT and legitimate connection, leading to the importance of proposing features based on the context of a connection.

Therefore, our scope mainly includes five objectives:

1. **To understand and profile APT campaigns from the industry reports:** To start our research, we aim to understand the APT campaigns throughout all cycles with their indicator of compromise (IoCs), TTPs, and most commonly abused protocols.

2. **To construct a threat model for network-based APT attacks:** To construct an APT network-based model that extends the current APT models and overcomes the shortcomings in terms of evasion techniques of command and control (C&C) used by an APT campaign based on technical analysis of real samples.

3. **To investigate the possibility of detecting APTs from the domain names:** Before the malicious C&C is established, the local network should be able to verify that the requested domain is legitimate and not APT. This should be a first line of defence from the network perspective to mitigate the attacks, which is one of our main targets. However, it may not be sufficient, which requires traffic-based detection as the next objective.

4. **To provide early detection as possible under different scenarios of malicious traffic:** We aim to detect stealthy and evasive C&C traffic from their early establishment

until exchanging information. The approaches should incorporate various lines of defence to detect C&C.

5. **The defence should preserve the privacy of users :** Privacy concerns force enterprises to investigate with minimum access to the data. Therefore, the next generation of NIDS should balance the technology behind detection and the privacy guidelines.

1.3 Thesis Overview

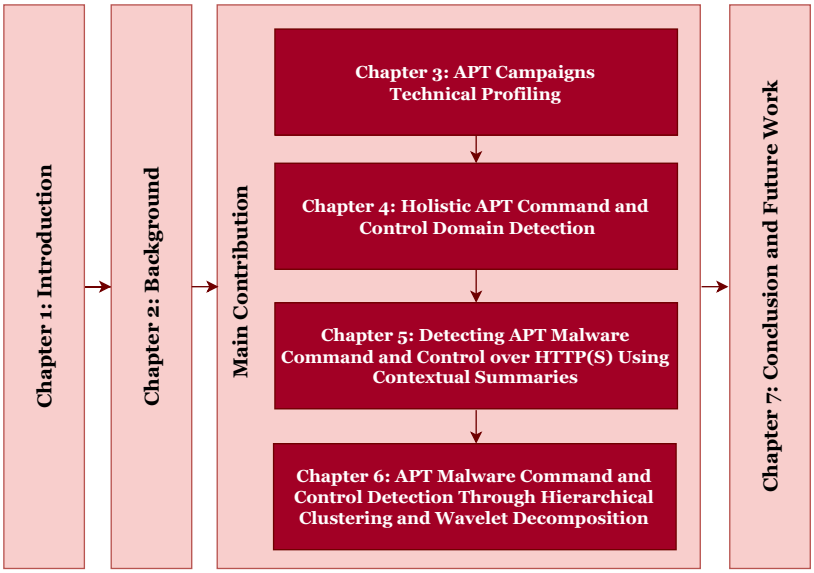


Figure 1.1: Thesis overview.

The thesis is organised (Figure 1.1), as follows:

Chapter 2: This chapter starts with a survey of recent worldwide incidents of APTs in multiple sectors. Then, we introduce three attack models with some discussions on how these are related to our scope. We present the state of art research conducted on APT campaign network-based detection. These papers investigate a wide range of protocols, including HTTP, HTTPS, DNS, P2P and custom protocols used by RAT. This is followed by a discussion around the gaps that need to be filled in the current research and possible future trends. Next, we present background on the techniques that we used in this thesis, including supervised and unsupervised machine learning for cybersecurity and digital signal processing.

Chapter 3: In this chapter, we select 33 APT campaigns based on a fair distribution over the past 22 years to observe the evolution of APTs over time. We focus on their communications over multiple channels with a wide range of evasion techniques. Furthermore, the analysis includes the APT campaign delivery method, vulnerabilities, backdoors, protocols, obfuscation, payloads, and the total number of FQDNs and resolved IP addresses. One of the most popular protocols to deploy evasion techniques is using HTTP(S) with 81% of APT campaigns. HTTP(S) can evade firewall filtering and pose as legitimate web-based traffic. DNS protocol is also widely used by 45% of APTs for DNS domain resolution and tunnelling. Then we describe the related TTPs used by APT campaigns from the network perspective. Based on industry reports and our technical analysis, we describe and discuss those TTPs related to DNS, traffic and alternative channels.

Chapter 4: We present the most comprehensive study to date of the usage of domains in the context of APT Command and Control (C&C) infrastructure. We cover 63 APT campaigns spanning the last 13 years until August 2020 by reviewing 125 public reports and 146 threat intelligence pulses by 35 leading security organisations. Based on that, we propose a detailed threat model for APT C&C usage, which focuses, in particular, on evasion techniques. We leverage the gained insight to study the effectiveness of existing and novel features of domain names and their DNS infrastructure for detecting APT C&C domains. We built HawkEye, a system to classify domain names requested in PCAP files. HawkEye includes modules for robustly parsing and retrieving DNS data, which is sometimes technically challenging, and for extracting features and classifying domains. We find that a holistic feature set including largely orthogonal features performs the best, achieving an accuracy of 98.53% and a macro average F1-score of 90.38% and an FPR of 0.48% for unseen APT campaigns, compared to the state of the art, which performs at 96.95%, 76.81%, and 0.68%, respectively.

Chapter 5: This chapter introduces PairFlow, a novel multipurpose network flow format which captures the necessary fields of the connections between host pairs of interest, over a granular time tuned by security practitioners. These fields are extracted via three main components, i.e., Tracking, Aggregation, and Encapsulation. Next, we build EarlyCrow a feature-based classifier that benefits from the contextual fields introduced by PairFlow.

EarlyCrow generates four sets of data focused on connections, hosts, destinations, and URLs. Each of these has its features grouped to form a ContextualSummary. The ContextualSummary has multidimensional features that help in building more informative random forest trees used for classification. We evaluate the effectiveness of EarlyCrow on unseen APTs, obtaining a headline macro average F1-score of 93.72%, and an accuracy of 98.11% with an FPR of 0.74%. Our results outperform the baseline, which stands at 60.29% and with no false positive rates.

Chapter 6: This chapter provides an approach regardless of the protocol type. This can be useful for those APT operators who use raw TCP. It also benefits enterprises if they have strict standards on exposing DNS or HTTP-based features. Therefore, we build NightVision which only accesses the time series of network data without diving into the header of the application layers (DNS, HTTP, TLS ... etc). NightVision adopts the Discrete Wavelet Transform (DWT) to build a matching filter as a template, which is popular in radar applications for specific malicious TTP. We also adopt the popular use of ECG to identify heart disease using DWT. Many features, such as statistical ones, can be extracted for each frequency band and select the relevant bands for malicious behaviour.

To enhance NightVision, we also introduce enterprise-based gauges that can be tailored for a specific organisation. We mainly profile the traffic volume, non-application and scanning activities using a combination of Agglomerative Hierarchical Clustering (AHC) for the training, and a decision tree model for the testing time. Various extracted features are introduced to feed the clustering and decision tree models, including those that track TCP flags activities using Finite State Machine (FSM) that mainly distinguish normal behaviour from SYN, ACK and FIN scan attacks. The average F1-score of our system is 80.09%, indicating its overall performance. It also demonstrates a high accuracy rate of 97.71% and a low FPR of 0.25%, compared to the state of the art baseline, which performs at 67.61%, 1.61%, and 95.82%, respectively.

Chapter 7: We summarise the thesis results and present the limitations and future work.

1.4 Contributions

- Profiling 33 APT campaigns based on industry reports. We discuss their malware delivery methods, vulnerabilities, backdoor families, and their evasion techniques Over TCP/IP Protocols for C&C configurations. Our investigation shows that 81% and 45% of APT campaigns exploit HTTPS and DNS protocols. Also, we find that 60.6%, 54.5% and 27.2% use fallback and encrypted channels as well as dynamic DNS, respectively.

- A detailed, evidence-based analysis of the use of domains in the context of APT C&C infrastructure. This includes technical analysis based on domain and traffic artefacts. We also present the first publicly available, curated dataset of APT domains used for C&C infrastructure. The collected dataset follows the best practice of implementing causal analysis to troubleshoot datasets.

- The implementation of HawkEye, a classifier to detect APT C&C domain requests from PCAP files, crawling live DNS and WHOIS data where necessary. Hawk-Eye presents new features for malicious domain detection targeted for APTs. We evaluate the classification performance of new and existing features for the APT C&C domain detection under different scenarios. Our evaluation indicates that the most effective approach involves using holistic features. With this approach, we achieve an accuracy of 98.53%, a macro average F1-score of 90.38%, and a low FPR of 0.48% when detecting unseen APT campaigns. In comparison, the baseline achieves lower performance, with accuracy, F1-score, and FPR values of 96.95%, 76.81%, and 0.68%, respectively.

- We introduce PairFlow, which summarises a PCAP into contextually relevant fields, including packet behaviour, domain and URL list, TLS/SSL certificates and client and server cipher suits and extensions, User-Agent, status code, and content type for HTTP. PairFlow generates four main output files for consumption by NIDSs.

- We implement EarlyCrow, a system to detect evasive malicious communication, in particular from APTs and botnets. EarlyCrow leverages the context of communication through four perspectives, including PairFlow, host, destination, and URL. We evaluate

the classification performance of new and existing features for malicious traffic detection under different scenarios, distinguishing ATP, botnet and legitimate traffic. EarlyCrow defends well against unseen APTs that encrypt their traffic with HTTPS, obtaining a headline macro average F1-score of 93.72%, and an accuracy of 98.11% with an FPR of 0.74%. In comparison, the state of the art stands at 60.29% and with no false positive rates.

- We design NightVision, an approach that can capture attacks regardless of the protocol types. This includes non-HTTP(s) protocols, TCP and UDP-based or even non-application protocols. NightVision can also meet the standard of enterprise, which requires not accessing application features such as the header of HTTP or DNS. The evaluation of NightVision includes several malware families, including APTs, botnets, scans, and DDoS. We also evaluate NightVision against unseen malware and whether it sustains performance over time. Our system obtains a macro average F1-score of 80.09% and an accuracy of 97.71% with an FPR of 0.25% compared to the state of the art which performs at 67.61%, 95.82%, and 1.61%.

1.5 Publications

Published Papers

- **Alageel, Almuthanna**, and Sergio Maffeis. "HAWK-EYE: holistic detection of APT command and control domains." In the Proceedings of the 36th annual ACM symposium on applied computing, Security Track, pp. 1664-1673. 2021.
  DOI: *10.1145/3412841.3442040.*
  Included in: Chapter 4.

- **Alageel, Almuthanna**, and Sergio Maffeis. "EARLYCROW: Detecting APT Malware Command and Control over HTTP(S) Using Contextual Summaries." In the 25th International Conference, ISC 2022, December 18–22, 2022, Proceedings, pp. 290-316. Cham: Springer International Publishing, 2022.
  DOI: *10.1007/978-3-031-22390-7_18.*
  Included in: Chapter 5.

Planned Papers:

- **Alageel, Almuthanna** and Sergio Maffeis. "APT Network-based Industrial and Technical Profiling." In the Journal of Computer and Security, Elsevier, ISSN 0167-4048 (To be submitted)
  Included in: Chapter 3

- **Alageel, Almuthanna** and Sergio Maffeis. "NIGHTVISION: APT Malware Command and Control Detection Through Hierarchical Clustering and Wavelet Decomposition" In USENIX Security 2024 (To be submitted)
  Included in: Chapter 6

Chapter 2

Background

In this chapter, we explore the history of APTs and highlight the objectives they achieved. We also examine the current APT lifecycles adopted in previous works. Following that, we review the literature of APT detection related to NIDS, and identify the research gaps that need to be covered. Then, we demonstrate techniques used in this Thesis to defend against APTs.

2.1 Overview

Since 2006, a large-scale cyber-espionage APT campaign called APT 1, a.k.a Comment Crew, has been compromising more than 140 organisations in twenty different sectors, including state administrations, energy, health, scientific research entities, aerospace, satellites, telecommunication, IT and financial services [3]. More than ninety organisations had failed to detect the malicious activity of APT 1 for an average of a year, while other organisations had been challenged to detect over 40 malware families used by APT 1 for up to four years and ten months. 87% of these organisations are headquartered in English-speaking countries with 6.5 TB of compressed data exfiltrated for more than ten months from a single organisation without detection. The espionage operation collected business plans, senior management emails, system design, simulation technologies and user credentials. From 2011 to 2013, APT 1 expanded C&C servers to establish more than 937 with 988 FQDNs resolved with unique IP addresses and to

transmit its traffic over HTTPS with thirteen different X.509 certificates [3].

From the beginning of 2007 until September 2014, APT 28, a.k.a Fancy Bear, scanned 1.7 million vulnerable IPs in Ukraine alone against several sectors such as government entities, telecommunication and aerospace companies [26]. The same campaign was famous for espionage against the Georgian military and other Eastern European states. The campaign had been successfully hidden from detection by mimicking legitimate domains, frequently updating to open C&C multichannel using DGA for its Fall-back Channels [27].

In mid-2009, the Operation Aurora campaign targeted more than 34 organisations, including Yahoo, Symantec, Morgan Stanely, and Google, that could not detect the infiltration for over six months [12]. Another campaign called APT 32, a.k.a OceanLotus or Operation Cobalt Kitty, delivered fileless payloads, i.e. in-memory, to hide their traces from forensics engineers [28]. The operation transferred 46 binary files and 24 scripts via several C&C channels, including Denis and Goopy Trojan, PowerShell script and outlook macro backdoors, Cobalt Strike, Don't-Kill-My-Cat evasion tool, custom NetCat and IP tools [29]. The stealthy C&C channel operated over DNS tunnelling to evade both NIDS and Firewalls [28].

Undetected for over seven years [30], a well-resourced cyberespionage group called APT 29, a.k.a CozyDuke, was found to infiltrate the US White House successfully, in addition to defence, energy and financial institutions in Western Europe and China [31] at the end of 2015. APT 29 is known for deploying many malware toolsets, most of which belong to the Duke malware family [30]. APT 3, a.k.a Buckeye, has been targeting nearly 84 organisations, including government departments in the US, the UK and Hong Kong, in low and slow mode since 2009 [32]. The campaign employed zero-day exploits through spearphishing to drop a remote access Trojan called Pirpi [32]. APT 3 is known to employ a vast arsenal of malware and TTP, including remote access tools (RAT), backdoors, keyloggers and Lazagne to extract passwords from the current application and exfiltrate all data back to C&C server [32].

Another stealthy and extremely successful campaign [33] is called Wekby or APT 18. The operation started in 2009 and has been known to use DNS as a medium to communicate with C&C server via HTTPBrowser, gh0st RAT and Pisloader for over six years without a detection [34].

Another cyber espionage, APT 37, utilised many zero-day vulnerabilities through spearphishing links or attachments, strategic web compromise (SWC) and Torrent file-sharing from 2012 until 2018 in support of military activities of the state against South Korea, Japan, and the Middle East [35]. The campaign delivered different malware families, including SLOWDRIFT, KARAE and POORAIM malware and TTP toolsets to exfiltrate highly classified data to cloud services and then to wipe the master boot record (MBR) of the organisation's data centre [35].

2.2 APT Malware

APT malware are those malicious tools known to be used by APT campaigns. They usually do not participate in DDoS attacks, sending spam or propagating to other hosts to spread infections [36]. APT malware is classified as targeted malware with different functions to communicate with C&C. For instance, a RAT is typically composed of a builder, stub, and controllers. The builder initiates a new instance stub upon the infection. The stub stays on the victim with preconfigured FQDN or IP to communicate to RAT's controller placed at the C&C server [37]. Trojans, spyware, and keyloggers may also be composed to connect C&C at a low profile. However, some malware, such as DarkCommet, includes these functions in one ecosystem [36], which may capture the audio, explore files and drop malicious tools through visiting URLs [38]. Griffon, used by FIN 7, can gather information, load Meterpreter, and take screenshots [39].

Once APT actors drop malware on infected hosts, they may not start with their manual instructions immediately. Therefore, several malicious activities can be done automatically to prepare the necessary information for the APT actors to serve their following steps. For instance, malware connects automatically to the C&C server, and other automatic operations are followed, such as information gathering, dropping more malicious tools or payloads, and establishing fallback channels. Therefore, it is crucial to detect these malicious tools initially, whether the actors start controlling them or not, as long as they are connected to C&C servers.

2.3 APT Lifecycle

Understanding how an APT campaign behaves from the first day until the mission is accomplished can be crucial in order to identify the attack surface in each stage. This also helps in selecting appropriate detectives or even protective controls against such a threat. Lockheed Martin Cyber Kill Chain® is considered as the first model that was able to describe APT proposed in 2011 [2]. Cyber Kill Chain® is then followed by Mandiant [3] and MITRE ATT&CK [4], both of which are increasingly accepted among the research community. However, there are two further models proposed by Vries et al. [40] and Attack Pyramid by Giura and Wang [41], each of which describes an APT as a normal attack lifecycle without any APT propriety, therefore, we omit these two models from our discussion for the brevity.

2.3.1 Lockheed Martin Cyber Kill Chain®

Hutchins et al. propose Cyber Kill Chain [2], which is part of a broader Intelligent Driven Defence framework funded and adopted by Lockheed Martin [2]. The model is widely adopted by several papers, including [42, 43, 44, 45]. It describes APTs in seven phases in one direction for one time, as depicted in Figure 2.1.

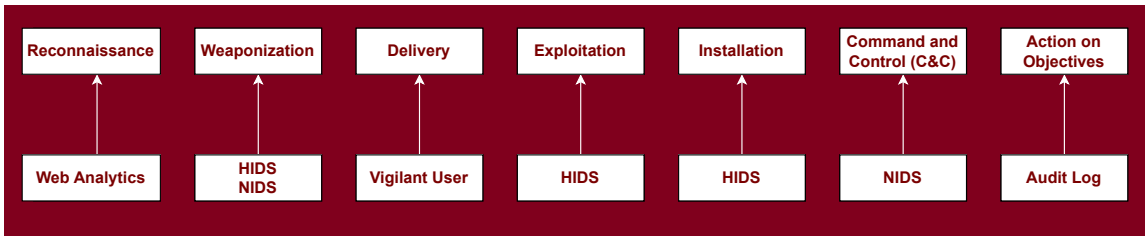


Figure 2.1: Lockheed Martin Cyber Kill Chain detective controls [2].

Cyber Kill Chain starts with the reconnaissance stage, which includes passive reconnaissance, scanning and enumeration. Then, the adversary starts developing malware and coupling it with exploits in the weaponisation stage. The malicious payload is then attached to a medium at the delivery stage, whether that medium could be a malicious link to a website that hosts the malicious payload or attached with a file, e.g. PDF or MS Word. When a target is vulnerable

to that exploitation (stage 4), an installer component attached to malware tends to establish a backdoor at stage five and be able to communicate back with the threat actor through the C&C server in stage six. The last stage is where the action can be made, which varies according to the APT objectives from the beginning, including espionage, data exfiltration and disruption. Hutchins et al. recommend a matrix of security measures against each stage, including detection, prevention, disruption, degrading, and deceiving, with respect to stage order. We illustrate in Figure 2.1 a course of action in terms of detective controls recommended by the authors [2].

2.3.2 Mandiant Attack Model

Mandiant Attack Model is proposed by FireEye to analyse APT 1 [3]. The model is adopted by many researchers, including [46, 47] that will be discussed in Section 2.4. This section will highlight the similarities and differences between the Mandiant and Cyber Kill Chain Models.

The Mandiant model begins with initial reconnaissance, which matches the first stage on the Cyber Kill Chain. Then, a threat actor initially compromises a host in stage 2, which matches stages 3-5 in the Cyber Kill Chain, i.e. delivery, exploitation and installation. It is worth noting that Mandiant does not consider weaponisation as well as merging three stages from the Cyber kill chain into one stage. This is because Mandiant focuses on defence strategy more than considering the full picture from the adversary's perspective. Next, the adversary establishes a foothold once a backdoor is already installed, which is equivalent to C&C - stage six - in Cyber Kill Chain.

The next four stages are performed n times by the adversary until an organisation takes defensive action upon detecting the malicious activities. The first stage of the n cycle stages is privilege escalation, where an adversary either delivers another zero-day exploit or passes the hash to take over a root account, for instance. Next, the adversary uses built-in operating system commands or a custom one to understand the local environment intellectually. The adversary moves around the network as a privileged and legitimate user who can evade all detective controls. The adversary can then launch an OS scheduler, a persistent technique in

the following stage, to install further backdoors or to exfiltrate data according to a specific schedule in the final stage as depicted in Figure 2.2. The iterative cycle might be conducted multiple times in order to open fall-back channels and maintain APT persistence.

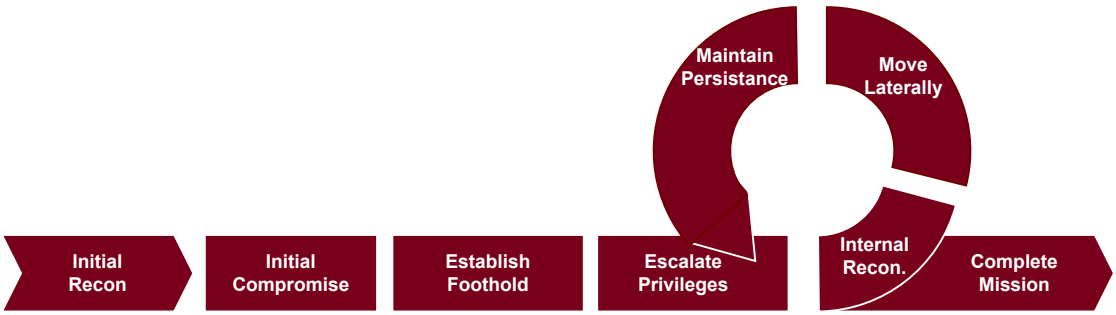


Figure 2.2: Mandiant attack model [3].

2.3.3 ATT&CK ™ Model

While Cyber Kill Chain and Mandiant APT attack models are presented in an abstract view and are focused on the lifecycle of an APT campaign, MITRE ATT&CK presents a matrix for enterprise to describe tools, tactics and procedure (TTP) of an APT campaign. The matrix and its TTPs are based on the observation of tens APT campaigns [4]. The TTP are presented across twelve stages as depicted in Figure 2.3. The total number of TTPs is 130, which an APT can select from such a pool. For example, APT 1 used 22 TTPs across all stages. It is worth paying attention to the new stages not presented by other models, such as defence evasion, credential access, discovery and collection. The defence evasion stage covers evasion techniques from the hardware to the application level. However, our analysis in Chapter 3 leads to an expanded view of network evasion techniques based on our observations. Internal reconnaissance in other models is divided here into discovery and collection, where the former represents an early stage of internal reconnaissance and the latter refers to the late one where accounts, files and emails, for example, are collected and sent back to C&C servers in C&C stage.

As we can notice from the description above and shown in Figure 2.3, the matrix presents the TTP with respect to each stage, but the stages are not chained to form a lifecycle. Therefore,

ATT&CK Matrix might be used as a companion and a pool of TTPs to other lifecycle models to overcome the absence of subsequent behaviour of an APT in order to provide proper security measures.

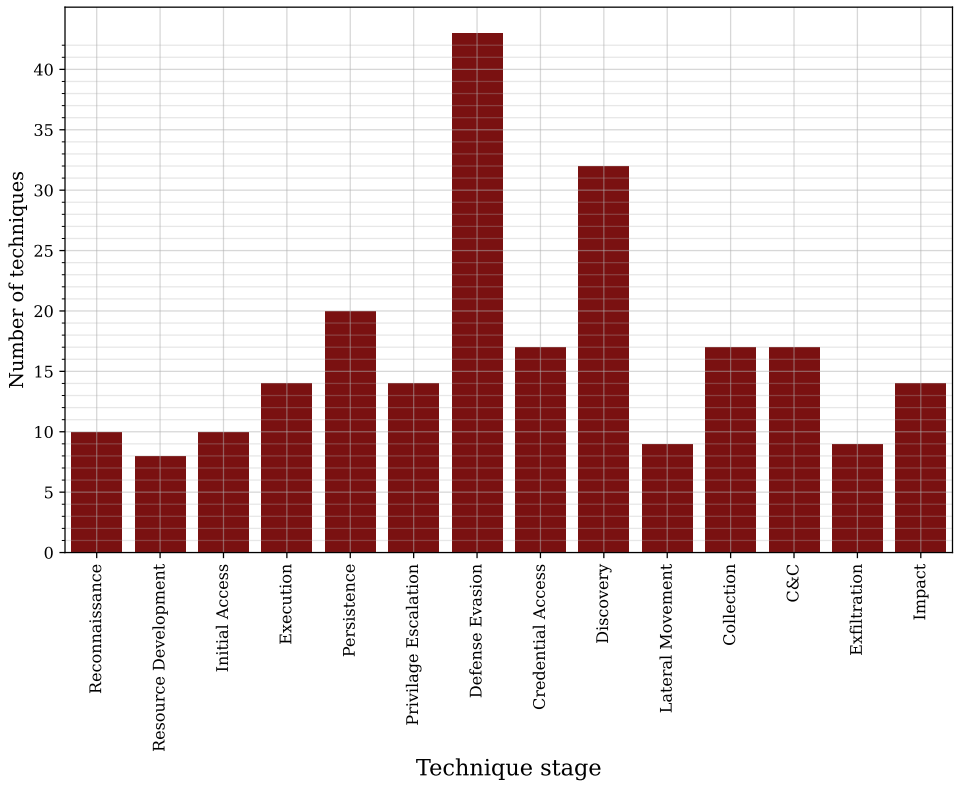


Figure 2.3: MITRE ATT&CK stages [4].

2.3.4 Discussion on APT Models

We note that Cyber Kill Chain takes the full picture from the adversary's point of view. For instance, stages 1-4 are out of the target's network perimeter. Establishing a foothold on the model is represented in stage six, while multiple subsequent actions are conducted and not presented well. Furthermore, unlike Mandiant, the cyber kill chain does not represent the cyclic nature of APT, where an APT starts expanding inside the target's network. In addition, the Cyber kill Chain merges the internal reconnaissance and lateral movement stages of the Mandiant model under the action on the objective stage.

Mandiant fits for research that focuses on host intrusion detection and endpoint protection more than the network perspective. This is because the Mandiant attack model does not explicitly

describe C&C stage, as we can see from Figure 2.2, which is our scope in this thesis. As regards to ATT&CK Matrix, it does adopt a lifecycle scheme. Instead, a pool of TTPs is presented in a concrete way and lacks abstraction for future TTP.

Due to the need for an APT campaign network-based attack model that considers the following requirements :

- C&C stage should be included in the proposed APT life cycle to identify when C&C is needed for an adversary and why.
- C&C stage should be divided into multiple stages to provide the details of attack behaviour. Stages may include DNS resolution, locating C&C servers and malicious C&C traffic.
- Evasion techniques at C&C stages reported by MITRE ATT&CK need to be considered based on real evidence, such as dynamic resolution, data obfuscation and fallback channels.

Based on these requirements, we propose two threat models in Chapters 4 and 5.

2.4 Literature Review

Published works on APT detection are quite limited. They are even far less when the scope focuses on network-based detection. There are two major surveys on APT detection published in the past. Both Alshamrani et al. [48] and Singh et al. [49] cover papers related to all APT stages. However, most covered papers are not precisely related to APTs due to the miss-definition. For instance, they include DDoS or botnets as APT. This is not accurate according to our discussion in the previous section. Therefore, organising the most relevant works to our scope is relatively challenging. However, we organise these papers in the remainder of this chapter (Table 2.1), according to the proposed detective controls of Lockheed Martin Cyber Kill Chain described in the previous section, limiting to recent research related to NIDS. Therefore, we omit papers that exclude APT traffic from their scope.

Table 2.1: APT campaigns research scope based on Cyber Kill Chain model.

Paper	**Reconnaissance**	**Weaponisation**	**Delivery**	**Exploitation**	**Installation**	**C&C**	**Objectives**
Marchetti et al [50]						✓	✓
Friedberg et al [51]						✓	✓
Zhao et al [5]						✓	
Barcelo et al [52]						✓	✓
Lu et al [53]						✓	
Vance [54]						✓	
Siddiqui et al [55]						✓	
Mathew et al [47]						✓	✓
Albanese et al [56]						✓	
Brogi and Tong [57]					✓	✓	✓
Atighetchi et al [58]	✓	✓				✓	✓
Vries et al [40]		✓	✓	✓	✓	✓	✓
Bhatt et al [59]		✓	✓	✓	✓	✓	✓
Milajerdi et al [46]		✓	✓	✓	✓	✓	✓
Giura and Wang [41]		✓	✓	✓	✓	✓	✓
John and Kinkelin [60]		✓	✓	✓	✓	✓	✓
Chandran et al [61]		✓	✓	✓	✓	✓	
Ioannou et al [45]		✓	✓	✓	✓	✓	
Wang et al [62]		✓	✓	✓	✓	✓	✓

2.4.1 Network-Based APT Detection

This section covers the recent works for detecting APT traffic using only network data. APT varies from one campaign to another, and many different protocols may be abused. They may use HTTP, DNS, Peer to Peer (P2P), Flow, or custom C&C. In Table 2.2, we summarise the scope of each paper in terms of protocols and the attack objectives.

2.4.1.1 HTTP-based Detection

Friedberg et al. [51] introduce a statistical anomaly detection model to detect APT based on a white list approach. The approach is based on a model M that can discover the relationships in the log line of the HTTP header from multiple systems. The system model M consists of four stages including search pattern $\mathbb{P}$, event classes $\mathbb{C}$, hypothesis $\mathbb{H}$ and rules $\mathbb{R}$. First, log-atom Log_a is extracted from the HTTP header and associated with a timestamp to extract log event Log_e. Then fingerprint vector $\vec{F_p}$ is generated after extracting n-gram from L_a into multiple search pattern $\mathbb{P}$ Now each feature in $\vec{F_p}$ is classified into their respective classes. Event class E is a 2-tuple of mask $\vec{E_m}$ and a value $\vec{E_v}$.

$$\vec{E_m} = p_1 ... p_n \quad \text{where} \quad p_i = \begin{cases} 1 & \text{if P at feature i is relevant} \\ 0 & \text{if P at feature i is irrelevant} \end{cases}$$

$$\vec{E_v} = p_1 ... p_n \quad \text{where} \quad p_i = \begin{cases} 1 & \text{if P at feature i is enforced} \\ 0 & \text{if P at feature i is prohibited} \end{cases}$$

This is not a final classification but a filter to make an assumption about the relevant features, which will be considered as a null hypothesis H_0 out of many hypotheses and to run a binomial test against the threshold level α. Now a set of rules $\mathbb{R}$ receives each stable H where $\mathbb{R} \subset \mathbb{H}$, i.e. generating rules of the correlation with a threshold level greater than or equal to α.

Table 2.2: Malicious traffic detection.

Paper	C&C Data Source or Protocol						Generic	Actions on Objectives Stage	
	Flow-based	HTTP	Encrypted	P2P	DNS	Custom		Data Exfiltration	Internal Recon.
NIDS									
Albanese et al [56]				✓				✓	
Barcelo et al [52]		✓							
Friedberg et al [51]		✓					✓		
Lu et al [53]	✓					✓			
Marchetti et al [50]	✓		✓					✓	
Siddiqui et al [55]	✓	✓				✓			
Vance [54]	✓						✓		
Zhao et al [5]	✓				✓				
IDS									
Chandran et al [61]	✓						✓		
Milajerdi et al [46]						✓	✓		✓
SIEM-like									
Giura and Wang [41]		✓			✓		✓		
Mathew et al [47]							✓		

Barcelo et al [52] present a comparison of three classification algorithms, i.e. genetic programming [63], decision trees and support vector machine to detect APT network traffic based on HTTP traffic and header. They simply label the anomalous data point on which their *anomalous score* > *threshold* and altogether form a *darkset* for validated data points by human experts and *greyset* for unknown behaviour. The dark set is used for training and testing, while the grey set is only used for testing. Examples of selected features are duration, bytes, HTTP reply code, HTTP method, server and client addresses and URL connections. The URL is further broken into features such as Top Level Domain (TLD), vowels and constants ratio, sequence of vowels, constants, numbers and symbols of Fully Qualified Domain Name (FQDN). Anomalous score (A_i) can be calculated based on the rare values of the following combination.

$$\mathrm{A_i} = \sum_{j=c1}^{cN} \omega_j \cdot A_{ij} \tag{2.1}$$

Where A_{ij} is the anomalous score for each feature, and w_j refers to the attribute relevance.

2.4.1.2 DNS-based Detection

DNS is used to resolve internal or external domains. Botnets, malware and APTs are increasingly using DNS to evade network intrusion detection in order to establish and locate C&C servers. Malware and RAT widely use Dynamic DNS, DGA, and other DNS techniques, investigated in Chapter 3, to locate and communicate with C&C servers. Zhao et al. [5] propose a DNS and IP traffic feature-based approach that is placed on the network edges. First, a technique is used to identify malicious C&C domain. Then, suspicious IPs are extracted and monitored accordingly. Using 14 DNS features and other IP-based ones, the system can compute the reputation score for a given host, as depicted in Figure 2.4.

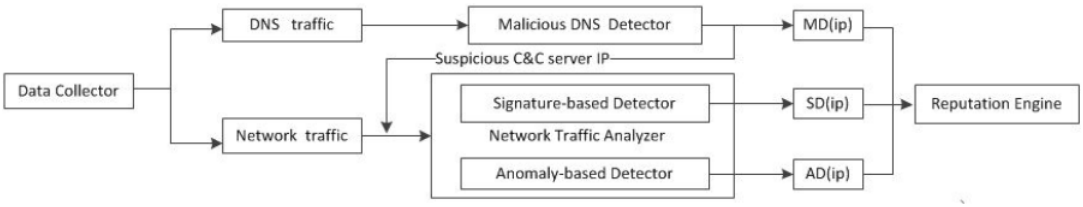


Figure 2.4: Reputation engine proposed by [5].

2.4.1.3 Custom C&C Detection

Thus far, we have seen several techniques that can establish C&C in APT campaigns. However, APT may use custom C&C. For instance, APT 1 uses custom C&C using RAT, such as Gh0st. Authors in [53, 46, 55] propose their approaches by considering custom C&C in their dataset for evaluation. We will discuss these papers in the following section.

2.4.1.4 Packet and Flow-based Detection

Some works focus on detecting malicious behaviour regardless of the characteristics of a specific protocol. Vance [54] proposes a NetFlow-based statistical model to detect C&C and data exfiltration. The proposed approach analyses and monitors the flow by creating a baseline for the stream of the packet size over a time window. For incoming flows with respect to IP address and port number, they measure the deviation using sketch-based change detection [64]. However, the framework is simple and assumes APT exceed the average of the legitimate volume, which may be the case for attacks like some variants of DDoS.

Lu et al. [53] propose a method based on observing time-based features and train a machine learning model for the prediction. They propose features including maximum time intervals, query, and response TCP numbers. According to [53], 40% of APT flow remains in-session for 1 second while only 5% remains for 15 seconds. In their evaluation, they mix 36 captures from the Contagio malware database [65] for malicious NetFlow with normal traffic collected from their campus networks. In addition, they synthesised malicious behaviour according to [55]. They reach 3.77%, 3.87%, 3.6% false positive rates for KNN, SVM and gradient boosting

decision tree (GBDT), which are quite high. Also, the F1-measure is not reported, which is essential for unbalanced datasets.

Siddiquie et al. [55] use a fractal dimension to correlate TCP features to detect APT traffic. First, they extract the raw features of the TCP session using Tshark [66] on PCAP file. Next, they compute the prior correlation fractal of anomalous and normal samples $fd_{prior_{anom}}$ and $fd_{prior_{norm}}$, respectively. Then, they load a set S of data points in the main memory. Using a data point p, where $p \subset S$, they calculate the posterior correlation fractal dimension $fd_{posterior_{anom}}$ and $fd_{posterior_{norm}}$. After taking the absolute difference of prior and posterior values of both anomalous and normal samples, a classification can be determined whether if $fd_{anom} < fd_{norm}$ then p is an anomaly, otherwise normal.

Marchetti et al. [50] develop a framework to rank the suspicious hosts involved in data exfiltration. The framework computes a score for each internal host S_t at time t. S_t score for each host comprises the distance (s_t^1) from the feature space centroid $c(X_t)$ at time t, the magnitude of movement (s_t^2), which reflects the rapid increase of the upload for a host at time t, and unlikelihood of movement direction (s_t^3) with respect to its history in the feature space. The feature vector for each host $x_t(h)$ only considers three features, i.e. number of bytes (x_t^1), number of flows (x_t^2) and number of destination (x_t^3). The final suspicious-host score S_t linearly combines s_t^1, s_t^2, s_t^3 as follows:

$$\mathrm{S_t} = \sum_{j=1}^{3} (q_t^j \cdot s_t^j) \tag{2.2}$$

Note that the combined scores are normalised with two-sided quantile q. However, the approach might be vulnerable to poisoning attacks over a long period so that an adversary can manipulate statistical inference, leading to a higher misclassification rate.

2.4.2 APT Compensating Controls for NIDS

In this section, we will discuss detecting APT traffic using a combination of NIDS and other technologies.

2.4.2.1 HIDS

Few works claim that host intrusion detection (HIDS) may enhance network-based performance and vice versa [61, 46]. Chandran et al. [61] present a classification model to detect APT at testing time. They collect data from CPU and memory usage, system32 files, open ports, domain names, and network captures. With the raw features of these logs, they use random forest for classification.

Milajeradi et al. propose Holmes, real-time correlation-based detection of suspicious information flows between processes and I/O resources (files, pipe and sockets) [46]. The main objective of Holmes is to track alerts coming from IDS and to report the presence of the APT campaign. Holmes tracks stealthy APT behaviour in multi-stage with data representation in a provenance graph and a High-Level Scenario Graph (HSG).

The provenance graph is a dependence graph that represents data received from audit logs [67]. Due to the storage cost to store the dependence graph, a graph-based reduction technique called source dependence is implemented, which reduces total events up to 19x with an optimised version graph technique [68]. HSG can be seen as an abstract of a provenance graph, and it summarises the APT campaign to allow the incident response team to react in real time. Each set of nodes and edges of the provenance graph matched ATT&CK TTP is represented by a node on HSG, while the edges represent the prerequisites between TTPs to reduce the noise to reduce FPR.

An example of a TTP chain across APT stages that summarised by HSG is that `untrusted _read(S,P)` for initial compromise, `c2(P,S)` for establish foothold, `sudo_exec(F,P)` for privilege escalation, `sensitive_command (P,P')` for internal reconnaissance, `send_internal (P,S)` for move laterally, `destroy_system (F,P)` for complete mission and finally clear_logs (P,F) for clean up tracks. However, the prerequisites to validate TTP are quite limited, i.e. 16 prerequisites, one for each TTP across seven stages for APT.

Finally, the attack chain is then represented on a 7-tuple, each of which is assigned a severity

level from low to high. The qualitative level is converted to a quantitative one based on the NIST severity rating scale [69]. Accordingly, Holmes ranks HSGs based on the overall score of the weighted product of a 7-tuple, i.e. w_i for i^{th} tuple is assigned by a security administrator.

However, works based on the provenance graph in [46, 67, 68] are designed based on limited reported attacks to be detected by host-based intrusion detection. Therefore, IPS or even SOC engineers will block the attack, whether this attack is a part of APT campaigns or not. The critical issue about APT is their large number of scenarios. In addition, the threshold and the noise reduction are determined manually for the seven dataset scenarios. Because of the generalisation issue, an adversary can extend their next TTP to evade correlation detection. In addition, Holmes is designed with rule-based, i.e. specific sets of prerequisites to validate whether a TTP belongs to the APT kill chain. An adversary can change his steps to evade Holmes and compromise further hosts, therefore breaking Holmes's rules.

2.4.2.2 SIEM-Like

Giura and Wang [41] present an APT detection framework based on the attack pyramid model. According to [41], the APT campaign determines a targeted goal G, which is located at the top of the attack pyramid model. The attack pyramid is based on the attack tree proposed by B Schneier [70]. For the adversary, there are multiple paths in order to reach G. These paths represent an attack tree with G as the root node.

Each node in the tree may be located in two dimensions. The first dimension of their model represents the TCP/IP layer, which is either application, network, users, physical or other planes. The second dimension represents the APT stages, i.e. exfiltration, data collection, operation, exploitation, delivery, and reconnaissance. The detection framework begins with collecting events from sensors for each pyramid plan. The correlation rules generate the context received by the alarm system to apply the detection rules. The detection rules evaluate each context based on their risk and confidence levels and update the threshold accordingly. Finally, the APT incident alarm may be raised once the context falls on the alarm zone.

However, the threshold of detection rules in the framework is updated in each context, which keeps it vulnerable to poisoning attacks. Second, the threshold of risk and confidence level of their detection rules is based on signature-based, which is vulnerable to zero-day and even simple evasion attacks. Also, the implementation of their framework is not clear. The APT scenarios, attacks, and evasion techniques are not explicitly specified.

Mathew et al. [47] design a tool called Event Correlation for Cyber Attack Recognition (ECCARS) on the top of Snort, an intrusion detection system [71]. The tool's mission is to find whether such an attack is part of a multi-stage attack. ECCARS is a real-time situation awareness based on guidance templates. The role of guidance templates is to correlate events with respect to their semantics and IP addresses. The event fusion by semantic correlation receives an attack event A paired with a stage S_i in order to build a complete attack tree A_T for such a multistage attack M_A. An example of attack events paired with multiple-stage is `Recon_Sniffing, Intrusion_User, Escalation_Service and Goal_Corruption`. The semantic correlation is validated by IP-address correlation, i.e. the transition of $E_1 \rightarrow E_2$ is valid only if $IP_{destination}(E_1) == IP_{source}(E_2)$ can hold.

However, the solution does not contain a detection process. The approach is purely about awareness rather than detection. They assume from the beginning that Snort will report intrusions and scanning attacks. Then, ECCARS ranks these threats to specific values. In addition, the correlation relies on source and destination IP addresses which are vulnerable to IP spoofing or packets coming from Tor or VPN. IP-address correlation-based could also be vulnerable against target multi-stage attacks over a long period of time, which frequently change their IPs with respect to each stage.

2.4.2.3 Moving Target Defence (MTD)

Albanese et al. claim that stealthy P2P botnets are known to be used for data exfiltration in APT campaigns [56]. They design a system called DeBot which is based on the moving target defence (MTD) technique to change the attack surface frequently. The main objective of DeBot is to detect the attempt of data exfiltration. DeBot implements several detectors

across the network and collects flow-based traffic from different points in order to highlight any suspicious hosts with anomalous behaviour that is associated with data exfiltration. They use periodogram analysis to detect the periodic behaviour of data exfiltration traffic of the APT campaign compared to normal traffic.

2.4.3 Discussion

We cover some works that attempt to detect APT traffic with different approaches and data sources. These approaches are either feature-based or rule-based. We also conclude that, in the context of APT detection, evaluating an approach against a real data set or semi-real data set is an essential requirement to prove the effectiveness of such an approach. It is also important to report TPR, FPR, Precision, and F1-measure, all of which are important for NIDS.

Table 2.3: Literature-based works results.

Paper	Techniques	Dataset	Metrics (%)							Scalability
			Acc.	TPR	TNR	FPR	FNR	Precision	F1	
Albanese et al [56]	Reinforcement Learning	N/A	–	–	–	1-7	–	–	–	M
Barcelo et al [52]	GP/SVM	Real Network	–	84.85	97.36	2.64	15.15	–	–	L
Chandran et al [61]	Random Forest	N/A	99.8	–	–	–	–	–	–	L
Friedberg et al [51]	Statistical Correlation	Three datasets [72]	–	80	–	3	–	–	–	L
Giura and Wang [41]	Heuristic Rules	Real Network		–	–	85.3	–	–	–	M
Lu et al [53]	Fuzzy Logic	Contagio	–	–	–	3.6	3.07	–	–	M
Mathew et al [47]	Heuristic Rules	N/A	–	85	–	–	–	85	–	M
Milajerdi et al [46]	Graph-based	DARPA TC [73]	–	100	–	–	–	100	100	S
Siddiqui et al [55]	Correlation Fractal	Contagio	94.42	93.58	94.43	5.57	6.41	15.02	29.99	L
Vance [54]	Sketch-based	Real-Network	–	–	–	1.28-1.93	–	–	–	M
Zhao et al [5]	Statistical-based	Real Network	–	95.7-96.3	–	1.3-1.7	–	–	–	L
Marchetti et al [50]	Statistical Correlation	Real Network	–	–	–	–	–	–	–	L
Papers without experimental results										
Atighetchi et al [58]	Semantic/Ontology	–	–	–	–	–	–	–	–	L
Bhatt et al [59]	Heuristic Rules	–	–	–	–	–	–	–	–	M
Brogi and Tong [57]	Heuristic Rules	–	–	–	–	–	–	–	–	S
Ioannou et al [45]	Markov	–	–	–	–	–	–	–	–	M
John and Kinkelin [60]	Statistical/ Heuristic Rules	Theoretical	–	–	–	–	–	–	–	M
Vries et al [40]	Heuristic Rules	–	–	–	–	–	–	–	–	S
Wang et al [62]	Gene-Based	–	–	–	–	–	–	–	–	S

2.4.3.1 Issues in the evaluation

In the previous two sections 2.4.1 and 2.4.2, we discuss multiple research on detecting APT traffic. Several evaluation metrics of binary classification are not reported for imbalanced datasets. We also notice that works on [47, 5, 51, 52, 55] have high TPR at the expense of other metrics such as precision or F1-measure. One of the reasons behind these results is the imbalance of data, i.e., the majority of data belongs to a legitimate class. For instance, precision in [55] is pretty low, i.e. 15.02%, while the proposed approach reaches 93.58% as TPR with 29.99% F1-measure. If this approach is applied in practice, it will cause tremendous false alarms, which in turn, keeps a SOC engineer from neglecting any further alarms. Albanese et al. [56] report low FPR without reporting the TPR. Marchetti et al. [50] provide a reasonable approach to detect data exfiltration by the host. However, they neither report the false positive rate nor F-1 measurement at least.

Another common issue we have observed on several papers is the dataset being used for evaluation. For instance, Giura and Wang [41], Mathew et al. [47], Vance [54], and Zhao et al. [5] do not confirm if the dataset has APT traces or not and the type of TTP used. A similar issue is observed in Chandran et al. [61] where the authors report 99.8% accuracy of their random forest with the absence of reporting the dataset and attack variants. Friedberg et al. [51] only detect APT attacks by monitoring HTTP header on the semi-synthetic dataset generated based on [72], discarding common scenarios where an adversary use HTTPS, which prevents their system from accessing the required features. Also, we identify in Chapter 5 features based on the HTTP header are insufficient to detect APT. For Holmes [46], it is not clear if it can generalise, i.e. the rule-based method is static and tailored to detect seven scenarios represented in the DARPA TC dataset [73]. In addition, their tool suffers from scalability issues. For instance, only 4 hosts can be stored on 64 GB memory and monitored over one year. The rest of these works [40, 45, 57, 58, 59, 60, 62] neither provide empirical results nor evaluate on real traces. However, Lu et al. evaluation [53] is similar to the one by Siddiqui et al. [55] in terms of using the Contagio APT dataset. Although they are the only two works that present different types of TTP for several APT campaigns, neither paper provides precision with either F1-measure.

2.4.3.2 Rule-based

The advantage of rule-based approaches is their fast performance and accuracy for specific attacks. However, rule-based approaches can be evaded by learning these rules or even simpler ones, such as those activities out of the rules' radar. Rule-based is designed to detect specific scenarios in [46] [67], so it is necessary to avoid overfitting to be able to detect attack variants. Context-based approaches can improve rule-based generalisation with lower FPR because of the ability to track several scenarios over a long period. However, they might be extremely expensive to apply to detect APT if the information is not summarised over a time window. At the beginning of this chapter, we discuss that APT campaigns are persistent and the operation may take up to months or years. Hence, collecting data from multiple planes while storing and constructing graphs for each host over months is computationally expensive [46]. As a result, the attack surface might be expanded, and the amount of data to be processed and stored can grow rapidly.

2.4.3.3 Feature-based

Therefore, we believe feature-based machine learning models are more practical now and in the future. The industry's next generation of detective controls uses machine and deep learning models such as anti-malware products introduced by security vendors like CrowdStrike, Sophos and Cylance [74]. Feature-based approaches require a deep understanding of multiple attack vectors on the network level to construct better features. For instance, IP-related features are vulnerable to IP spoofing, Tor and VPNs. Time-related features are vulnerable to the long and persistent nature of APT. Volume-related features are vulnerable to stealthy and low attacks. Therefore, relying on a single type of feature category may be insufficient to capture the various TTPs used by APTs. In addition, if the dataset is collected from a sandbox environment, some features may cause strong bias. Mariconti et al. recommend avoiding biased features, such as geolocation features, if the selected malware belongs to one campaign [75].

In the next sections, we discuss the machine learning models used in this thesis. We highlight

the importance of constructing features with high information gain rather than using raw fields directly from logs, as we presented in Chapters 4 and 5. We also explore techniques based on clustering and digital signal processing to develop new features with high information gain required in Chapter 6.

2.5 Machine Learning

Machine learning is a subfield of artificial intelligence that is broadly described as a machine's ability to replicate intelligent human behaviour. A machine learning model starts with data, which can include numbers, images, and text. The model's parameters will be automatically tuned based on the training data to provide the desired performance. Some data is excluded as a testing set, which is used to assess the machine learning model's accuracy when dealing with new data. As a result, we can apply the model to a wide range of data sets in the future.

Machine learning models are categorized into supervised, unsupervised and reinforcement learning. Our work in this thesis does not require reinforcement learning. Therefore, it is excluded from this section. There are common terms used interchangeably in the machine learning literature that we need to highlight to clear the confusion. The inputs of a model can also be called predictors, independent variables or features, while the output may also be called responses or dependent variables, page 28 of [76].

2.5.1 Supervised Machine Learning

The aim of supervised machine learning is to learn a mapping from input x to output y. In another way, we will have training set $\mathcal{D}$ such that $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^{N}$, where X_i represents a feature vector, y_i is the label and N is the number of training examples. When the label y_i is categorical, the problem we aim to solve is known as classification, while it can be regression if y_i is real-valued [7]. In this thesis, we focus on the classification problem. We aim to train a model so its output $y \in \{0, ..., C\}$. For binary classification, $C = 2$, while for multi-label classification,

$C > 2$. In the classification problem, we aim to achieve generalization by predicting the new data classes.

Several models return probability distribution over possible labels to achieve such a task [7]. In the training set, we calculate $p(y|X, \mathcal{D}, M)$, where X is the test input in this case, D is still the training set, and M is the model. For the binary classification, $p(y = 0|X, \mathcal{D}, M)$ for the first class and $p(y = 1|X, \mathcal{D}, M)$ for the second one [7]. Therefore, the probability for each class is conditional on the training and testing sets. In the end, we report the best prediction to the true label $\hat{y}$ by:

$$\hat{y} = \hat{f}(X) = \arg\max_{c=1}^{C} \quad p(y = c|X, \mathcal{D}, M) \tag{2.3}$$

Specific models can perform well for classification tasks. These models can be grouped as parametric or nonparametric. Parametric models rely on the tuned parameters for each variable on the training data to identify the decision boundary. Examples of models in this category include Linear Regression, Naive Bayes and Neural Networks. To grasp the idea of the parametric models, let us consider a mapping function for a line which is used for linear regression:

$$b_0 + b_1 \times x_1 + b_2 \times x_2 = 0 \tag{2.4}$$

b_0, b_1 and b_2 are the biases or coefficients that control the intercept and slope of the variables x_i. Now, the task in these kinds of models is to learn the estimation of these coefficients from the training examples to separate classes.

Nonparametric models rely more on informative features, which requires several techniques to improve the basic features to be stronger predictors before the training time. In this section, we limit our discussion to non-parametric models, especially tree-based ones, since the decision tree and random forest are popular models used in NIDS [19, 75, 77, 78, 61, 79].

2.5.1.1 Decision Tree Model

The decision tree model can learn the decision rules from the data features [6]. For example, In order to approximate a sine curve as a regression problem, decision trees learn from data a series of if-then-else rules. As the tree branches out, the more complicated the decision rules get, and the better the model fits the data [6].

Figure 2.5 shows an example of a training decision tree on the Iris dataset as a classification problem. The dataset comprises three classes of irises, i.e. Setosa, Versicolour, and Virginica, as labels, and four features, Sepal Length, Sepal Width, Petal Length and Petal Width [80]. As shown in Figure 2.5, the decision tree starts with a root, which is split into two nodes, decision nodes (right) and leaf nodes (left). So the root node is able to separate some samples on the leaf node, which belongs to Setosa, based on a condition of Petal Width <=0.8. The rest of the samples will go to the decision node and so on. The learning process here is to learn which feature to consider first, splitting conditions, and their threshold to split the data optimally.

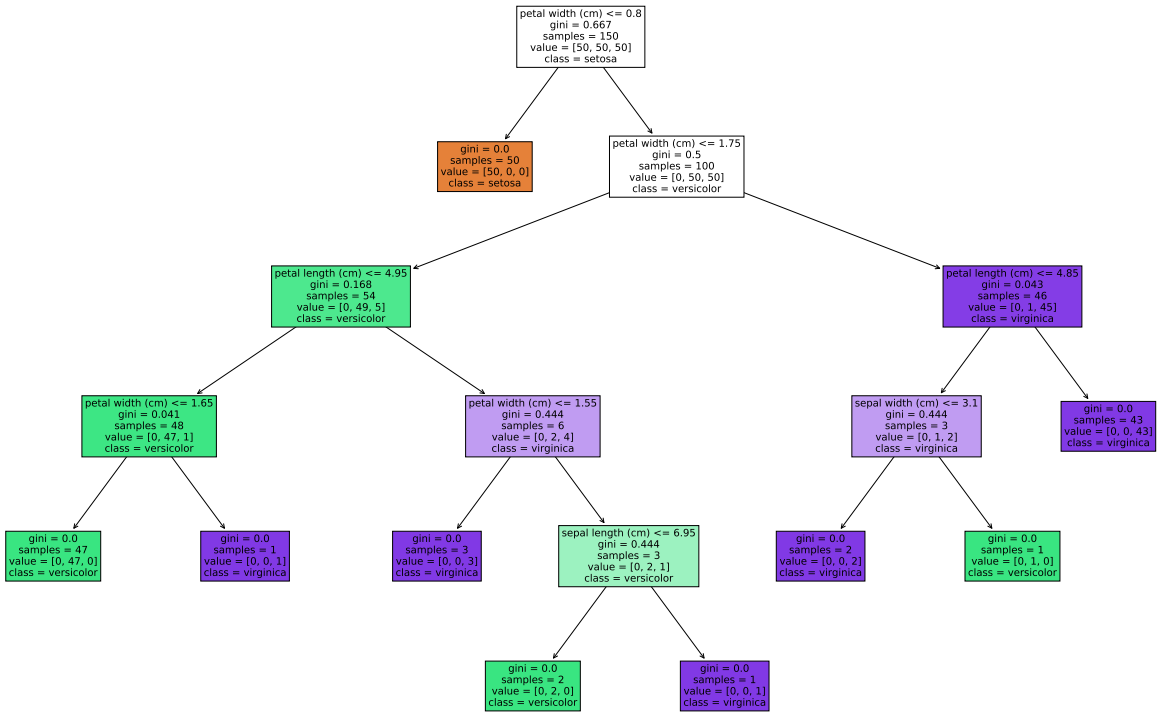


Figure 2.5: Decision tree example on Iris dataset [6].

The decision tree chooses the split that maximises the information gain. Gini index, Cross

entropy, or Log Loss can be used to quantify the gain, page 686 of [81]. We chose the default in the scikit-learn library, which is the Gini index, and defined as follows:

$$Gini\ Index = \sum_{i=1}^{n} p_i(1 - p_i) \tag{2.5}$$

Where p_i is the probability of class i of the samples at each level. So if we assume the Iris dataset is balanced, p_i= 0.33 at the root level since we have three classes. Then we recalculate the p_i at every level with the remaining samples for each class. The Higher index refers to high uncertainty about the randomly picked point. We keep calculating the Gini index for every possible split and choose the minimum one. As the split points are determined by ranking the data points, the decision tree model is not sensitive to monotone transformations of the inputs. It is fairly robust to outliers, scales well to large data sets, and can be adjusted to handle missing inputs, all of which contribute to their widespread adoption, page 550 of [7]. There are, however, limitations associated with the decision tree model. The primary downside is its high sensitivity to the training data, which results in high variance and may fail to generalise [7]. One of the recommended methods to overcome this drawback is to use random forest [7], which is discussed in the next section.

2.5.1.2 Random Forest Model

Random Forest is an ensemble model that combines several decision trees, which is why it is called a forest, using the bagging technique. The bagging technique stands for "bootstrap aggregating", page 551 of [7]. It adopts two random processes to build a model. Given a dataset, first, the model randomly selects samples to extract a new dataset from the original one such that the new dataset size equals the original one, but the sampling technique is based on bootstrapping, page 588 of [76]. Bootstrapping is a sample selection method with a replacement which means when a sample is drawn from a population, the same sample is returned back to the population, which means there is a possibility to draw the same sample [76]. The model creates a new dataset for each decision tree. For instance, if we select 10 trees, the model generates 10 datasets, one for each decision tree model.

The second random process is aggregating, where the features are randomly selected. Each new dataset will keep a random subset of the original features and remove the others [76]. To determine the number of random features to select, $\sqrt{\#\ of\ Features}$ or $\log_2(\#\ of\ Features)$ are the most popular, where the former is the default in scikit-learn library [82]. To predict a new sample, we will pass the new sample to each tree and record the result. Next, we take the majority vote for the classification problem and the average for the regression one [76].

To summarise the random forest improvement over the typical decision model. First, bootstrapping mitigates the decision tree's sensitivity to the original dataset. Second, the random feature selection decreases the correlation between the decision trees, which reduces the variance.

2.5.2 Unsupervisied Machine Learning

In this type of machine learning, we aim to discover knowledge from the data, such as finding or describing patterns among data and splitting them into common characteristic clusters. We only take the input in this setting such that $D = \{X_i\}_{i=1}^{N}$. Unsupervised machine learning models can be categorised into association rules, cluster analysis, self-organisation maps and component analysis [76]. We will focus on cluster analysis, particularly Hierarchical clustering since it is used in Chapter 6.

2.5.2.1 Clustering Analysis

The clustering process involves grouping a set of objects so that they are more similar than other objects in different groups, page 906 of [7]. The output of the models can be partitional clustering, where the output is partitioned into disjoint sets. Another possible output is hierarchical clustering, which organises the clusters as a tree. For the input, we assume objects can be presented as data points. Each data point is determined by a set of features. In clustering analysis, we tend to cluster the data points based on their similarity, which is called similarity-based clustering. The input to the model is an $N \times N$ distance metric.

2.5.2.2 Hierarchical Clusters

Hierarchical clustering is divided into two approaches: bottom-up (agglomerative clustering) and top-down (divisive clustering), page 893 of [7]. Both methods input the model with a dissimilarity matrix **D** between objects. For instance, for objects i and j, a distance metrics include $d_{i,i}$ and $d_{i,j} \geq 0$. To compute the attributes for each object [7]:

$$\Delta(x_i, x_{i'}) = \sum_{j=1}^{D} \Delta_j \left(x_{ij}, x_{i'j}\right) \tag{2.6}$$

Another common function is to use squared Euclidean distance:

$$\Delta_j(x_i, x_{i'}) = (x_{ij} - x_{i'j})^2 \tag{2.7}$$

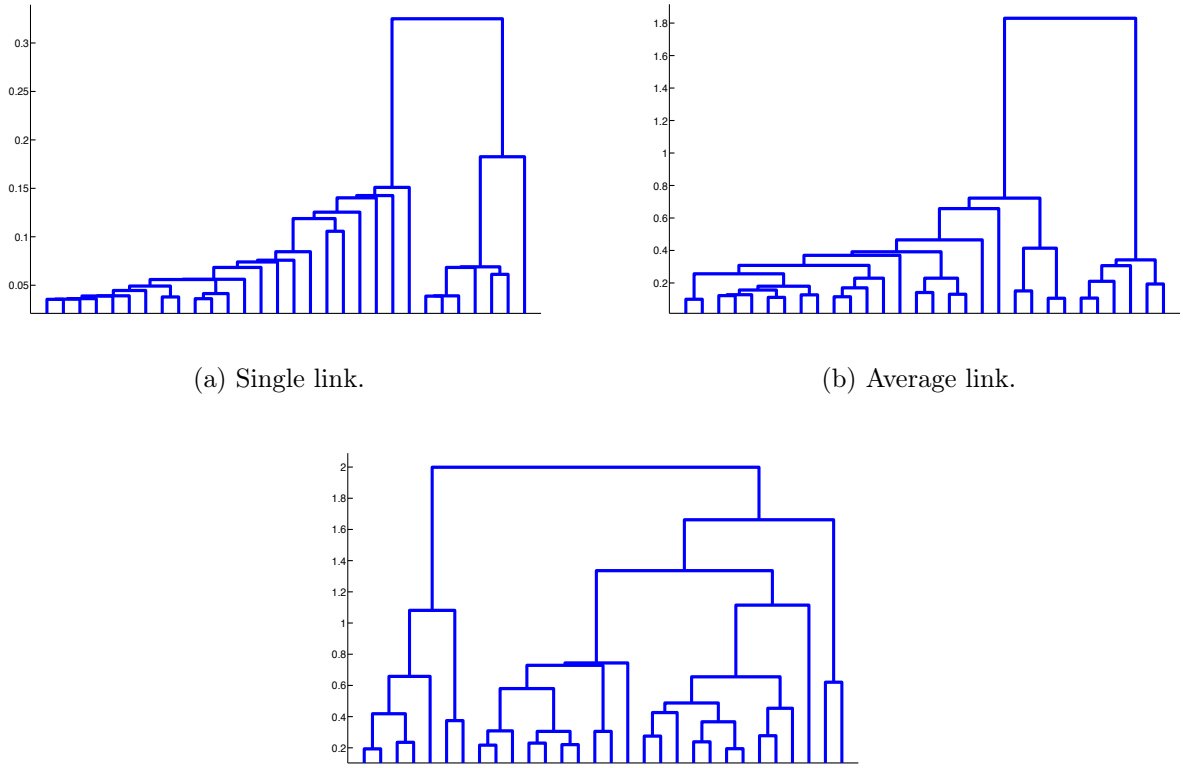
For categorical variables, we set 1 as a distance if the feature is different and 0 otherwise. Then we use hamming distance all over features to calculate the dissimilarity score [7] such that:

$$\Delta(x_i, x_i) = \sum_{j=1}^{D} I \left(x_{ij} \neq x_{i'j}\right) \tag{2.8}$$

2.5.2.3 Agglomerative hierarchical clustering

Agglomerative hierarchical clustering (AHC) is a bottom-up approach where two connections are measured in terms of their similarity. The similarity is quantified using the Euclidean distance between every pair of objects. Therefore, each object will be measured against all others independently. As a result, AHC will produce $n * (n-1)/2$ leaf nodes.

Next, each pair is grouped into one cluster according to their close proximity. This process, called linkage, uses the similarity distance in the previous step as an input to decide which pair should be grouped into one cluster. Then, each pair of nodes is recursively used in the linkage process to be merged into the larger cluster until we have one large cluster at the head of the binary tree, called a dendrogram.



(a) Single link.

(b) Average link.

(c) Complete link.

Figure 2.6: Dendogram produced by AHC showing three linkage methods [7].

As shown in Figure 2.6, there are three linkage methods. Figure 2.6.a shows a single link where the algorithm search for the nearest neighbour. For instance, if we have two groups, G and H, the distance can be defined as follows [7]:

$$d_S(G, H) = \min_{i \in G, i' \in H} d_{i,i'} \tag{2.9}$$

Figure 2.6.b shows the complete link method. This method searches for the furthest neighbour [7] such that:

$$d_C(G, H) = \max_{i \in G, i' \in H} d_{i,i'} \tag{2.10}$$

Finally, the third method is to take the average, which compromises between the single and

complete link methods [7] defined as follows:

$$d_C(G, H) = \frac{1}{n_G n_H} \sum_{i \in G} \sum_{i' \in H} d_{i,i'} \tag{2.11}$$

where n_G and n_H refer to the number of objects in groups G and H, respectively.

2.5.3 Evaluation Metrics

Several commonly used evaluation metrics are used in this thesis to determine whether the detection models are effective during the experimental sections. Since our problems are classification, we consider macro and weighted precision, recall, and F-measure in addition to accuracy and FPR.

As discussed in Section 2.4.3, reporting these metrics is crucial for an unbalanced dataset, which is common in intrusion detection, to meet the practical standard used for SOC analysts.

We describe and define the performance metrics as follows:

- **True positive** (TP)

  The outcome of a sample where the detection model correctly predicts the malicious sample as malicious.

- **False positive** (FP)

  The outcome of a sample where the detection model falsely predicts the legitimate sample as malicious.

- **True negative** (TN)

  The outcome of a sample where the detection model correctly predicts the legitimate sample as legitimate.

- **False negative** (FN)

  The outcome of a sample where the detection model falsely predicts the malicious sample as legitimate.

- **Accuracy** (A)

  The total number of correctly predicted samples, either malicious or legitimate.

$$A = \frac{\sum TP + \sum TN}{\sum TP + \sum FP + \sum TN + \sum FN} \tag{2.12}$$

- **Precision** (P)

  Measures the ability of the detection model to present relevant predictions. The higher the score, the higher its ability to present relevant predictions. In this thesis, the relevant prediction predicts if a malicious sample is a malicious sample.

$$P = \frac{\sum TP}{\sum TP + \sum FP} \tag{2.13}$$

- **Recall** (R)

  The rate of correctly predicted samples is malicious. Since the objective of the thesis is to detect malicious samples, the objective of the evaluation is also to focus on malicious samples.

$$R = \frac{\sum TP}{\sum TP + \sum FN} \tag{2.14}$$

- **F1-measure** (F)

  The test for accuracy considers both recall and precision when calculating the score. It calculates the harmonic average of precision and recall. In the presence of unbalanced datasets, the F1-measure helps to determine how precise and robust the detection model is in predicting the samples correctly while reducing false predictions.

$$F = \frac{2 \times P \times R}{P + R} \tag{2.15}$$

  In this thesis, we apply the above evaluation metrics in two settings, weighted and macro. Weighted metrics calculate the score for each class and take the average weighted (W_i) proportionally to the number of samples. For macro metrics, it finds the score also for each class, but it takes their unweighted mean ($\frac{1}{n}$). Both settings are important to report

for NIDS, especially the macro one, which can reflect the performance on unbalanced datasets. We summarise the two settings of evaluation metrics as follows:

- **Weighted Precision** (wP)

$$wP = P_1 \times W_1 + ... + P_n \times W_n = \sum_{i=1}^{n} P_i \times W_i \tag{2.16}$$

- **Weighted Recall** (wR)

$$wR = R_1 \times W_1 + ... + R_n \times W_n = \sum_{i=1}^{n} R_i \times W_i \tag{2.17}$$

- **Weighted F-measure** (wF)

$$wF = F_1 \times W_1 + ... + F_n \times W_n = \sum_{i=1}^{n} F_i \times W_i \tag{2.18}$$

- **Macro Precision** (mP)

$$mP = \frac{1}{n} \sum_{i=1}^{n} P_i \tag{2.19}$$

- **Macro Recall** (mR)

$$mR = \frac{1}{n} \sum_{i=1}^{n} R_i \tag{2.20}$$

- **Macro F1-measure** (mF)

$$mF = \frac{1}{n} \sum_{i=1}^{n} F_i \tag{2.21}$$

2.5.4 Limitation of Deep Learning for APT

Deep learning models are, in general, parametric models which rely on a large number of parameters. To learn the estimation of these parameters, a deep learning model requires a large

number of examples, e.g. millions, to achieve such a task. Nowadays, these kinds of models show success in many domains, including image recognition, text generation and speech recognition. It also successfully predicts security events [83] and investigates host-based sequence attacks [84]. However, deep learning would not be the best option in the context of network-based APT detection, where a campaign may use different techniques for the domain or their traffic with very few real examples. In addition, deep learning struggles to explain how it detects malicious behaviour, which is important for SOC engineers to investigate further and improve their defence systems. Therefore, random forest remains popular to be used in NIDS [19, 75, 77, 78, 61, 79].

2.6 Digital Signal Processing

Data interpretation is topic-dependent. Useful information can be extracted by considering in which domain data are encoded in the first place. For example, financial data are recorded over time (time-domain), images are taken and stored in pixels that refer to the spatial domain, and audio is captured in recording frequency like human hearing ability. One of the best practices for using Digital Signal Processing (DSP) is to consider the encoded data and provide interpretation based on that particular domain, not by others. For example, looking at the frequency domain of financial data does not provide explainable information. However, there is another use of the frequency domain for this case. Expensive operations such as convolution in the time domain may be replaced with multiplication in the frequency domain, which utilizes faster and simpler multiplication rather than the expensive convolution [85].

Network traffic is recorded, encoded, and explained in the time domain. We will use the frequency domain for the expensive operation. Another important aspect is to remove noise. Third, the matched filter can be achieved, which is known to be successful in catching a specific signal, as we will discuss in Section 2.6.2. However, our explainable features and signatures for malicious techniques are mostly based on the time domain.

For the above reason, to transform from the time domain to the frequency domain, the Fourier

transform is known to be the most efficient method. The method also provides an inverse function to go back from the frequency to the time domain. However, if the undersampling is applied to the time domain, the frequency domain produced by the Fourier transform suffers from aliasing, meaning a higher frequency wraps around to the lower frequency. Therefore, the output of Fourier will generate a distorted time domain traffic when the inverse Fourier transform is applied. Since the information of network traffic is encoded in the time domain, a verification of the reversible operation of the Fourier transform and its inverse should be verified before further analysis.

In this section, we explain the Fourier transform, the proper sampling rate selection, digital filters, and the wavelet transform. Next, wavelet is an advanced technique that considers both time and frequency domains for the presentation. We will use the wavelet transform as the main framework to generate the waveform-based signatures for several operations in egress connections (i.e. from an internal host to a remote host). This is important to provide an efficient privacy-preserving method to distinguish legitimate communication from malicious connections.

2.6.1 Fourier Transform

Discrete Fourier Transform (DFT) is a mathematical technique that converts an input signal in the time domain $x[i]$ to the frequency domain $X[i]$. The DFT is an intermediate step to extract key information, including the frequency, phase, and amplitude of the component sinusoids [85]. The sinusoids are used for representation purposes because of their flexibility to identify the amplitude of the frequency bins and perform mathematical operations such as multiplication with a low-pass filter to achieve time-domain decomposition later. DFT is described by the following formula:

$$\begin{aligned} X[k] &= \sum_{i=0}^{N-1} x[i] e^{-j\omega_k i} \\ &= \sum_{i=0}^{N-1} x[i] \left[\cos\left(\frac{2\pi k i}{N}\right) - j \sin\left(\frac{2\pi k i}{N}\right) \right] \end{aligned} \tag{2.22}$$

When x has a length of N points, DFT decomposes it into real X and imaginary X with $N/2+1$ cosine and sine waves for each, respectively. From Equation 2.22, we can extract the basis functions of the real and imaginary X as follows:

$$X_R[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi ki/N) \tag{2.23}$$

$$X_J[k] = -\sum_{i=0}^{N-1} x[i] \sin(2\pi ki/N) \tag{2.24}$$

The length for each sinusoid wave is similar to the original signal, which is N points. X_R represents the cosine waves and X_J represents the sine waves as given by Equations 2.23 and 2.24. The frequency and wave shape of X_R and X_J are fixed, while the amplitude and phase can change. The amplitudes of $N+2$ sinusoids will be added to form the frequency domain with a length of N points. Therefore, the final waveform in the frequency domain X will be scaled amplitudes of the sine and cosine waves of the original signal x. The default DFT outputs frequency spectrum in rectangular notation, which is difficult to interpret and find useful information. Therefore, we need to convert it to polar notation, which transforms X_R and X_J into magnitude X_M and phase X_P. If we consider X_R and X_J as two vectors, by applying simple properties of the addition of two vectors, we can have a third vector X_M, which has a phase X_P as follows:

$$\boxed{X_M[k] = \sqrt{X_R[k]^2 + X_J[k]^2}} \tag{2.25}$$

$$\boxed{X_P[k] = arctan\left(\frac{X_J[k]}{X_R[k]}\right)} \tag{2.26}$$

2.6.2 Digital Filters

A Digital filter is a key tool to separate and restore the original signal in DSP. In Chapter 6, we need to use filters to separate the evasive signal from the malicious traffic and to identify

network events based on those signatures. A digital filter can be described through its impulse, step, and frequency responses. The impulse response is the output of a system when the input is an impulse signal [85]. Similarly, step response is the output of a system when the input is a step signal. Step response can be calculated by finding the integration (continuous signals), or running sum (discrete signals) of the impulse response since the step signal is just an integral of the impulse signal. Finally, the frequency response is the DFT of the impulse response.

To design a digital filter, an input signal can be convolved with the digital filter's impulse response which is called the filter kernel. This type is called Finite Impulse Response (FIR). Many filters fall in FIR, including moving average and windowed-sinc. Another way to implement a digital filter is by recursion. Each output sample is calculated by the sum of weighted input samples and the current output sample. As a result, the impulse response of the recursive filter is infinitely long. This filter category is called Infinite Impulse Response (IIR). Examples of IIR filters are Single pole and Chebyshev. Digital filters can be used in time or frequency domains. When the information is encoded in the time domain, moving average and single pole can be used. If the information is encoded in the frequency domain, windowed-sinc and Chebyshev are the options [85].

Filters may have one of the four standard frequency responses based on the bands' selection to pass and block. Low-pass filters mean the passband belongs to the lower frequency components, whereas the stopband starts from the higher ones. The high-pass filters are the opposite operation. The addition of low and high-pass filters produces band-reject, while the convolution of the two filters results in band-pass. Band-reject and band-pass specify a range of frequencies to either be blocked or to pass. To design these types, we should have low and high-pass filters. It should be started with a low-pass filter and then transformed to a high-pass through spectral inversion [85].

One of the common applications for digital filters is to use a matched filter. In Figure 2.7, an example of a matched filter is applied to identify if there is a square component in the input.

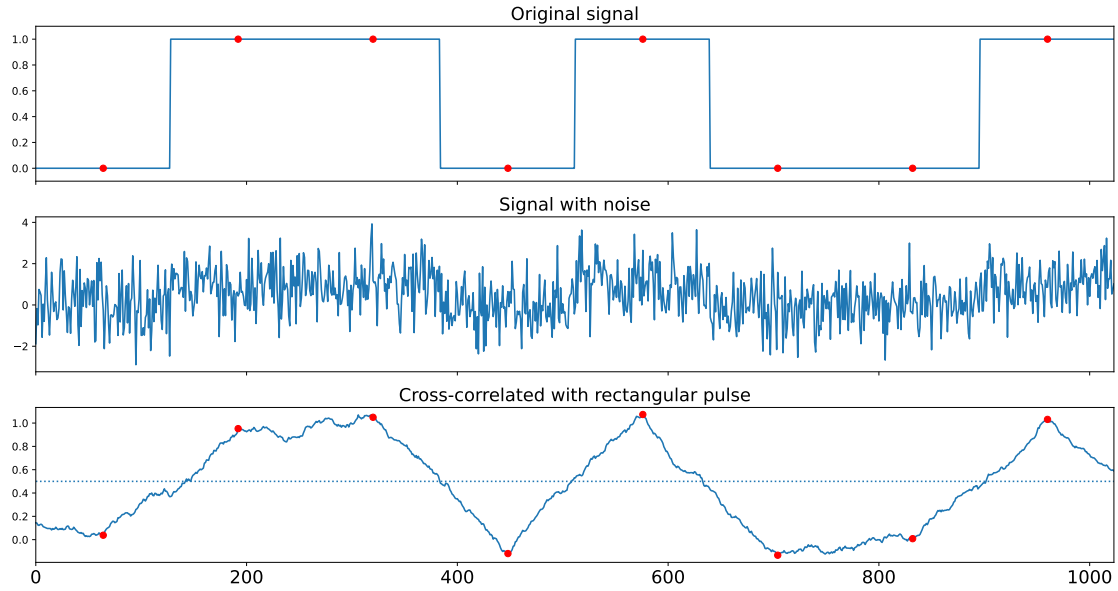


Figure 2.7: An example of matched filter using cross-correlation [8].

2.6.3 Discrete Wavelet Transform

We discussed previously the Fourier Transform limitation when the data is encoded in the time domain. Wavelet transform was proposed to consider the time and frequency domain at the same time. It is known to be used for many applications, including financial data, which is close to the network data as a time-series problem. In this section, we define Wavelet Transform and its constituent component. The mathematics symbols in this section follow P. Addison [9].

2.6.3.1 Wavelet Functions

The Wavelet transform is an approach that converts an input signal into its constituent components while considering its time and frequency features [9]. The output will be a group of waves that represent the input signal at different scales or levels. To generate these waves, a wavelet function $\Psi(t)$ convolves with the input signal at different scales. There are multiple wavelet functions that can be used, such as Mexican hat, Daubechies, Haar and Symlets. For now, let us consider the Mexican hat to explain the concept. The Mexican hat is a second derivative of the Gaussian wave depicted in Figure 2.8. Two parameters are needed, which are the dilation a and translation or the location b.

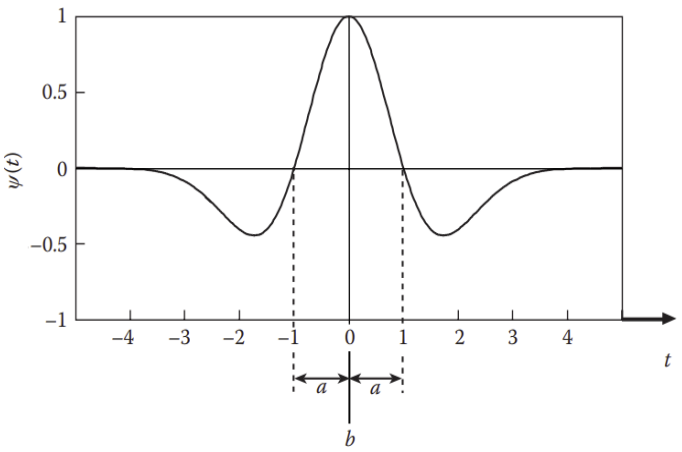


Figure 2.8: Mexican hat wavelet [9].

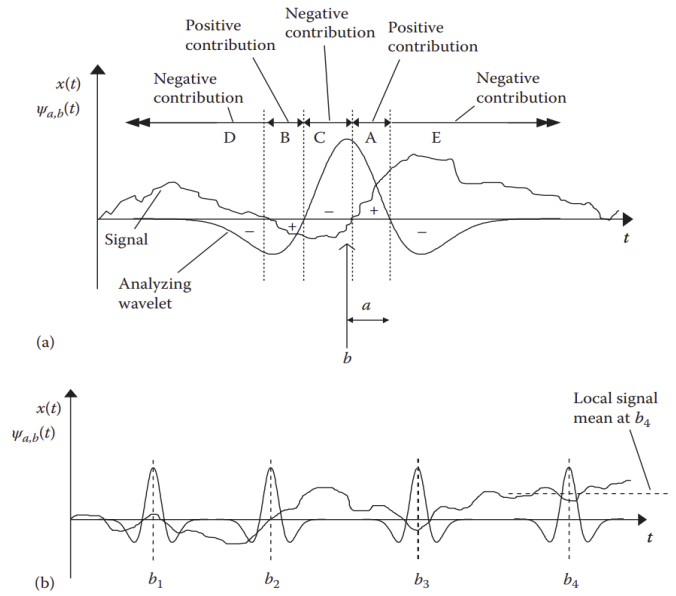


Figure 2.9: An example of wavelet dilation and translation on the signal [9].

When we select the a value, we calculate the convolution of the wavelet function and the input signal at location b. Figure 2.9 illustrates the positive and negative contribution of a signal when compared with the Mexican hat wavelet. These contributions will be the output for the same location. Then, we move to location $b+1$ and iterate the same process. One condition is important when selecting b. The locations should not be overlapped, which means each wavelet must be orthogonal to the other one. For the discrete wavelet transform, a discrete dyadic grid is commonly chosen for its orthogonality property. Therefore, a and b are chosen proportionally, producing m for the scale and n for the location. The wavelet function is defined as follows:

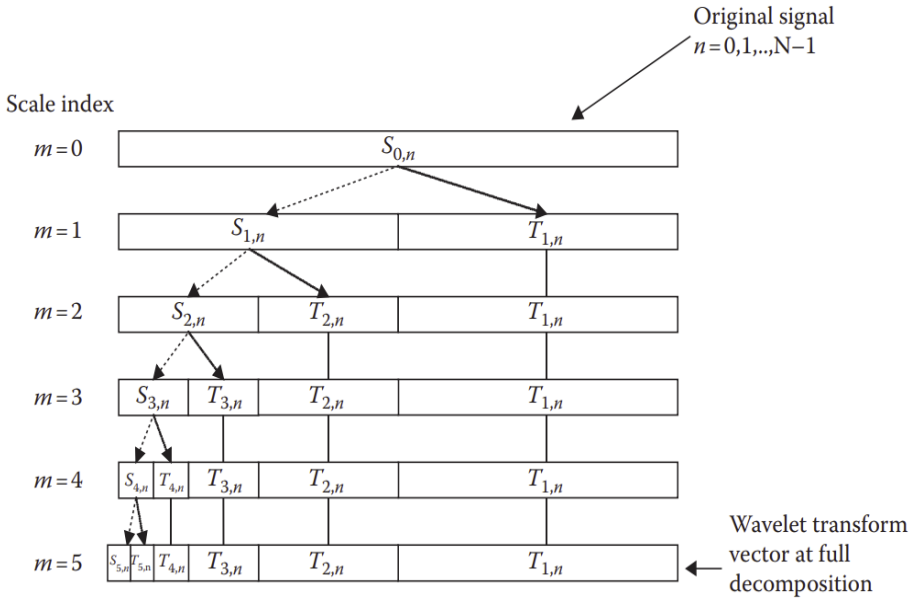


Figure 2.10: An example of discrete wavelet transform multiresolution algorithm [9].

$$\Psi_{m,n}(t) = \frac{1}{\sqrt{a_0^m}} \Psi\left(\frac{t - nb_0a_0^m}{a_0^m}\right) \tag{2.27}$$

For the discrete dyadic grid, typical values for a and b are 2 and 1, hence we will have:

$$\Psi_{m,n}(t) = 2^{-m/2}\Psi(2^{-m}t - n) \tag{2.28}$$

Now, the output for each scale involves two signals: the first one is the *approximation coefficients* $S_{m,n}$ (low pass) and the second one is the *wavelet coefficients* $T_{m,n}$ (high pass).

2.6.3.2 Multiresolution and Reconstruction Algorithms

In this section, we discuss the details of generating $S_{m,n}$ and $T_{m,n}$ in the top-down and bottom-up approach, where the former is called multiresolution and the latter is the reconstruction algorithms [9]. Figure 2.10 visualizes the process of the multiresolution algorithm that we aim to formulate in this section.

Let us define the scaling function $\phi(t)$ in the same way as wavelet function in Eq. 2.28:

$$\phi_{m,n}(t) = 2^{-m/2}\phi(2^{-m}t - n) \tag{2.29}$$

By involving the scaling function with the input, we can generate the approximation coefficients:

$$S_{min} = \int_{-\infty}^{+\infty} x(t)\phi_{min}(t)\, dt \tag{2.30}$$

Now, we can obtain a continuous approximation of the signal $x_m(t)$ at scale m by the approximation coefficients factored by the scaling functions:

$$x_m(t) = \sum_{n=-\infty}^{+\infty} S_{m,n}\phi_{m,n}(t) \tag{2.31}$$

We can reconstruct the original signal $x(t)$ by combining the approximation and wavelet coefficients as:

$$x(t) = \sum_{n=-\infty}^{+\infty} S_{m,n}\phi_{m,n}(t) + \sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty} T_{m,n}\Psi_{m,n}(t) \tag{2.32}$$

We consider the right part as a signal detail $d_m(t)$ at scale m:

$$d_m(t) = \sum_{n=-\infty}^{+\infty} T_{m,n}\Psi_{m,n}(t) \tag{2.33}$$

Next, we plug eq. 2.31 and 2.33 in eq. 2.32:

$$x(t) = x_{m_o}(t) + \sum_{m=-\infty}^{+\infty} d_m(t) \tag{2.34}$$

So, the multiresolution representation can be given as follows:

$$x_{m-1}(t) = x_m(t) + d_m(t) \tag{2.35}$$

Now, we can build a scaling function $\phi(t)$ at one scale from a number of scaling equations at

the previous scale as follows:

$$\phi(t) = \sum_k c_k \overbrace{\phi(2t-k)}^{Contracted\ version\ of\ \phi(t)} \tag{2.36}$$

where c_k is the scaling coefficient and k is the shifted integer step. Eq. 2.36 is called a *scaling or dilation equation.*

Wavelet function with compact support

$$\Psi(t) = \sum_k (-1)^k c_{N_k-1-k}\, \phi(2t-k) \tag{2.37}$$

We consider the first part of eq. 2.37 as *reconfigured scaling coefficient* (b_k) as follows:

$$b_k = (-1)^k c_{N_k-1-k} \tag{2.38}$$

Where the sum of b_k is zero. Now, eq. 2.37 can be rewritten as:

$$\Psi(t) = \sum_{k=0}^{N_k} b_k \phi(2t-k) \tag{2.39}$$

From eq. 2.29 and 2.36, we can examine the wavelet at scale index $m+1$, the following is true:

$$\overbrace{2^{-(m+1)/2}\phi\Big(\frac{t}{2^{m+1}-n}\Big)}^{Eq.\ 2.29 with\ m+1} = \overbrace{2^{-m/2}\, 2^{-1/2} \sum_k c_k \phi\Big(\frac{2t}{2\times 2^m} - 2n - k\Big)}^{Eq.\ 2.36 with\ m+1} \tag{2.40}$$

So we can write eq. 2.40 more compactly:

$$\phi_{m+1,n}(t) = \frac{1}{\sqrt{2}} \sum_k c_k\, \phi_{m,2n+k}(t) \tag{2.41}$$

So, the wavelet function with reconfigured scaling coefficient (b_k) can be obtained as follows:

$$\Psi_{m+1,n}(t) = \frac{1}{\sqrt{2}} \sum_k b_k \, \phi_{m,2n+k}(t) \tag{2.42}$$

From eq. 2.30, approximation coefficients at scale index $m+1$ are given by:

$$S_{m+1,n} = \int_{-\infty}^{\infty} x(t) \, \phi_{m+1,n}(t) dt \tag{2.43}$$

By substitute $\phi_{m+1,n}(t)$ with eq. 2.41:

$$S_{m+1,n} = \int_{-\infty}^{\infty} x(t) \overbrace{\left(\frac{1}{\sqrt{2}} \sum_k c_k \, \phi_{m,2n+k}(t) \right)}^{Eq. 2.41} dt \tag{2.44}$$

We can rewrite this as follows:

$$S_{m+1,n} = \frac{1}{\sqrt{2}} \sum_k c_k \underbrace{\left(\int_{-\infty}^{\infty} x(t) \, \phi_{m,2n+k}(t) \right)}_{Sm,\, 2n+k} dt \tag{2.45}$$

Now, we can construct the *multiresolution decomposition algorithm* for low pass (eq. 2.46)and high pass (eq. 2.47) filters as follows:

$$\boxed{S_{m+1,n} = \frac{1}{\sqrt{2}} \sum_k c_k \, S_{m,\, 2n+k} \; = \; \frac{1}{\sqrt{2}} \sum_k c_{k-2n} \, S_{m,k}} \tag{2.46}$$

$$\boxed{T_{m+1,n} = \frac{1}{\sqrt{2}} \sum_k b_k \, S_{m,\, 2n+k} \; = \; \frac{1}{\sqrt{2}} \sum_k b_{k-2n} \, S_{m,k}} \tag{2.47}$$

If we are at a specific scale and we have $S_{m,n}$ and $T_{m,n}$, we can go back to the previous one $S_{m-1,n}$. This is called the *reconstruction algorithm.* Recall eq. 2.35, we substitute the smoothing function $x(t)$ with eq. 2.31 and signal details $d_m(t)$ at scale m with eq. 2.33. We obtain the following equations:

$$S_{m-1,n} = \sum_n S_{m,n}\phi_{m,n}(t) + \sum_n T_{m,n}\Psi_{m,n}(t) \tag{2.48}$$

$$S_{m-1,n} = \overbrace{\frac{1}{\sqrt{2}}\sum_k c_k\ \phi_{m,2n+k}(t)}^{Eq.2.41} + \overbrace{\frac{1}{\sqrt{2}}\sum_k b_k\ \phi_{m,2n+k}(t)}^{Eq.2.42} \tag{2.49}$$

2.7 Conclusion

In this chapter, we have discussed how APT campaigns have stayed undetected for several months. We shed light on the excessive usage of FQDNs to locate C&Cwith HTTP(S) protocol. Then, we explore three APT models, including Cyber Kill Chain, Mandiant and ATT&CK models. We identify the need to develop a new threat model focus from the network perspective. Then, we discuss the related works for APT detection that focus on NIDS. The current approaches require improvement in methodology, datasets and evaluation metrics. Our proposed NIDS in this thesis is based on random forests for classification and other models for feature extraction. Therefore, we explain the algorithm of tree-based and clustering-related models. Following that, evaluation metrics used in this thesis are defined. Finally, we define and discuss DSP techniques, including Discrete Fourier and Wavelet transforms, which are used for feature extraction in Chapter 6.

Chapter 3

Investigation of APT Network-based Tactics, Techniques and Procedures

In the previous chapter, we explored the APT lifecycle and defined each stage. We also discussed recent research related to NIDS against APT, botnets and other malware families at the C&C stage. Throughout our discussion, we highlighted the importance of identifying the requirements to detect APTs, which lacks in the literature. To create an effective APT detection strategy, it is important to examine the Tactics, Techniques, and Procedures (TTPs) that have been reported by the industry. These TTPs can be difficult to classify as either malicious or legitimate. Therefore, when developing an approach for the next chapters, it is necessary to take into account the specific context of the attack explained in this chapter.

For example, we identify that every campaign relies heavily on the APT domain to resolve the IP address to connect to C&C.An adversary might choose to use a host located in the same country as the targeted enterprise, exploit the brand name associated with the enterprise, and frequently change the IP address to avoid detection. Consequently, the characteristics, lexical usage, and infrastructure-related features of APTs should be interconnected and relevant to the targeted enterprise. For this reason, we discuss here different usage of domain-based TTPs in this chapter and design HawkEye in Chapter 4, based on our study here to defend against such malicious behaviour.

In the case of APT traffic, the TTPs can result in a significant overlap in statistical features between APTs and legitimate connections. This highlights the importance of proposing features that are based on the contextual information of a connection. Based on our investigation in this chapter, we design two systems (EarlyCrow and NightVision) to defend against APT traffic in Chapters 5 and 6, respectively.

To study the malicious TTPs accurately, we rely on technical reports from the industry to profile APT campaigns, with a particular emphasis on the TTPs for each campaign. We review 118 reports released by the industry since 2001. We select 33 APT campaigns based on a fair distribution over the past 22 years to observe the evolution of APT over time. We describe many TTPs relevant to network security. Some TTPs are not detailed in the industry reports from the traffic behaviour perspective. Therefore, we resort to our datasets to provide details on their behaviour and how they evade network defences. For convenience, we keep the description of the datasets in their respective chapters on page 136, page 184, and page 222. We also focus on the usage of TTPs and identify their evasion techniques, protocols, payloads, obfuscation and channels. Lastly, we report the most prevalent network-based TTPs employed by APTs. We organise these 13 TTPs according to their techniques involving DNS, traffic, and communication channels. We provide specific examples based on our own datasets, as well as additional references from the industry.

Based on our findings, we determine that conventional malware or multi-stage attacks are typically carried out by individuals or small teams. In contrast, APTs are orchestrated by a collective of exceptionally skilled individuals and are often financially backed by governments. In our examination, we observe that all APT campaigns persist in utilizing HTTP in order to avoid detection by NIDS. This highlights the importance of detecting malicious connections that employ the HTTP protocol. However, it should be noted that not every single connection will necessarily employ HTTP, but rather every APT campaign will utilize HTTP at some point during their attack lifecycle.

We also find that 81% of APT campaigns depend on HTTPS to bypass NIDS that rely on HTTP in plaintext features. In Chapter 5, we will design our approach to identifying HTTP(S)

connections. Since the HTTP protocol typically requires DNS resolution, 45% of campaigns employ DNS protocols, while the remainder use HTTP with a preconfigured IP address. Additionally, 24.2% of these campaigns distribute their malware through spearphishing links, which necessitates the consideration of detecting malicious URLs. In Chapter 4, we introduce another line of defence to detect APT campaigns from the start using DNS data. Other protocols are also can be used, such as SMTP, P2P, SOCKS5, SMB, and FTP, with percentages of 24%, 21.2%, 18.1%, 18.1%, and 18%, respectively. These protocols require a generic approach to defend malicious traffic, which we will introduce in Chapter 6.

3.1 Analysis of APT Campaigns

Several researchers argue whether APT is a special case of a multi-stage attack or another malware family. According to Khattak et al. [86], Aurora was a specialized botnet with a cyber espionage objective against Google. Zhao et al. [5] define one primary difference that makes APT malware different from Botnet. While the former's purpose is cyber espionage or data exfiltration, the latter is destruction, such as denial of service attacks. On the other hand, Vormayr et al. [87] describe Blackenergy, Duqu 2.0 and Regin as Botnet and APT alongside Conficker and Phatbot with a minor difference in whether the attack is targeted or not.

Ussath et al. [88] confirm that APT is a campaign and present a survey of APT campaigns based on 22 reports in terms of initial compromise methods, their lateral movement techniques and C&C protocols used to transmit the C&C payloads. However, details on C&C protocols and other settings are not available, as we will discuss in this chapter. Authors in [42] provide an analysis of 4 campaigns with respect to each stage of the Cyber Kill Chain model, without focusing on persistent and evasion properties, a wide range of protocols and a variety of channels. This leads us to limit our analysis to the network level of an APT campaign to illustrate how persistent, evasive and stealthy APT traffic is, and how we can design a robust network-based intrusion detection accordingly.

Last but not least, the primary source of information comes from industry, due to the relative

monopoly on information related to APT campaigns. According to Lemay et al., [89], there is no alternative to industrial resources on this topic because targeted organisations request forensics services from industrial labs, who might publish this confidential information upon the client's permission.

3.2 APT Campaigns From Network Perspective

For the above reasons, we conduct a survey of 33 APT campaigns presented in Tables 3.1 and 3.2, and we provide an analysis of these campaigns from the network perspective. First, we focus on the variant names for the same campaign, followed by their points of entry and the delivery methods. Next, we explore some evasion techniques for most campaigns. Then, we describe what payloads are exchanged between the victims and C&C server and how they are obfuscated. All these characteristics are provided based on IoC, public reports and evidence materials. Sources include, among others, Bitdefender, ESET, Trend Micro, Sophos, McAfee, F-Secure Lab, Symantec, FireEye Mandiant, Kaspersky Lab, Checkpoint and evidence from the Department of Homeland Security in the USA.

Table 3.1: APT Campaigns From Network Perspective - Part I.

APT Campaign	Delivery Method	Vulnerability	Backdoors
APT 1, a.k.a Comment Crew	Spearphishing attachment [3].	N/A	Standard Backdoors: Poison Ivy, Gh0st Beachhead Backdoors: Beachhead Family i.e. WEBC2 (>16 variants) e.g. WEBC2-QBP AURIGA, BANGAT, BISCUIT, BOUNCER, CALENDAR, COMBOS, COOKIEBAG, DAIRY, GLOOXMAIL, GOGGLES, GREENCAT, HACKSFASE, HELAUTO, KURTON, LONGRUN, MACROMAIL, MANITSME, MINIASP, NEWSREELS, SEASALT, STARSYPOUND and SWORD [3].
APT 2, a.k.a Putter Panda	Spearphishing attachment abc.scr Dropper To install 4H RAT [90].	N/A	3PARA RAT, 4H RAT, httpclient, pngdowner [90].
APT 3, a.k.a Gothic Panda a.k.a Pirpi Buckeye	Spearphishing attachment Malicious RAR Spearphishing Link Browser Exploit [91].	For initial access: Flash SWF file CVE-2015-3113 †. For privilege escalation: CVE-2014-4113 † [92]. Others: CVE-2015-5119 † CVE-2010-3962 † CVE-2014-1776 † CVE-2014-6332 [93].	Backdoor.Pirpi (12 variants) [32], SHOTPUT, Backdoor.APT.CookieCutter and PlugX [91].
APT 10 a.k.a menuPass	Spearphishing to deliver EvilGrab or ChChes [94].	N/A	HTRAN, ZXProxy, ZXPortMap, PlugX, PoisonIvy, QuasarRAT, RedLeaves [95].
APT 12	Spearphishing Attachment [96].	MS Word CVE-2012-0158 [96].	RIPTIDE , HIGHTIDE [96], THREBYTE and WATERSPOUT.
APT 15, a.k.a Ke3chang Vixen Panda	Spearphishing attachment/URL [97]. IE injection technique used by HTTP-based, Backdoors (BS2005 Malware) [97].	MS Word CVE-2010-3333, Adobe PDF Reader CVE-2010-2883 [97].	RoyalCli, BS2005 and RoyalDNS [98].
APT 16, a.k.a EPS	Spearphishing Attachment MS Word [99] and URL (DOORJAMB.Tools, IRONHALO or ELMER [100].	MS Word: CVE-2015-2545 †, Windows: CVE-2015-2546 † [100], Windows privilege escalation CVE-2015-1701 [99].	IRONHALO ELMER with two variants [99].
APT 17 a.k.a Deputy Dog	Spearphishing URL or Social Networks (Cleaver)[101].	N/A	BLACKCOFFEE [101].
APT 18, a.k.a Wekby, Dynamite Panda	Spearphising Xyligan RAT [33].	N/A	Xyligan.rat, gh0st.rat, HcdLoader.rat, Pisloader [34, 33].
APT 19 a.k.a. Codoso	Waterhole [102] Spearphishing Rich Text Format (RTF) macro-enabled MS Excel [103].	Windows CVE 2017-0199 [103].	N/A
APT 27 a.k.a Emissary Panda, Threat Group-3390, LuckyMouse	Spearphishing attachment (ZIP archive) [104].	CVE-2014-6324 [105], CVE-2017-11882 [106]. For privilege escalation: Jave SWCs CVE-2011-3544, JBoss CVE-2010-0738 [104].	HyperBro a.k.a Backdoor.Win32.HyperBro (Trojan.backdoor.rat), Trojan.Win32.Generic [106], PlugX [104], (Trojan.backdoor.rat), HttpBrowser (Trojan.backdoor.rat) [106].
APT 28 a.k.a Sednit Sofacy Fancy Bear	Spearphishing URL-shortener HIDEDRV rootkit with Downdelph backdoor [107, 27].	Java CVE-2015-2590 †, Flash CVE-2015-3043 †, CVE-2015-5119 †, CVE-2015-7645, Word CVE-2015-1641 †, CVE-2015-2424 † [108], IE CVE-2014-4076 [107]. For privilege escalation: CVE-2015-2387 †, CVE-2015-1701 † [108].	Downdelph, CHOPSTICK, CORESHELL, Komplex, Zebrocy, JHUHUGIT and Sofacy [109, 110].

Continued on the next page

Table 3.1 – (Continued)

APT Campaign	Delivery Method	Vulnerability	Backdoors
APT 29 a.k.a CozyBear CozyDuke	Spearphishing attachment and link [111].	PDF Acrobat: CVE-2013-2729, MS Office: CVE-2009-3129, CVE-2015-2424, CVE-2015-1641, CVE-2010-3333, CVE-2014-1761, CVE-2012-0158, Adobe Flash CVE-2016-4117 CVE-2016-7855 and for Privilage Escalation: CVE-2016-7255 [111].	CosmicDuke, CloudDuke, HammerDuke, SeaDuke, SeaDaddy, PinchDuke, OnionDuke, MiniDuke, HAMMERTOSS [30] and Cobalt Strike [112].
APT 30	Spearphishing attachment to drop BACKSPACE [113].	N/A	FLASHFLOOD, NETEAGLE, SHIPSHAPE, BACKSPACE SPACESHIP [113].
APT 32, a.k.a OceanLotus	Spearphishing attachment (RTF Document) [114].	CVE-2017-11882. For privilege escalation: CVE-2016-7255 [114]	Cobalt Strike, Denis [29], KOMPROGO, OSX OCEANLOTUS.D, PHOREAL, SOUNDBITE, WINDSHIELD [115].
APT 33, a.k.a Elfin Shamoon	Spearphishing Link [116].	WinRAR CVE-2018-20250 [117]. For privilege escalation: CVE-2017-0213 [118].	Shamoon, TURNEDUP, AutoIt [117] and POWERTON [118].
APT 34, a.k.a OilReg	Spearphishing attachment [119].	MS office: CVE-2017-11882, Malicious RTF CVE-2017-0199 [119].	ISMAgent malware Plink utility to create tunnels to C&C server [120]
APT 35, a.k.a Majic Hound	Spearphishing attachment and link to deliver Pupy RAT or MagicHound.Rollover [121].	N/A	IRC bot (MagicHound.Leash) MPK.trojan.rat[121].
APT 37, a.k.a ScarCruft	Spearphishing attachment and link Torrent (KARAE) [35].	Adobe Flash: CVE-2016-4117, CVE-2018-4878. MS Word: CVE-2017-019 [35].	NavRAT, HAPPYWORK, Final1stspy, DOGCALL, CORALDECK, ROKRAT, SHUTTERSPEED, SLOWDRIFT, WINERACK and KARAE [35].
admin@338	Spearphishing attachment (LOWBALL malware) [122].	N/A	Bubblewarp, LOWBALL and PoisonIvy and Custom (PoisonIvy) [122].
Blockbuster a.k.a Operation Flame, Lazarus Group, HIDDEN COBRA	Spearphishing attachment [123].	N/A	HOPLIGHT, KEYMARBLE, HARDRAIN, FALLCHILL.rat, Bankshot.rat, BADCALL, WannaCry, Volgmer and Proxysvc [124].
Cobalt Group a.k.a Cobalt Spider	Spearphishing attachment [125].	CVE-2017-11882, IE: CVE-2018-8174, CVE-2017-8570, CVE-2017-0199, CVE-2017-8759 [126]. MS Word: CVE-2015-1641 , Adobe Flash: CVE-2016-4117 [125].	Odinaff, $Odinaffg1$, $Odinaffgm$, Batel, Gussdoor, Ammyy [125] and More_eggs [126].
Dragonfly 2.0	Spam campaign Waterhole attack, i.e. iFrame is used forward to another website hosting Lightsout [127].	With using Lightsout exploit kit against Java 6: CVE-2012-1723 Java 7: CVE-2013-2465 and IE: CVE-2012-4792 CVE-2013-1347 [127].	Backdoor.Oldrea, Trojan.Karagany [127].
Duqu 2.0	Spearphishing attachment [128].	MS Word with embedded TTF: CVE-2011-3402 †, CVE-2014-4148 †. For lateral movement: CVE-2014-6324 † [128].	N/A
Elderwood, a.k.a Operation Aurora, Sneaky Panda	Waterhole (IFRAME) to forward to the server that hosting exploit [129].	CVE-2012-0779 †, CVE-2012-1875 †, CVE-2012-1889 †, CVE-2012-1535 †, CVE-2011-0609 †, CVE-2011-0611 †, CVE-2011-2110 †, CVE-2010-0249 † [129].	Backdoor.Briba, Backdoor.Ritsol, Backdoor.Nerex, Backdoor.Linfo, Backdoor.Wiarp, Backdoor.Vasport, Backdoor.Darkmoon, Trojan.Hydraq (Aurora), Trojan.Naid, Trojan.Pasam, Packed.Generic.379, Packed.Generic.374 [129].
FIN 7, a.k.a Carbanak	Spearphishing attachment malicious DOCX and RTF [130].	N/A	Carbanak, POWERSOURCE, TEXTMATE and HALFBAKED [130].

Continued on the next page

Table 3.1 – (Continued)

APT Campaign	Delivery Method	Vulnerability	Backdoors
Leviathan, a.k.a APT 40	Spearphishing attachment and link [131].	N/A	AIRBREAK, FRESHAIR, BEACON, Gh0st, PHOTO (Derusbi) , BADFLICK , China Chopper, PluX (Sogu) [131], Cobalt Strike and BLACKCOFFEE [132].
Naikon APT	Spearphishing attachment to drop naikon backdoor [133].	CVE-2012-0158, CVE-2010-3333 [133].	Naikon backdoor, RARSTONE, SslMM, Sys10, xsPlus, MsnMM, Sakto [133], WinMM, WininetMM [134].
Patchwork a.k.a Dropping Elephant MONSOON	Spearphishing attachment and link Drive-by-Download i.e. Adobe Flash Update [135].	CVE-2014-4114, CVE-2012-1856, CVE-2017-0199, CVE-2017-8570, CVE-2015-1641 to deliver Badnews backdoor [135].	AutoIt, Unknown Logger [135], QuasarRAT [136], TINYTYPHON [137], Socksbot, NDiskMonitor and Badnews [135].
Sandworm Team a.k.a BlackEnergy Quedagh [138].	Spearphishing attachment [138].	CVE-2010-3333 [138].	BlackEnergy [138].
Strider a.k.a PROJECTSAURON [139].	Plugged USB sticks [139].	N/A	Remsec: Loader: MSAOSSPC.DLL Lua modules: Pip backdoor and HTTP backdoor [140].
Regin	Spearphishing attachment [141].	N/A	Backdoor.Regin [141].
Red October, a.k.a Cloud Atlas [142].	Spearphishing attachment [143].	MS Excel: CVE-2009-3129, MS Word: CVE-2010-3333, CVE-2012-0158, Rhino Java: CVE-2011-3544 [143].	LHAFD.GCP [143], Zakladka a.k.a winupdate.dll and WNFTPSCAN [144].

Table 3.2: APT Campaigns From Network Perspective - Part II.

APT Campaign	Evasion Tech.		Other C&C Settings					Note
	DNS	HTTP	Channels	Protocols	Obfuscation	Payload	# FQDNs / C&C IPs	
APT 1, a.k.a Comment Crew	Hijacked FQDNs, Dynamic DNS [3].	HTTPS, Mimicked MSN Messenger, Jabber/XMPP, Gmail Calendar [3].	C&C Over HTTP, C&C over DNS [3].	HTTP, HTTPS, FTP [3].	Base64, single-byte XOR, TLS, SSL and 3DES [3].	GDOCUPLOAD, GETMAIL, LIGHTBOLT, MAPIGET [3].	(2551)/ (849) [3].	In addtion, APT 1 establish >937 C&C servers [3].
APT 2, a.k.a Putter Panda	Dynamic DNS [145].	N/A	abc.scr, a dropper, installs 4H RAT to open the first channel to connect to C&C server. The next channel is initiated with 3PARA followed by several channels established by Httpclient and Pngdowner [90].	HTTP [90][145].	RC4, 16-byte XOR, DES (CBC) With MD5 hash key of a string in HTTP request back to C&C 1-byte XOR with 0xBE [90].	N/A	(>57)/ (>32) [90].	
APT 3, a.k.a Gothic Panda a.k.a Pirpi Buckeye	N/A	HTTP Proxy and HTTP Cookie field [91].	The main channels established by Pirpi [93].	HTTP, HTTPS, SOCKS5 and FTP [93].	N/A	Keylogger, RemoteCMD, PwDumpVariant, OSinfo, ChromePass, Lazagne [32], Customized Mimikatz, Dsquery [91].	(5)/ (2) [92].	
APT 10 a.k.a menuPass	Dynamic DNS [94].	Cookies embedding in HTTP, HTTPS [95].	Through legitimate access to many Managed IT Service Providers (MSP) [94], or embedding data in cookies field in HTTP header [95].	HTTP, HTTPS, FTP [95].	AES, RC4, MD5 and Base64-encoding [95].	MimiKatz, PwDump6 and certutil [95].	(102)/ (25)[94].	
APT 12	N/A	N/A	Over HTTP [96].	HTTP [96].	RC4 [96].	N/A	N/A	
APT 15, a.k.a Ke3chang Vixen Panda	Dynamic DNS [97] and RoyalDNS uses DNS for C&C [98].	HTTP with COM interface IWebBrowser2 [98].	Third Party DNS Service "Nwsapagent"[98].	RoyalDNS uses TXT Record in DNS protocol to send payloads [98].		Spwebmember, custom keylogger, Mimikatz, other network scanning and enumeration tools [98].	(11)/ (22)[97].	
APT 16, a.k.a EPS	N/A	HTTP Beacon and HTTPS [100].	Variants of ELMER communicate with two different C&C locations [100].	HTTP and ELMER beacons over HTTPS [100].	N/A	N/A	(2)/ (2) [100].	
APT 17 a.k.a Deputy Dog	N/A	N/A	C&C distribution over Microsoft TechNet and Social Media[101].	HTTP [101].	N/A	N/A	N/A	BLACKCOFFEE holds a URL for the actor profile or thread at Microsoft TechNet or Social media to retrieve C&C IP address which will be updated once it is exposed[101].

Continued on the next page

Table 3.2 – (Continued)

APT Campaign	Evasion Tech.		Other C&C Settings					Note
	DNS	HTTP	Channels	Protocols	Obfuscation	Payload	# FQDNs / C&C IPs	
APT 18, a.k.a Wekby, Dynamite Panda	DNS as a covert channel [34].	HTTPS [146].	HcdLoader used for lateral movement and data Exfiltration over HTTP, HTTPBrowser, and Pisloader used DNS requests as a channel in TXT records [34].	HTTP, HTTPS [146] and DNS [147].	Pisloader uses base32-encoded [147].	HTTPBrowser sends Keystrokes [33].	N/A	Xyligan establish the foothold, then install Hcdloader to provide command-line access [33].
APT 19 a.k.a. Codoso	N/A	N/A	HTTP over port 22, SCP, SFTP over port 22, HTTPS and DNS [103]	HTTP [103]	Base64, single-byte XOR keys [103].	Cobalt Strike [103].	(4)/ (4) [102].	
APT 27 a.k.a Emissary Panda, Threat Group-3390, LuckyMouse	N/A	N/A	N/A	HTTP, HTTPS and DNS [104].	Base64 Encoding, Metasploit's Shikata Ganai, encoder and LZNT1 [106].	ASPXSpy (webshell) China Chopper, OwaAuth (to steal Exchange's Passwords) and Mimikatz [148].	(5) [106]/ (2) [148].	Main C&C bbs.sonypsps[.]com with resolved IP, i.e. belong to Router, at Ukrainian ISP network, that was hacked to pass malware's HTTP request [106].
APT 28 a.k.a Sednit Sofacy Fancy Bear	DGA used by Sofacy.WinHttp [27]	Custom (CHOPSTICK) over 80/443 [107].	First Channel: Downdelph over HTTP [27]. 2nd Channel: CHOPSTICK over HTTP. 3rd Channel: CORESHELL [107]. and Zebrocy over HTTP, SMTP, and POP3. Komplex and JHUHUGIT use HTTP Post and HTTPS while XTunnel uses SSL/TLS with RC4 [110].	HTTP, HTTPS, SMTP and POP3 [107, 109, 110].	CORESHELL uses Base64 encoding [107].	N/A	N/A	APT 28 uses POP3 to communicate with GMAIL services to allocate FQDNs and C&C locations [107].
APT 29 a.k.a CozyBear CozyDuke	N/A	Domain Fronting over HTTPS places malicious domains in the HTTP header and legitimate ones in the TLS header [149].	First CozyCar over HTTP/HTTPS, Second PowerShell Script, Third (SeaDaddy) over 443 [31], Tor-meek for domain fronting and Multihop Proxy [149].	HTTP, HTTPS, RDP (3389), NetBios (139), SMB (445), FTP [30].	Base64 Encoding [111].	Backdoors for importing persistent PowerShell scripts [149] and exfiltrating data [111].	(18)/(31) [30].	CozyCar is embedded with an alternative C&C channel to Twitter, MiniDuke uses Google Search if Twitter approach [30] failed.
APT 30	N/A	NETEAGLE over HTTP proxy post request, or UDP (6000) BACKSPACE disable firewall [113].	NETEAGLE connect with proxy over HTTP post beacons. If failed, RDP over TCP (7519) [113]. BACKSPACE uses one domain to receive an update and another one for a backup with two more run-hide configurations [113].	HTTP and RDP [113].	FLASH- FLOOD uses zlib, byte-rotation, and XOR NETEAGLE uses RC4 [113].	SPACESHIP and SHIPSHAPE exfiltrate data [113].	(5>) / (N/A) [113].	SHIPSHAPE provides persistence by propagating through removable devices of the infected network against air-gap setting [113].
APT 32, a.k.a OceanLotus	DNS Tunnelling for C&C and Data Exfiltration [29, 28].	Custom (Outlook macro backdoor over SMTP [29]), (JavaScript over HTTP 14146 [114] or HTTPS [150].	Denis [29] and SOUNDBITE exfiltrate over DNS packets while PHOREAL uses ICMP and WINDSHIELD communicate over raw TCP sockets [115].	HTTP, HTTPS [150] , DNS [29] , P2P over SMB and ICMP [115].	AES-256(CBC) [114], RC4 [151], RSA256 and Base64 encoding [115].	Mimikatz and data exfiltration [29].	(62)/(22) [151].	Backdoors exploits Microsoft Outlook [29].

Continued on the next page

Table 3.2 – (Continued)

APT Campaign	Evasion Tech.		Other C&C Settings					Note
	DNS	HTTP	Channels	Protocols	Obfuscation	Payload	# FQDNs / C&C IPs	
APT 33, a.k.a Elfin Shamoon	N/A	HTTP Proxy [118].	Shamoon uses HTTP, PoshC2 may uses proxy to C&C server over HTTP/HTTPS [118] and NanoCore over 6666 [116]	HTTP, HTTPS and FTP [117].	AES, base64 encoding, DES [118].	SniffPass, DarkComet, ProcDump, Mimikatz, PoshC2 and data exfiltration [117].	(69) / (69) [117].	Shamoon is responsible for C&C as well as data destruction processes. PoshC2 has multiple functions, including C&C, search for passwords, Netstat and pass the hash to gain access without plaintext [117].
APT 34, a.k.a OilReg	DNS tunnelling [120] and DGA [119].	N/A	First 'dolt' function in the PowerShell script initiates the first channel, Then ISMAgent uses DNS tunnelling instead of HTTP [120].	HTTP, HTTPS, DNS, FTP [152].	Cryptographic Data Encoding [152].	keylogger.KEYPUNCH, screenshot.CANDYKING, Tool.Plink to create tunnels Tool.netscan.SoftPerfect, Tool.netscan.GOLDIRONY[152].	(10) / (14) [152, 153].	
APT 35, a.k.a Majic Hound	N/A	HTTPS [121].	Over 4443, 3543 [121].	HTTP, HTTPS, FTP, IRC and SOAP [154].	base64, AES[121].	Pupy variant of Mimikats, CWoolger for keylogging, FireMalv for stealing credentials of Firefox Data Exfiltration [121].	(39)/(97) [121].	
APT 37, a.k.a ScarCruft	N/A	HTTP POST headers for data exfiltration [35].	First, HAPPYWORK and KARAE connect with C&C to receive further backdoors and open further channels CORALDECK connects with C&C through HTTP POST headers DOGCALL communicates with C&C and exfiltrates through cloud services, NavRAT uses SMTP POORAIM overall AOL Instant Messenger for C&C ROKRAT uses a variety of C&C channels including HTTPS and cloud services. SLOWDRIFT is used for cloud services [35].	HTTP, HTTPS, SMTP and P2P [35].	CORALDECK uses RAR protected with password DOGCALL uses single-byte XOR Finalspy uses Base64 Encoding [35].	N/A	N/A	Not only APT 37 exfiltrate credential data but also the microphone data, snapshot of virtual machines [35].
admin@338	N/A	N/A	First, LOWBALL open a C&C channel over HTTPS and can also use Cloud services [122]. The next level for C&C traffic is performed by BUBBLEWRAP over HTTP/HTTPS or SOCKS [122].	N/A	N/A	HTTP, HTTPS and SOCKS [122].	N/A	

Continued on the next page

Table 3.2 – (Continued)

APT Campaign	Evasion Tech.		Other C&C Settings					
	DNS	HTTP	Channels	Protocols	Obfuscation	Payload	# FQDNs / C&C IPs	Note
Blockbuster a.k.a Operation Flame, Lazarus Group, HIDDEN COBRA	N/A	HTTP HTTPS (SSL) HTTPS (fake TLS) BADCALL disables Windows firewall [124].	WannaCry uses Tor for C&C communication Volgmer, TYPEFRAME, Proxysvc, KEYMARBLE, BADCALL uses HTTPS. However, TYPEFRAME, AuditCred, and BADCALL connect to a proxy server RATANKBA and Proxysvc are used for data exfiltration KEYMARBLE has multiple channels for exfiltration Bankshot uses HTTP channel for exfiltration [124].	HTTP, SMTP and RDP [124].	Symmetric stream cipher e.g. AES, RC4 and Caracachs and XOR [155].	The main payload is for data exfiltration in addition, customised password dump tools, batch scripts are uploaded [123].	(2) /(3) [156].	HARDRAIN, FALLCHILL, BADCALL use fake TLS Bankshot to create a fake TLS handshaking with the use of a public certificate [124]. 45 unique malware families during this operation [123].
Cobalt Group a.k.a Cobalt Spider	DNS Tunnelling [125].	HTTPS [125].	Cobalt Groups uses Plink to open an SSH channel. In addition, the group uses Cobalt Strike for a variety of channels, i.e. HTTP, HTTPS, DNS and VNC [157] over remote framebuffer (RBF Protocol), and it can send over one channel and received from another channel	HTTPS, DNS and P2P SMB [125].	N/A	Cobalt Strike is responsible for multiple payloads including keylogging, PowerShell scripts and data exfiltration [125].	N/A	
Dragonfly 2.0	N/A	Backdoor.Oldrea uses base64 encoded string of HTTP Get or RSA with HTTP Post [127].	First, Backdoor.Oldrea communicates with C&C server. Second, Trojan.Karagany uses a live connection to Microsoft or Adobe websites if available [127].	HTTP [127].	RSA, Base64 Encoding and XOR [127].	Trojan.Karagany is implanted to conduct internal reconnaissance, then Backdoor.Oldrea exfiltrates Credentials, Emails, OWA address book, processes, and infrastructure info. and classified documents [127] [158].	(N/A) / (13) [159]	The group extends their work after Dragonfly 1.0 [127].
Duqu 2.0	N/A	Bidirectional HTTP Proxy, embed C&C traffic inside JPEG or GIF over HTTP or inside driver files of SMB with knocking mechanism for tunnelling or fake TCP/IP packets to specific IP [128].	Duqu 2.0 uses a variety of channels, including HTTP, HTTPS and tunnelling SMB/RDB network pipes over HTTPS[128].	Outside LAN: HTTPS while inside LAN: SMB network pipes or RDP[128].	Symmetric stream cipher e.g. AES[128] and symmetric block cipher e.g. Camellia 256 and XXTEA [128].	Exfiltrate highly classified documents related to nuclear program[128].	N/A	The threat actor penetrates a certificate authority in Hungary and is able to generate legitimate certificates[128].
Elderwood, a.k.a Operation Aurora, Sneaky Panda	N/A	N/A	All backdoors connect to a shared C&C infrastructure [129].	N/A	N/A	N/A	(18) / (N/A) [129].	

Continued on the next page

Table 3.2 – (Continued)

APT Campaign	Evasion Tech.		Other C&C Settings					Note
	DNS	HTTP	Channels	Protocols	Obfuscation	Payload	# FQDNs / C&C IPs	
FIN 7, a.k.a Carbanak	Embedding data in TXT record [130].	N/A	POWERSOURCE embed C&C traffic in TXT record of DNS packet [130]. If POWERSOURCE is detected, then TEXTMATE opens another channel using the same technique, another C&C channel over GoogleDoc [130]. Finally, Ammyy Admin tool, a legitimate tool is used as C&C channel [160].	Remote Desktop Protocol (RDP)HTTP HTTPS DNS [130]	N/A	N/A	N/A	
Leviathan, a.k.a APT 40	NanHaiShu uses, Dynamic DNS With Third Party [161].	HTTPS [131].	Derusbi opens two channels, an HTTP beacon and HTTPS channel over 31800, several channels established by Gh0st, AIRBREAK and FRESHAIR [131].	HTTP, HTTPS [131].	XOR	N/A	N/A	APT 40 develop several custom tools and malware, including HOMEFRY. China Chopper is implanted as a web shell to brute force passwords, upload and downloads files packed with UPX and commands are sent over HTTP Post [131]. APT 40 targets VPN credentials [131].
Naikon APT	Dynamic DNS [134].	HTTPS [133].	Naikon backdoor drops all other backdoors to deploy a variety of channels, RARSTONE, which opens the first channel over HTTPS, then SslMM is installed to send the keystrokes and footprinting info. over two channels, Sys10 open another channel over HTTP to collect local IP addresses WinMM opens two channels to add more persistence over HTTP [133].	HTTP, HTTPS, FTP and TFTP [133].	N/A	HDoor is uploaded to the victim for internal reconnaissance over SOCKS5 proxy service FTP is used for data exfiltration [133].	N/A	
Patchwork a.k.a Dropping Elephant MONSOON	N/A	HTTPS	QuasarRAT opens multiple channels over SOCKS5 proxy and FTP [136], Socksbot opens another channel over SOCKS5 proxy to run Powershell scripts, BADNEWS uses RSS feeds and Github for C&C [135], TINYTYPHON is mainly used for data exfiltration [137].	HTTP, HTTPS, RDP, FTP and SOCKS5 [162]	QuasarRAT, NDiskMonitor use AES AutoIt base64 encoding, TINYTYPHON [137] and BADNEWS uses XOR [135].	NDiskMonitor collect username, files and directories and send it back to C&C server Unknown Logger spread through USB to disable security tools, records keystrokes, and collect usernames and IP addresses [135].	N/A	
Sandworm Team a.k.a BlackEnergy Quedagh [138].	N/A	HTTP, POST requests [138].	BlackEnergy is known to operate over multiple channels with [138].	HTTP POST requests getp and plv fields for plugin getpd and for binaries download [138].	Base64 Encoding [138].	The payload includes highly classified documents and layouts of Ukrainian SCADA and government. In addition, BlackEnergy captures keystrokes and obtains credentials[138].	N/A	BlackEnergy has the ability to create botnets for destructive attacks DDoS [138].

Continued on the next page

Table 3.2 – (Continued)

APT Campaign	Evasion Tech.		Other C&C Settings					Note
	DNS	HTTP	Channels	Protocols	Obfuscation	Payload	# FQDNs / C&C IPs	
Strider a.k.a PROJECTSAURON [139].	DNS Tunnelling	HTTP over ICMP	Remsec open four channels to maximise the persistence over DNS, HTTP, HTTPS and SMTP with TCP, UDP or ICMP [140].	HTTP, ICMP, DNS and SMTP [140].	Remsec uses multiple encryption schemes including RC5(CBC), Remsec.Null session pipes AES(CBC) and RSA [139].	Remsec obtains keystrokes with its module Sauron, network infrastructure layout performs ARP scanning, and exfiltrates data[140].	HTTP backdoor holds several URLs to locate C&C servers [140].	
Regin	N/A	HTTP over TCP, UDP and ICMP [141], and HTTPS over proxy of other victims in internal networks [163].	Regin opens multiple channels among internal networks (P2P) over network pipes SMB and ICMP raw socket that communicate with the machines on the border which are acting as a router to forward traffic to C&C servers over HTTP or HTTPS [163].	HTTP, HTTPS, SMTP, SMB [163] and ICMP raw socket [141].	XOR and RC5 [141].	Data exfiltration, Regin also obtains keystrokes and footprinting info [163].	(N/A)/ (4)[163].	
Red October, a.k.a Cloud Atlas [142].	N/A	Zakladka uses POST request [144]	Red October backdoors allow a victim to connect to C&C through a chain of proxies with different locations [143]. Recently, it can also open HTTPS channels to connect cloud services for C&C communications [142]. WNFTPSCAN Exfiltrate data to remote FTP servers [142].	HTTP, HTTPS and FTP [144].	Zakladka uses RC4 and base64 encoding	Data Exfiltration modules such as WNFTPSCAN, GetFileReg, and FileInfo do not interact with C&C server directly [144].	(>60) / (10) [143].	

At the beginning of this chapter, we briefly define the APT and start arguing whether APT refers to a special type of malware or a well-resourced campaign. After the survey presented in Tables 3.1 and 3.2, we have seen how a single APT campaign is equipped with multiple malware, including RAT, backdoors or trojans. As expected, only APT 35 and Sandworm use botnets out of 33 APT campaigns. From now on, we refer to the malware used by APT for C&C communications with confirmed IP addresses as APT malware unless the industry reports it as a botnet. Table 3.2 shows the persistence and stealthiness properties of APT malware, which increase the chances of evading defences over a long period of time.

There are remarkable differences between an APT and other malware, such as botnets or ransomware, that use C&C communication. For instance, when APTs initially compromise a target and drop a backdoor to communicate with C&C servers, they tend to implant other RATs and further backdoors to continue their operation with other C&C servers. In practice, once a SOC engineer discovers a backdoor or a list of IoCs belonging to an APT, an incident response plan will be carried out to contain the incident. However, with the persistence property, an APT tends to implant several backdoors against the same organisation, each of which exploits different zero-day vulnerabilities [3].

For instance, Elderwood APT launch their attack through zero-day exploits of CVE-2012-0779, CVE-2012-1875, and CVE-2012-1889, followed by another five zero-day exploits in one month against the same target in order to deliver backdoor Trojans [129] and that is a sign of highly intelligent and knowledgeable engineers who in charge of these campaigns with unlimited resources. In addition, we notice another example of how APT differs from typical malware or attacks represented by CozyDuke, a.k.a APT 29. The campaign uses a modular malware platform that includes the command execution module, password stealer, NTLM hash stealer, downloader, loaders and backdoors such as HammerDuke OnionDuke, CosmicDuke, and SeaDuke. As a result, the campaign might continue without detection and complete espionage and disruption objectives. Nevertheless, the APT campaign carefully develops its own custom malware, FQDNs and IP addresses and is able to exfiltrate over time to stay in stealthy mode against the same target.

3.3 Network-based Characteristics of APTs

In this section, we focus on the network characteristics of the APT campaigns covered in the previous section. Table 3.4 summarises the APT campaign in terms of their adaptation of protocols that carry out C&C traffic, the tool type of C&C, and the most frequent evasion techniques adopted for APT campaigns. We notice that most APT campaigns use DNS and HTTP(S) for their operations through their attack period. It does not mean every malware uses the two protocols, but many frequently use them. Because of the excessive usage of these two protocols, we dedicate Chapter 4 and 5 to defend against APT campaigns. For other protocols used by different malware families, the proposed NIDS should work with protocol-independent. We will use traffic analysis of packets' arrival times in Chapter 6 as the last line of the defence for such a task.

Table 3.3: Evasion techniques over TCP/IP protocols.

APT Campaign	C&C Protocols								C&C Tools			C&C Evasion TTP									
	HTTP	HTTPS	DNS	SMTP	SOCKS5	SMB	FTP	P2P	RAT	Bot.	Backdoor	Proxy	HTTP Embedding	Obfuscation	DNS Tunnelling	DGA	Dynamic DNS	Multi-Stage Channels	Fallback Channels	Multi-hop Proxy	Multi-Layer Encryption
APT 1	✓	✓	✓	✓			✓		✓		✓			✓		✓	✓	✓	✓		✓
APT 2	✓								✓		✓			✓				✓	✓		✓
APT 3	✓	✓	✓		✓				✓		✓	✓		✓				✓	✓		✓
APT 10	✓	✓	✓		✓		✓		✓		✓	✓	✓	✓				✓	✓		✓
APT 12	✓								✓		✓	✓	✓	✓				✓			
APT 15	✓		✓						✓		✓			✓							
APT 16	✓	✓							✓		✓	✓		✓				✓			
APT 17	✓										✓			✓				✓	✓		
APT 18	✓	✓	✓						✓		✓			✓				✓	✓		✓
APT 19	✓	✓	✓			✓	✓	✓	✓		✓			✓				✓	✓		✓
APT 27	✓	✓	✓						✓		✓		✓	✓				✓	✓		✓
APT 28	✓	✓	✓	✓	✓				✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
APT 29	✓	✓	✓	✓		✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
APT 30	✓								✓		✓	✓	✓	✓				✓			
APT 32	✓	✓	✓			✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
APT 33	✓	✓		✓					✓		✓			✓				✓			
APT 34	✓		✓				✓		✓		✓		✓	✓	✓	✓	✓	✓	✓		✓
APT 35	✓						✓	✓		✓				✓							
APT 37	✓	✓		✓		✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
admin@338	✓	✓			✓				✓		✓			✓							
Blockbuster	✓	✓		✓					✓		✓	✓	✓	✓				✓	✓	✓	✓
Cobalt Group	✓	✓	✓			✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.4: Evasion Techniques Over TCP/IP Protocols

APT Campaign	C&C Protocols								C&C Tools			C&C Evasion TTP									
	HTTP	HTTPS	DNS	SMTP	SOCKS5	SMB	FTP	P2P	RAT	Bot.	Backdoor	Proxy	HTTP Embedding	Obfuscation	DNS Tunnelling	DGA	Dynamic DNS	Multi-Stage Channels	Fallback Channels	Multi-hop Proxy	Multi-Layer Encryption
Dragonfly 2.0	✓	✓										✓		✓							
Duqu 2.0	✓	✓										✓		✓							
Elderwood	✓	✓							✓			✓		✓							
FIN 7	✓	✓	✓						✓					✓	✓			✓	✓		
Leviathan	✓	✓	✓			✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Naikon APT	✓						✓		✓		✓	✓	✓	✓				✓			
Patchwork	✓	✓			✓		✓		✓		✓	✓	✓	✓				✓	✓		✓
Sandworm	✓				✓				✓	✓	✓		✓	✓				✓	✓		✓
Strider	✓	✓	✓	✓							✓			✓			✓	✓	✓		✓
Regin	✓	✓		✓							✓	✓		✓				✓			
Red October	✓	✓					✓				✓		✓	✓				✓		✓	

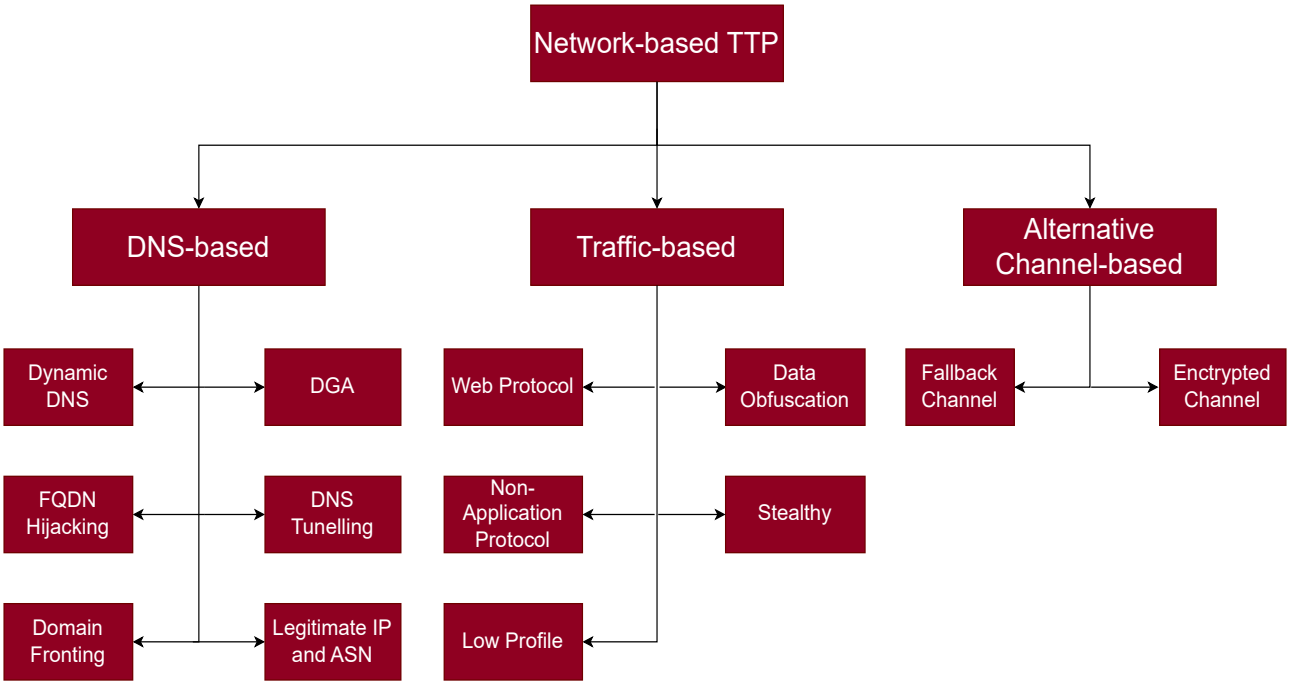


Figure 3.1: Network-based TTPs taxonomy.

3.4 Taxonomy of Network-based TTPs Used by APTs

Figure 3.1 illustrates the taxonomy of the network-based TTPs based on our findings. The first level is categorized based on the common usage by users. For example, DNS-based is normally used for retrieving the DNS resources, such as the IP address of the remote server. In APTs, the possible techniques used to locate C&C servers include Dynamic DNS, FQDN Hijacking, Domain Fronting and stealthy DGA. However, we include DNS tunnelling in this category as it appears to a SOC analyst as DNS packets. The second category relates to traffic-based techniques. We limit this category to our findings based on our datasets with additional information based on the technical reports. This includes web protocol, data obfuscation, non-application protocol, DNS over HTTPS (DoH), low profile, and stealthy behaviour. DoH remains in this category since it uses HTTPS and is not visible to a SOC analyst unless they have the key to decipher the packets at the web proxy. The final category is the possible channels that can be used during an attack. The two techniques frequently used for APTs are the encrypted and fallback channels. We keep these two techniques under the channel-based category because they can act as the backbone for other TTPs, and enhance the difficulty of detecting them.

3.5 DNS-based TTPs

In this section, we discuss the TTPs related to DNS. As introduced in Chapter 1, our first line of defence is to stop the APT traffic before it begins. Malicious domain detectors can be evaded using DNS-based TTPs. In Chapter 4, we will develop techniques to defend against this category.

3.5.1 IP Addresses and ASN

Analysts in [134] introduce an analysis of Naikon APT and observe that 99% of the campaign's domains use two IP addresses, and it can reach up to 51 IPs. This suggests to identify that counting the IP addresses used for a single domain over a time window might be useful. However, if we consider the autonomous system number (ASN) for a given IP address, then we could identify a malicious cluster. For example, in Naikon APT, six clusters are identified in South Korea (Seoul), China (Kunming and Jinma), Thailand (Bangkok) and the USA (Denver), with 22 city DNS resolutions referring to Kunming in China. [104] reports that threat actors of APT 27 frequently change the IP address of C&C domain, but within the same subnet. To confirm that, we need to collect APT domains and investigate historical data, which will be part of our work in Chapter 4.

3.5.2 Hijacked FQDNs

While the classic DNS spoofing attack targets the DNS recursive resolver's cache to forward the request to a malicious IP, FQDN hijacking targets the legitimate web server. FQDN Hijacking is the process of hijacking an FQDN with a qualified registered name and legitimate use [3]. First, threat actors penetrate the web server hosting a legitimate website. Then, they implant a backdoor on their web pages and send spear-fishing emails with a link to the victim. After the victim is hacked, the backdoor is preconfigured to communicate to C&C using the hijacked FQDN. For the network defences, the hijacked FQDN is a legitimate one, and the NIDS or SOC team cannot discover that the legitimate domain becomes a malicious one. We notice that APT 1 [3], APT 27 [106] and APT 34 [152] frequently use such a technique for their C&C operations.

3.5.3 DNS Tunnelling

APT 32, APT 34, Cobalt Group, and Strider campaigns use DNS tunnelling. A threat actor might embed restricted protocols inside a legitimate one, such as DNS, HTTP and SSH. The idea of DNS tunnelling is to communicate to C&C server via DNS query [164]. For instance, APT 32 a.k.a OceanLotus, uses DNS Tunnelling to avoid NIDS and to bypass Firewalls [28]. The destination IPs are legitimate DNS servers, i.e. Google and OpenDNS [28]. However, the malicious domain is embedded in the packet, which will be unpacked at some intermediate points to forward the traffic accordingly [165]. For instance, Pisloader sends a beacon periodically, with setting flags including response, recursion desired and recursion, and in case of any additional flags are set, the packet will be discarded [147]. The obfuscated based64 payloads are attached on the same string with the C&C server domain in a 4-bytes length [147]. The C&C server responds in TXT record of the DNS packet with the same settings [147].

3.5.4 Dynamic DNS

Since 2016, Dynamic DNS with third-party services has been essential for several APTs [94], including APT 1 [3], APT 2 [90], APT 10 [95], APT 15 [97], APT 18 [34], Leviathan [131] and Naikon [133]. APT actors frequently register new subdomains under sharing zones with other clients. APT 1, for instance, registered hundreds of subdomains over the years and frequently changed the resolution of FQDN to new IP addresses of C&C servers by logging into the service via a web-based interface and updating instantly [3]. NIDS and the security team view the malicious FQDN using Dynamic DNS as a domain from a public service, which makes it difficult to classify it as malicious.

3.5.5 Stealthy Domain Generation Algorithms (DGA)

DGAs generate a large number of pseudo-random variations of domain names using algorithms based on arithmetic, hash functions, wordlists, and permutations [166]. Normally, DGA is used as a backup plan for the botnets to communicate back to a master bot [167]. Zeus and Cryptolocker are pretty famous for using binary DGAs modules attached to the malware to algorithmically generate domains (AGDs) until C&C channel is established [167]. Once the

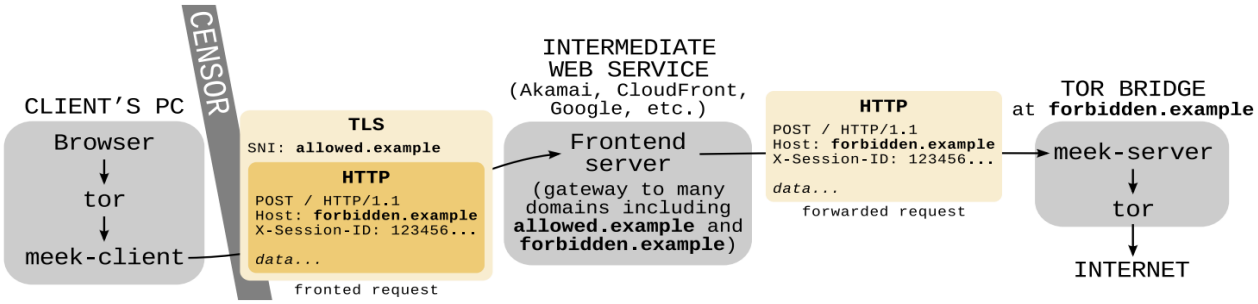


Figure 3.2: An example of domain fronting [10].

host is infected by malware with the DGA module, the module will start sending a random DNS query with a seed value known upfront by the threat actor. The communication will then be initiated with the C&C server if the DNS query matches a registered domain. In the APT context, APT 28 and APT 34 are the main examples of campaigns using DGA. Backdoor uses DGA to create more domains upon the request of C&C servers in hours and destroy it within 24 hours [168]. These make it hard to predict what domain names will be used, and even when the algorithm is successfully reverse-engineered, take-downs are expensive as only a small fraction of a large number of generated domains is effectively registered by an adversary.

3.5.6 Domain Fronting and Multi-hop Proxy

Domain fronting is an evasive technique that conceals the location of the remote C&C server [10]. The backdoor implanted in a victim machine pretends to communicate with a legitimate host over HTTPS. The legitimate host is only an intermediate step to communicate with the true destination. This chain may include multiple hops. The domain outside of the HTTPS request is a legitimate one, while the inside the HTTPS is the true destination, which is encrypted (Figure 3.2).

We observe that APT 29 [31] and Blockbuster [124] use domain fronting and multi-hop proxy. APT 29 connects to the onion router (TOR) with the domain fronting technique, i.e., a domain included in the TLS header refers to Google services. The destination domain (TOR multiple hops) is hidden in the HTTP header until it connects back to a threat actor [149]. To update the FQDN for newer C&C servers, the Duke malware may receive the IP address and exfiltrate over Twitter and Microsoft One Drive [30]. In addition, the TOR tunnel provides remote access to the victim by using Server Message Block (SMB) and NetBIOS. Throughout this operation, the

HTTP POST header appears to the NIDS as a legitimate URL with unsuspicious parameters [149].

3.6 Traffic-based TTPs

After APT operators successfully evade the defences and locate the C&C server, they deploy several traffic-based TTPs to continue undetected. This section discusses the TTPs related to the traffic itself. Our second and third lines of defence (Chapter 5 and 6)) aim to detect such techniques and separate those legitimate from APTs.

3.6.1 Web Protocol

In order to evade detection and network filtering, adversaries may communicate by utilising application layer protocols typical of online traffic. The protocol data exchanged between a client and server will contain instructions for the remote system and, typically, the responses to those instructions. Internet communication protocols like HTTP(S) [90] and WebSocket [169] may be widely used. Many different kinds of information can be hidden in the various fields and headers of an HTTP(S) packet. An attacker can utilise this to communicate with target systems under their control while appearing to be legitimate network traffic.

In our dataset described in Chapter 5, APT28 uses a malware called Zebrocy. The malware has been observed spreading through maliciously produced Word documents with malicious macros and Dynamic Data Exchange (DDE)-based via social engineering or attachments targeting governments [170]. In our analysis, we confirm that the information of the infected hosts is collected and then sent back to 220.128.216[.]127 using HTTP with the POST method. In Listing 3.6.1, we extract the HTTP header and identify the usage of WinHTTP, a library often used as a web crawler or bot.

```
POST /search-sys-update-release/base-sync/db7749sc.php?next=C4BA3647 HTTP
   /1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: */*
```

```
User-Agent: Mozilla/4.0 (compatible; win32; winHttp.winHttpRequest.5)
Content-Length: 793558
Host: 220.158.216[.]127

dbgate=Path: C:\Users\admin\AppData\Local\Temp\8
   ac4e164b463c313af059760ce1f830c19b0d5a28ec80554e8f77939143e24e.exe
```

Listing 3.1: Zebrocy pushes collected information of the infected host using POST method with WinHTTP library.

3.6.2 Data Obfuscation Through Protocol Impersonation

A common tactic used by adversaries to prevent analysis and conceal C&C activity is to forge protocol or web service communication. In order to conceal their C&C traffic, adversaries may masquerade as genuine protocols or online services. To trick security tools into thinking the following traffic is encrypted with SSL/TLS or to make their traffic appear to originate from a trusted source, attackers can spoof an SSL/TLS handshake. For example, A false TLS handshake is generated by Bankshot using a public certificate to camouflage C&C network communication [171]. Cobalt Strike simulates HTTP while concealing malicious data by adding it to the URI, hiding it in headers, parameters, or transaction bodies, or using a combination of these methods [125]. Okrum uses a protocol that is quite similar to HTTP for C&C communication, but it conceals the actual messages in the Cookie and Set-Cookie headers of the HTTP requests that it makes [172]. In our dataset, we commonly observe protocol impersonation through fake TLS over HTTP. In Figure 3.3, we observe beaconing Mivast and Sakula malware adopt such a technique while the traffic is sent as impulses in periodic time slots.

However, the protocol impersonation technique may include customized protocol. For instance, SolarWinds use the Orion Improvement Programme (OIP), where their users can provide feedback on the quality, functionality, and usefulness of their products in order to guide future development. Data about errors and other unusual occurrences in the system is also compiled [173]. Therefore, The SUNBURST Backdoor is able to blend in with legal SolarWinds activity because it disguises its network traffic as the OIP protocol and keeps its reconnaissance results within legitimate plugin configuration files [174].

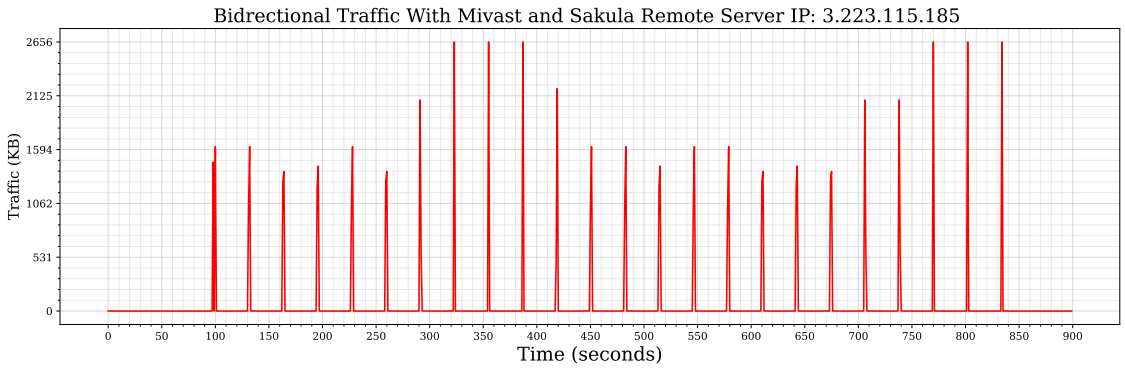


Figure 3.3: Mivast and Sakula malware gather information and send it back to the operator periodically through data obfuscation through protocol impersonation.

3.6.3 Non-Application Protocol

An adversary may use a non-application layer protocol for communication between a host and a C&C server or between infected hosts inside a network [175]. Numerous protocols can be considered. Network layer protocols like Internet Control Message Protocol (ICMP), transport layer protocols like User Datagram Protocol (UDP), session layer protocols like Socket Secure (SOCKS), and redirected/tunnelled protocols like Serial over LAN (SOL) are all used as examples.

Host-to-host communication via ICMP is one example. Since ICMP is included in the Internet Protocol Suite, all IP-compatible hosts must support it. However, unlike TCP and UDP, it is not as widely monitored and could be used by attackers looking to conceal conversations [176]. In Figure 3.4, we show an APT operator using NanoCore interacting with the victim using raw TCP only.

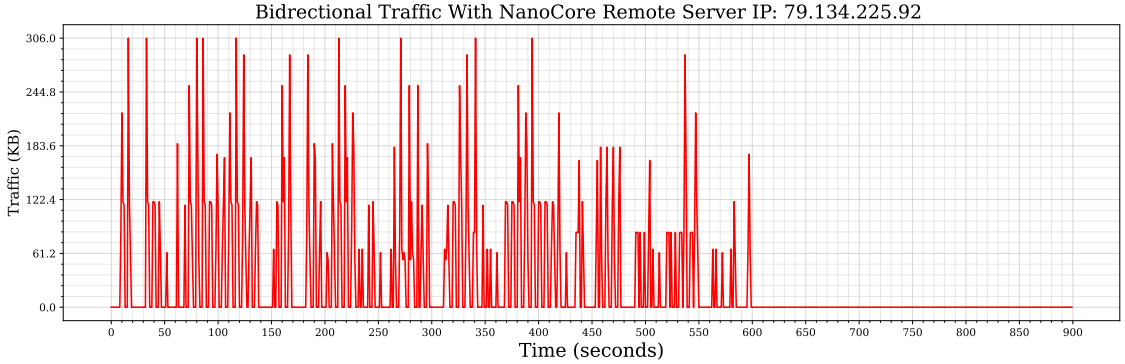


Figure 3.4: NanoCore executes remote commands and collects information over Raw TCP.

3.6.4 Stealthy behaviour

Many reports blend stealthy behaviour with a low profile. However, our findings show that stealthy behaviour should be categorized as a different concept. Stealthy behaviour is normally used for uploading a malicious payload. That means some packets are sent/received in a burst within a short period. For example, based on our analysis, njRAT exchanges data with the APT C&C within 76.53 seconds to drop the payload and exfiltrate initial information. In Figure 3.5, we show the difference between the stealthy behaviour and the legitimate one. We notice that the volume of the burst does not exceed the legitimate threshold. In our example, the maximum magnitude of the burst for a single second is 7,614, compared to 18,630 KB for the legitimate one. This means the stealthy packets consume only 40.87% of the legitimate threshold. However, we can see that the njRAT stealthy packets exchange data nearly without pausing after executing remote commands at the beginning. Once the mission is done, no packets are exchanged in the same time window.

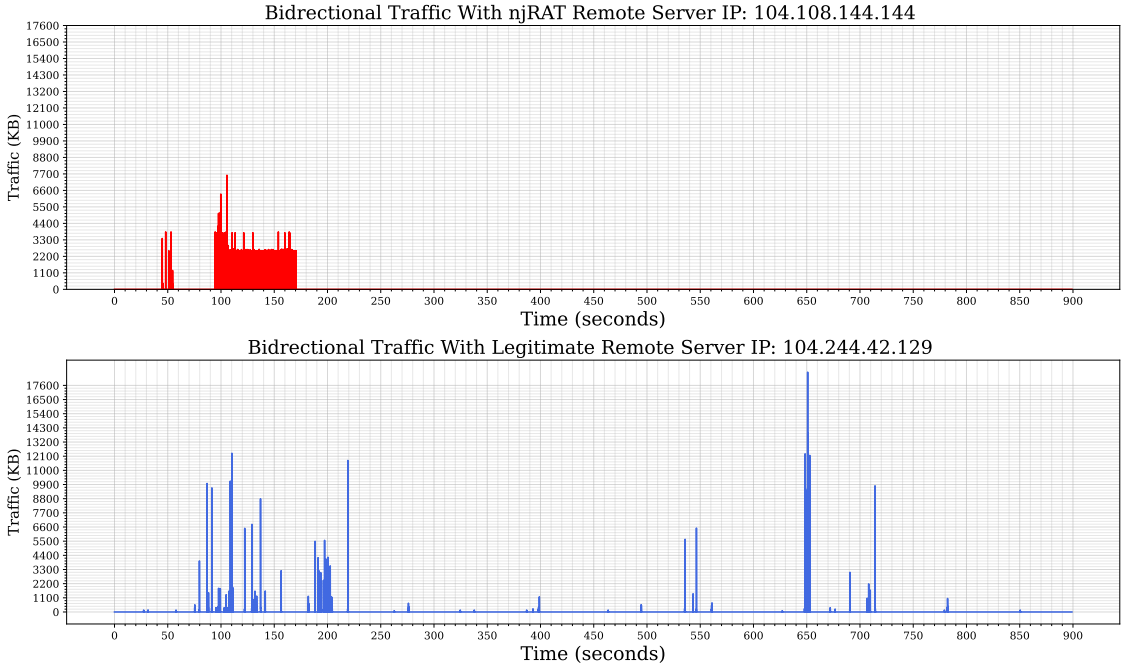


Figure 3.5: Stealthy malicious APT StringPity compared legitimate behaviour.

3.6.5 Low Profile

Most of the reports produced by governments and security vendors emphasise that the APT traffic normally operates at a low profile. That means an adversary is aware of the typical

traffic volume in an enterprise network. An adversary may guess based on the number of hosts connected in the internal network. During the initial compromise stage, an adversary may gather the information around the network to minimize their traffic at a low profile.

We can see the low magnitude of the malicious traffic for most TTPs in the figures presented in this chapter. However, let us take another example that stays in low profile mode while it does not use any previous traffic-based TTPs other than the non-application TTP. Figure 3.6 shows the behaviour of Remcos. Remcos is a highly developed RAT that allows remote control and monitoring of any Windows machine running XP or later. Data is collected and transmitted to C&C servers, including OS version, user access, and processor revision number and architecture [177].

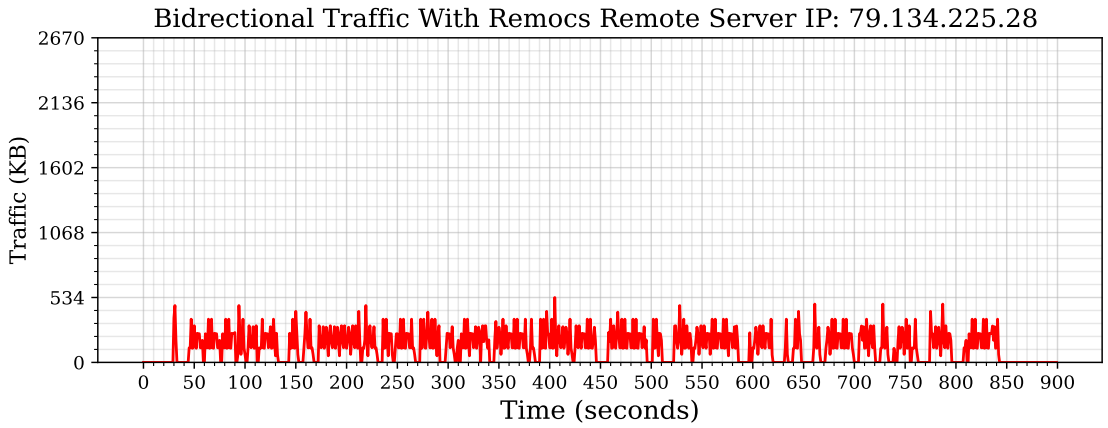


Figure 3.6: Remcos transfers the collected data of the infected machine.

3.7 Alternative Channel-based TTPs

In this section, we describe the most common techniques used by most APTs. As described before, APT establish alternative channels for either persistence or NIDS obstruction. Our experiments in Chapter 5, page 161, find that fallback and encrypted channels are the most commonly used.

3.7.1 Encrypted Channel

It is well-known that several malware families use encryption for their communication. However, a multi-level encryption channel can be adopted in the APT settings. A common technique is to use Base64 encoding to disguise the remote commands using another encoding scheme such as UTF, ASCII or, simply, rotate by 13 places (ROT13). Next, the output is XORed the plaintext with the pad, and encrypt the output using symmetric encryption algorithms such as block ciphers (e.g. AES) or stream ciphers (e.g. RC4). Later, when the traffic is carried over a trusted VPN, SOCK5, or TLS, the local network cannot identify the real information even after deciphering the connection. In this way, the local NIDS can be evaded as it sees the traffic came from public encrypted services such as GitHub, Dropbox, GoogleDrive, or any account on the social media platform.

3.7.2 Fallback Channel

APT operators establish alternative channels with different IP addresses for two purposes. First, maintain persistence by providing a backup channel if one is detected and blocked. Second, stay undetected by following a divide-and-conquer strategy, which is the main reason for using fallback channels as a backbone to enable other TTPs deployment. We focus on this goal for the rest of our discussion and analysis here. As we know that APT operators carry out their different operations, such as payload download, controlling the victims with remote commands, crawling the information of the infected network and exfiltrating data.

In practice, it is nearly impossible with the current defences in highly secure environments to accomplish these operations without getting detected. Therefore, APT campaigns establish many remote servers to communicate with and split the communications among them to appear several legitimate behaviours from the enterprise point of view. For example, the first channel can be dedicated to downloading an internal reconnaissance tool by a combination of stealthy and web protocol TTPs. In addition, a new domain is sent using one of the DNS-based TTPs to evade malicious traffic and domain detectors.

In Figure 3.7.a, we plot the planned traffic to be exchanged with C&C server. However, Figure 3.7.b-e depicts the actual behaviour, which is split over four C&C servers during the same

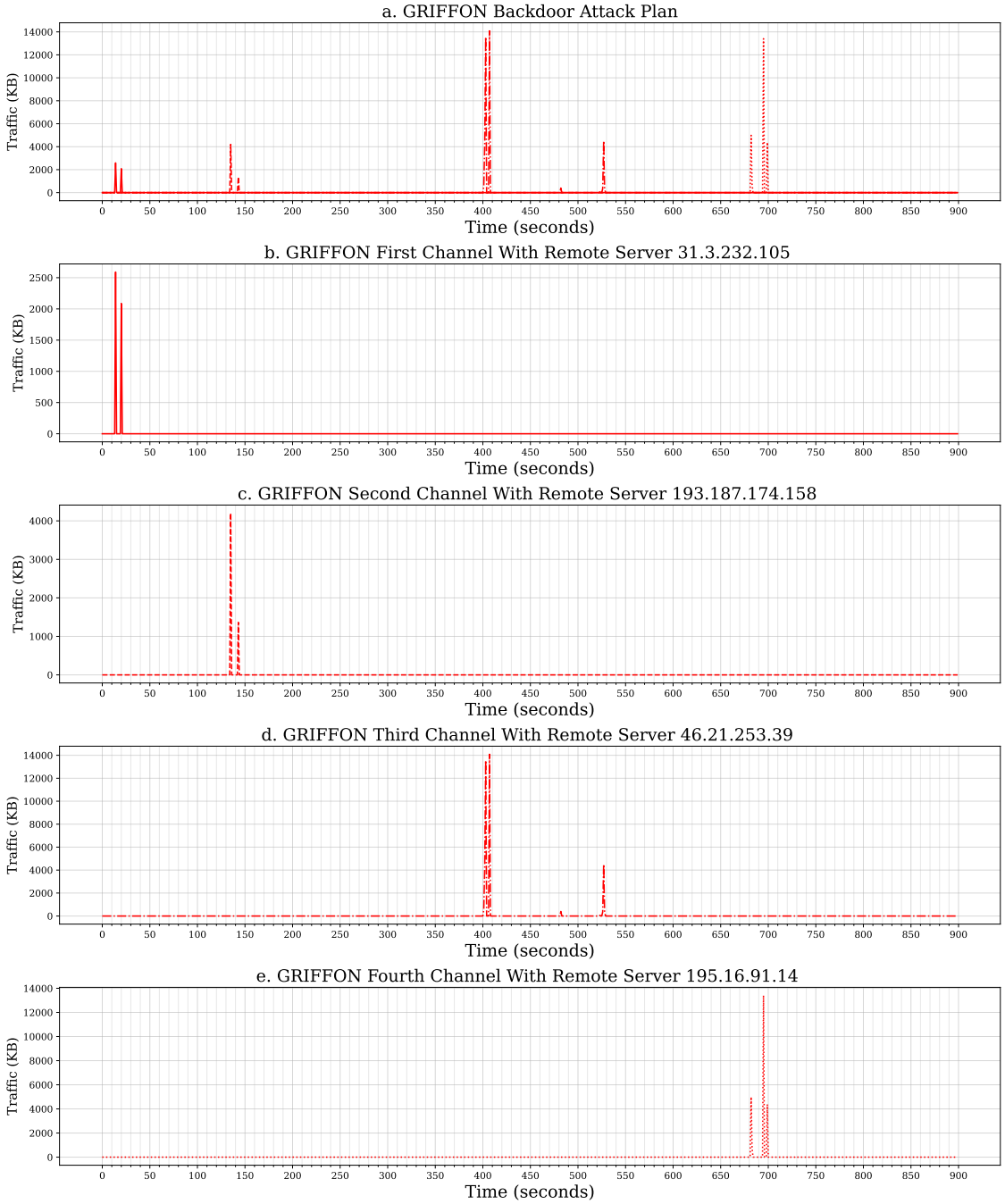


Figure 3.7: GRIFFON divides its communication with its C&C over four channels.

15 minutes. Now, the first channel carries out the remote commands and retrieves initial internal reconnaissance information of the infected network. Just before being idle (Figure 3.7.a), another domain is sent for the next channel. With resolving the domain name sent during the first channel, a second channel starts to communicate to another C&C server. The second channel continues retrieving more internal information with another domain sent for the next channel. Now, the third channel transfers a payload with two large impulses. Further tools are also sent over the last channel. Finally, APT operators prepare the tools needed for

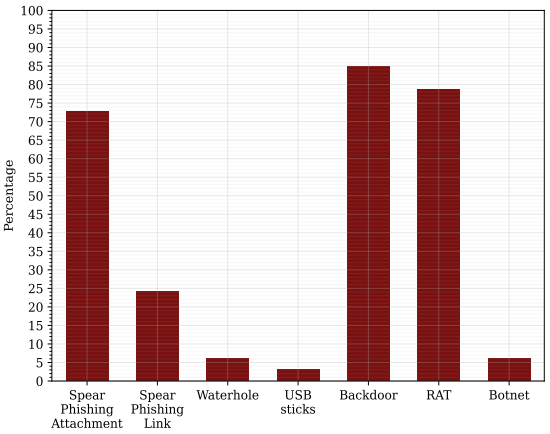
further tasks. For example, they can keep a logger and transfer the usernames and passwords, or they can continue with the lateral movement stage to control another victim in the same network.

3.8 Discussion

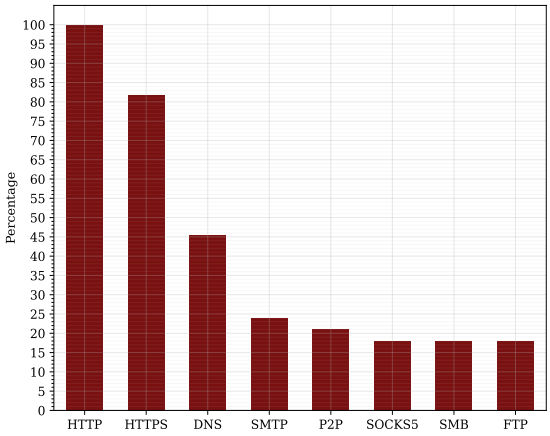
In this section, we summarise our findings based on our study, which includes our investigation of 118 technical reports supported by our real investigation based on our collected datasets introduced in respective chapters. We notice that over the past 20 years, spearphishing links have been popular to deliver the backdoors until now, which emphasises the importance of developing a detective approach to defend against malicious domains, which will be addressed in Chapter 4. Spearphishing attachments are also popular and can break the current host-based intrusion detection, which is out of the thesis's scope, and we leave it for future work.

In the previous chapter, we discuss if the APTs are a single malware or an arsenal of malware. In this chapter, we reported the backdoors for each campaign to confirm that APTs used a collection of malware, for instance, up to 26 different malware families used by APT 1. Some of these backdoors are communicating with the remote C&C server. There is a chance to find if an organisation is under an APT campaign attack if we are able to detect the malicious traffic by such malware. We find that 84.8% of APT campaigns use malware that is described as backdoors and 78.7% are RAT while only 6% are botnets as depicted in Figure 3.8.a. That clarifies that APTs rarely use botnets for their attack and rely heavily on customised backdoors and RATs. For this reason, we will collect some APT malware to evaluate our work in detecting their malicious traffic in Chapter 5 and 6.

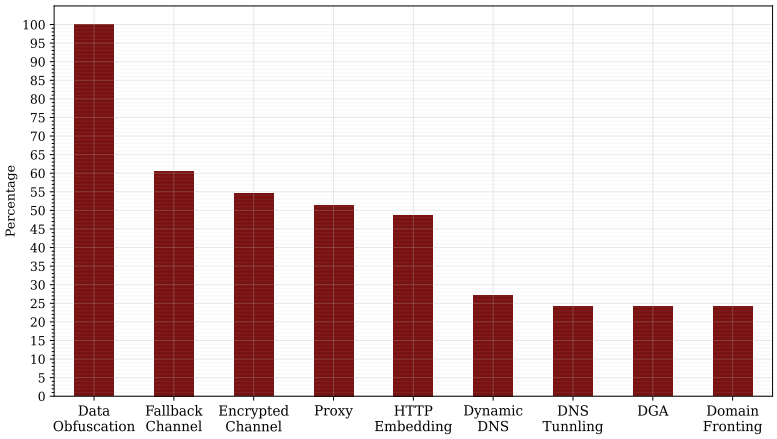
Since our scope in this thesis is to defend against APTs using network data, we limit the rest of the discussion to our findings related to the popular protocols and TTPs used to communicate to C&C server.



(a) Most popular delivery methods.



(b) Most popular protocol usage.



(c) Top TTPs usage.

Figure 3.8: Investigation summary based on 33 APT campaigns.

3.8.1 APTs Popular Protocols

Figure 3.8.b shows that all APT campaigns have continued using HTTP since 2001 to evade NIDS detection, emphasising that we need to carefully detect malicious connections that use HTTP protocol. That does not mean every single connection will use HTTP protocol, but it means at least every APT campaign will use HTTP once at a time. 81% of APT campaigns rely on HTTPS to bypass NIDS that rely on HTTP plaintext features. In Chapter 5, we will show our approach to detecting HTTP(S). Since HTTP protocol normally starts with DNS domain resolution, 45% of APTs use DNS protocols, while the others use HTTP with a preconfigured IP address.

Also, 24.2% of APT campaigns deliver their malware using spearphishing links, which we should also consider to detect malicious URLs. In Chapter 4, we present the first line of defence to

stop APT campaigns from the beginning. SMTP, P2P, SOCKS5, SMB and FTP come next with 24%, 21.2%, 18.1%, 18.1%, and 18%, respectively. Therefore, we need to design a generic approach that can detect malicious traffic regardless of the protocol type, which is presented in Chapter 6

3.8.2 APTs Popular Network-based Evasion Techniques

To design a reliable approach to detect APTs, we study their usage of TTPs that enable them to bypass NIDS. Figure 3.8.c shows that APTs typically adopt data obfuscation through protocol impersonation. We show an example of Mivast and Skula malware in Figure 3.3 impersonating legitimate HTTPS. The next popular TTP is the fallback channel to split the traffic volume over multiple C&C servers, which is used by 60.6% of APT campaigns. This percentage is increasing over time to evade detective approaches that rely on volume-based features, as we have seen in Section 3.7.2, which are been quite a popular approach in the past two decades.

The multi-level encrypted channel is also popular and is used by 54.5% of APT campaigns to evade decryption if the TLS traffic is blocked unless the decrypt key is available for the web proxy. Therefore, even if the traffic is decrypted, another layer is still held, such as encoding technique or block cipher, as described in Section 3.7.1. The next popular TTP is domain fronting and multi-hop proxy, which is more than half (51.5%) of APT campaigns exploit such technique together and 24.2% only use domain fronting, to conceal the location of the remote C&C server as described in Section 3.5.6. Other DNS-based TTPs are also found, with 27.2% using dynamic DNS, and 24.2% exploiting DGA or DNS tunnelling. This leads to the importance of considering the detection of malicious domains and UDP-based traffic. We consider such behaviours in Chapters 4, 5, and 6.

3.9 Conclusion

At the beginning of this chapter, we highlight the importance of developing a successful APT detection strategy, which can be achieved by, first, studying the network-based TTPs. These TTPs pose a challenge when it comes to distinguishing between malicious and legitimate activ-

ities. Consequently, when formulating approaches for the upcoming chapters, it is imperative to consider the particular context of the attack as discussed in this chapter.

In this chapter, we analyse 33 APT traffic campaigns (Tables 3.1 and 3.2) in terms of many features of TTPs, including evasion techniques, protocols, payloads, obfuscation and channels. We observe several APT campaigns use zero-day vulnerabilities, and we denote that in Table 3.1 with a (†) symbol. For instance, Stuxnet uses four zero-day exploits, while Elderwood uses eight [129]. However, other APTs such as Taidoor reuse exploits [129]. These different exploits provide multiple persistence against targeted organisations. In addition to that, we discuss some evasion techniques used by a wide range of campaigns and summarised in Table 3.4.

We conclude that typical malware or multi-stage attacks are run normally by individuals or small teams. On the other hand, APTs are launched by a group of highly skilled people and mostly funded by governments. Finally, we define the most popular network-based TTPs used by APTs. We focused on HTTP(S) and DNS protocols and categorized 13 TTPs related to these protocols since HTTP(S) and DNS are the popular protocols among APTs with 81% and 45%, respectively. We present several examples based on our datasets in addition to further resources from the industry. In the next chapters, we present our defences against APTs based on detecting the malicious use of these TTPs.

Chapter 4

Holistic APT Command and Control Domain Detection

In Chapter 2, we discussed how APTs often invest significant resources per victim and launch low-volume and targeted attacks over long periods. The low volume and targeted nature of APT attacks complicate detecting APTs using only traffic analysis. In this Chapter, we aim to protect networks from APT attacks effectively by creating a first line of defence based on APT domain detection. However, finding a public APT domain dataset to analyze and evaluate domain detection is challenging. According to [89], the security industry is the primary source of information on APTs. In some cases, targeted organizations request forensic services from security companies, which publish (part of) their confidential findings after obtaining client consent.

In this Chapter, we present the most comprehensive study to date of the usage of domains in the context of APT Command and Control (C&C) infrastructure. Our research covers 63 APT campaigns spanning 13 years until August 2020, by reviewing 125 public reports and 146 threat intelligence pulses provided by 35 leading security organizations, including FireEye Mandiant, Palo Alto Networks, Symantec, US-CERT and many others. The results of this analysis led us to propose a detailed threat model for the usage of C&C by APTs, focusing primarily on evasion techniques (Section 4.1).

As a basis for objective evaluation, we collect and make an extensive dataset of APT C&C

domains publicly available based on the campaigns we reviewed. We leverage the gained insight to study the effectiveness of existing and novel features of domain names, and their DNS infrastructure, for detecting APT C&C domains. The detection of malicious domains is a common problem when fighting attacks such as botnets or phishing campaigns. Botnets often use DGAs, discussed in Section 3.5.5, to protect their C&C channels from disruption. Character level features [178] and Non-Existent Domain (NXDomain) responses [179, 16, 180] were shown to be effective in detecting DGA domains. Some APTs use domains that look similar to DGA-generated domains, so character-level features are also relevant to their detection. On the other hand, APTs prioritize stealth and are unlikely to generate a large number of NXDomain responses, making the corresponding detection technique not relevant. Overall, we have not found typical DGA domains in our analysis of APT campaigns in Section 4.1.

Phishing campaigns can be detected by their choice of domain names, or by analyzing the look, structure and behaviour of phishing web pages themselves [181, 182]. We note in Section 4.1 that APT domains may serve innocent pages by default, preventing the latter kind of analysis. Phishers who choose deceptive domain names do so with the goal of confusing an end-user into thinking that the deceptive domain is, in fact, the legitimate target. APTs who adopt deceptive domains for their C&C communications instead do so in order to avoid automated and manual detection. Yet, phishers and APTs adopt some similar techniques, and we shall leverage lexical-based phishing domain detection techniques to build APT-specific detectors.

In order to evaluate the performance of the various feature sets we studied, we built HawkEye, a system to classify domain names requested in PCAP files. HawkEye includes modules for robustly parsing and retrieving DNS data, which is sometimes technically challenging, and for extracting features and classifying domains. Finally, we compare and discuss the detection performance of different sets of features for known APT campaigns and unseen ones. We find that a holistic feature set including largely orthogonal features performs the best, achieving an accuracy of 98.53%, an $mF1$ of 90.38%, and a low FPR of 0.48% when detecting unseen APT campaigns. In comparison, the baseline achieves lower performance, with accuracy, $mF1$, and FPR values of 96.95%, 76.81%, and 0.68%, respectively, for unseen APT campaigns.

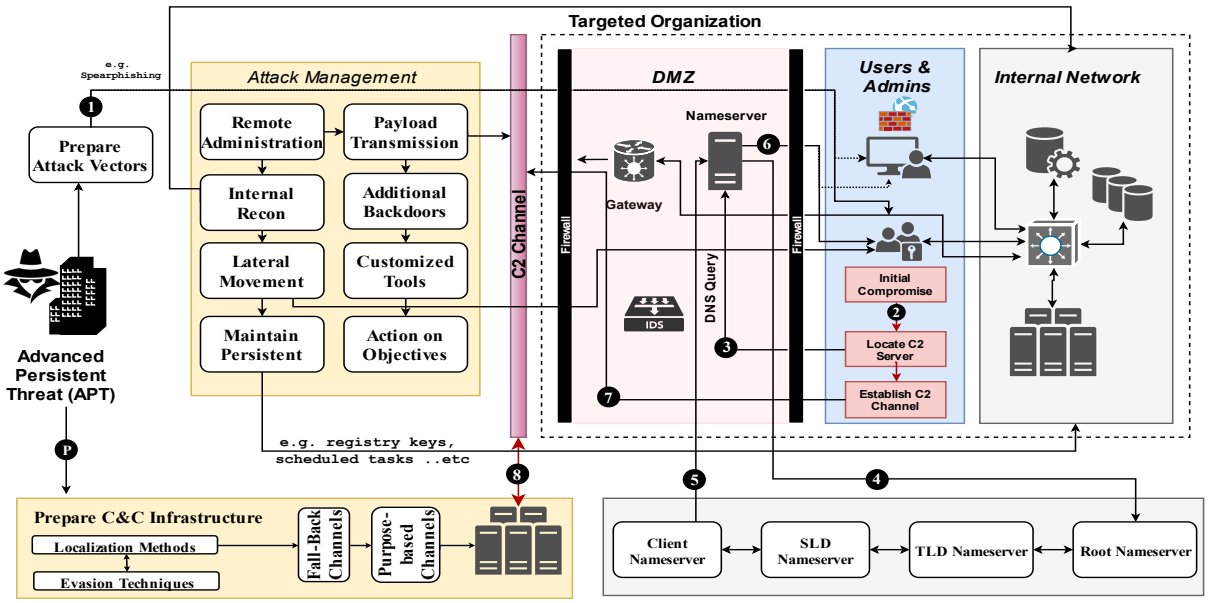


Figure 4.1: APT Threat Model. Prior to infection, APTs prepare the C&C infrastructure and the initial compromise vectors. Once exploitation begins, APTs move to the attack management phase, which includes C&C localization by infected hosts.

4.1 Threat Model

Understanding how APTs behave, from the first day of a campaign until the mission is accomplished, is crucial in order to identify the attack surface at each stage and helps select appropriate detective- and protective controls. The Lockheed Martin Cyber Kill Chain could be considered as the first model that was able to describe Advanced Persistent Threats (APT) proposed in 2011 [2]. That was followed by the Mandiant [3] and MITRE ATT&CK [4] frameworks, both increasingly accepted among the security community as we discussed in details in Chapter 2.

In Figure 4.1, we propose a more specific APT model, focusing on the localization and establishment of the C&C channel, which we use to inform the design of HawkEye. The threat model describes the typical lifecycle for most APTs from the network perspective, and in particular how an APT smoothly penetrates these defences and manages to hide their activities over time through stealthy C&C channels. The attacker activities are divided into three main phases.

4.1.1 Prepare C&C Infrastructure

APTs tend to prepare their infrastructure over multiple years. They normally establish a sophisticated network for fall-back channels support, carefully selecting the locations of their C&C infrastructure and the localization method (Figure 4.1, ❿). These decisions are based on the properties of their targets. For example, if the target is a government entity in a specific country, C&C servers are likely to be located in the same country, and to use domain names that mimic government-like domains. Using domain names that blend in with the business of the target, or that could plausibly belong to its technical infrastructure is a common technique to evade IDS detection (which needs to avoid false positives), and to mislead SOC engineers into overlooking innocent-looking alerts. For instance, DarkHydrus [183] used `symanteclive[.]download` (with nameserver `ns102.kaspersky[.]host`) to localize its C&C server. It also registered `owa365[.]bid` to imitate Outlook Web Access (OWA), `kaspersky[.]science`, `fortiweb[.]download` to masquerade as Kaspersky and Fortinet products, `data-microsoft [.]services`, `Akamai[.]agency` and `windowsdefender [.]win` to blend in with network and cloud infrastructure. Other APT campaigns also use this technique, with CobaltGroup using `api.outlook[.]kz`, APT 41 using `macfee[.]ga` and `kasparsky[.]net`, and APT39 using `win7-update[.]com`. As a further measure to avoid C&C detection, APTs tend to reserve C&C domain names exclusively for that purpose, and set them up to display clean pages as their default content.

4.1.2 Prepare Attack Vectors

This stage (Figure 4.1, ❶) includes selecting the targeted organization, information gathering, customizing malware and tools, identifying vulnerabilities and delivering a malicious payload to the targeted host, using techniques such as spear phishing, whaling or strategic web compromise (SWC). We consider domains used exclusively as part of this phase as out of the scope of our investigation.

4.1.3 Attack Management

At this stage, APTs have partial control of the target and can perform malicious activities, such as *remote administration* using SSH or another tunnelled protocol. Meanwhile, they can also

transfer more payloads, over different time windows, binary files, PowerShell script, RATs and post-exploitation tools. At the same time, the campaign is sneaking deeper into the network through *internal reconnaissance* and *lateral movement*, potentially using superuser accounts which appear legitimate to the targeted network.

These actions are controlled via the C&C channel. In a typical instance, once the target has been compromised through a decoy document (Figure 4.1, ❷), malware starts locating the C&C through the domain that is hard-coded in the configuration block of its main binary (❸). Next, the victim issues a DNS query for that domain to the local nameserver. This request is initiated by operating system processes without the need to evade a browser or a web application firewall. If the requested record is not in its cache, (❹), the local nameserver communicates with root, TLD, SLD and enterprise-level nameservers to obtain them (❺, ❻). The contacted domain may display legitimate and harmless web pages, or not even display any content. In the context of APTs, the `A` record normally points to a C&C server, and a typical APT campaign changes the `A` record periodically to avoid frequent connections to the same IP (❼). For example, the APT C&C domains `windowsdefender[.]win` and `micrrosoft[.]net` changed their `A` records every 24 and 48 hours during June - July 2018 and August - September 2019 (using nameservers `ns102.kaspersky[.]host` and `ns11025.ztomy[.]com`), providing each time a fresh IP. Also, Strider APT aggressively changed the `A` records for `bikessport[.]com` to 203 distinct IPs from February until December in 2018, despite the nameserver providing only a single `A` record at a given time. We also notice that, at the opposite end of the spectrum, some APTs make substantial reuse of IP addresses. For example, the Donot APT C&C subdomains `jasper`, `qwe`, `alter`, `genwar`, `param`, `car` and `bike` of `.drivethrough[.]top` share the same IP address from April 2019 until July 2019.

While both phishing or botnet campaign infrastructures tend to exhibit large numbers of `A` or `NS` records (compared to RFC recommendations) as a defence against take-downs [184], we notice that APT C&C domains tend to deploy even fewer such records than popular legitimate domains do. Also DNS `TXT` records can be abused by APTs tools. For example, RoyalDNS of APT 15 (Ke3chang) used `TXT` records to send malicious payloads [98], HTTPBrowser and Pisloader of APT 18 (Wekby) used them to exfiltrate data [34], and also FIN7 (Carbanak) embedded data in them [130].

To *maintain persistence* over the network, APTs adopt many techniques, including altering

registry keys, scheduling tasks and using additional malicious domains for fall-back channels, in case one of the earlier ones had been detected and taken down by defenders (❽). We even noticed taken-down domains being later released back to the market and used again for the same campaign. For example, although `maccaffe[.]com` is a typosquatting of `mcafee.com` registered on October 2019, its historical records reveal that the domain has been resolved to 99 distinct IP addresses from August 2013 until December 2019. Similarly, `0ffice[.]com` and `hotmai1[.]com` have been active since 2008 and were still active at the time of writing this chapter, despite violating common registrar regulations.

Some of the features we will select in Section 4.3, such as the length of a domain, or its decomposition into meaningful parts are *semantic*, in that they reflect the choice of a particular domain name. Other features are *contextual*, in that they express the organization of the DNS infrastructure that supports the C&C operation for the chosen domain. We also collect *hybrid* features, which extract semantic properties from the DNS infrastructure (Section 4.4).

4.2 HawkEye

In order to detect if a domain name being accessed by a monitored host belongs to APT C&C infrastructure, we have implemented HawkEye, a system that processes PCAP files to extract and classify relevant domains. Our classification is based on a Random Forest classifier, which uses the *semantic*, *contextual* and *hybrid* features described in Sections 4.3 and 4.4. In order to evaluate our classifier, and facilitate further research in the area, we built the Hawk-Eye Dataset, which collects a large number of features for the C&C domains of the APT campaigns of Section 4.1.

4.2.1 Architecture

HawkEye is composed by the *Parser, Crawler, Preprocessor* and *Classifier* modules. The Parser inputs a PCAP file and extracts from DNS queries the Fully Qualified Domain Names (FQDNs) to be classified. Next, these FQDNs are segmented into the host, Entity Level Domain (ELD) and public suffix, as explained in Section 4.3.1. The output of the Parser is used directly to extract semantic features, and is passed to the Crawler, to retrieve contextual

and hybrid features. For active campaigns, live WHOIS and DNS information suffices, whereas for older campaigns, including those with expired registration or run by a sinkhole operator, we query the historical information from SecurityTrail [185] as discussed in Section 4.5.1. The WHOIS collector sub-module requests registration information from WHOIS servers and parses these files from text or JSON format. This component can parse 23 different WHOIS formats, covering the vast majority of the Alexa top 500K domains. The DNS collector sub-module handles the Resource Record (RR) responses we need to process. RRs are individually obtained with independent requests, as several nameservers disable `ANY` queries for security reasons and to comply with RFC8482 [186]. The RData payload of different RRs have different formats: HawkEye implements parsers for all the RData and zone file formats relevant to any domain in our datasets. The data obtained by the Crawler is passed to the Preprocessor, which normalizes data, fills in missing values and encodes categorical and ordinal features in order to meet the requirements of different machine learning models. Finally, the Classifier labels each FQDN, according to the chosen classifier. For the heterogeneous features considered here, we found Random Forest to be the most appropriate model.

Figure 4.2 shows the flow of data in HawkEye, from the FQDN being analyzed to the classification result. Further details on the individual stages are provided in Sections 4.5.1, 4.3, 4.4 and 4.6.

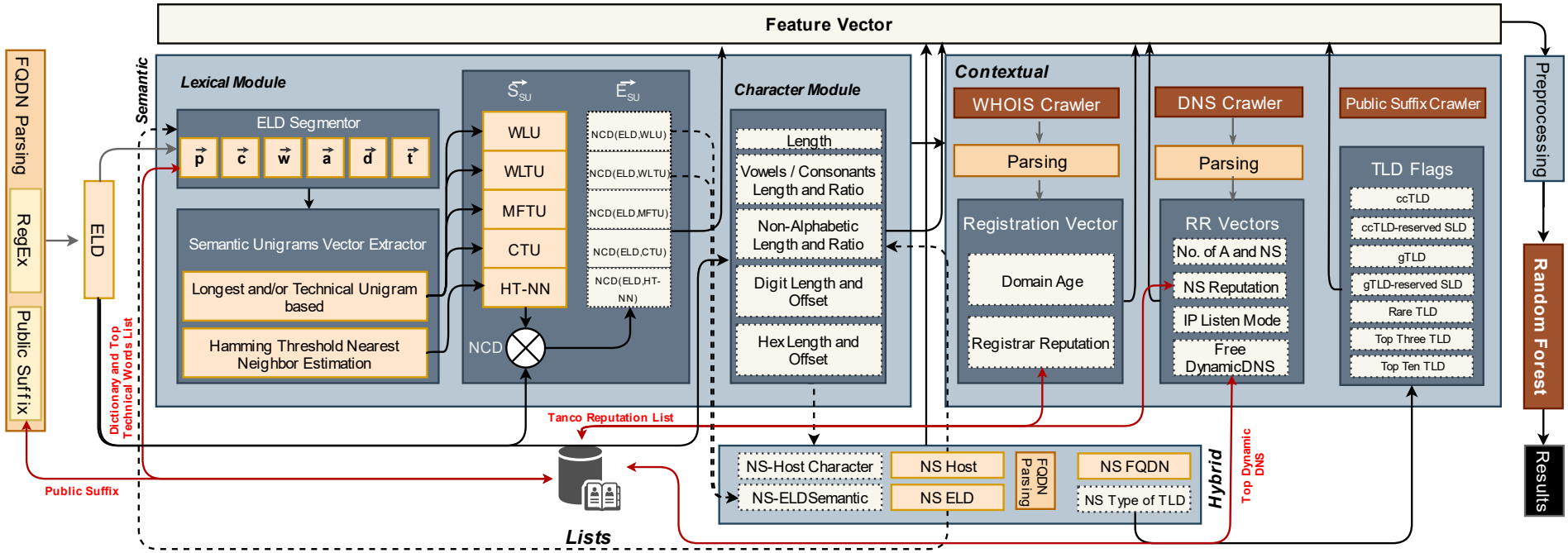


Figure 4.2: HAWKEYE data flow. $\vec{p}$, $\vec{c}$, $\vec{w}$,$\vec{a}$ $\vec{d}$ and $\vec{t}$ are the results of segmenting an ELD with respect to popular, common, worldwide, ASCII, English and technical vocabularies (see Section 4.3.3). Light-shaded boxes denote elements of the feature vector.

4.3 Semantic Features

Semantic features, such as the length of a domain, or its decomposition into meaningful parts, reflect the choice of a particular domain name. These aim to capture DGA-wordlist- and phishing-like techniques used by APTs, including typosquatting (`telagram[.]net`), TLD squatting (`microsoft[.]store`), and the use of technical words (`accounts-google[.]com`). A selection of the semantic features described below are summarised in Table 4.4.

4.3.1 ELD Identification

As a preliminary step, we split each FQDN into an *apex* domain (the registrable part) and a possibly empty *host* prefix. Next, we split the apex into its largest *public suffix* [187], and its *ELD*. The ELD represents the part of the domain name chosen by the entity owning and controlling it. For example, the Strider APT domain `ping.sideways[.]ru` has host `ping`, apex `sideways[.]ru`, and ELD `sideways`, whereas `mynetwork` is the ELD of the APT33 domain `mynetwork[.]ddns.net`. Identifying the ELD precisely, and in particular avoiding the pitfall of including dynamic DNS, public hosting providers or a reserved SLD, is important. Several APTs use dynamic DNS providers, such as `ddns.net` above, to bypass domain detection techniques which overlook this distinction. Moreover, one may want to extract character and lexical features such as the ones described below, or those used in related work (length, number of vowels, string similarity, etc.) only from the ELD, and not, for example, from the public suffix of a domain. Further examples of ELD identification are presented in Table 4.1.

4.3.2 Character Features

The length [188, 180, 189, 184] and the number of vowels [180] of an FQDN have proven to be useful features for malicious domain detection. However, in the area of APT, a high overlap of ELD length between APTs and legitimates is found, presented in Figure 4.3. Therefore, we propose further features, including the number of consonants, the ratio of vowels and consonants to length, and a boolean feature for whether or not a word starts with a vowel. We apply these six features to the apex and ELD. From the ELD, we also extract the total number of digits, the longest sequence of digits and the maximum digit offset, and similarly for hexadecimal strings

Label	FQDN	ELD	TLD
	I. gTLD		
APT41	apt41 [.]industries	apt41	[.]industries
APT28	irf [.]services	irf	[.]services
Silence	layout [.]network	layout	[.]network
DarkHydrus	Kaspersky [.]science	Kaspersky	[.] science
FIN7	ns1 [.]website	ns1	[.]website
Legitimate	fitznerblockchain [.]consulting	fitznerblockchain	[.]consulting
Legitimate	chirashi [.]marketing	chirashi	[.]marketing
	II. ccTLD		
Strider	gtf [.]cc	gtf	[.]cc
Strider	utc-wien [.]at	utc-wien	[.]at
Strider	ping [.]sideways [.]ru	sideways	[.]ru
Silence	others [.]in	others	[.]in
SilverTerrier	ArmyDepartment [.]us	ArmyDepartment	[.]us
SmokeScreen	fmchr [.]in	fmchr	[.]in
	III. SLD-gTLD		
APT2	kaspersky[.]firewall-gateway[.]net	kaspersky	[.]firewall -gateway [.]net
APT1	johnford985 [.]appspot [.]com	johnford985 [.]appspot [.]com	[.]appspot [.]com
APT16	carwiseplot [.]no-ip [.]org	carwiseplot [.]no-ip [.]org	[.]no-ip [.]org
Legitimate	insagirl-toto [.]appspot [.]com	insagirl-toto [.]appspot [.]com	[.]appspot [.]com
	IV. SLD-ccTLD		
APT28	mail [.]byegm [.]web [.]tr	byegm	[.]web [.]tr
APT29	acciaio [.]com [.]br	acciaio	[.]com [.]br
APT28	eposta [.]basbakanlik [.]qov [.]web [.]tr	qov	[.]web [.]tr
Legitimate	americamovil-mx-livechat -client-pro [.]azurewebsites [.]net	americamovil-mx-livechat-client-pro	[.]azurewebsites [.]net
Legitimate	olshelmore [.]catholic [.]edu [.]au	olshelmore	[.]catholic [.]edu [.]au
Legitimate	city [.]kawasaki [.]jp	city [.]kawasaki [.]jp	[.]kawasaki [.]jp
	Case Study for SLD-gTLD: Dynamic DNS with 3rd Party		
APT18	longshadow [.]dyndns [.]org	longshadow	[.]dyndns [.]org
APT35	mychannel [.]ddns [.]net	mychannel	[.]ddns [.]net
APT33	mynetwork [.]ddns [.]net	mynetwork	[.]ddns [.]net
FIN7	doddyfire [.]dyndns [.]org	doddyfire	[.]dyndns [.]org
Machete	derte [.]ddns [.]net	derte	[.]ddns [.]net
Legitimate	monobservatoire [.]ddns [.]net	monobservatoire	[.]ddns [.]net
Legitimate	kf2da [.]ddns [.]net	kf2da	[.]ddns [.]net
Legitimate	alsarh [.]dyndns [.]org	alsarh	[.]dyndns [.]org
Legitimate	infoquest [.]dyndns [.]org	infoquest	[.]dyndns [.]org

Table 4.1: HawkEye examples on ELD identification.

of even length [189]. We also record the percentage of digits in the ELD string [15]. Finally, we collect new features by counting the number and ratio of non-alphabetical characters in an ELD.

4.3.3 Lexical Features

Before extracting lexical features, we try to split a domain name (specifically, an ELD) into its constituent words, if possible.

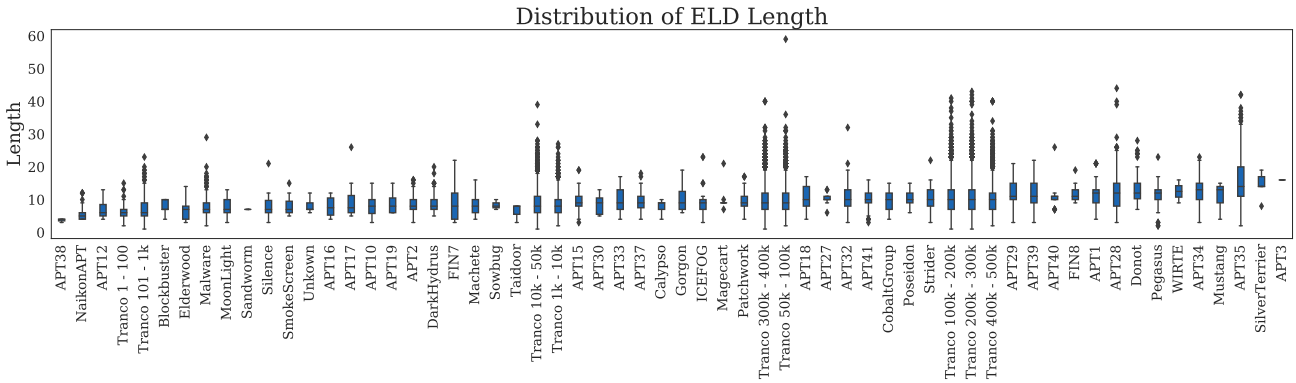


Figure 4.3: Campaigns are sorted by median length. APT and legitimate domain campaigns are interleaved. The upper and lower caps refer to the 90th and 10th percentile, whereas the upper and lower sides for each box represent the 75th and 25th percentile. Finally, the mid-line inside boxes are median values, and diamonds refer to outliers.

4.3.3.1 Vocabularies

In NLP, typical approaches to segment a sentence into tokens include dedicated APIs such as `nltk.tokenize` [190], or regular expressions with explicit word separators, such as whitespace, comma, colon, and period. However, many of the domains we consider concatenate multiple words together without any separator, and such techniques cannot be applied directly. Even previous work on word segmenting for English language OCR and spelling correction [191, 192] is ineffective in our case, because of the frequency of non-English and brand words in the APTs setting. To the best of our knowledge, there is no specific technique to segment domain names semantically.

Although the English language is commonly used for conveying meaning through the choice of a domain, also brands, technical words, and the ASCII representations of words in other languages are pervasive. It is trivial to identify `apple.com` as the word "apple", matching an English word, a brand name and a popular domain [193]. The case for `howstuffworks.com` is more subtle: it can be segmented as the popular domain "howstuffworks", or as the English words "how", "'stuff" and "works". Similarly, `yandexmail.ru` should be segmented as "yandex", an ASCII sequence denoting another popular domain, and the English word "mail".

Based on the intuition above, we collect a number of vocabularies as a basis for domain name segmentation: note that we do not use these directly to determine the maliciousness of a domain.

Set	ELD (.TLD)	Popular	Common	Domains ASCII	English-Dictionary	WLU
			I. Bitsquatting			
APT35	telagram ([.]net)	None	[t]	[tel, agr]	[lag, ram]	lagtelagrram
Legitimate	telegram ([.]org)	None	[t]	[telegram]	[telegram]	telegram
			II. DGA-Like Alexa-based			
APT34	egoogle ([.]org)	[google]	[google]	[ego]	[ego]	google
Legitimate	google ([.]org)	[google]	[google]	[google]	[go]	google
			IV. DGA-Like Dictionary-based			
ICEFOG	sportsnewsa ([.]net)	None	[rt]	[sports, news]	[sport, news, a]	sports
Legitimate	sportsmansoutdoorsuperstore ([.]com)	[so]	[rt, so, t, t]	[sports, mans, out, ors, rst]	[sport, man, so, door, super, store]	sportsuperstore
			V. DGA-Like Random Character			
APT28	message-id8665213 ([.]com)	None	[6, 6]	[mes, sage, -, 86, 6, 5, 21, 3]	[message]	message
Legitimate	xn—-8sbkahkuskl1n ([.]com)	None	None	[-, -, -, -, bka, hku, skl, 1]	[a, us]	bkahkuskl
			VI. Dynamic DNS With Third Party			
APT15	ensun ([.]dyndns[.]org)	None	None	[ens]	[sun]	sunens
Legitimate	erogamescape ([.]dyndns[.]org)	None	[ca]	[erogamescape]	[game, cape]	erogamescape
			VII. Technical Process			
APT28	accoounts-google ([.]com)	[google]	[t, google]	[acc, ooun, -, google]	[a, go]	google
Legitimate	googleaccountlogin ([.]com)	[google]	[google, t, in]	[google, acc, unt, logi]	[go, account, log, in]	account
			VII. Typosquatting			
APT34	miedafire ([.]com)	None	[ed]	[eda, fire]	[a, fire]	fire
Legitimate	mediafire ([.]com)	None	[mediafire]	[mediafire]	[media, fire]	mediafire
			II. TLD Squatting			
WIRTE	microsoft ([.]store)	[microsoft]	[microsoft]	[microsoft]	[soft]	microsoft
Legitimate	microsoft ([.]com)	[microsoft]	[microsoft]	[microsoft]	[soft]	microsoft
			IX. Phishing			
Patchwork	yahoomail ([.]pw|[.]support)	[yahoo, mail]	[yahoo, mail]	[yahoo, mail]	[a, mail]	yahoo
Legitimate	(mail[.]) yahoo ([.]com)	[yahoo]	[yahoo]	[yahoo]	[a]	yahoo

Table 4.2: Examples of APT ELD segmentations across different character and lexical evasion techniques.

Inspired by [194], who collect two lists with the 8 and 100 most popular domains, we build non-overlapping vocabularies with the ELDs of the **top 100 (popular), 1k (common),10K (worldwide) and 500k (ASCII)** Alexa domains. Our goal is to increase the coverage of brands, abbreviations and non-English-words, which are common in domains which may be targeted by phishing-like, typosquatting or similar impersonation attempts. We also build a vocabulary of 350 **technical terms**, which could be useful to an APT for impersonating a process or piece of infrastructure (as discussed in Section 4.1), including terms such as: *update, DNS, mail, support, account, CDN, API, cloud.* Finally, we build an **English vocabulary** of the 10k most common English words from the Cambridge dictionary. In Table 4.2, we present our segmentation methods for different evasion techniques used by APT.

4.3.3.2 Segmentation Features

In order to segment a target ELD with respect to a vocabulary, we collect all the ordered, non-overlapping matches against a regex which contains the vocabulary words in decreasing

length order (to favor longest matches). Once we have segmented an ELD, we join the list of matches, and we save the length of the resulting unigram, as well as its ratio to the ELD length as features. These features attempt to quantify how much of the ELD is semantically related to an entity of interest. For example, a fake ELD `yandexm4il-cdn` would be segmented by the union of common and technical vocabularies into the unigram `yandexcdn`, yielding a 65% match.

For completeness, we also mimic a lexical feature proposed in [15] by saving the length and ratio to the ELD of the longest substring of the ELD that has a match in the English vocabulary.

4.3.3.3 Unigrams and NCD$_e$ Features

Besides direct segmentation as described above, we propose five more flexible methods to select a target unigram for comparison with a given ELD. We then compute the *Entropy Normalized Compression Distance (NCD$_e$)* between the ELD and each unigram as proxies for similarity.

Consider that we have two strings S_1 and S_2, their concatenation S_1S_2, and a compressor C with C(S), a function returning the number of bytes of the compressed strings.

C(S_1S_2) will retain (nearly) the size in bytes as C(S_1) when $S_1 = S_2$. In other words, as S_2 resembles S_1, the compressor will encounter more redundancy, resulting in C(S_1S_2) bytes getting closer to the number of bytes of C(S_1). The following formula describes this process:

$$NCD_e(S_1, S_2) = \frac{C\big(S_1S_2\big) - min\{C(S_1), C(S_2)\}\big)}{max\{C(S_1), C(S_2)\}} \tag{4.1}$$

Such that $0 <= NCD_e(S_1, S_2) <= 1$, where 0 indicates the full match, and 1 is the exact opposite. In HawkEye, we use NCD$_e$ to produce features that help to identify domain various spoofing attempts described in Table 4.2.

Now, after segmenting the ELD with respect to the popular, common, worldwide, ASCII, English and technical vocabularies, the *Weighted Longest Unigram* (**WLU**) collects the longest unigrams, concatenating multiple unigrams if they have the same length. The WLU preserves the order of match from the ELD and does not include overlapping matches. Concrete examples of this unigram and the ones below are provided in Table 4.3. The *Weighted Longest Technical Unigram* (**WLTU**) instead is a restricted form of WLU using only the technical vocabulary.

The *Most Frequent Technical Unigram* (**MFTU**) also only considers technical terms, but the segmentation is based on a re-sorting of the vocabulary according to the frequency of each term in the training set, cutting the vocabulary at a threshold τ. The *Concatenated Technical Unigram* (**CTU**) is a variant of the MFTU that does not consider frequency. Algorithm 1 describes the **WLU** and the other variants. $\vec{S}_{SU}$ and S_{fqdn} refer to a semantic unigram vector and a string of FQDN, respectively. M refers to the matrix which includes all features in the dataset. Flag 1 and Flag 2 store the longest values. $\vec{S}_{\omega}$ refers to a vector that includes the current segmented ELD lexicon based on a particular set used for segmentation with keeping their index, i.e. feature name.

Campaign	ELD.SUFFIX	ELD Segmentation Vectors			Unigram Extractions			ELD Neighbor	
		$\vec{p}$	$\vec{a}$	$\vec{t}$	WLU	MFTU	CTU	HT-NN	h
APT34	go0gle[.]com	None	None	None	gole	go	None	google	1
APT28	n0vinite[.]com	None	[vin, ite]	None	vinite	None	None	novinite	1
APT18	google-ana1ytics[.]com	[google]	[google, -, ana, 1, tic,]	[google]	google	google	google	google-analytics	1
APT35	facebook[.]in	None	[face, boe]	None	face	None	None	facebook	1
APT28	euronews24[.]info	None	[euronews, 24]	None	euronews	None	None	euronews21	1
DarkHydrus	hotmai1[.]com	None	None	None	hot*	None	None	hotmail	1
Pegasus	foxlove[.]life	None	[love]	None	love	None	None	foxlive	1
APT34	go0gle[.]com	None	None	None	gole	None	None	google	1
APT41	kasparsky[.]net	None	[kas, pars]	None	pars	None	None	kaspersky	1
APT1	newsesport[.]com	None	[news, espo]	None	sport	None	None	newsreport	2
APT28	updatecenter[.]name	None	[update, center]	[update, enter]	updatecenter	update	updateenter	irdatacenter	3
APT35	gmail-com[.]xyz	[mail]	[gmail]	[gmail]	gmail	mail	gmail	email-hog	3
APT15	englishedu-online[.]com	None	[english, u-on, line]	[online]	english	None	online	englishhelponline	4
APT37	securytingmail[.]com	[mail]	[securyti, gmail]	[gmail]	securyti	mail	gmail	security4arabs	7
APT39	update-microsoft[.]space	[microsoft]	[update, microsoft]	[update, microsoft]	microsoft	updatemicrosoft	updatemicrosoft	updatesmartphone	8
APT28	office365-onedrive[.]com	[office]	[office, 36, 5-, one, drive]	[office]	office	office	office	office365wholesale	8
APT32	googleuserscontent[.]org	[google]	[google, user, cont]	[google, user, content]	content	googleuser	googleusercontent	localguidesconnect	9
APT18	microsoft-outlook[.]org	[microsoft, ok]	[microsoft, -, outlook]	[microsoft, outlook]	microsoft	microsoftoutlook	microsoftoutlook	configuraroutlook	10
APT28	adobe-flash-updates[.]org	[adobe]	[adobe, -, flash, -, upd, ates]	[adobe, flash, update]	update	update	adobeflashupdate	wholesaleautoplates	11
FIN7	windowsupdatemicrosoft[.]com	[microsoft]	[windows, update, microsoft]	[windows, update, microsoft]	microsoft	windowsupdatemicrosoft	windowsupdatemicrosoft	windowsactivatorloader	14

* In this example, **WLU** selects only the unigram "hot" from the English dictionary ELD segmentation vector $\vec{d}$, not reported in the table due to space limitations.

Table 4.3: APT domain segmentation: examples from the Hawk-Eye dataset.

The last unigram we consider is the *Hamming Threshold Nearest Neighbor* (**HT-NN**), which consists in the ELD from the ASCII vocabulary which is closest in the Hamming distance to the ELD of the FQDN being classified, such that:

$$\arg\min_{\alpha} HT - NN = \alpha(ELD^{Imitating}, ELD^{Input}) \tag{4.2}$$

where α refers to the minimum threshold as a distance to a legitimate domain. In Algorithm 2, we describe the hamming threshold nearest neighbour estimation on malicious names according to their 1-hamming distance to the top reputed domain while maintaining the same length. Besides collecting NCD_e between the ELD and the HT-NN, we also collect the Hamming distance itself as a feature. This captures the intuition that APT operators may choose domain names that are subtle alterations of existing ones, such as replacing one character with a similar-looking one. For example, a Hamming distance of 1 captures the APT domain `go0gle[.]com`, which impersonated `google.com`.

4.4 Contextual and Hybrid Features

Contextual features are orthogonal to semantic features, and express the organization of the DNS infrastructure that supports the C&C operation for the chosen domain. We expect the low-and-slow nature of APTs to differentiate their infrastructure from that of generic malware, or regular domains. *Hybrid* features collect semantic features from the DNS infrastructure. Selected contextual and hybrid features are reported in Table 4.4.

4.4.1 Contextual Features

We observed in Section 4.1 that APT C&C behaviour involves other aspects of the DNS infrastructure besides the choice of an ELD. Hence, we proceed to consider orthogonal features, based on the domain infrastructure used by APTs.

Algorithm 1: Generating semantic unigrams vectors.

Result: $\vec{S}^{SU}$

def *WeightedLongest(technique: str, SEV: list)*:

Initial($length, flag1, flag2$);
$j = 0$;
while $j \neq len(SEV)$ **do**
  if $|SEV_j| > length$ **then**
    $length \leftarrow |SEV_j|$;
    $technique \leftarrow |SEV_j|$;
    $flag_1 \leftarrow SEV_j|$;
  else if $\big(|SEV_j| == length\big) \wedge \big(|SEV_j| \neq (flag_1 \vee flag_2) \big)$ **then**
    $technique \leftarrow technique + |SEV_j|$;
    $flag_2 \leftarrow |SEV_j|$;
  $j+ = 1$;
end
return technique;

def *UnigramsInclusion(TU: STR, SEV: list)*:

Initial(TU);
$j = 0$;
while $j \neq len(SEV)$ **do**
  $TU \leftarrow TU + SEV_j$;
  $j+ = 1$;
end
return TU;

Start: $M^{ELD} \leftarrow$ fqdn_segmentation(S_{fqdn});
$i = 0$;
$\vec{MFT}$=histogram($M^{ELD}_{train\_set}$,hist_threshold);
while $i \neq len(M)$ **do**
  Initial($length, WLU, WLUT, CTU, MFTU, flag_1, flag_2$);
  $\text{sev}_i = \{\text{M}_i^{\vec{p_i}}, M_i^{\vec{c_i}}, M_i^{\vec{w_i}}, M_i^{\vec{d_i}}, M_i^{\vec{t_i}}\}$;
  $S_i^{\omega} \leftarrow \bigcup_{sidx=1}^{len(sev_i^{sidx})} sev_i^{sidx}$;
  $S_{WLU}^{SU} \leftarrow WeightedLongest(WLU, S_i^{\omega})$;
  $S_i^{\omega} \leftarrow S_i^{\omega} - \bigcup_{sidx=1}^{len(sev_i^{sidx}-1)} sev_i^{sidx}$;
  $S_{WLUT}^{SU} \leftarrow WeightedLongest(WLUT, S_i^{\omega})$;
  $S_{CTU}^{SU} \leftarrow UnigramsInclusion(CTU, S_i^{\omega})$;
  $S_i^{\omega} \leftarrow Segmentation(ELD, MFT)$;
  $S_{MFTU}^{SU} \leftarrow UnigramsInclusion(MFTU, S_i^{\omega})$;
  $M_i^{S_{SU}} \leftarrow S_i^{SU}$;
  $i+ = 1$;
end

Algorithm 2: Hamming threshold nearest neighbor semantic estimation.

Result: $\arg\min_{\alpha} f(x) := \alpha(ELD^{Imitating}, ELD^{Input})$
def *MatchingLength(ELD: STR)*:
    j=0;
    while $j \neq len(M^R)$ **do**
        if $|M_j^R| == |ELD|$ **then**
            $ELDList = ELDList.append(M_j^{ELD})$
        end
    end
    return $ELDList$;
def *MinimumHammingDist(ELD: STR,ELDList: list, α: int)*:
    j=0;
    while $j \neq len(ELDList)$ **do**
        if $hamming(ELD, ELDList_j) == \alpha$ **then**
            $NNHT = ELDList_j$;
            break;
        end
        j+=1;
    end
    return $NNHT$;
Start: $M^{ELD} \leftarrow$ fqdn_segmentation(S_{fqdn});
$i = 0$;
while $i \neq len(M)$ **do**
    Initial($ELD^{Imitating}, ELD\_LIST$);
    $ELD\_LIST = MatchingLength(M_i^{eld})$;
    $ELD^{Imitating} = MinimumHammingDist(M_i^{eld}, ELD\_LIST, 1)$;
    $M_i^{NNHT} \leftarrow ELD^{Imitating}$;
    $i+ = 1$;
end

4.4.1.1 Domain Suffix

As discussed in Section 4.3.1, we hypothesize that the type of TLD used by a domain, and in particular the presence of a public SLD denoting virtual hosting or dynamic DNS may be relevant to detect C&C domains. Hence, we propose four boolean features recording the kind of public suffix of a domain: ccTLD, gTLD, ccTLD, and gTLD with a reserved SLD. In addition, we propose three features, recording if a public suffix belongs to the top three TLDs (`.com`, `.net`, `.org`), top ten TLDs or bottom 100 TLDs based on the frequency observed in the training fraction of our dataset.

4.4.1.2 Domain Age

WHOIS servers are responsible for providing domain registration information for public use, including the owner name, primary nameserver, admin email, registrar, creation and expiry dates. WHOIS information has been used to detect malicious domains in the past, for example, in [184, 188, 195]. However, recent increases in privacy regulation and concerns from the users have led WHOIS servers to severely restrict the amount of information divulged, therefore several features that have been used in the past are no longer widely available. The domain age is computed as the difference between the expiration and creation date for a given domain, expressed in months [188]. This feature is present in the historical dataset and is also currently available when querying up-to-date WHOIS information.

4.4.1.3 Registrar Reputation

This feature approximates the reputation of the Registrar URL parsed from a WHOIS file by its ranking in the Tranco List [193]. We used Tranco instead of Alexa as the latter has been found vulnerable to poisoning [193].

4.4.1.4 DNS Resource Records

In Section 4.1.3, we discussed some examples of DNS abuse by APTs. Here, we consider related DNS features relevant to APT C&C domain classification. The `A` resource record is used to communicate the IPv4 addresses a domain resolves to. The `NS` record, for communicating the nameservers storing the zone file for the domain. We use, as features, the count of `A` and `NS` entries found in the respective responses for a candidate domain being classified.

4.4.1.5 Nameserver Reputation

This feature reports the Tranco-based reputation of the apex of the `NS` of a domain. The idea is that some APTs will not be able to meet the stringent anti-fraud requirements of highly ranked DNS providers, and resort to less popular ones. However, several APT campaigns are able to comply, and have used, for example, `domaincontrol.com` by GoDaddy.

4.4.1.6 Use of Free Dynamic DNS

Dynamic DNS (DDNS) is an approach to update the mapping of a domain to different IPs quickly and automatically. A known technique for domain flux [78, 196] is to use *Free DDNS Hosting* for malicious purposes, where the ELD is controlled by the attacker but the reserved SLD refers to the provider itself [197, 194, 78, 196]. Several APT campaigns, such as APT32, APT 37, APT 41, FIN7 and SilverTerrier [198], instead abuse DDNS in a different way, by directly delegating the nameserver of a C&C domain to a free DDNS provider. Therefore, we collect a list of the top 40 free DDNS providers and extract a boolean feature to record if the `NS` of a domain is in the list.

4.4.1.7 IP Listen Mode

Some APT campaigns configure their C&C domain to be resolved to the *loopback address* (`127.0.0.1`) or to a *non-routable meta-address* (`0.0.0.0`) [199] in order to let the victim connect to an attacker-controlled processes listening on the same machine, on a specific open port. We extract a feature recording if the `A` of a domain is internal or non-routable.

4.4.2 Hybrid Features

The Hybrid feature set consists of semantic features extracted from entities retrieved as part of the contextual analysis of a domain. For the ELD of a `NS`, we add lexical features similar to the ones described in Section 4.3, although we only use WLU and WLTU to identify the longest (technical) unigrams for computing the NCD_e. Finally, we add character features, including consonant length, non-alphabetic length, max digit and hex length, digit and hex offset for both the ELD and the Host of the `NS`.

The Semantic, Contextual and Hybrid features reported in Table 4.4 are combined as the *Holistic* feature set.

ID	Feature	Type	New?	ID	Feature	Type	New?
			I. Semantic Features				
1	Apex consonants len.	N	✓	14	ELD max digit len.	N	[189]
2	Apex consonants ratio	N	✓	15	ELD max digit offset	N	[189]
3	Apex len.	N	[8, 10, 14, 23, 34]	16	ELD Hex len.	N	[189]
4	Apex vowels len.	N	[180]	17	ELD max Hex offset	N	[189]
5	Apex vowels ratio	N	[180]	18	Longest meaningful string	N	[15]
6	Apex start with vowels	B	[180]	19	NCD_e(ELD, CTU)	N	✓
7	ELD's WLU len.	N	✓	20	NCD_e(ELD, HT-NN)	N	✓
8	ELD's WLTU len.	N	✓	21	NCD_e(ELD, MFTU)	N	✓
9	ELD's WLU pct	N	✓	22	NCD_e(ELD, WLTU)	N	✓
10	ELD's WLTU pct	N	✓	23	NCD_e(ELD, WLU)	N	✓
11	ELD non-alphabetic len.	N	[180, 189]	24	Technical words pct	N	✓
12	ELD non-alphabetic ratio	N	[180]	25	Top 1k pct	N	✓
13	ELD digit len.	N	[180]	26	Top 10k pct	N	✓
			II. Contextual Features				
27	Domain age	N	[189, 78, 200]	34	FQDN's Rare TLD	B	✓
28	Free Dynamic DNS-NS	B	✓	35	IP listen mode	N	✓
29	FQDN's ccTLD	B	[184, 78]	36	No. of IP	N	[200]
30	FQDN's gTLD	B	[184, 78]	37	No. of NS	N	[184, 189]
31	FQDN's reserved SLD	B	✓	38	NS reputation	N	[189]
32	FQDN's Top 3 TLD	B	✓	39	Registrar reputation	N	[189]
33	FQDN's Top 10 TLD	B	✓				
			III. Hybrid Features				
40	NCD_e(NS-ELD, NS-WLU)	N	✓	57	NS-Host max Hex offset	N	✓
41	NCD_e(NS-ELD, NS-WLTU)	N	✓	58	NS-Host consonants len.	N	✓
42	NS-ELD WLU len.	N	✓	59	NS-Host consonants ratio	N	✓
43	NS-ELD WLU pct.	N	✓	60	NS-Host non-alphabetic len.	N	✓
44	NS-ELD WLTU len.	N	✓	61	NS-Host non-alphabetic ratio	N	✓
45	NS-ELD WLTU pct.	N	✓	62	NS-Host startwith vowels	B	✓
46	NS-ELD digit len.	N	✓	63	NS-Host vowels num	N	✓
47	NS-ELD max digit len.	N	✓	64	NS-Host vowels ratio	N	✓
48	NS-ELD max digit offset	N	✓	65	NS len	N	[184]
49	NS-ELD max Hex len.	N	✓	66	NS-rareTLD	B	✓
50	NS-ELD max Hex offset	N	✓	67	NS-ccTLD	B	✓
51	NS-ELD non-alphabetic len.	N	✓	68	NS-gTLD	B	✓
52	NS-ELD non-alphabetic ratio	N	✓	69	NS Top 3 TLD NS	B	✓
53	NS-Host digit len	N	✓	70	NS Top 10 TLD	B	✓
54	NS-Host max digit len.	N	✓				
55	NS-Host max digit offset	N	✓				
56	NS-Host max Hex len.	N	✓				

Table 4.4: HawkEye features (N: numerical, B: boolean).

4.5 Dataset

4.5.1 Data Collection

As part of our analysis of APT campaigns from Section 4.1 we collected 5687 FQDNs and 2984 unique domains, which were used as part of APT C&C infrastructure. After the quality control described in Section 4.5.3, we kept 3894 FQDNs and 1894 domains. For each sample, we collected metadata recording the specific APT campaign ID, the period of activity, the report source, and other relevant data such as the IP addresses of C&C servers. We omitted domains that are not explicitly attributed to C&C, such as the ones used as part of an initial

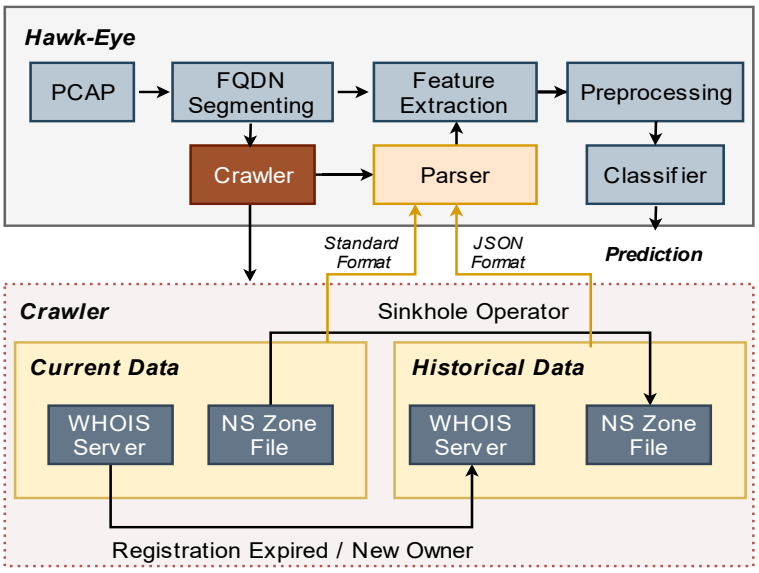


Figure 4.4: Hawk-Eye can also retrieve the historical data for a domain for further investigations.

compromise phase via spear phishing. Some APT reports provide ranges of malicious domains: in such cases, we collected only one concrete example to limit redundancy.

As a primary source of comparison against APT domains, in November 2019, we collected 47k legitimate domains randomly selected from the Tranco 1M list following a distribution heavily skewed towards the highest-ranking sites. We collected one order of magnitude more legitimate domains than APTs in order to support the unbalanced training of classifiers. Although it is easy to collect legitimate domains, we did not attempt to reflect the actual frequency of APTs in typical traffic, which is unknown and likely to be vanishingly small. Instead of a campaign ID, we assign to each legitimate domain a label representing its ranking bracket, such as `Alexa 101-1000`. To further compare APT domains against different malicious domains which may use related evasion techniques, in July 2020, we collected 6804 phishing FQDNs with 2523 unique domains labelled as valid and active from PhishTank [201].

In our experiments, we use three combinations of these datasets: *Hawk-Eye APT and Legitimate* (**HEAL**), *Hawk-Eye APT and Phishing* (**HEAP**), and *Hawk-Eye APT, Legitimate and Phishing* (**HEALP**).

4.5.2 Challenges

For malicious domains, we tried to obtain the features relative to the time when their respective campaigns were active, by querying the SecurityTrail API [185] for historical records, as current live data may not reflect the actual domain ownership and configuration whilst malicious. Several factors contribute to complicate the analysis of historical data, which is crucial to obtain reliable features. Some malicious domains were kept online by a sinkhole operator for more than 3000 days [202]. Others were taken down and subsequently released and repurchased for legitimate or malicious purposes [202]. While inspecting historical DNS data, we found that most malicious domains have tens of `A`, `NS`, `MX`, `SOA` and `TXT` records. For instance, the domain `ceofanb18.mipropia[.]com` used by Machete [203] has more than 600 nameserver configurations since December 2017, whereas `corporatefaxsolutions[.]com` used by Carbank resolved to 433 different IP addresses since July 2015.

In order to zoom in on the most likely attack configuration, we select historical data in a given APT campaign time window, and we prioritize matches for `NS` reported in threat intelligence feeds for the same campaign or an overlapping one. For example, that leads to choosing `ns1621.ztomy[.]com`, `ns2621.ztomy[.]com` and `brit.ns.cloudfronts[.]services` for DarkHydrus APT. This is a manual and time-consuming process, due to the insufficient analysis of NS records in the published reports. Once we identify the validity window of the nameserver used during the campaign, we are able to filter relevant `A` records within that time interval. If we identify several records with different dates during an interval, we select the ones closest to the date the domain was reported. We proceed in a similar way for historical WHOIS records.

4.5.3 Missing Values

We have adopted a rigorous methodology to handle missing data in our datasets. In order to reduce the bias introduced by imputation, we omit features for which at least 20% of the values are missing. For example, WHOIS features such as registrant email, country, etc adopted in [189] are missing from up to 40% of the data, due to the adoption of new privacy rules on WHOIS files for gTLD domains, and because of ccTLD WHOIS servers that choose not to report that information to the public. We also removed features related to `CNAME`, `MX`, `SOA` and `TXT` which were missing from the majority of legitimate and malicious domains. For some

FQDN, we could not find any nameserver, although the domains were not taken down. In such cases, we assumed the APT campaign used dynamic DNS services during the attack for domain resolution, and then deleted the records for anonymity. Recent domains tend to lack historical data to capture such behaviour, yet we noticed several domains using this technique. In a similar way, we excluded from our dataset entire campaigns (FIN 8, APT 38 and others) where every single domain missed at least one feature.

4.6 Results

We now evaluate and discuss the detection performance of HawkEye, using the feature sets defined in Sections 4.3 and 4.4. Since there is no publicly available tool that detects APT C&C domains, we cannot compare our approach with an accepted detection baseline. Instead, we used established features from the literature (those with a citation in Table 4.4) to create an additional *Literature Baseline* feature set for comparison. We compare the various feature sets on three detection tasks: APT versus legitimate (HEAL), APT versus phishing (HEAP), and APT versus non-APT, that is, both legitimate and phishing (HEALP). While the classes of the HEAP dataset are mostly balanced, APT domains are 4% of the samples in the HEAL and HEALP datasets. Hence, the results reported in Table 4.5 and 4.7 are the *weighted average* across the two classes, approximating the performance on a balanced dataset. We also report the macro F1 score ($\boldsymbol{mF}1$), which reflects the existing 4%/96% bias.

We also present two different configurations for our experiments, which are known and unseen APT campaigns. The former is the typical configuration to test the machine learning-based classifiers, while the latter mimics the challenge of the real-world setting. A comparison between the two configurations may help evaluate the generalization and robustness of our features. Then, we focus on the performance of the unknown APT campaign classification and analyze the feature importance of the three datasets.

4.6.1 Linear Transformation Test

One of the most effective feature-reduction techniques that could reduce training time is obtaining a set of principle variables. This is a powerful method for summarizing a high number

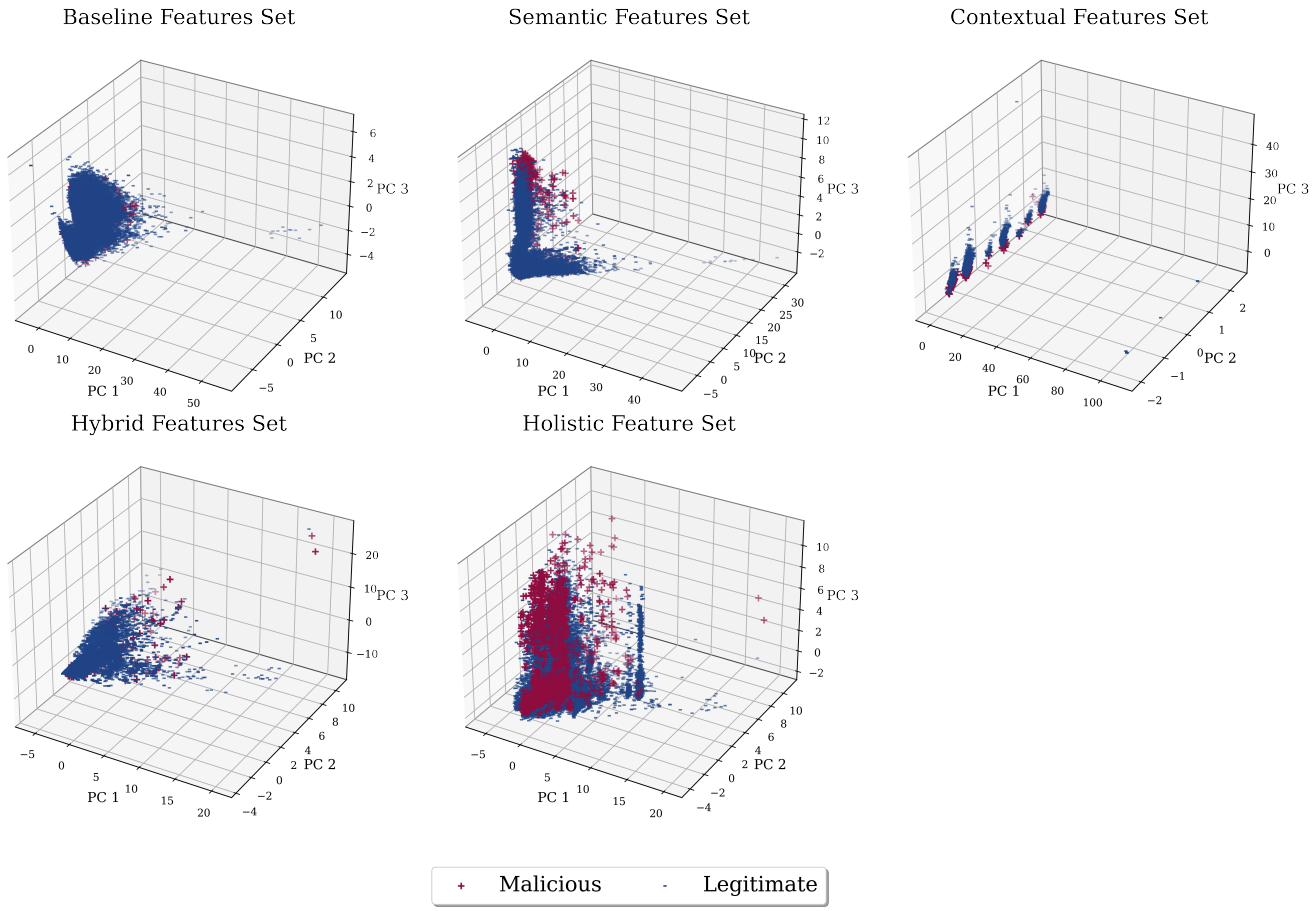


Figure 4.5: PCA on the training set shows all feature sets cannot be separated linearly. However, the Holistic Feature Set is promising.

of features that correlate linearly into the most significant information, typically two or three eigenvectors (principal components). PCA can be useful if the problem can be separated linearly. If this is achieved, several ML models can be useful, including k-NN and SVM, with low training time.

In Figure 4.5, the PCA analysis of several features based on their categories shows that APT is highly overlapped with legitimate domains. This confirms our need to use all categories and random forests instead of those ML models that search for clear decision lines, as we discussed in Chapter 2.

4.6.2 Known APT Campaigns Classification Performance

The best overall performance of HawkEye is obtained by the Holistic features set, with respectively 98.98%, 90.26% and 98.82% $wF1$ for the HEAL, HEAP and HEALP datasets. Due

Classifier Name	*Macro*			*Weighted*				
	mP	*mR*	*mF*1	*wP*	*wR*	*wF*1	*FPR*	*A*
I. Dataset: HEAL								
HawkEye-Semantic	89.73	69.18	75.55	96.68	97.03	96.51	0.38	97.03
HawkEye-Contextual	94.55	80.90	86.37	98.01	98.11	97.94	0.28	98.11
HawkEye-Hybrid	93.88	84.33	88.47	98.23	98.32	98.22	0.38	98.32
HawkEye-Holistic	**96.24**	**91.28**	**93.60**	**98.98**	**99.01**	**98.98**	**0.27**	**99.01**
Literature Baseline	91.93	76.47	82.29	97.44	97.63	97.37	0.39	97.63
II. Dataset: HEAP								
HawkEye-Semantic	74.88	74.44	74.58	74.98	75.04	74.92	19.68	75.04
HawkEye-Contextual	83.31	83.28	83.30	83.47	83.48	83.48	14.80	83.48
HawkEye-Hybrid	86.34	86.29	86.31	86.46	86.46	86.46	12.02	86.46
HawkEye-Holistic	**90.09**	**90.32**	**90.18**	**90.33**	**90.25**	**90.26**	**10.44**	**90.25**
Literature Baseline	82.21	82.17	82.19	82.38	82.39	82.38	15.72	82.39
III. Dataset: HEALP								
HawkEye-Semantic	86.33	62.95	68.64	96.10	96.67	95.88	0.36	96.67
HawkEye-Contextual	92.69	79.31	84.65	97.84	97.98	97.79	0.37	97.98
HawkEye-Hybrid	93.04	83.19	87.42	98.16	98.26	98.15	0.40	98.26
HawkEye-Holistic	**95.12**	**89.62**	**92.18**	**98.81**	**98.86**	**98.82**	**0.33**	**98.86**
Literature Baseline	89.15	74.94	80.33	97.22	97.49	97.21	0.52	97.49

Table 4.5: Known APT campaigns classification performance.

to a large number of legitimate samples in our dataset, which is the case in the real world, *wF*1 difference between sets and the baseline is slightly small. As discussed in Chapter 2, SOC engineers focus on *mF*1 to reflect the real performance against attacks. HawkEye with Holistic feature set shows a larger difference to its benefit. *mF*1 records at 93.60%, 90.18% and 92.18 with Δ compared to the baseline 11.31%, 7.99%, and 11.85% for the three datasets, resepectively.

In HEALP, the Hybrid features set performs the top among other individual sets, including the baseline with 88.74%, 86.31%, and 87.42% of *mF*1 for the three datasets. The Contextual feature set slightly performs lower than the Hybrid features set, with 81.83%, 80.79%, and 80.83% of *mF*1. We will revise the stability of the Hybrid and the Contextual feature against the unknown APT campaign in the next section. The Semantic features set obtains *mF*1 of 74.99%, 73.88%, and 67.55%. However, its FPR of 0.41% is the lowest, which shows that they can recognize the legitimate domain better than other sets. This can be confirmed when semantic features struggle in HEAP, where the classifier confuses between the APT and phishing domain, where there is an overlap of evasion techniques used in the semantic perspective. The Semantic feature contributes to the Holistic one by reducing the FPR and better recognising the legitimate samples.

4.6.3 Unseen APT Campaigns Classification Performance

In this experiment, we are interested in testing HawkEye against unseen APT campaigns to test the generalization and robustness of our features against ones. This can help identify if a set of features, such as contextual or hybrid, can be robust, e.g., different domain ages or lexical analysis of their nameservers. In the context of known APT campaigns (Section 4.6.2), if the campaign started in November 2018, many of these domains may fall under the same age group, which facilitates the classifier to distinguish them from legitimate ones. However, if a new campaign establishes its DNS infrastructure in September 2023, it may confuse the classifier. A similar scenario may arise for hybrid features when the same campaign generates new domains with the same nameserver. The lexical analysis of the nameserver is more likely to detect the new domains but for the same APT campaigns. In Table 4.6, we report the unseen APT campaigns experiment with an approximate split of 70% APT samples for training and 30% for testing, which we mirrored for the other classes.

Training Sets		**Testing Set**	
APT1 (CommentCrew)	APT40 (Leviathan)	AnimalFarm APT	Higaisa
APT12 (IXESHE)	APT41 (DoubleDragon)	APT Big Bang	ICEFOG APT
APT15 (ke3chang)	APT Robotic	APT C37	KONNI APT
APT16 (EPS)	LazarusGroup	APT C39	Microcin APT
APT17 (Deputy Dog)	Calypso APT	APT CYBEC TIA	MuddyWater APT
APT18 (Wekby)	CobaltGroup (CobaltSpider)	APT Pirate Panda	Mustang Panda APT
APT19 (C0d0so0	DarkHydrus	APT10 (menuPass)	Operation Transparent
APT2 (PutterPanda	Elderwood (OperationAurora)	APT23	Pierogi APT
APT27 (EmissaryPanda)	FIN7 (Carbanak)	APT34 (OilReg)	Scarlet (Mimic APT)
APT28 (SednitSofacy)	Machete El (Machete)	APT35 (MajicHound)	StrongPity3 APT
APT29 (CozyDuke)	NaikonAPT (MsnMM)	BITTER APT	Trident APT
APT3 (Gothic)	Patchwork	Chinese mAPT	Turla APT
APT30	Pegasus	Deep Panda APT	
APT32 (OceanLotus)	Poseidon Group	Donot APT	
APT33 (Elfin, Shamoon)	Sowbug	Enfal APT	
APT37 (ScarCruft)	Strider (PROJECTSAURON)	Etirehni APT	
APT39 (Chafer)	Taidoor	Gamaredon	

Table 4.6: Training/testing split for the unseen APT campaigns experiment.

In this experiment, the best overall performance of HawkEye is obtained by the Holistic features set, with respectively 90.38%, 88.56%, and 89.15% $mF1$ for the HEAL, HEAP and HEALP datasets. The baseline performs at 76.81%, 79.06%, and 73.05% $mF1$. Therefore, the Δ between $mF1$ of the Holistic feature set and the baseline are 13.57%, 9.5% and 16.10%. As expected, all classifiers performances are dropped against unseen APT campaigns. However, the highest drop is found in the baseline presented by Δ when compared with the Holistic features set. It proves that holistic is more stable for the known and unseen compared to the literature.

While semantic features are the least susceptible for new campaigns, hybrid features are the

Classifier Name	*Macro*			*Weighted*				
	mP	*mR*	*mF*1	*wP*	*wR*	*wF*1	*FPR*	*A*
I. Dataset: HEAL								
HawkEye-Semantic	88.85	68.76	74.99	96.58	96.96	96.43	0.41	96.96
HawkEye-Contextual	93.19	75.46	81.83	97.47	97.63	97.33	0.30	97.63
HawkEye-Hybrid	88.00	77.68	81.95	97.21	97.45	97.25	0.71	97.45
HawkEye-Holistic	**93.32**	**87.85**	**90.38**	**98.46**	**98.53**	**98.48**	**0.48**	**98.53**
Literature Baseline	85.89	71.62	76.81	96.55	96.95	96.58	0.68	96.95
II. Dataset: HEAP								
HawkEye-Semantic	74.21	73.74	73.88	74.30	74.36	74.24	20.11	74.36
HawkEye-Contextual	80.76	81.07	80.79	81.21	80.86	80.91	21.00	80.86
HawkEye-Hybrid	85.24	84.45	84.70	85.11	85.01	84.92	10.04	85.01
HawkEye-Holistic	**88.48**	**88.69**	**88.56**	**88.72**	**88.65**	**88.66**	**11.76**	**88.65**
Literature Baseline	79.31	78.92	79.06	79.37	79.40	79.34	16.38	79.40
III. Dataset: HEALP								
HawkEye-Semantic	84.37	62.19	67.55	95.89	96.56	95.74	0.41	96.56
HawkEye-Contextual	92.15	74.55	80.83	97.43	97.63	97.32	0.33	97.63
HawkEye-Hybrid	90.03	77.94	82.81	97.52	97.73	97.52	0.53	97.73
HawkEye-Holistic	**94.16**	**85.24**	**89.15**	**98.40**	**98.48**	**98.40**	**0.35**	**98.48**
Literature Baseline	83.30	67.89	73.05	96.23	96.77	96.28	0.69	96.77

Table 4.7: Unseen APT campaigns classification performance.

highest. Semantic features exploit the nature of the character and lexical analysis of a domain which is unlikely to be significantly drifting in the future. Hybrid features can benefit from the known campaigns based on the shared character and lexical analysis of the shared nameservers. Nevertheless, hybrid still outperforms semantic performance, even for unknown campaigns. One reason is that several campaigns still use weak reputations of nameservers or dynamic DNS.

We found that training on the current campaigns and detecting their upcoming attacks reduce the false positives for all sets. This can be helpful for an enterprise whenever they discover they are under a specific or few APT campaign(s) attacks. A SOC analyst may only need to push the current IoC to Hawk-Eye for the training set, so the new APT domains will be under the classifier's radar. Now, since the performance on unseen APT campaigns is also reasonable, it can also transform unseen APT campaigns into known ones. For instance, when a new campaign is discovered through Hawk-Eye, a SOC analyst may collect the newly detected APT domains and push them again to the training set. In this way, the FPR would decline gradually through time. This process can be easily automated without any human intervention. However, it could lead to poisoning attacks, meaning those false positive domains may confuse the classifier over time. Therefore, we recommend verifying the new malicious domains flagged by Hawk-Eye manually from time to time. We leave the automated process for future work.

4.6.4 Feature Importance

In Figure 4.6, we plot the feature importance for the Holistic classifier and for the literature baseline. Different features turn out to have different importance for different tasks. We focus our analysis on the HEALP dataset of Unseen APT campaigns, which, including both APT, legitimate and phishing domains, is the most relevant to detecting APT domains in the wild.

For convenience, we divide features into three groups based on importance thresholds of 0.10, 0.05 and 0.025. The first group only contains two features. The first is Domain age (#27), which is known to be an effective feature to detect malicious domains [189, 78, 200], and remains relevant for APTs. The second is $\text{NCD}_e(ELD, WLU)$ (#23), one of the new features we propose, with mean and standard deviation for APT 0.136 ($\pm$0.087), phishing 0.121 ($\pm$0.01), and legitimate 0.047 ($\pm$0.077). It captures the well-known fact that some APT and phishing domains attempt to resemble popular ones without exactly matching them. In the next group, 34.46 %, 1.99%, and 2.05% of APT, phishing and legitimate domains use free DDNS-NS (#28), respectively. We also observed that some of these nameservers have no reputation (#38) based on appearance in the Tranco list: 23.93 % for APTs, 25% for phishing and 14% for legitimate. Finally, the average number of IPs (#36) used by APTs is 1.07 ($\pm$0.40), phishing 1.78 ($\pm$1.63) and legitimate 2.25 ($\pm$2.67). The importance of these novel features confirms our hypotheses from Sections 4.3 and 4.4.

The third group contains a number of features that still contribute to classification, but that may be effective only for a smaller subset of domains. We verified that performance degrades by omitting each of these features. They include registrar reputation (#39) and variants of NCD_e using *MFTU, HT-NN* and *NS-WLTU, NS-WLU*.

4.6.5 Features Cumulative Distribution

In Figure 4.7, we provide more details on the top features with their Cumulative Distribution Frequency (CDF) on the testing set of the HEALP dataset. We reported those features with importance thresholds of 0.025, similar to 4.6.4. For contextual features, nearly 95% of APT age domains are equal to or below 120 months, whereas 64% and 60% of phishing and legitimate age ones have the same age. This confirms our analysis where APT age domains take considerable

time before being suspended, although they are easily convicted by Law once detected, unlike phishing. According to [204], it is hard to prove criminal activity for phishing domains because it targets mostly public users who do not have the resources to pursue the case.

This is also can be the case for the nameserver and registrar's reputation. 58% of legitimate domains use nameservers with a reputation rank equal to or below 20k. On the other hand, APTs and phishing, where few of them, 25% and 44%, use reputable nameservers. Regarding the registrar's reputation, 52% of APT domains are hosted by registrars ranked nearly 18.7k or below, whereas 76% and 74% of legitimate and phishing domains have the same range. For the number of IPs, as mentioned earlier, the APTs tend to use fewer IPs than legitimate. Approximately 89% of legitimate and phishing domains configure a single IP at a given time, whereas 4 IPs or below are deployed for phishing and legitimate ones.

For Semantic features, we found that NCD_e(ELD, WLU) is nearly 98.3% for both APT and phishing that have a similarity of 0.3 or below, whereas 99.7% for the legitimate ones. That reflects that the NCD_e(ELD, WLU) can flag most APT and phishing domains as close enough to those legitimate ones.

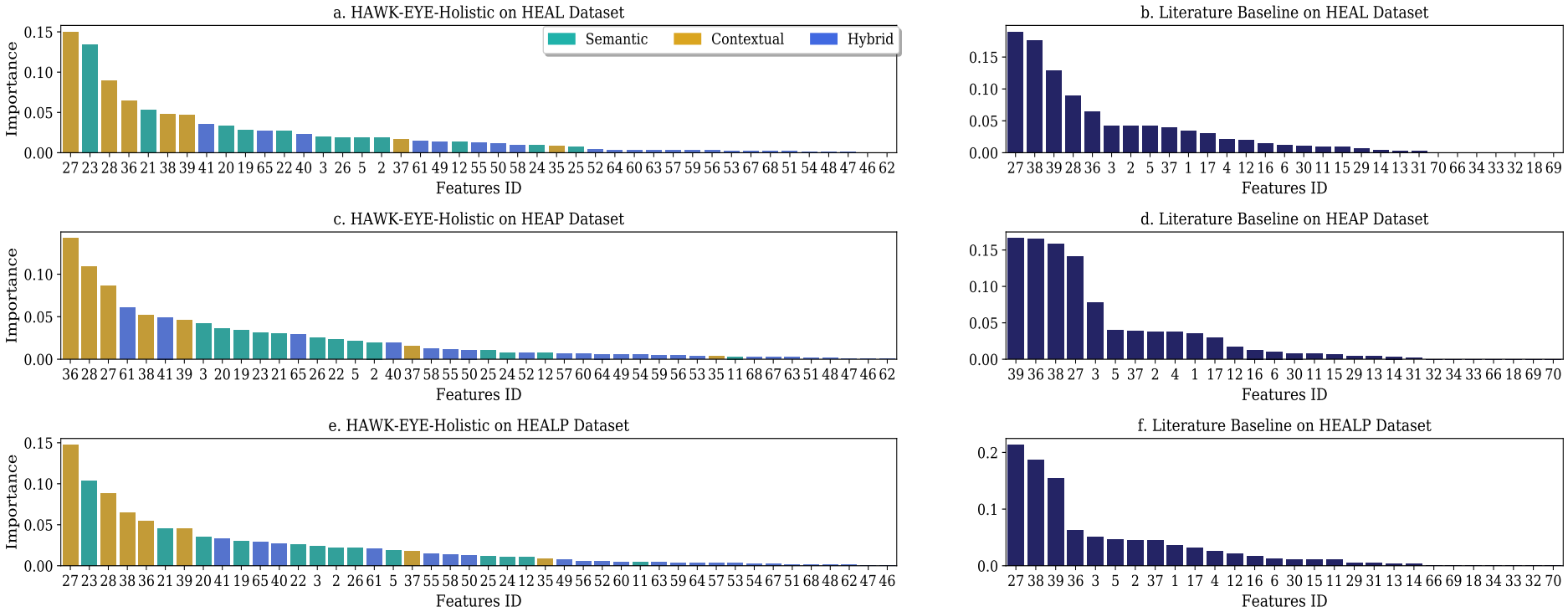


Figure 4.6: Feature importance: comparison between HawkEye-Holistic and Literature Baseline across all datasets for unknown APT campaigns.

Next, NCD_e(ELD, MFTU) shows that at 20% threshold, APT domains use the most frequent technical unigrams trained based on our training dataset with similarity equal to or below 0.4 for those unseen APT campaigns, whereas 11.7% and 1.2% for the phishing and legitimate ones. A similar technique of using technical words is also used for the nameservers. NCD_e(NS-ELD, NS-WLTU) identifies that nearly 60% of APT domains use nameservers with technical words close to 0.3 of similarity, compared to 45% for legitimate and phishing nameservers. This can be more obvious in NCD_e(ELD, CTU), where 38.7% of APT domains combined technical words regardless of their frequency in our training set with similarity equal to 0.4 or below, while roughly 25% and 8% for phishing and legitimate domains. Similar findings are confirmed for NCD_e variants of NS-WLU and WLTU.

4.6.6 Discussion

The HawkEye-Holistic (HH) feature set consistently outperforms the Literature Baseline (LB). For the HEALP dataset, the FPR, macro recall and precision of the LB, standing respectively at 0.69%, 36.47%, and 69.23% are much worse than the corresponding figures for HH, which stand at 0.35%, 70.83% and 89.55%. The 4% positive rate of HEALP that produces these numbers does not reflect the actual frequency of APT domains in corporate traffic: the performance gap between HH and LB increases as the positive rates decline, as can be seen by the larger gap in $mF1$ as opposed to $wF1$. In fact, $mF1$ decreases with the number of positives, and at the limit (no positives: we are monitoring a clean network) only the FPR matters.

Our split of training and test set by campaign prevented us from using cross-validation techniques. Due to the limited number of APT domains available for training and testing, we did not set apart a validation set, and so we did not optimize the RF parameters. Instead, in order to include a sufficient number of legitimate domains and better approximate the legitimate distribution, preventing an artificial separation between the APT and legitimate class, we chose to use an unbalanced data set. In fact, the training error for a class-balanced subset of HEAL is 0, whereas the $mF1$ is 90%, indicating overfitting. With the current class imbalance, the training error is much closer to the testing error. Increasing the ratio of negatives much further negatively affects the recall for APTs, as legitimate samples overwhelm the model.

Figure 4.8 shows the means of the top 6 features of the contextual, semantic and hybrid feature

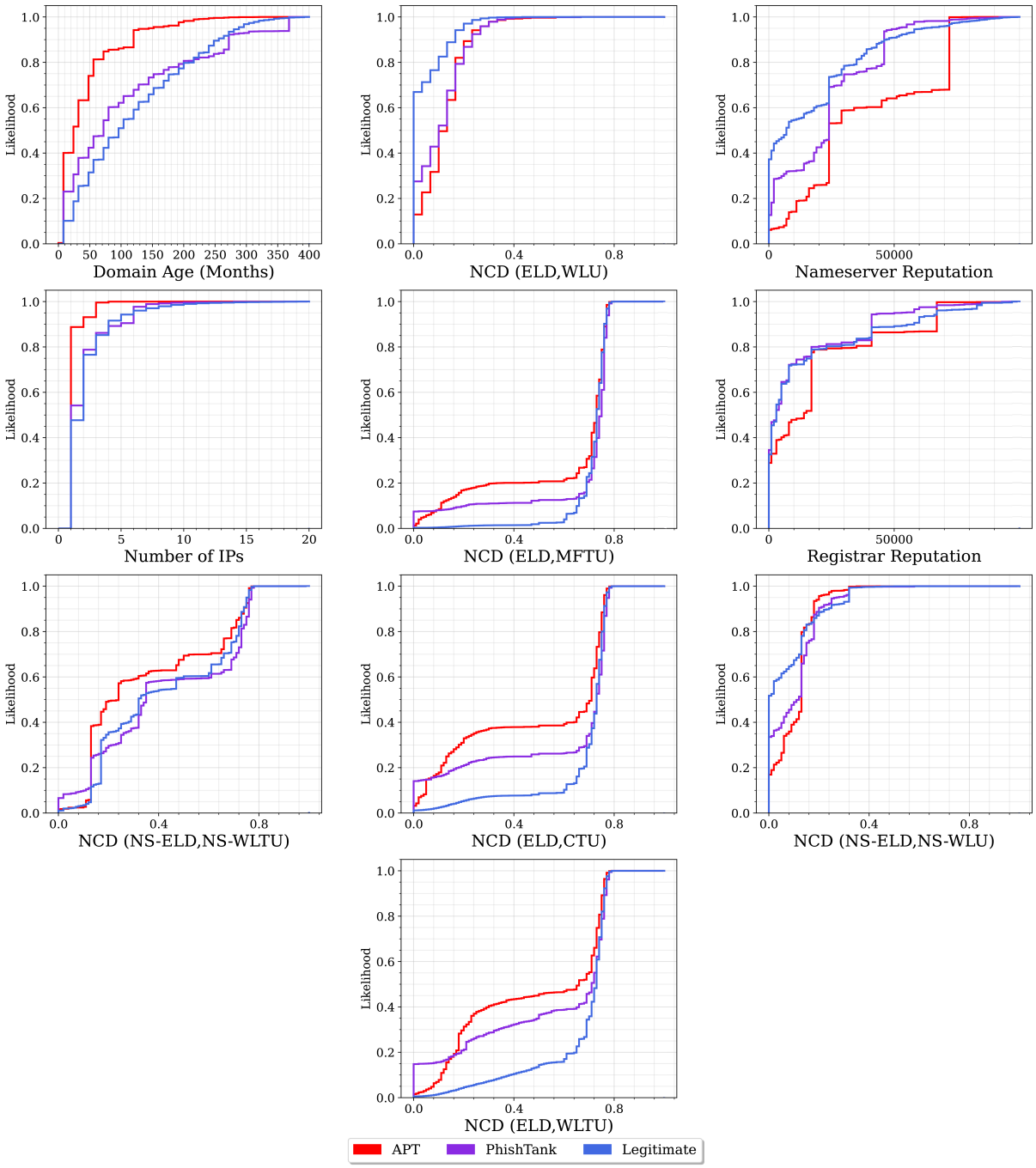


Figure 4.7: CDF for the top features on testing set of HEALP.

sets for each label class. The highest mean of each feature is normalized to 1.0, and the other means are scaled accordingly. We can see that contextual features play a substantial role in distinguishing APTs from both legitimate and phishing domains, whereas semantic and hybrid features single out legitimate domains but present less differences among malicious domains. The strength of contextual features suggests that domain configuration is a weak spot in the infrastructure of APTs. Character and lexical features tend to have a more mixed performance. Lexical analysis is quite effective in detecting malicious domains posing as legitimate ones, but

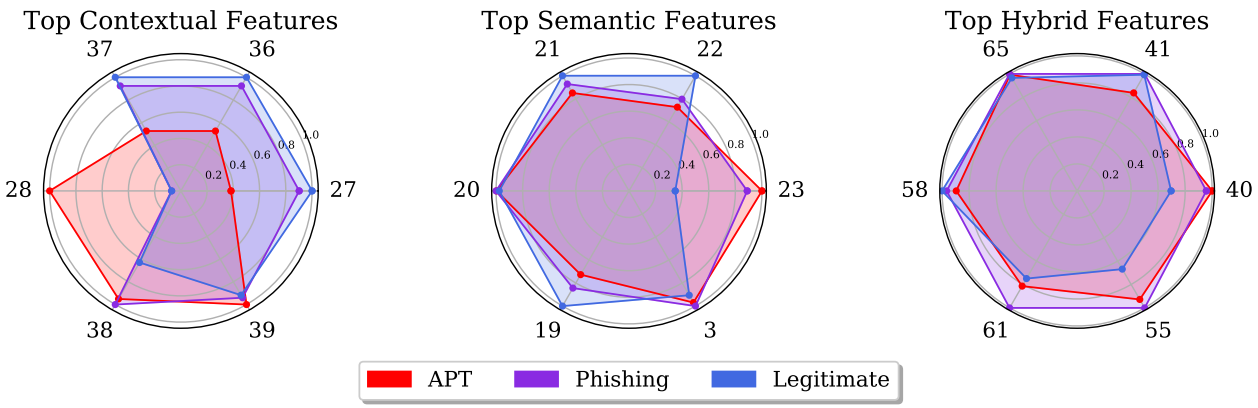


Figure 4.8: Top features comparison across APT, phishing and legitimate domains.

we also note that it is prone to false positives, as several relatively popular legitimate domains have the same characteristics as malicious ones. Some representative examples are reported in Table 4.8. Overall, our results confirm that in the case of APTs, a holistic approach is best suited for malicious domain detection.

Domain	Rank	Domain	Rank
googletagmanager.com	24	`mp3-youtube.download`	5595
`youtube-nocookie.com`	201	`microsoftazuread-sso.com`	11132
`ssl-images-amazon.com`	1011	`microsoftonline-p.com`	12337
`googlezip.net`	1223	`youtubeconverter.io`	16313
`onmicrosoft.com`	2808	`facebook-danger.fr`	34322
`firefoxchina.cn`	4433	`allfacebook.com`	27426
`symantecliveupdate.com`	4407	`microsoftemail.com`	73976
`mcafeemobilesecurity.com`	4729	`kasperskycontenthub.com`	160661

Table 4.8: Legitimate domains resembling APT domains.

4.6.7 Limitations

Although the FPR of HH at 0.35% can be considered low in the context of the malicious domain detection literature, we think it is still too high for HawkEye to be used as a standalone detector in practice, as the number of false positives for a much higher (realistic) negative rate would be excessive. As it stands, HawkEye can be used as a pre-filtering step to substantially reduce the traffic to be scrutinized for APT presence, or it can be employed in forensic investigations, where a limited window of traffic needs to be analyzed to detect actual APT C&C traffic.

The legitimate domains in our datasets are taken from a website ranking list, and consist mostly of apex domains. Legitimate sessions of end-user traffic would likely include a large number of domains, including a host, and would be better suited for a comparison with APT domains, which predominantly include hosts. To avoid introducing a bias in our experiment, we limited ourselves to consider apex features for FQDNs: we leave it to future work to build a dataset including legitimate host features. We experimented with adding host features to the current feature sets, but they flattered APT detection performance and overwhelmed the other features. For example, the average length of hosts is very close to 0 only for the legitimate class of HEALP, which is not reflective of real-world traffic.

Another limitation of our approach is that our segmentation and unigram techniques do not take into account the fuzzy matching of words, where characters may be added and removed, thereby missing out on detecting further typosquat variants of popular brands and domains. In fact, the HT-NN unigram based on Hamming distance is only able to handle character replacement. For example, the APT domain `samrsung[.]com` is misclassified as legitimate but it could plausibly be caught by a fuzzy segmentation, matching it as a variant of the popular domain `samsung.com`. We leave this extension of HawkEye to future work.

Finally, our legitimate and phishing domains were collected during specific and limited intervals of time, whereas the APT domains span 13 years. This may introduce a form of temporal bias, although we have mitigated that by not using the same APT campaigns for both training and testing. In general, HawkEye should be periodically re-trained in order to adapt to the drift in actual adversarial and legitimate behaviour.

4.7 Related Work

There is very limited previous work on detecting APT domains [5, 188], and it tends to suffer from a lack of details on the features used and on the composition of datasets. As discussed in Sections 4.3 and 4.4, several features we considered were inspired by previous work on malicious domain detection, and even more have been considered in the literature. Domain length [180, 182, 184, 189, 78] , % of numerical characters [189, 180], length of meaningful strings [15], vowels and consonant characteristics [180], and TTL [15, 180], NXDomain responses [179, 16, 180] have proven particularly useful to detect DGAs. Notos [194] is a detection

system which combines domain ranking with geolocation information based on BGP and AS. In our analysis of APT campaigns, we noticed that APTs are able to locate their infrastructure worldwide, and even inside the country hosting the targeted organization, so we did not attempt to geolocate domains. Exposure [15] (botnet and phishing), Pleiades [16], and HinDom [179] (DGAs) also use features based on the repeated interaction between a client and the requested domains to detect malicious domains. Features include daily similarity, repeating patterns and access ratio features, which are less likely to be effective with the low, slow and stealthy connections used by APTs.

PREMADOMA [189] helps registries (in the specific case, the `.eu` ccTLD registry) to prevent malicious domain registrations. The most characteristic features of [189] are related to the selection of name servers, contact email addresses and phone numbers. APTs often use bulletproof hosting providers or less reputable ccTLDs, which do not have an incentive to deploy measures against malicious domain registration. Unlike [189], HawkEye focuses on clients rather than registries or registrars, yet it could be useful to the latter as an aid for proactively identifying suspicious registrations.

YATT [184] is a browser-based framework to prevent users or adware from accidentally accessing typosquat domains. The framework includes WHOIS and DNS features such as a total number of typos in nameservers, TXT Google verification and registration date. However, their WHOIS crawler only considers the `.com` format, excluding the substantial number of domains hosted on other TLDs. As described in Section 4.2.1, HawkEye's crawler is able to handle the WHOIS format for most TLDs' WHOIS servers of the Public Suffix List [187], and may help extend the coverage of YATT.

MADE [78] is a SOC-like enterprise solution to analyze logs received from firewall, antivirus and web proxies, and detect malicious communications. It uses machine learning to assign a risk score to each connection. However, MADE mainly considers HTTP malicious connections, including URL parameters, User Agents features and domain-based features. We included some features from MADE in the literature baseline set, and we show that detection improves when those are combined with our new features.

Finally, PDNS [200] is a host-based malicious DNS detection system to detect encrypted DDNS requests. Since DNS is encrypted, PDNS mainly considers the location of DNS requests, in addition to GUI, UI and Web communication DDLs. This proved to be an effective strategy

against botnets, where there is substantial traffic, but does not apply naturally to APTs, where the establishment of C&C is just an initial stage which employs several evasion techniques, as discussed in Section 4.1.

4.8 Conclusions

In summary, our main contributions in this chapter are:

- A detailed, evidence-based analysis of the use of domains in the context of APT C&C infrastructure.
- The implementation of HawkEye, a classifier to detect APT C&C domain requests from PCAP files, crawling live DNS and WHOIS data where necessary.
- The first publicly available, curated dataset of APT domains used for C&C infrastructure.
- New features for malicious domain detection, targeted for APTs.
- A thorough evaluation of the classification performance of new and existing features under different scenarios.

To the best of our knowledge, HawkEye is the first system that attempts to detect C&C domains used by APTs at the network level, and ours is the first dedicated dataset publicly available. By leveraging a number of new and existing features captured at different levels (domain name, WHOIS, DNS records), our best classifier achieves a promising level of performance. A number of novel features introduced by us contribute to achieving the best performance. HawkEye is a prototype built in Python, focusing on robustness, modularity and generality and designed to test different domain detection hypotheses. We envisage that a high-performance tool based on HawkEye could work as a parallel component of a network intrusion detection system, but we leave a study of performance and deployment issues to future work.

Chapter 5

Detecting APT Malware Command and Control over HTTP(S) Using Contextual Summaries

After we create our first line of defence in Chapter 4 to examine the APT domains before the connection is established, we are now interested in detecting if an adversary successfully evades HawkEye and begins their C&C traffic. Malware, such as botnets, that rely on C&C for communication are well studied, and are targeted by several NIDSs. Typical botnet behaviour includes launching DDoS, propagating quickly to other hosts, internal scanning, or sending spam [14]. In the past, anomaly-based and NetFlow feature-based were able to detect such behaviour and filter it out, detecting the abnormality of a connection over a relatively short period of time, as discussed in Chapter 2.

In Section 3.4, we explored several TTPs deployed by APT operators to keep a low profile and facilitate incoming targeted and stealthy attacks over a long period of time. Therefore, relying on the traffic frequency, beaconing behaviour, or NXDOMAIN generated by DGA in many botnets is insufficient to detect APT malware. Also, the existing defences that adopt feature-based machine learning or deep learning [17, 18, 19, 20, 21, 22] to detect non-targeted attacks, are much less effective at APT detection, as we will discuss in Sections 5.1, 5.4 and 5.6. As discussed in Section 2.5.4, deep learning requires a vast dataset to train a classifier, but

real APT data is scarce. Machine learning with feature extraction is a promising candidate, but we need first to identify the tailored features that will work specifically for APTs, as we have proposed in Chapter 4 for DNS infrastructure.

The quote says "Information is power only if you can take action with it" [205]. All network data is recorded in PCAP format, which is difficult to follow and store in practice. A number of tools have been proposed to extract the desired fields and specific context for network and security operations. The state of the art in processing PCAPs to produce logs for detection and forensics purposes is not adequate to fully capture APTs. Using NetFlow or Web Proxy logs miss several essential fields, which limit APT-related feature extraction. In addition, considering the sparse logs files generated by tools such as Zeek/Bro[206] causes overhead to understanding the contextual behaviour of their file formats. Apart from raw PCAP, these data formats do not provide a way to access the packet-level data points, including bytes, timestamps, and the protocol, which hinders the researcher from providing important features relevant to APTs. For instance, we notice that APTs connections have considerably larger mean delta packets interarrival times. Likewise, the data packet exchange idle time of APTs is significantly different from legitimate traffic. Nevertheless, these features do not suffice to clearly separate the classes of interest, as some legitimate connections, browsing-like, have similar behaviour.

That leads to further investigation of other attributes, such as the number of failed HTTP packets, which is more prominent in APTs. Hosts may use several User-Agents, some of which are already reported as an IoC for malware using the exact string, which may conflict with the list of UA's a host. In addition, the list of FQDNs, Resource Records, and URLs accessed by users is vital to be tracked. The type of client cipher suits used by the host through a specific time may reveal a malicious behaviour of an APT malware pre-configured to be using particular settings. We discuss the issue of the current data format further Section 5.1.1.

In this Chapter, we propose a novel threat model that highlights the importance of the context around the malicious connections, and suggests traffic attributes of TTPs which help APT detection (Section 5.1). We focus on two cases in our threat model and introduce the first public measurement study of malware used by APTs and botnet C&C communication to investigate the deployed TTPs. We leverage our measurements to identify the features necessary to recognize such TTPs at the network level, and compare them with existing features from the literature. We found that the use of evasive TTPs leads to significant overlap with legitimate

behaviour, confusing the decision boundaries based on known features (Section 5.2).

Based on this analysis, we build EarlyCrow, a tool to detect APT activity in network traffic. EarlyCrow utilise PairFlow, a novel multipurpose network flow format which captures the necessary fields of the connections between host pairs of interest, over a granular time tuned by security practitioners (Section 5.3). EarlyCrow generates four sets of data focused on connections, hosts, destinations, and URLs. Features from these sets are grouped to form a ContextualSummary. The ContextualSummary has multidimensional features that help in building more informative random forest trees used for classification as described in Section 5.4.

We evaluate EarlyCrow on traffic from APT-like malware excluded from the measurement study and training set in order to test generalization and mimic a real-world scenario (Section 5.6). Fresh malware samples are also investigated to confirm the feature importance identified by our measurement study on the training set. We also investigate how the performance of EarlyCrow is affected by different deployment scenarios, where it has visibility on HTTP traffic or where it can only observe opaque HTTPS traffic. EarlyCrow defends well against unseen APTs that encrypt their traffic with HTTPS, obtaining a headline $mF1$of 93.72%, and accuracy of 98.11% with FPR of 0.74%. In comparison, the state of the art stands at 60.29% and with no false positive rates.

5.1 Threat Model

Defining a relevant threat model and focusing on a narrow set of attacks are recommended best practices when proposing a novel NIDS [207]. There are several ways to analyze the threat model for APTs at the network level. While the previous Chapter focuses on the DNS infrastructure, this Chapter focuses on APTs traffic. We identify four popular cases, each of which may deploy at least one C&C TTP, as depicted in Figure 5.1.

In Case I, the infected machine contains APT malware with a hard-coded FQDN. The malware issues a DNS query to resolve the FQDN to an IP address. The subsequent communication to the C&C server can be via HTTP or HTTPS. After that, the malware may initiate a *fallback channel*, another popular TTP used by APTs [208], using either of the strategies described in Cases I-IV, only this time no longer for the initial communication. In Case II, the APT malware connects to a URL whose domain component is a hard-coded IP address, in order to bypass malicious domain detectors, and its fallback channel can be established using the *DNS over HTTPS* (DoH) TTP [209], as in CobaltStrike [210], which is used by SUNBURST [211]. Case III is similar to Case I in using a hard-coded FQDN, but the subsequent communication uses raw TCP rather than HTTP during the malicious operation. Case IV is similar to Case II in using direct IP without DNS resolution, but then uses raw TCP communication as in Case III. Both Case III and IV may use fallback channels with various TTPs, although not including those related to HTTP(S).

Additional TTPs introduced by MITRE and relevant to APTs can be combined with the use of a fallback channel: *web protocol* [212] where an adversary may use HTTP to avoid network filtering and mimic legitimate and expected connections, *non-application protocols* [213] such as Raw TCP, UDP or ICMP, *encrypted channel* [214] to hide C&C malicious content, *fast flux* [215] (a sub-technique of *dynamic resolution*) to obtain different IPs for the same FQDN, and *data obfuscation through protocol impersonation* [216] to impersonate legitimate use of HTTP or to mimic a trustworthy entity using a fake SSL/TLS certificate. Further definitions and details of these TTPs are already explained in Chapter 3.

This Chapter focuses on Cases I and II, where at least one malicious HTTP(S) connection exists between the infected host and C&C server. Cases III and IV are left to Chapter 6.

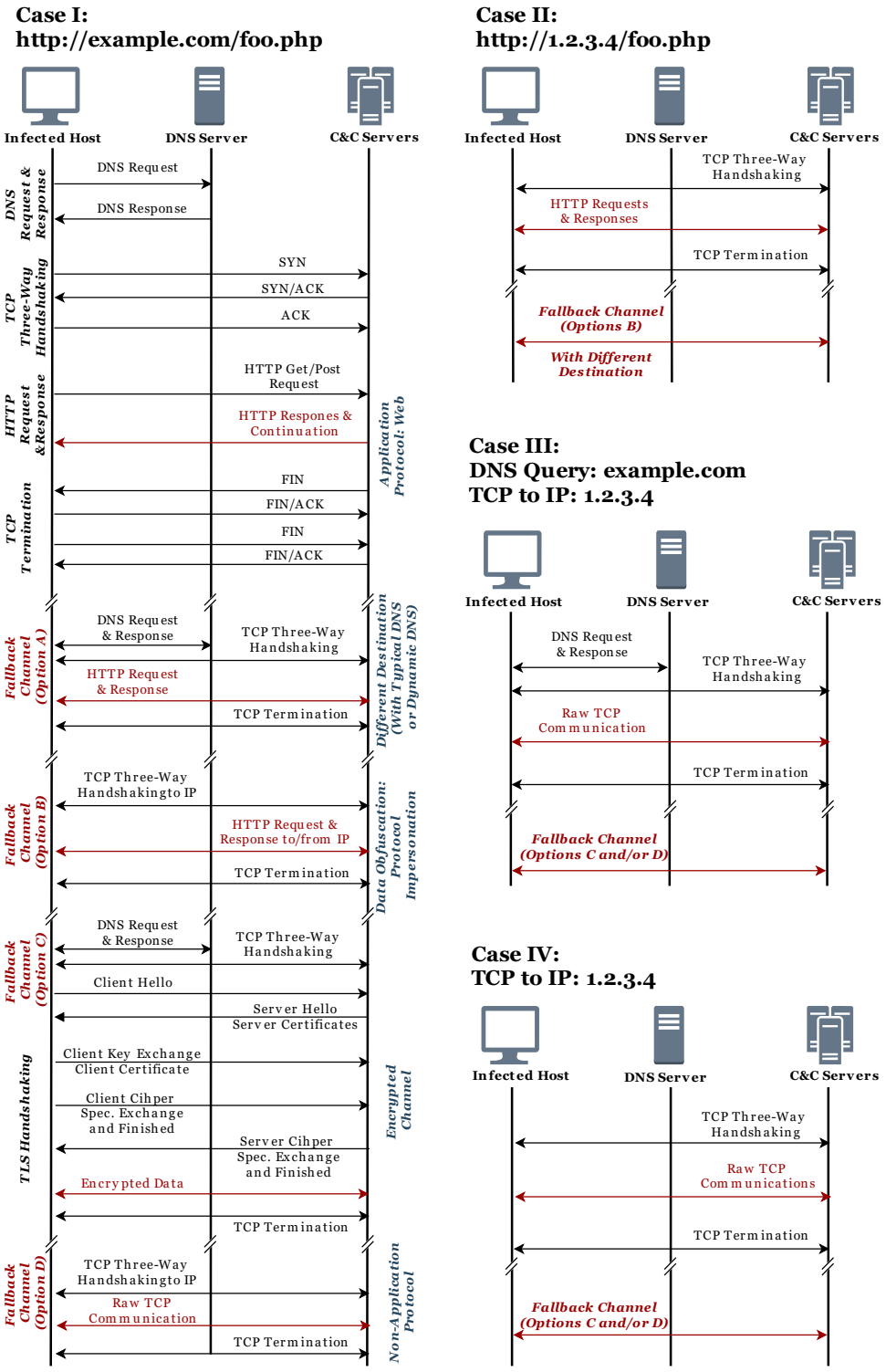


Figure 5.1: Types of APT malware initial communications. Fallback channels may use one or more TTPs (blue text). Malicious data packets are represented by red arrows.

5.1.1 Current Data Formats Issues

PCAP is the de-facto format for network captures, and it provides the raw data at the network level. However, several NIDSs use different data formats built on top of PCAP to improve

detection performance. Current data formats hinder security practitioners from extracting specific fields to capture the IoC, such as URLs and IP addresses. Many informative fields have to be summarised for the end-to-end connections to provide the contextualisation to extract instructive features that reduce FPR. Examples of these fields include A and nameservers resource records, HTTP status codes, content types, and protocols used during the connection.

NetFlow is a popular data format, which is defined in RFC 3954 [217]. It provides the network operator with IP flow information on their network for diagnosis and troubleshooting purposes [217]. Several packets between a source and destination with common properties, such as protocol and port, are encapsulated in one flow for a specific time window. Security analysts and researchers widely use NetFlow to detect malicious activities. However, NetFlow does not provide salient information to a flow, such as FQDN, accessed URL, UA, SSL/TLS settings, and HTTP status codes counts. In addition, a flow cannot include more than one protocol, which may increase the challenge for security analysts and NIDS to detect long, low-mode malicious activities.

Tranalyzer [218] is another lightweight flow generator and packet analyzer developed for researchers and practitioners. It enhances the capabilities of NetFlow and aids analysts in analyzing massive packet dumps. It also provides access to drill down to the specific flows or even individual packets. However, Tranalyzer separates flows according to their protocol, which does not provide contextual data between pairs of endpoints.

Contextual Flow [219] is another flow-based format designed to detect encrypted malicious traffic without decryption. The contextualization attaches the TLS parameters and UA to the flow. However, Contextual Flow lacks accessing the details of packet arrival time that forms such a flow, which may weaken the ability to analyze stealthy attacks during a flow. Therefore, HTTP responses behaviour cannot be analyzed. FQDN resource records and domain age are not included in a flow, requiring other data formats for malicious domain detection. The contextualization is also lacking from tracking the previous flow, which can help to capture those attacks successfully and evade NIDS, but can be reconstructed over time for detection.

Other popular data formats provide part of the required fields to capture APT attacks. *Passive DNS* was introduced by F. Weimer [220] to detect malicious connections resolved by DNS. Its output captures DNS responses and extracts resource records without involving authoritative name servers. However, relying on DNS resource records and their TTL is insufficient to detect

malicious connections, including APTs. *Web Proxy* is also a popular data format for parsing HTTP requests and responses and log-related information in a tabular format. Examples of web proxy popular generators include Microsoft Forefront TMG [221], and Squid [222]. APTs may use HTTP(S), raw TCP, or other protocols. Therefore, web proxy logs may not be the best to reconstruct APT attacks.

Zeek/Bro [206] is a network intrusion detection system run in real-time. However, several systems use Bro to compile PCAP files and generate more than 20 data formats to be used as input for proposed systems. These logs include connection status, application statistics, DNS, DHCP, HTTP, SSH, etc. However, Bro does not generate a flow-based format that tracks the story of a given connection over a period of time; instead, it provides a sparse format that needs considerable time to extract data related to a connection.

5.1.2 TTP Relevant Data

In order to define a convenient data format which can be used by feature-based NIDS for APTs communications detection, it is essential to understand the threat hunting of APTs in practice. To track TTPs at the network level, a security researcher needs to collect 'static' IoCs for known APTs or manually identify the likelihood of specific traffic-based TTPs based on enterprise network behaviour. Security vendors publish IoCs of discovered APTs, which may present a usage of TTPs like DGA or Dynamic DNS. These IoCs help security engineers to investigate if their enterprise is compromised by an APT campaign. Another source of information beyond IoC is the industry reports. These reports can explain the traffic behaviour of the infected enterprise. For example, an APT campaign may frequently use a non-application protocol or a fallback channel, which is unusual traffic for the infected enterprise.

In this section, we present the two methods, IoC-like and traffic data, to study the APTs based on such information, which is presented in Section 5.2. This is also important when we propose our data format in Section 5.3.

5.1.2.1 IoC-like Data

The proposed data format should provide enough information to extract IoCs from PCAPs. These include FQDN, IP, URL, UA, and Encryption Settings. APT campaigns attempt to reserve one or more *FQDN(s)* to locate C&C servers. They may mimic the targeted organisation interest or use Dynamic DNS, which is another TTP[223], to communicate back to C&C servers [36, 23]. The resolved FQDN holds at least one A resource record and many records for popular web servers. However, some APTs provide many A resource records to provide a fallback channel for the next sequence connection.

URL is known to be used as an IoC and used in HTTP-based malware detection [224, 78, 21, 225]. Some APT malware downloads an executable file or passes another malicious FQDN, IP, or configuration commands in the new URL parameters in subsequent requests. A typical URL structure includes FQDN, nested folders (which we will referr to as *depth*), filename, parameters and values with a delimiter (&) to separate between them and (=) to assign value to the parameter, and encoded strings which typically contain %-encoding.

User-Agent (UA) is also another IoC used in malicious traffic detection [226, 78, 219]. Different browsers are available to surf websites, and they vary in supporting technologies such as frames, images, and video content. Web developers design websites to fit the majority of browser rendering abilities. For malware, it is common to find typos, outdated versions, or even leaking information [227]. Other malware has been found to use an empty string (e.g., Trojan.dropper), unknown string (e.g., H-worm, Win32/Sality.3, Win32.QQP), or inconsistent string (Win32/Alman, Trojan.Win32.Dropper.aa) [228, 219].

The last IoC to check is the *Encryption Setting*. Some APT malware uses malicious TLS certificates to impersonate trusted entities (*data obfuscation: protocol impersonation TTP*), or to hide the traffic payload from an IDS. It is vital to check cipher suit settings to find whether the adopted one is insecure, unusual, or weak, which is a common sign of malware [229].

5.1.2.2 Traffic Data

Although multiple traffic-based TTPs were used by APTs in the past, it is challenging to capture them by configuring NIDS with naive and straightforward rules, which may raise the

FPR sharply. Hence, we need to consider how malicious packets are sent at the contextual level. First, we need to cover the details of HTTP requests and responses, and then the traffic behaviour of all protocols used for the same flow. *HTTP request and response context* involves consecutive HTTP transactions composed of request and responses. A request is mainly characterized by the URL, method type (e.g. GET, POST .. etc), UA and Referrer. Response headers specify the settings of the content type, Cookie and status codes. To detect APT malware, we need to efficiently store that information between two endpoints in one flow and enable the NIDS to extract valuable statistics at the packet level.

Due to the stealthiness and low-mode operation for APTs, we also need to provide a way to investigate *Traffic behaviour*. This can be achieved by storing the packets' arrival time, their length and other related information presented in Section 5.3. Such a summary needs to cover the control and data planes of TCP, UDP and ICMP packets. With this summary on datapoints, NIDS designers can catch APTs TTPs such as *fallback channel* and using *non-application protocols*. For instance, a host contacting three different destinations after only one DNS query can be a sign of infection by the fallback channel technique. Another scenario of TTP, the *non-application protocol*, is when the APT malware opens a legitimate-looking HTTP connection but that is followed by a sequence of malicious raw TCP packets as depicted in Figure 5.1.

5.2 Traffic Measurements

We provide several measurements taken on the training set summarised in Table 5.3, page 184, and described in Section 5.5. Since our objective is to detect APTs at the early stage, all measurements are observed during the first 15 minutes of each connection, which is the longest duration allowed by the Any.Run sandbox[1]. In Section 5.6.2, we will investigate these measurements and other proposed features, to see if they generalize to unseen malware.

[1]https://any.run

5.2.1 Traffic Statistical Measurements

Statistical end-to-end observations may highlight the evasive behaviour of APTs compared to legitimate actors. The presence of a slight deviation may reflect malicious use of three TTPs, including *non-application protocols*, *data obfuscation through protocol impersonation* and *web protocol*. Since this study focuses on malicious HTTP(S) usage, we measure the HTTP packets ratio across all classes. Other related protocols are also measured, including raw TCP and DNS ratios. Legitimate connections show a positive linear relationship between DNS and HTTP packets (Figure 5.2). With every additional page requested by a user, such packets are exchanged with a remote web server in order to fetch additional resources. For APTs, we notice that DNS ratios are half or less than for legitimate or botnets, respectively. 95.2% of APTs do not exceed a 0.19 DNS ratio, compared to 0.38 for legitimate and 0.46 for botnets.

Next, we focus on DNS requests and conclude that almost no malicious behaviour exceeds the legitimate, except for Conficker botnets, which use the DGA technique. 84% of APT or botnet traffic issues 2 or 6 requests at most, while legitimate traffic can generate up to 18. Once a domain is resolved to one or more IPs, a typical APT avoids requesting another DNS for the rest of HTTP communication unless they plan to establish another *fallback channel*. Another useful feature is the raw TCP ratio, which helps detect the *non-application protocols* TTP: a high ratio indicates the adversary uses HTTP as camouflage while still heavily relying on raw TCP. It is extremely rare for an APT to have a raw TCP ratio lower than 48.84%, whereas we observed minimum ratios of 2% of legitimate, and 0% of botnets.

Since we focus on the early stage of connections originating from the victim side, we found that around 70.58% of APTs receive 3.35 times more data than they send to the remote server, compared to 1.45 and 0.75 for legitimate and botnets, respectively. This is consistent with the threat model in [23], where an adversary uploads more tools on the victim's machine at the beginning of an APT campaign to continue other operations such as lateral movement, unlike botnets, which may show more data exfiltration behaviour. We also examine the number of resumed connections. Legitimate HTTP usage typically increases the number of resumed connections. Once the web resources are downloaded and the keep-alive time has expired, the TCP connection is terminated with FIN. Upon clicking another link, even for the same website, a new TCP three-way handshake is initiated. We count that as a resumed connection. With a web caching service, the scenario remains similar, although the server is contacted via a proxy

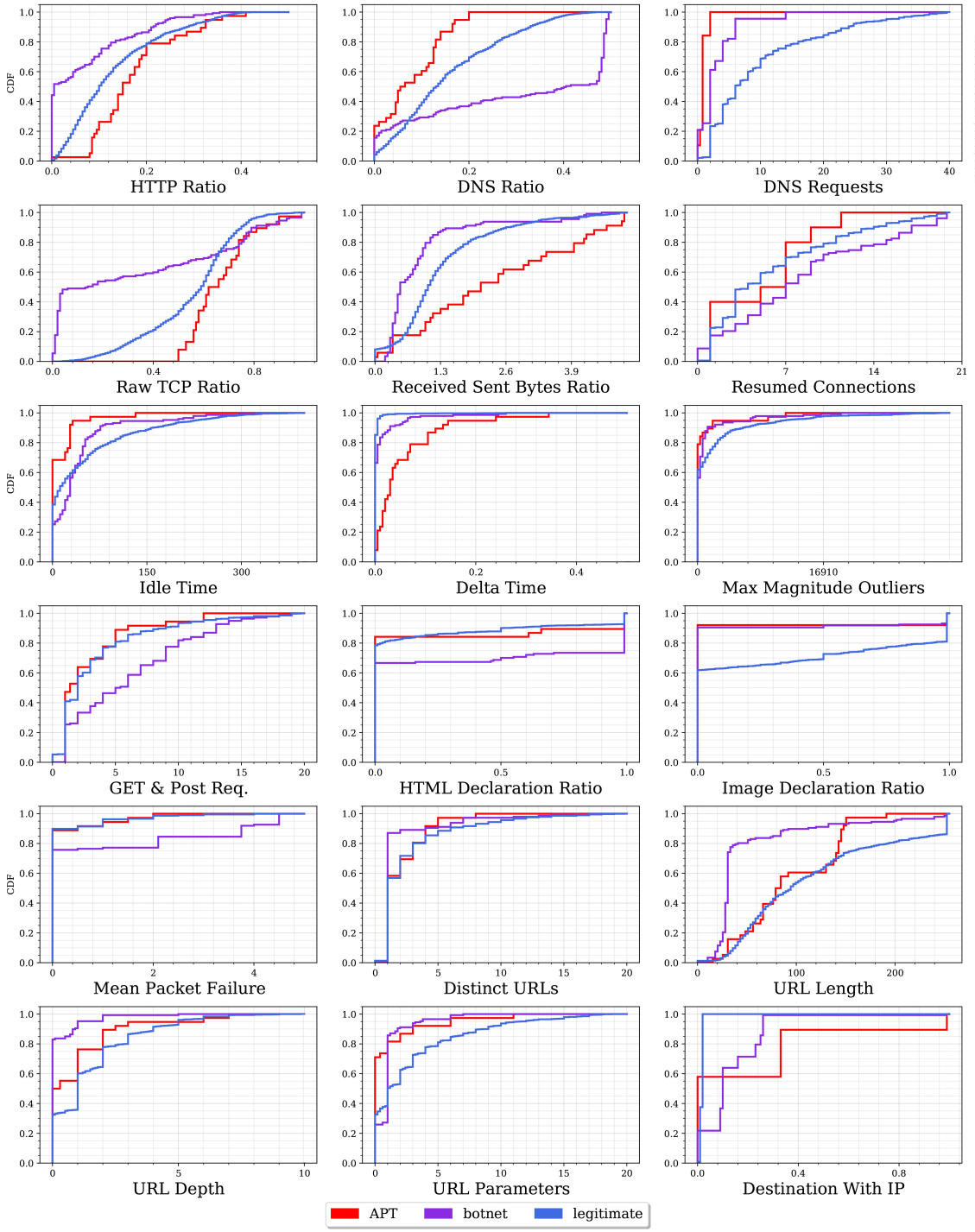


Figure 5.2: Measurements for APT, botnets, and legitimate connections.

or content delivery network (CDN). While legitimate and botnets connections may easily be resumed up to 21 times, APTs tend to terminate less (roughly 50% less). It seems plausible that APTs avoid frequent connection termination and resumption to increase stealthiness.

5.2.2 Time-based Measurements

We measure *stealthiness and low profile* of APTs by monitoring time-based features. *Delta*, the packet inter-arrival time between a remote server and a host, is estimated based on the arrival time difference between packets, independently from their protocol. For 94.73% of cases, we found the mean delta time in seconds to be at most 23.5×10^{-2} for APTs, 6×10^{-2} for botnets and 0.5×10^{-2} for legitimate. Hence, APTs may act slower than botnets and legitimate by up to 4 and 47 times respectively.

A new metric, data packet exchange *idle time*, is proposed to measure the time difference between actual data packets. We found APTs idle time to be 3 and 6.57 times shorter than botnets and legitimate: 92% of cases have an idle time of at most 28, 84, and 184 seconds, respectively. Once APTs establish a communication channel, they send bursts of data packets (low idle time), then pause communications (high delta) until the next burst. We also measure the maximum magnitude of outliers which exceed the Simple Moving Average (SMA) with respect to the predefined bins described in Section 5.4.2.2. We found that for 84% of the cases, the maximum magnitude for APTs (0.338 KB) is half the one for botnets (0.676 KB), and 10% of the one for legitimate (3.33 KB). These three time-based features partially capture the *low and stealthy profile* of APTs compared to botnets or legitimate.

5.2.3 Remote Web Server

Analysis of contacted web servers may help identifying the *web protocol* and *fallback channel* TTPs. Typical web servers mostly adhere to best practices in setting up their HTTP configurations. APTs appear to be more professionally configured than botnets, but not as much as legitimate ones. For instance, the *packet failure* rate for legitimate servers and APTs (HTTP responses with status codes 4xx and 5xx) is relatively low. To be precise, 90% have at most one packet failure, while the botnets may receive as many as five. Total GET and POST requests are less similar. 92% of APTs and legitimates have 9 and 10 or less, respectively while the botnets have up to 14.

We also investigate the ratios of content types declarations. We focus on the ratios of HTML and images, since these are most frequently used in HTTP connections. 73% of APTs, legitimate

and botnets declare HTML 2%, 2% and 98% of the time, so APT behaviour, in this case, is similar to legitimate. However, due to the possible use of the *data obfuscation through protocol impersonation* TTP, we found that APTs and botnets are less likely to declare image type, which is not the case for web browsing activities: 70% of legitimate declare images 30% at most during a connection, while it is zero for both APTs and botnets.

Next, we measure the URL characteristics, due to their proven effectiveness in detecting malicious web servers. Measuring the distinct URLs accessed in a given network may highlight the rich number of web pages which is more likely to be legitimate [78, 19]. We observe that APTs invest heavily in legitimate-looking pages, to evade NIDS that rely on URL-based features. For example, we find that 87% of botnets query only one URL, while legitimate and APTs query up to five and four, respectively. APTs have more resources than botnets in general. As depicted in Figure 5.2, 90% of APTs have 3 nested folders (depth), close to legitimate, which is 4, while botnets have 1 at most. URL parameters differ even more: 87% of APTs and legitimate use 3 and 7, while botnets use only 1. Following that, URL length is determined by the length of FQDNs, depths, filenames, parameters, values, fragments, and strings. 90% of legitimate URL lengths are 249 or less, whereas APTs and botnets are up to 145 and 109. Finally, APTs deploy a fallback channel in several ways, as discussed in Section 5.1. We measure the number of HTTP(S) connections established to an IP without a previous domain resolution. 57.89% of APTs reached 32% of C&C with IP only, while it is 9% and 1% for botnets and legitimates. Therefore, it is unusual for legitimate to perform such behaviour, while it is more common for APTs and occasional for botnets.

In the next section, we will describe PairFlow, which allows the NIDS designer to quickly pivot flows into many profiles such as host, destination, URL profiles. It also allows being used by those malicious domains or IP detectors. Instead of detecting one flow according to their initiating and termination of TCP, protocol-based or time window, it digests all information to extract features later based on the whole context over time. So it is vital to collect all proper IoC and traffic-based TTPs known to be used for malicious detection.

5.3 PairFlow

PairFlow is a data format that allows the NIDS designer to quickly pivot flows into many

profiles such as host, destination, URL profiles. PairFlow data can also be used by detectors of malicious domains or IPs. Instead of detecting one flow according to the initiation and termination of TCP, protocol-based or time window, PairFlow digests all information to extract features later based on the whole context over time.

5.3.1 Architecutre Overview

PairFlow receives raw PCAP data and stores these packets in a buffer until a time window of size t has passed. The buffer sends the current granular data of a time window of all connections at an enterprise network to the *Tracking* module to group unique pairs and label related packets (Figure 5.3, ❶). A unique pair refers to any (possibly bidirectional) connection observed between a host on the local network and a remote server. We take the *source* of the pair to be the local host, and the *destination* to be the remote server. Next, the *Aggregator* module adds a PairFlow ID and time window (Figure 5.3, ❷) to the flow data. The Aggregator module is also responsible for marking packets according to their plane, extracting the domains and HTTP fields. Next, the *Encapsulation* module groups all these pieces of information contextually (Figure 5.3, ❸), so that all possible TTPs in Figure 5.1 can be analyzed later. Therefore, each pair of connections has a comprehensive description of their packets behaviour (described in Section 5.3.4.1), HTTP settings, accessed domains, and cipher suites setting. Finally, PairFlow outputs four additional JSON files which can be used by any external classifier (Figure 5.3, ❹).

5.3.2 Tracking

5.3.2.1 Packets Retrieving.

The tracking module identifies all unique pair connections on the network and filters out those using non-IP protocols. For each unique pair connection, PairFlow tracks, bidirectionally, all packets related to a pair. These packets are designated with an initial Flow ID. The Flow ID holds unchanged for all packets during the same time window for a given pair connection. Each packet will maintain its individual index for the aggregation step later. Packets with the same Flow ID may also use different protocols. Therefore, each one has a one-hot encoding flag

called Encoding Protocol Flag (EPFLAG) used later for further filtering. These flags started with EPFLAG_Protocol, where a protocol is a subset of {TCP, UDP, DNS, ICMP, HTTP, SSL/TLS}.

5.3.2.2 DNS Requests and Responses.

The tracked packets do not include DNS requests and responses, which are responsible for locating the IP address needed to establish a connection. That is due to the pair connection being between the host and the DNS server, which is different than the destination. Similar to [219], to track these DNS packets, a destination of the present pair will be used as a Local PTR to find all DNS response packets from the PCAP repository. Once found, the DNS response resource records will be used to find all related DNS requests. Now, Any packets belonging to the pair connection are attached and sorted according to their arrival time. Those packets outside of time window are not included.

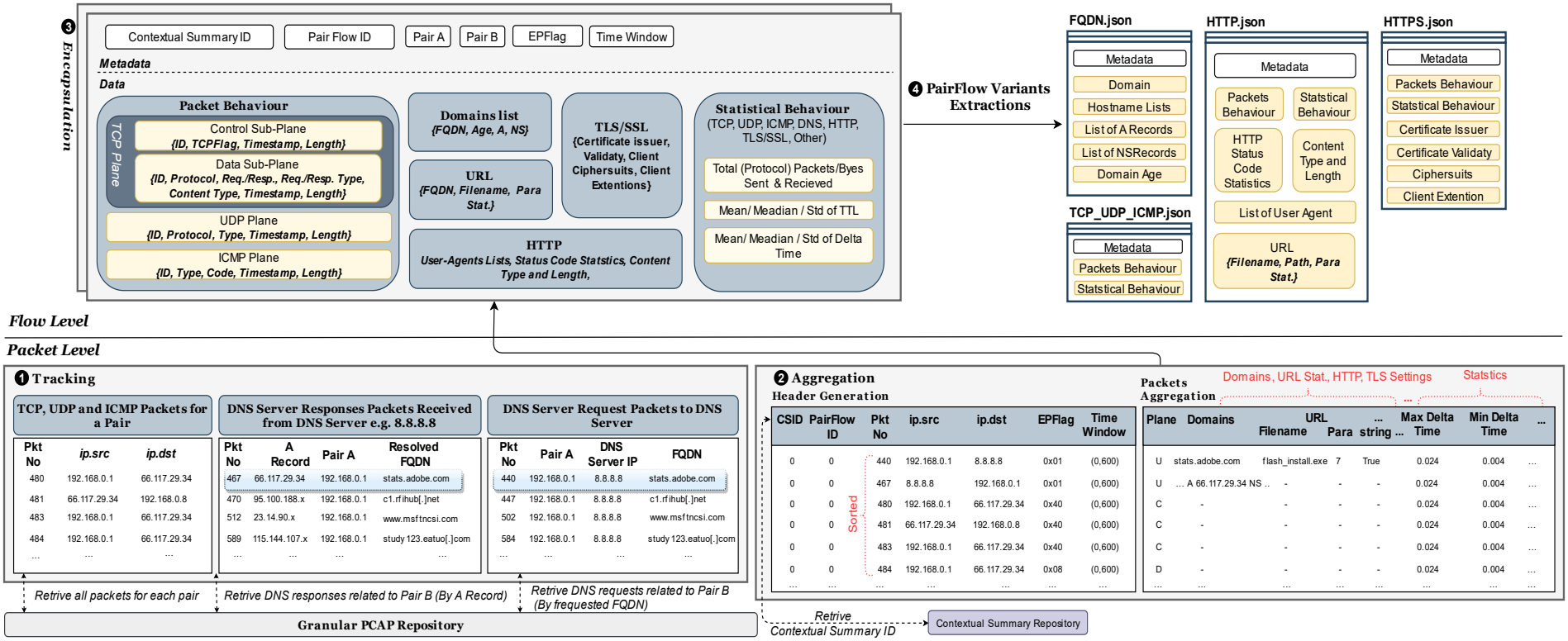


Figure 5.3: Overview of the PairFlow workflow.

5.3.3 Aggregation

5.3.3.1 Header Generation.

Besides the individual packet ID from the PCAP, every packet is also designated with a Flow ID composed of a ContextualSummary ID, described in Section 5.4.4, and a PairFlow ID. The former is unique for the lifetime of a pair, while the latter is unique for a time window. Any packets from that PairFlow will always have the same Flow ID. To assign the PFID, the Aggregation module will check the ContextualSummary repository to find if the pair has been processed in the past. If so, the incoming PFID will be the last used PFID for the same pair and ContextualSummary ID, incremented by 1. Otherwise, a new and unique ContextualSummary will be created, and the PFID will start with 0.

5.3.3.2 Packets Aggregation.

The Aggregator module creates a PairFlow to store PairFlow ID, sorted packet index, pair connection, time window, EPFlag, FQDNs, URL, UAs, SSL/TLS settings, and initial flow-based statistics. The initial flow-based statistics include the number of protocol-based packets (i.e., TCP, UDP, ICMP, HTTP, SSL/TLS, DNS packets), total (encrypted) bytes, and total (encrypted) bytes sent/received. Time-based statistics include packet Time to Live (TTL) and delta packets interarrival time max/min/median and the flow duration at the same time window. Similar to [230], we separate TCP packets into data and control packets to be used later in the encapsulation process. Finally, preprocessed flows are dispatched to the encapsulation step for further processing.

5.3.4 Encapsulation

The encapsulation phase explicitly groups packet behaviour, FQDN and URL, HTTP(S) and initial statistical behaviour implicit in preprocessed flows, in order to make contextual information readily available. The data types involved include a list of strings and tuples, Boolean and numeric fields, as shown in Table 5.1.

5.3.4.1 Packet behaviour

Packet behaviour encapsulates all packets according to their protocol type (TCP, UDP, and ICMP) in a list of tuples. The first element is the packet index for traceability of a given packet inside the original PCAP for further investigation.

The *TCP plane* involves the control and data sub-planes as shown in Figure 5.3. Each packet in the data sub-plane holds protocol name, request/response and their type, content type, timestamp, and packet length for each packet. For example, an HTTP request packet can be described as (460854, 'HTTP', 'Request', 'GET', 'Empty Content', 1066.51, 383) and its response (460895, 'HTTP', 'Response', 200, 'text/javascript', 1066.86, 429). This helps the upper system work on time series traffic and monitor the anomaly for a given PairFlow. Further packet-level statistical analysis such as counting GET/POST, HTTP Response types, content analysis can be achieved as described in Section 5.4.2. The control sub-plane provides the behaviour of the initial connections before the data exchange begins, the TCP continuation, or the termination of the TCP connection. For example, when TCP establishes a connection with three-way handshaking, it will summarise SYN, SNYACK, ACK packets as follows (72095, '0x02', 215.73 sec, 74),(72126, '0x12', 215.78 sec , 70 B), (72127, '0x10', 215.78 sec, 66 B). Then it will follow a stream of packets with TCP flag = 0x10 (ACK) until the connection is disconnected with flag FIN. This will be useful for analyzing any problem with time series or monitoring the discontinuity of such a PairFlow as we can see in Section 5.4.3.

UDP plane records all UDP-based packets with protocol name, packet type, timestamp, and packet length. For example, if there are two packets for DNS which are request and response for a specific domain, they will be summarised as follows: (21160, 'DNS', 'DNS Request', 141.44 sec, 75 B), (21219, 'DNS', 'DNS Response', 141.54, 547 B). *ICMP Plane* is follows the same format to the *UDP plane*. However, the type and code are reporting ICMP settings for each packet. The plane can be helpful for any classifier detecting ICMP-based attacks such as scanning activities.

5.3.4.2 FQDN and URL

As depicted in Figure 5.3, *Domain list* encapsulates all FQDNs related info in a list of tuples. Each tuple holds an FQDN, its A and NS resource records, and the domain age extracted from

the WHOIS file. This helps malicious domain detectors, which often rely on FQDN name, relative DNS zone, and WHOIS files. *URL* encapsulates each relevant element of URL during a connection in a tuple which includes FQDN, web page filename,the number of parameters, values and fragments, and whether it contains encoded strings or not.

5.3.4.3 HTTP(S)

HTTP encapsulates HTTP-level information for a given connection, in particular, distinct HTTP server names, status codes, content types and User Agents. *TLS Protocols* summarises the security settings between a client and server. Cipher suites for both client and server are stored in a list. Cipher suites includes the key exchange/agreement (e.g. RSA, Elliptic-curve Diffie–Hellman (ECDH), Elliptic Curve Digital Signature Algorithm (ECDSA)) , authentication (e.g. RSA), block/stream ciphers (e.g. AES, RC4) with their block cipher mode (e.g. CBC) and message authentication (e.g. MD5, SHA-x). Extension types are also listed for each connection which summarises the cipher suite settings such as extended master secret, session tickets, and Elliptic Curve (EC) point formats. Supported Groups are also stored, known as the EC setting (e.g., secp256r1, secp521r1).

5.3.4.4 Initial Statistical behaviour

It is important to summarise statistically a few essential fields. We calculate max, min, mean packet TTL, delta packets interarrival time, and duration for a given PairFlow. We also calculate the total (encrypted) bytes and the ratio of sent/received (encrypted) bytes. Max, min, median of cipher suites bytes, and server and client extension bytes are also calculated. We also provide a statistical summary of individual protocol numbers of packets such as raw TCP, raw UDP, ICMP, DNS, HTTP, TLS, and SSL.

In Table 5.1, we summarise all fields produced by PairFlow with their datatype. Table 5.2 presents all features used by EarlyCrow with their ID to be referred throughout the discussion in Section 5.6. We also mark the features used in the literature and novel ones.

ID	Field	Type	ID	Field	Type
	I. Informative Fields				
1	Flow ID	N	17	HTTP Servers	LS
2-3	Source & Destination	S	18	Status Codes	LS
4	Packet Data Points	LT	19	Content Type	LS
5	EPFLAG	S	20-21	Server and Client Ciphersuit	LS
6-12	EPFLAG raw TCP, raw UDP, ICMP, DNS, HTTP, TLS and SSL	B	22-23	Server and Client Extension Type	LS
13	FQDN	LS	24-25	Client and Server Signature Algorithm and Hash	LS
14	Nameservers Resource Records	LS	26-27	Client and Server Supported Group	LS
15	A Resource Records	LS	28	ALPAN Next Protocol	LS
16	URL	LT	29	EC Point Format	LS
	II. Statistics Fields				
30	Total Bytes	N	44-48	TTL Max/Min/Mean/SD	N
31-32	Total Sent/Received Bytes	N	49-52	Delta Packets Interarrival time Max/Min/Mean/SD	N
33	Total Encrypted Bytes	N	53-56	Content Length Total/Max/Min/Median	N
34-35	Total Encrypted Sent/Received Bytes	N	57-59	Server and Client Cipher Suites Bytes Max/Min/Median	N
36-42	Number of raw TCP, raw UDP, ICMP, DNS, HTTP, TLS and SSL packets	N	60-62	Server and Client Extensions Bytes Max/Min/Median	N
43	Duration	N			

Table 5.1: Summary of PairFlow data fields (B: Boolean, LS: List of Strings, LT: List of Tuples, N: numerical).

5.3.5 Variants Extraction

PairFlow processing also exports four *variant* JSON files, which can be used by any external classifier, as depicted in Figure 5.3. FQDN.json includes all domains and their hostname lists that have been accessed during a given PairFlow. In addition, resource records such as A, NS are also included, and domain age is extracted from the WHOIS file, which appears to be useful for domain detection [23]. TCP-UDP-ICMP.json is dedicated to those classifiers that use time-series for detection [230, 231, 19]. All three planes are presented here in addition to related statistical fields such as packet TTL and delta packets interarrival time. HTTP.json is employed for those interested in detecting malicious HTTP connections [19, 78]. Other classifiers may deploy HTTPS.json for detecting encrypted communications without deciphering the traffic [219]. The following section will discuss how EarlyCrow used mostly the HTTP variant. A detailed study of the other variants goes beyond the scope of this thesis.

5.4 EarlyCrow

EarlyCrow detects malicious HTTP(S) attacks, and in particular APT malware. We now discuss the architecture of EarlyCrow, and how features are extracted and updated.

5.4.1 Architecture Overview

EarlyCrow is composed of four main processes, as depicted in Figure 5.4. First, it starts with buffering and dispatching using PairFlow. After the PairFlow HTTP variant is generated, these flows are preprocessed for profile pivoting to generate three profiles: Host, Destination, and URL. Then, two types of feature extraction follow (PairFlow and profile features) to form a ContextualSummary which is then classified by a random forest. When another PairFlow is received, it will follow the same workflow. A further step is required when the new PairFlow matches one of the previous ContextualSummary ID in the repository. The ContextualSummary updating process is responsible for updating the matched ContextualSummary to maintain the contextualization and reclassify again. In the following, we describe these four processes briefly.

5.4.1.1 Raw Data Buffering and Data Format Transformation

EarlyCrow maintains a blacklist of IPs and FQDNs in order to block connections as soon as a malicious entity is detected. Otherwise, it stores the packets temporarily in a PCAP repository (Figure 5.4, ❶). Once the time window has passed, these packets are dispatched to the PairFlow module which generates the HTTP variant for the current time window, as explained in Section 5.3.

5.4.1.2 PairFlow Preprocessing

Incoming PairFlows are preprocessed in two ways. First, each PairFlow is individually dispatched to Feature Extraction, to extract additional features from the flow perspective (Figure 5.4, ❷). Second, a list of PairFlow in the same time interval is pivoted to three different profiles. *Host profile* pivots all PairFlow using the source pair (A), *Destination profile* uses

the destination pair (B), and *URL-profile* pivots flows using the FQDN as an identifier. These profiles are dispatched to Feature Space Generation.

5.4.1.3 Feature Space Generation

We have two primary dimensions of PairFlow to obtain features (Figure 5.4, ❸). First, PairFlow features help to understand the overall connection (HTTP and DNS) in terms of the statistical behaviour of requests and responses, their temporal analysis, and content exchanged. Second, the profile-based features include the three profiles above: Host, Destination, and URL. The profile-based features help identify whether a received flow has abnormal behaviour with respect to those profiles. For instance, the Host profile may help identify those infected hosts who access two destinations (two different PairFlows) which may linked to the same APT. Together, they form the ContextualSummary (Figure 5.4, ❹), which includes features from different perspectives to help a classifier for accurate classification. For example, consider a fallback channel scenario, which is common in APT malware. Host A queries a domain and resolves with more than one IP. Hence A may establish multiple unique connections to more than one destination. Another case is when host A communicates to remote server B, which sends another IP address to communicate in parallel with host A, or suspends the first connection and relies on the second as the main channel. The ContextualSummary catches such behaviour, which would be missed by solutions using exclusively flow-based features.

5.4.1.4 ContextualSummary Updating Process

The ContextualSummary has an updating process for the next time window for the matching PairFlow (Figure 5.4, ❺). First, the updating component inspects the ContextualSummary repository to find if a pair has been processed for the incoming PairFlow (Figure 5.4, ❻). Then update rules are applied on ContextualSummary differently according to the two dimensions i.e. PairFlow and profile features (Figure 5.4, ❼).

The rest of this Section discusses in detail the feature space generation and how the ContextualSummary is formed.

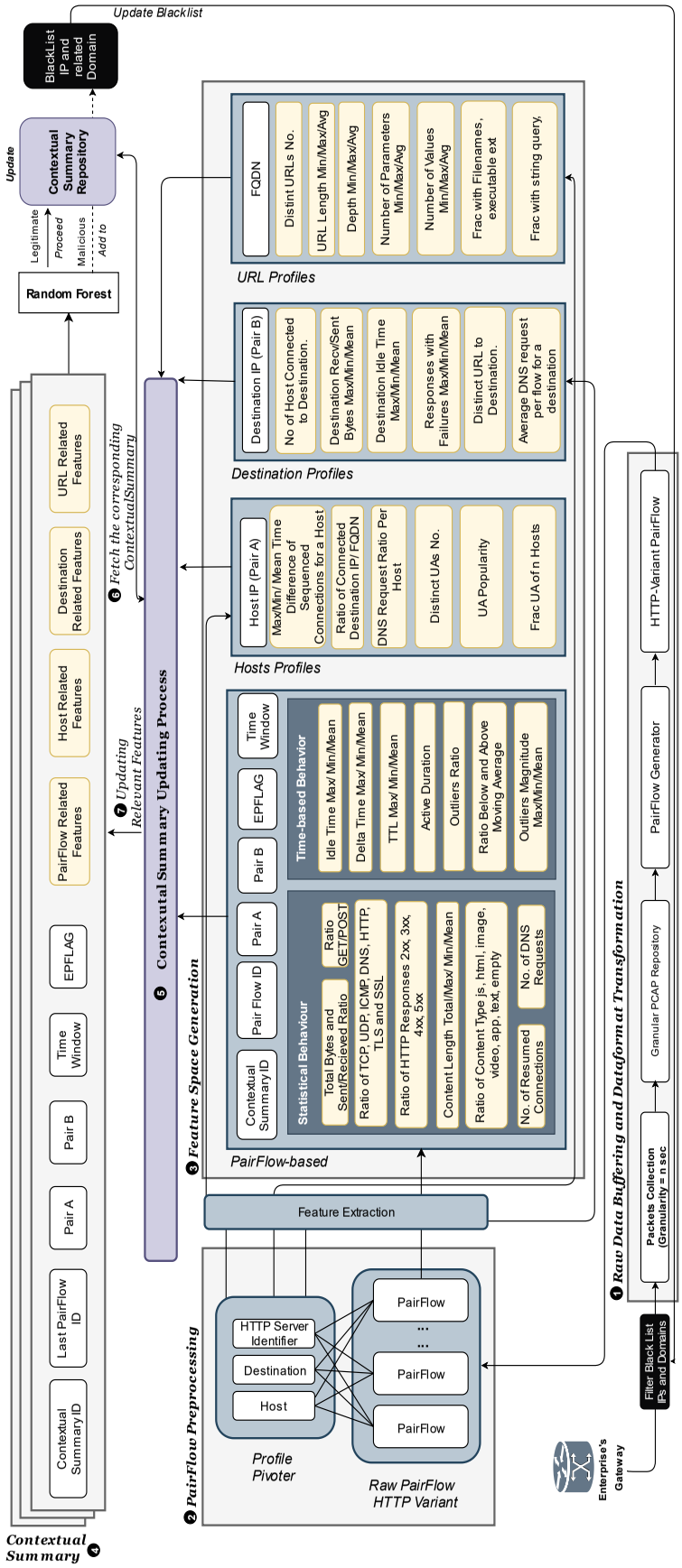


Figure 5.4: Overview of the EARLYCROW architecture.

5.4.2 Flow-based Features

EarlyCrow benefits from using the statistical features produced by PairFlow, which are presented in Section 5.3 and Table 5.1. It also extracts higher-level contextual features from the TCP and UDP planes (Figure 5.3, ❸).

5.4.2.1 Statistical behaviour

Identifying the total exchanged bytes can reveal the average for legitimate connections. Botnets are typically noisy and have higher total bytes during a flow, while APTs have the lowest total to keep their low profile. Another noticeable feature is using raw TCP ratio to detect the *non-application protocols* TTP. APTs tend to have the highest, and use HTTP as camouflage while still relying on TCP for many tasks. Furthermore, identifying the ratio of DNS packets may reveal APTs malicious use of HTTP because it tends to request a domain resolution one time during a connection. Also, the mean delta packets interarrival time may highlight the low profile of APTs because they are more likely to be significantly longer than legitimate connections.

From Data Sub-Plane in Figure 5.3, we calculate GET/POST requests and the fraction of status codes started with 1xx, 2xx, 3xx, 4xx, 5xx to identify the most salient behaviour of such a connection. It is also essential to analyze the status codes at the packet level to identify whether there are frequent HTTP packets failure due to HTTP server misconfiguration when connecting with APT malware. We also notice that measuring the number of resumed connections per flow may highlight the *web protocol* TTP of APT malware. Using the control sub-plane, we count the termination of TCP connection FIN-ACK (0x11) during a PairFlow instead of the sequence of TCP handshaking, i.e., SYN, SYN-ACK, ACK (0x02, 0x12, 0x10) due to the lower computation cost. However, to exclude a typical HTTP flow (e.g., browsing sessions) for reducing FPR, the number of DNS requests during a given PairFlow, using the UDP plane, may reveal such behaviour. EarlyCrow calculates the number of declarations of content types and their ratios to the others in the data sub-plane. Examples of considered types: JavaScript, HTML, image, video, application, and text. APTs and botnets typically have more HTML declarations than legitimate ones, for example due to the use of the *data obfuscation through protocol impersonation* TTP. A typical legitimate connection declares the type without

the need to redefine it every time unless another content type of a web page is used, such as images or videos. Moreover, APTs and botnets rarely use the image content type, which is frequently used in legitimate traffic.

5.4.2.2 Time-based behaviour

The challenge of time-based features is to identify APTs connections that operate in low-profile mode. First, we consider using a couple of time-based features from the PairFlow such as packet TTL, duration of the PairFlow, and delta packets interarrival time, which is the time difference of arrival packets including control, data, UDP, or ICMP packets. We can also measure the data packet exchange idle time using the data sub-plane, the difference between subsequent data packets' arrival time. We measure the max/min/mean data packet exchange idle time. Typical use of HTTP, such as web browsing, has a small gap between delta packet interarrival time and data packet exchange idle time. We suspect that APTs may tune data packet exchange idle time to be shorter/faster because they tend to make delta time to be slower to avoid frequent and unnecessary packets. Once they have to communicate, they send many data packets subsequently, stop for a longer time, and resume later with similar cycles. We investigate such APTs approach in Section 5.6.

We propose additional time-based features that attempt to measure the stealthy behaviour with time-series techniques. We present features based on the simple moving average (SMA). The purpose of the SMA is to average the data points over a time window of size t decided in advance, so that an analyst can identify when a data point is above or below such average. SMA_k can be described as follows:

$$SMA_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i \tag{5.1}$$

where p is the packet length, k is the number of previous data points in a time window, and n is the current data point. Since packets arrive asynchronously, in order to calculate an SMA we need to introduce a sampling rate such that packets arrived within two sampling events are combined together in a single point. For example, if the time window is one minute and we sample points every second, then $k = 60$ and if we receive two packets of length, respectively 128 and 32, between seconds 5 and 6, then $p_6 = 160$. After calculating the SMA, we can extract

the number of outliers and their ratio and magnitude. Outliers are those points two times above the corresponding SMA_k. Therefore, we can capture the stealthy behaviour of APTs, which has fewer outliers than legitimate and botnet traffic. However, it is also essential to find the number of packets below and above average. These features can capture the APTs that touch or slightly exceed the SMA, reflecting cautious operation.

5.4.3 Profile Features

Profiles features are generated based on all PairFlows with longer time windows, for example lasting several weeks, or even months. EarlyCrow queries the related info using a host IP, destination IP, and FQDN for Host profile, destination, and URL profile, respectively. The purpose of the host profile is to identify whether that host has a sign of infection such as discrepant information or fallback channel for example. The destination profile may reflect those destinations that an enterprise can access and avoid some false positives. The URL profile helps identify the typical use of a given FQDN. FQDNs commonly accessed without parameters or values, especially with GET as method type, could signal use of the Dynamic DNS technique to point to frequently changed IP addresses known to be used for APTs [23]. Long URLs with many parameters and values for a given FQDN could instead be legitimate URLs because many hosts request different URLs, indicating rich website access. The URL profile helps pinpoint the malicious use of HTTP protocol from their past behaviour. Yet, APT cannot be easily detected based on such single features, so these will only contribute in part to the final classification.

5.4.3.1 Host Profile

The purpose of the host profile is to investigate the effect of infection on a machine behaviour over $\hat{t}$ time which should be longer than the selected granularity t time for PairFlow. Benign hosts should have specific characteristics in terms of resumed connections, DNS requests per flow, time difference of sequence connections, type of UA used. When a host is infected with APT malware, its characteristics may move to another point further from the benign host centroid. For instance, it is suspicious for a host to initiate a connection by IP only, which is highly linked to a *fallback channel*.

EarlyCrow investigates the number of resumed connections per flow for each host. Legitimate HTTP usage typically increases the number of resumed connections because each time a web page is downloaded, the TCP connection is terminated with FIN. Then, when a host clicks on another link, even for the same website, a new TCP three-way handshake is initiated and completed, followed by a connection termination once the page is completely retrieved. The scenario remains similar for a web caching but with a different destination when a host contacts a proxy server or content delivery network (CDN). Hosts infected with APTs have lower resumed connections per flow than benign hosts. APTs tend to terminate less because HTTP(S) usage is for malicious use, and to stay stealthy, they attempt to avoid frequent connection termination. However, botnets have higher connections terminations than the legitimate and APTs due to their noisy nature. Similarly, we extracted the DNS request per flow to identify a host with lower DNS requests than expected, which is also a sign of APTs using dynamic resolution, DGA and data obfuscation through protocol impersonation TTPs. Hosts infected with botnets are likely to use excessive DNS requests even more than the legitimate ones because of DGA techniques [16] or C&C communication over DNS [232].

We also measure the Mean Time Difference of Sequenced Connections (MTDSC), which can help to identify *fallback channel*. MTDSC can be calculated as follows:

$$MTDSC = \frac{1}{n}\sum_{i=0}^{n} t_{i+1} - t_i \tag{5.2}$$

where n is the number of new connections and t are their timestamp. The input timestamp should be the first packet sent or received from Control, UDP, or ICMP planes for any PairFlow, where the source is the same host. Hosts infected with APTs may have a higher mean for MTDSC than those infected by botnets and benign hosts. They initiate new connections after a long time for their *fallback channel*, which affects the MTDSC even if a host has many normal browsing connections.

We also compute a ratio of connected destinations using IP only to those with FQDN. The feature can capture the APTs behaviour of using DNS requests to locate the IP address of C&C; once the first channel is established, APT malware sends another IP as a fallback channel and starts another three-way handshake. Botnets are less likely to use the same approach, while it is infrequent for legitimate HTTP to be accessed by IP only. As pointed out in Section 5.1,

any client that uses HTTP will have an optional UA in a request packet, and it could be a (non-)browser, malicious string, or just an empty. Similar to [78], we extract several features for UA, including the distinct number of UAs and their popularity among an enterprise.

5.4.3.2 Destination Profile

The destination profile analyzes the servers contacted by internal hosts to find the characteristics of the provided services. We are interested to find out if it is normal for a destination to have fewer/higher DNS requests, short/long data packet exchange idle time, high/low packet failure, sending/receiving dominant, and high/low resumed connections. For instance, we measure the number of DNS requests per flow for a destination to investigate if such destination is using *dynamic resolution, DGA or data obfuscation through protocol impersonation* TTPs. An APT destination usually has fewer DNS requests than usual. Once the domain is resolved and TCP establishment has been completed, it is rare to request more DNS packets. The legitimate use of HTTP(S) is to query the DNS packet every time they visit each page. Therefore, the number of DNS requests is directly proportional to HTTP packets.

It is also essential to measure the destination data packet exchange idle time to identify legitimate web servers with a reasonable time to be idle for browsing. The typical APTs destinations have lower data packet exchange idle time than both legitimate and botnets due to the need for continuous or beaconing communication with lower outliers over a long time. Again, the data packet exchange idle time here focuses only on the meantime of zero data exchange packets from a destination point of view without considering the control ones.

As pointed out in Section 5.1, some APTs use *protocol impersonation* such as HTTP as camouflage to communicate with C&C. Thus, identifying the packet failure for each destination is essential to find if the failure comes from the destination itself here defined in this section or from the PairFlow in Section 5.4.2.1. The objective is to find if a destination mimics web browsing activities while mainly communicating with the victims with raw TCP as *non-application protocol*. Therefore, many APTs destinations have server misconfiguration due to that fake need, causing higher packet failure (response with 4xx or 5xx) than legitimate. We also measure the destination received/sent bytes ratio. APT malware tends to send instructions and payloads to the infected machine more than data exfiltration at the beginning of the

APTs cycle. We measure each destination's receive/sent bytes ratio to identify the APT, and we find that the infected machine has a higher destination ratio than the legitimate ones. On the contrary, botnets tend to exfiltrate data more than sending payloads and instructions.

Another important aspect for each destination is calculating the number of resumed connections. Browsing behaviour has frequently more resumed connections than the APT ones. Each time a user moves to another page, three-way handshaking is established, incrementing the destination of the resumed connection by one. On the APT, once the connection is completed, it is rare to disconnect the communication and resume it later to avoid noisy TCP handshaking. Another important feature is to observe the number of hosts connected to each destination. Popular web servers and botnets destinations are routinely contacted by a considerable number of hosts. In contrast, APTs typically infect as few as possible hosts, hence receiving few connections to their destinations.

5.4.3.3 URL Profile

We present URL-based features which are separate from those in the destination profile, as many FQDN-based URLs share the same IP or vice versa, host several different IPs. The URL profile summarises the standard behaviour of URI resources and the traffic statistics for each FQDN or IP-based URL. We count here how many URLs are reached during a connection and how many are distinct. A malicious C&C server typically has fewer than a legitimate one. However, in case of evasive use of *web protocol*, APTs may have more distinct than botnets. We also check if a URL has a query string, filename, and whether it has an executable extension, then calculate the fraction of the number of each field compared to the distinct number of URLs. A legitimate URL is likely to possess a filename with a variety of extensions. Other statistical features, i.e., Min/Max/Mean, are also calculated on URL length, depth, number of parameters, values, and fragments. The legitimate URL Profile is typically the longest because it has more depth, parameters, and values. APTs come next, followed by botnets.

5.4.4 ContextualSummary

When all features are extracted from PairFlow, and profile-based features are prepared, the ContextualSummary module collects these features in one bundle to be dispatched to the classifier. When a new PairFlow is received, EarlyCrow checks the ContextualSummary repository to identify if the pair had been already processed in the past. If so, the PairFlow will be processed as described in the previous sections. Then, it will be dispatched to the updating process module to combine the new flow with the previous ones as described in the next section. The purpose is to track the same connection over time to catch malicious behaviour. For example, if a malicious actor bypasses EarlyCrow for the first flow, it will be tracked over time until it gets blocked. Indicators associated to positive detections may stay in the ContextualSummary repository and the blacklists for training the classifier. In Table 5.2, we summarise all features included in ContextualSummary.

5.4.5 ContextualSummary Updating Process

While PairFlows are stored in a repository, the ContextualSummary gets updated over time, using different rules for Host, Destination, and URL Profiles. If an incoming PairFlow has no associated ContextualSummary, a new one is created. Otherwise the new PairFlow is considered for feature extraction, causing an update of the corresponding features of the associated ContextualSummary. The time window is expanded with the new PairFlow to describe the overall time window covered by the ContextualSummary. However, updating profile-based features could cause higher time complexity because these profiles are to be updated for every different ContextualSummary. Therefore, new profile-based features are recalculated every $\hat{t}$ time, such that $\hat{t} > t$, where t is the selected granularity for EarlyCrow. For instance, we can configure $\hat{t}$ at 15 minutes in our experimental settings, which is higher than t by 50% if t at 10 minutes.

EarlyCrow considers different methods to update features according to their data type. Numerical features are updated by using a weighted average. As shown in Figure 5.4, each ContextualSummary stores the last PairFlow ID as a counter of previous ones to be used for the weighted average formula. EPFLAG-based features, Boolean data type, are updated with OR operation with an incoming one to summarise the overall protocol used during Con-

ID	Feature	New?	ID	Feature	New?
	I. PairFlow Based Features				
1	Total Bytes	[225, 17, 233]	28-31	No and ratio below and above AVG	✓
2	Sent/Received Ratio	[225, 17, 233, 78]	32-33	No and ratio outliers	✓
3-9	Ratio of raw TCP, raw UDP, ICMP, DNS, HTTP, TLS and SSL packets	✓	34-37	outliers Magnitude Max/Min/Mean/SD	✓
10-13	Ratio of HTTP Response packets with 2xx, 3xx, 4xx, 5xx	[78]	38-40	Data packet exchange idle time Max/Min/Mean	✓
14-15	Ratio of frequent GET and Post	[21, 78]	41	Active duration	✓
16-19	Content Length Total/Max/Min/Median	✓	42-45	Packet TTL Max/ Min/ Mean/ SD	✓
20-26	Ratio of Content Type Javascript, HTML, Image, Video, App, Text, Empty	[78]	46-49	Delta packets interarrival time Max/ Min/ Mean/ SD	Similar to [225]
27	No of resumed connections	✓	50	No of DNS request	✓
	II. Host Profile Features				
51-53	Max/Min/Mean Time Difference of Sequenced Connections	✓	59	Distinct UAs per host	✓
54	Ratio of Connected Destination IP only to FQDN	✓	60	nAvg UA Popularity	[20]
55-57	Max, Min, Mean of Resumed Connections per flow for a host	✓	61-62	Frac UA 1, 5	[20]
58	No of DNS request per flow for a host	✓	63	Ratio UAs	[20]
	II. Destination Profile Features				
64	No. of hosts connected to destination	[20]	72	No. of Distinct URLs associated to destination	✓
65-67	Destination Received /Sent Max/Min/Average	✓	73-75	Destination Max/Min/Mean Packets Failure	✓
68-70	Destination Data Packet Exchange Idle Time Max/Min/Mean	✓	76-81	Max/Min/Mean No and ratio of DNS request per flow for a destination	✓
71	No of resumed connections per flow for a destination	✓			
	II. URL Profile Features				
82	Frac URLs filename	[78]	94-96	URLs Values Max/Min/Mean	[225, 21, 78]
83	Frac URLs filename exe	✓	97-99	URLs Fragments Max/Min/Mean	[78]
84	Number of distinct extensions	[78]	100	Frac query	[78]
85-87	URLs Length Max/Min/Mean	[225, 21, 19, 78]	101	Num hasString	✓
88-90	URLs Depth Max/Min/Mean	[225, 21, 78]	102	Num of URLs and Distinct ones	[234]
91-93	URLs Parameters Max/Min/Mean	[225, 21, 19, 78]			

Table 5.2: EarlyCrow features. Note that features reused from the literature are computed from PairFlow data rather than from other data formats.

textualSummary. For instance, APTs often have the DNS packets at the first PairFlow, but not the subsequent one, as we discuss in Section 5.1. Therefore, updating the ContextualSummary does not reset EPFLAG for the DNS. For Host-Profile features, strings of UA are stored to accurately extract other related UA, such as the number of distinct UAs which cannot be updated without having access to their strings.

Label	#Packets	Set	Malware Families
Malicious	567,090	Training	Bitsadmin (0.09%), Carbank (0.05%), Conficker (27.56%), Mivast&Sakula (0.93%), NanoCore (0.13%), njRAT (28.45%), PlugX (0.11%) , Remcos (0.87%), Sogou (3.65%), Virut (9.59%), Zebrocy (0.98%),
		Testing (Unseen)	Ammyy (1.01%), ChChes (0.13%), CobaltStrike (0.39%), Dridex (0.23%), Emotet (0.02%), Empire (1.70%), FlawedAmmy (0.24%), ImminentMonitor (11.27%), MagicHound (0.40%), OnionDuke (0.14%), PoisonIvy (0.25%), Ramnit (0.21%), StrongPity (11.38%), Zeus (0.04%)
Legitimate	766,641	-	Training: 70%, Testing: 30%.

Table 5.3: Dataset characteristics used for measurements (training set) and for unseen malware evaluation (testing set).

5.5 APT Malware Dataset

APT malware attacks a few targets in discontinuous time-frames spanning months or years, unlike other malware and common attacks. Therefore, the chance of finding a real network infected with various APT campaigns is unrealistic. We resort to raw PCAP captures from two different honeypot networks, each of which includes legitimate, APTs and botnets C&C connections (Table 5.3). After PairFlow compiles the PCAPs and generates HTTP variant files, we build three combined datasets: APTs vs. Legitimate, botnets vs. Legitimate, and Malicious (APTs or botnets) vs. Legitimate. We then causally analyze these datasets and select appropriate mitigation techniques for bias and spurious correlations.

5.5.1 Captures

APTraces:

We run different active APT malware using Any.Run[2] sandbox machines to generate PCAP files. These malware families are known to be used by 48 APT campaigns[3] (Table 5.4). However, they are often temporarily inactive. Due to this, we run them during multiple time windows (October - November 2019, April 2020 - January 2021) until each campaign's activities are resumed, and their command and control are activated. Malware families included in this dataset are RATs (*njRAT, Imminent Monitor, CrossRAT, Mivast & Sakula, NanoCore, PlugX,*

[2]https://any.run

[3]Campaigns use each APT malware can be found at https://attack.mitre.org/groups/

PoisonIvy) and trojans (*Empire, OnionDuke, MiniDuke, Remcos, StrongPity, Zebrocy*). We have already discussed the meaning of APT malware and how they are related to campaigns in Section 3.2, 87. We also consider legitimate connections from the same sandbox to avoid data bias based on the victim machine, configuration settings, or temporal bias [235] against legitimate.

Malware Capture Facility Project (MCFP):

The MCFP[4] is a repository of 349 public captures of different malware families provided by Stratosphere Research Laboratory. We select malware used in APTs such as (*Magic Hound and Cobalt*), admin tools (*Ammyy*), and RATs (*njRAT*). We also add botnets captures that use HTTP(S) for C&C communication (*Conficker, Dridex, Emotet, Ramnit, Sogou, Virut, Zeus*) and normal traffic (CTU-Normal-12, 20-22).

Malware	Used in
Ammyy	FIN6 and TA505
ChChes	APT10
Cobalt Strike	APT10, APT19, APT29, APT32, APT37, APT41, DarkHydrus, Chimera, Cobalt Group, CopyKittens, FIN6, FIN7, Indrik Spider, Leviathan, Mustang Panda and Wizard Spider
Empire	APT19, APT33, APT41, CopyKittens, FIN10, Frankenstein, Indrik Spider, Leviathan, MuddyWater, Silence, Turla, Wizard Spider and WIRTE,
Imminent Monitor	APT-C-36
Magic Hound	APT35
MiniDuke	APT29
Mivast & Sakula	Deep Panad
NanoCore	APT33, Gorgon, Group5 and SilverTerrier
njRAT	APT41, Gorgon, Group5 and Transparent Tribe
OnionDuke	APT29
PlugX	APT3, APT10, APT27, APT41, DragonOK, GALLIUM, Higaisa, Mustang Panda and TA459
PoisonIvy	admin@338, APT1, APT10, DragonOK, Elderwood, GALLIUM, IndigoZebra, Moafee, Molerats, Mustang Panda, PittyTiger, Soft Cell and Tropic Trooper
Remcos	Gorgon
StrongPity	PROMETHIUM
Zebrocy	APT28

Table 5.4: A list of potential campaigns using APT malware presented in APTraces and MCFP.

5.5.2 Causal Analysis and Control Measures

Causal analysis is a recommended approach to make sure of data quality in machine learning, especially for predictive tasks. In this section, we diagnose our dataset based on the causal

[4]https://www.stratosphereips.org/datasets-overview

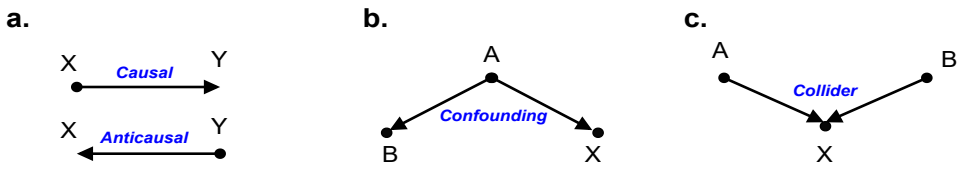


Figure 5.5: Causal reasoning relationships [11].

analysis and provide the control measures. We follow D. Castro et al. [11] work for the definition of causal analysis and methods to ensure data quality for our dataset.

5.5.2.1 Causal Analysis

A causal diagram is a directed acyclic graph (DAG) that represents the cause-effect relationship between variables considered in a predictive task, and helps elicit or mitigate bias from experiments. A link between a pair of edges can be causal or anticausal. A causal relationship is described by $P(Y|X)$ when $X \rightarrow Y$, where X is a sample and Y is the label. Literally, 'X is a direct cause of Y, which means modifying the value of X will change the likelihood of Y (Figure 5.5.a). Causal links refer to predicting the effect (Y) from the cause (X). In contrast, anticausal, $P(Y|X)$ when $Y \rightarrow X$, meaning anticausal links predict cause from the effect such as predict X (sample) from Y (label) as depicted in Figure 5.5.a. Clearly, anticausal is out of our scope because it does not influence the data-generating process of our dataset, which will use X (ContextualSummary) only on predicting Y (APTs, botnets and legitimate). Therefore, we build a causal model focusing on causal links as depicted in Figure 5.6. We attempt to balance between accuracy and clarity on the level of abstraction. We aim to identify the bias, including confounder and collider, in our dataset and control them. We will explain Figure 5.6 after introducing some related concepts.

Confounder may be an unadjusted common cause, while collider involves conditioning a common effect such as selection bias. Let us consider that A (say, PairFlow) is a common cause of B and X, a Destination Profile and a ContextualSummary, respectively (Figure 5.5.b). A is called a confounding factor, and it creates a spurious correlation between B and X, resulting in $B \not\perp\!\!\!\perp X$ (literally, "B is dependent on X"). However, they become independent when A is controlled: $B \perp\!\!\!\perp X|A$. Consider the instance when X is a common effect of both A and B (Figure 5.5.c). We call X as a collider in this case. Conditioning on X adds a connection

between A and B, as they can now explain away the influence of each other on the observed result, X (i.e., $A \perp\!\!\!\perp B|X$).

Due to the various capture environments for our considered dataset, it is vital to control collider and confounder caused by data shift. The data shift in causal direction can be population, annotation, acquisition shifts, or sample selection bias. Let us assume Z is a vector of the PCAP raw features. *Population Shift* may occur when $P_{train}(Z) \neq P_{test}(Z)$. We ensure to mitigate such bias by focusing on samples that belong to the same protocol - in our case HTTP(S) - have to be used in each sample. *Annotation Shift* is given by $P_{train}(Y|X) \neq P_{test}(Y|X)$ where samples are labelled based on different metadata. For example, APTrace may include a malicious flow that is also presented in MCFP with a different label source. In this case, control measures can be taken, such as relabelling all flows by referring to the same security intelligent platform source.

Acquisition Shift can occur due to different platform configurations, which may affect several features, including packet TTL, duration, and UA-based features. Data harmonization, such as extracting domain-invariant representations, is the foremost approach to mitigate such bias to validate $\vec{F} \perp\!\!\!\perp P_f$, where $\vec{F}$ is a feature vector, and P_f is the platform. *Sample Selection Bias* is when $P_{train}(X, Y) \not\equiv P_{test}(X, Y)$, meaning the training set does not represent the target population. We can avoid it with a random sample selection with stratified training and testing split for each class.

In Figure 5.6, we point out all possible cause and effect links, where some are shaded with specific colours reflecting the type of potential bias as annotated in the legend box. Y (Label) is a common effect of three causes, threat intelligence, IoC, and creator of raw capture. Y is a collider, and the potential bias is the *annotation shift* because we may have conflict among labels based on different sources. The platform configuration is a common cause of UA, Time, and Raw PCAP variables. In this case, we call the platform configuration variable a confounder that produces an association between all three variables. The type of shift between these variables is *acquisition shift* due to the different platforms that we collect our captures from. The encryption variable is also a confounder. It is a common cause of UA, URL, control, and data sub-planes. For instance, UA and URL are hidden inside the payload when encryption is applied. The details of the data sub-plane are not available without decryption, leaving the data plane with SSL/TLS packets. The potential bias is the *population shift*. Another population shift is caused

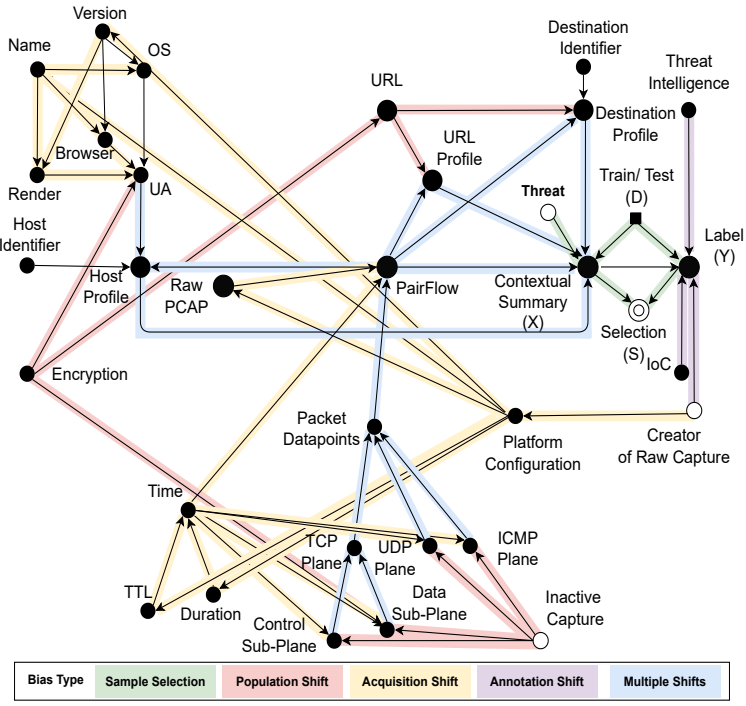


Figure 5.6: Causal model diagram. Empty circles denote hidden variables, whereas solid circles refer to explicit ones.

by the inactive capture variable, which is also a confounder of control, data, UDP, and ICMP planes. When a capture is collected, but there is no sign of attacks or communication with C&C it affects the behaviour of these planes to act as legitimate or relatively low number of malicious packets exchange. When two or above shifts are possible, we call such a link as *multiple shifts*. However, mitigating the individual bias will reduce the multiple shifts.

5.5.2.2 Causal Control Measures

We take several control measures to improve our datasets motivated by the causal model reported in Figure 5.6. *Population shift:* We exclude any flows that do not use HTTP from the inputs to the PairFlow-HTTP variant. We limit all captures to the same duration for the duration-related features to mitigate the bias of packet statistics and the duration of a flow itself. For instance, it is biased to compare a flow captured for 15 minutes to the other during 2 hours, which may influence the statistics for the benefit of a longer capture. We also consider recently recommended practices to troubleshoot datasets for IDS [236]. Therefore, we removed samples without data transfer or failed TCP connections to C&C servers.

Annotation Shift: The sandbox already generates an IoC file in APTrace. However, we label those missed via sandbox by manually checking every IP and domain using other threat intelligence platforms and APTs industry reports following the recommendation by [235]. For MCFP, we label the flows based on the original project and manually label them using historical data retrieved from security intelligence platforms, including *IBM X-Force Exchange*[5] and *AlienVault Open Threat Exchange*[6]. Finally, we need to mitigate the bias for the Host profile in EarlyCrow through various captures. Instead of considering source IP as a host identifier, we used a tuple of $(multiple\ labels, source)$ to avoid mixing data with the same internal IP but in a different network or capture. The multiple label field refers to the filename of a PCAP capture.

Sample selection bias: To mitigate the bias caused by APTs only as a malicious class, we added botnets C&C captures to measure if our system can detect both activities. Then we stratified the train and test process to ensure $P_{train}(X, Y) \equiv P_{test}(X, Y)$. In Section 5.6, we report the weighted and macro average F score due to the binary labels' imbalanced samples and shed light on malicious flow detection.

Acquisition Shift: The differences in platform configuration cause bias. We consider only features with domain-invariant representations to ensure $\vec{F} \perp\!\!\!\perp P_f$. Therefore, we neglect features on names and versions of OS, Browser, and renderer. We keep the distinct number of UA per host and its popularity-based features. Packet TTL is another domain variant, which is dependent on the platform. We omit features based on TTL, although we kept it available in PairFlow for the research community but not for our assessment in the next section.

5.6 Results

This section evaluates EarlyCrow performance on the three datasets described in Section 5.5, each with a split of 70% for training and 30% for testing. The same experiments are performed on a baseline, inspired by Made [78], which is a NIDS detecting C&C used by botnets, ransomware, and APTs. Since we focus on detecting Case I and II in Section 5.1, we omit inapplicable features, such as those related to domains, that would be missing for Case II.

[5]https://exchange.xforce.ibmcloud.com
[6]https://otx.alienvault.com

	Macro			*Weighted*				
Classifier Name	mP	mR	$mF1$	wP	wR	$wF1$	FPR	A
I. Dataset: APTs vs. Legitimate								
EarlyCrow	**94.20**	**93.69**	**93.89**	**97.55**	**98.41**	**98.21**	**0.40**	**99.17**
Baseline	92.40	89.09	90.45	95.52	98.35	96.85	0.49	98.75
EarlyCrow-HTTPS	92.85	93.18	92.79	97.98	98.34	96.53	0.51	99.03
Baseline-HTTPS	82.90	68.96	73.18	91.81	95.31	93.04	0.72	97.19
II. Dataset: Botnets vs. Legitimate								
EarlyCrow	**96.49**	**95.40**	**95.90**	**99.23**	**99.50**	**99.36**	0.48	**98.92**
Baseline	94.64	86.92	90.24	98.23	98.91	98.53	0.57	97.61
EarlyCrow-HTTPS	96.79	95.02	95.84	98.55	99.47	99.00	**0.42**	98.92
Baseline-HTTPS	90.73	80.76	84.79	96.10	98.66	97.28	0.96	96.39
III. Malicious vs. Legitimate								
EarlyCrow	**95.41**	**94.79**	**95.06**	**97.59**	**98.18**	**97.87**	**0.86**	**98.29**
Baseline	93.76	88.20	90.68	94.52	97.38	95.84	0.93	96.97
EarlyCrow-HTTPS	95.07	94.76	94.89	97.09	98.17	97.62	0.93	98.23
Baseline-HTTPS	91.21	78.66	83.47	90.60	95.53	92.57	0.95	95.13

Table 5.5: Known malware classification performance.

Classifiers are evaluated in two modes. First, HTTP-Mode, which assumes the administrator connects the NIDS to a web proxy to decrypt HTTPS and accesses features such as UA, HTTP response codes, content type, and URL. Second, HTTPS-Mode, where the administrator places the NIDS at the network edge, without deciphering HTTPS. Because of imbalanced classes of APT (3.9%) and botnet (8.3%) compared to legitimate, we focus on macro average F1-score ($mF1$), Precision (mP), and Recall (mR) in Tables 5.5 and 5.6.

5.6.1 Known Malware Classification Performance

In this experiment, we randomly split the training and testing sets ten times. Then, we take the average performance under two constraints. First, the malware should be presented in both sets. Second, the infected hosts and the destination C&C server should be unique and not leaked from training to testing.

EarlyCrow obtains the best performance with $mF1$ of 93.89%, 95.9%, and 95.06% for the three datasets. The performance of EarlyCrow continues to achieve higher than baseline for mP and mR. We also note that the baseline achieves almost similar for APT and botnet (-0.21 $\Delta mF1$), while EarlyCrow achieves better against the botnet (+2.01 $\Delta mF1$). This confirms that improving APT detection can help detect another type of malware, such as botnets. The FPR of EarlyCrow are minimised to 0.40%, 0.48% and 0.86%, which is also lower than the baseline. Therefore, The detection performance on known malware for EarlyCrow is not at

the expense of higher FPR, which is critical to observe for SOC analysts.

Even in HTTPS mode, which cannot take advantage of plaintext HTTP features such as headers or URL details, EarlyCrow still outperforms the baseline on both three tasks, scoring 92.79%, 95.84%, and 94.89% of $mF1$, respectively. Analogous to HTTP mode, EarlyCrow-HTTPS achieves better $mF1$ against botnets compared to APT (Δ +3.05%). Note that EarlyCrow can operate almost similarly on both modes on known malware. However, we observe the $\Delta mF1$ of the baseline between the two modes for the three tasks are 17.27%, 5.45%, and 7.21%, which is a significant loss compared to EarlyCrow, which are only 1.1%, 0.06%, and 0.17%. However, we will investigate that on unseen malware in the next section.

5.6.2 Unseen Malware Classification Performance

For this experiment, we train our classifiers on the training set used for our measurement study. Then we evaluate the performance against the unseen malware described in Table 5.3. EarlyCrow obtains the best performance with $mF1$ of 93.02%, 94.25%, and 93.11% for the three datasets. While the baseline can achieve better mP and mR in a single task, the other tasks Δ's range fluctuates from 23.04% to 19.79% compared to the range from 4.76% to 3.17% for EarlyCrow. The primary reason for the considerable gap is the baseline struggles to detect unseen malware samples. For example, mR against APTs is 75%, while EarlyCrow achieves 91.67%.

On all three tasks in HTTPS mode, EarlyCrow surpasses the baseline, achieving 93.72%, 87.12%, and 91.54%. Note that EarlyCrow can operate almost similarly on both modes on unseen malware for the first and third datasets. However, the $mF1$ is decreased when detecting only botnets with a margin (-7.13%) compared to a severe loss for the baseline (-35.95%). Also, the Δ between the mP and mR is relatively stable for the APT (1.87%). However, the gap is growing faster against botnets (14.01%). The same conclusion can be held for the baseline, where the difference is even larger (39.88%). We discuss of the importance of using HTTP features for APT, the chances to improve the detection against botnet in HTTPS mode, and how that affects the FPR performance in Section 5.6.7.

	Macro			*Weighted*				
Classifier Name	*mP*	*mR*	*mF*1	*wP*	*wR*	*wF*1	*FPR*	*A*
I. Dataset: APTs vs. Legitimate								
EarlyCrow	94.48	91.67	93.02	98.07	98.11	98.08	0.74	98.11
Baseline	**98.04**	75.00	82.33	96.37	96.22	95.63	**0.00**	96.22
EarlyCrow-HTTPS	94.68	**92.81**	**93.72**	**98.25**	**98.28**	**98.26**	0.74	**98.28**
Baseline-HTTPS	96.70	56.82	60.29	93.90	93.47	91.10	**0.00**	93.47
II. Dataset: Botnets vs. Legitimate								
EarlyCrow	**96.77**	**92.01**	**94.25**	**99.25**	**99.26**	**99.24**	0.19	**99.26**
Baseline	95.08	78.85	85.06	98.26	98.35	98.16	0.19	98.35
EarlyCrow-HTTPS	95.49	81.48	87.12	98.46	98.53	98.40	0.19	98.53
Baseline-HTTPS	48.25	50.00	49.11	93.14	96.51	94.79	**0.00**	96.51
III. Malicious vs. Legitimate								
EarlyCrow	94.77	**91.60**	**93.11**	**97.45**	**97.51**	**97.46**	0.93	**97.51**
Baseline	**95.89**	76.10	82.62	94.96	94.85	94.15	0.19	94.85
EarlyCrow-HTTPS	94.27	89.22	91.54	96.92	97.01	96.92	0.93	97.01
Baseline-HTTPS	95.22	54.76	56.18	91.44	90.53	86.86	**0.00**	90.53

Table 5.6: Unseen malware classification performance.

5.6.3 Features Diversity

Detecting APTs necessitates a spread of features, as presented in Section 5.1. In Figure 5.7, we show the extent to which additional features affect the performance of the various classifiers based on the second experiment (Section 5.6.2). The first 10% of features for EarlyCrow show rapid improvements in terms of precision but with poor recall. Also, stronger features between 48% and 62% can improve the performance of *mF*1 up to 92.26% for EarlyCrow-HTTPS. Adding more features afterwards increases the detection rate, enabling more unseen APTs to be detected. Furthermore, a detection system with diverse and strong features will require more time and resources to evade as opposed to one that relies on a few particular features [237].

5.6.4 Feature Importance

We investigate the feature importance of the experiment of the third dataset in HTTPS mode, which comprises APTs, botnets, and legitimate samples, because it is the one closest to a realistic scenario for APT hunting. Figure 5.8 illustrates the top features based on their information gain. MTDSC is an effective feature that shows that 82% of hosts infected with APTs and botnets spend up to 73.7 sec and 38.5 sec, which are higher than those for legitimate connections that spend up to 1.1 sec, confirming the typical HTTP browsing behaviour. The longer time for APTs indicates the use of *fallback channel*, which is generally established after a long

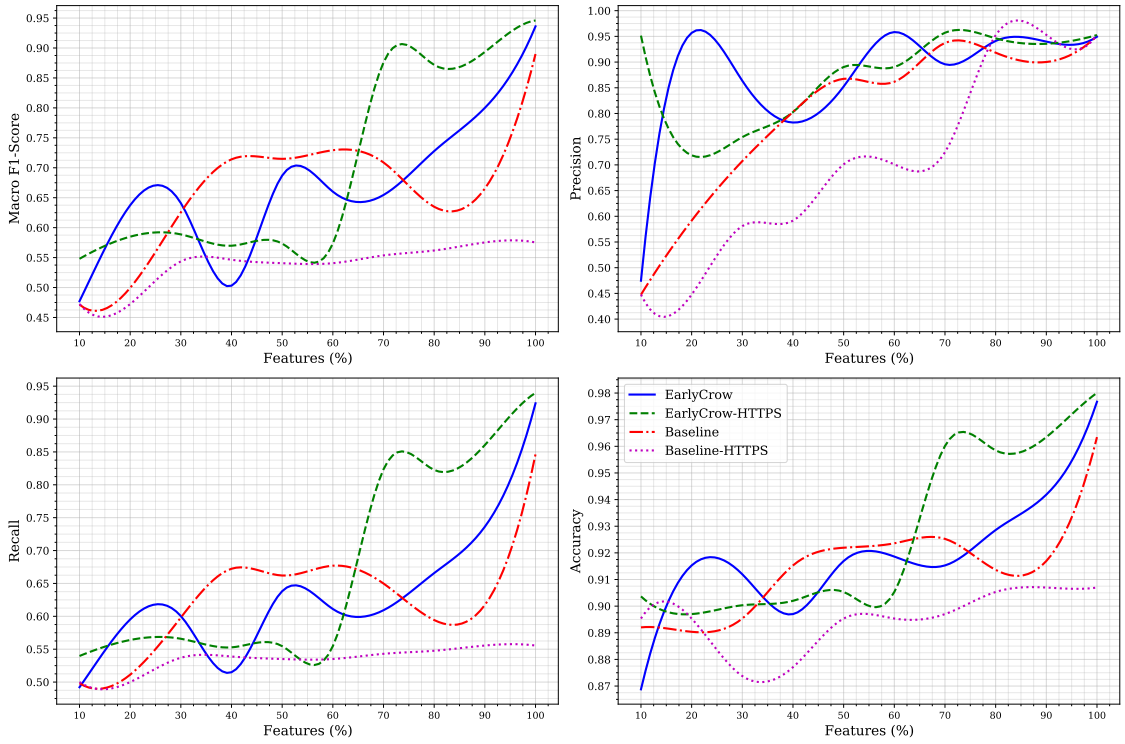


Figure 5.7: Effect of using only the top % of features.

time. Next, 90% of DNS requests for host per connection are lower in APTs than botnets and legitimate with 2, 6, and 19, respectively. Interestingly, hosts infected with APTs have higher connections reaching further destinations by IP without *domain resolution* as *fallback channels*. This is consistent with our measurement study in Section 5.2. 70% of APTs use such an approach with 88% or less for their connections while only 1% for the legitimate.

While 60% of legitimate connections are resumed six times or less, botnets and APTs are rarely disconnected, with two-thirds lower. This confirms the malicious connections are weakly imitating legitimate behaviour (*web protocol TTPs*). Next, the mean delta interarrival time between packets during a PairFlow shows that APTs and botnets are significantly slower than legitimate, i.e., at 95%, the mean delta time at most 33.5×10^{-2}, 46×10^{-2} and 0.5×10^{-2} sec, respectively. We confirm that APTs tend to switch from HTTP to raw TCP for malicious operations (*non-application protocols*). Within a PairFlow, we find 50% of APTs rely on 81.09% (58.35% for legitimate) of the whole exchange packets on raw TCP, indicating the adversary uses HTTP as camouflage while still relying on TCP for many tasks. Nonetheless, the APTs and botnets are faster regarding the difference between data packets. APTs and botnets tend to be shorter/faster than legitimate, where 90% of them take 104, 124, and 168 sec, respectively.

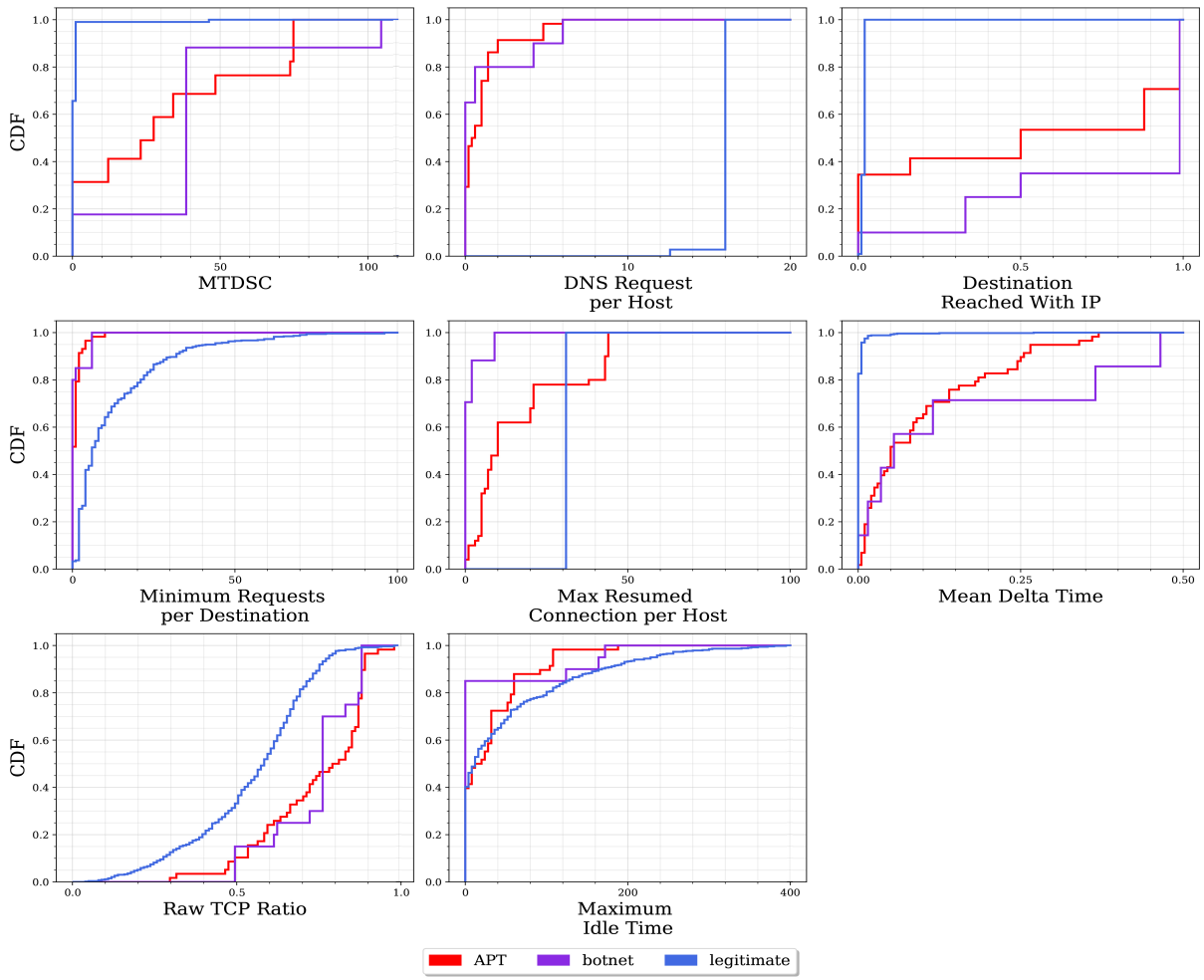


Figure 5.8: Cumulative distribution of top features gains on the testing set for EarlyCrow-HTTPS.

5.6.5 Attacks Investigation

In Table 5.7, we breakdown the results of EarlyCrow-HTTPS on unseen malware. 92% of unseen malware are detected with at least one C&C communication, and 64% of different malware are fully detected. However, one server belonging to StrongPity is not detected at HTTPS. We found that StrongPity does not use a fallback channel, and its measurement reflects a legitimate one. At HTTP mode, EarlyCrow managed to detect StrongPity because of its malicious URL characteristics, such as using `.exe` file extension and lack of rich web server (i.e., No. of URLs distinct) compared to the data exchanged. Also, some C&C servers belonging to OnionDuke and Zeus manage to evade detection. These servers are established as fallback channels with minimum data transfers, evading many features. Since the malware is detected on a specific machine, we recommend a SOC analysis to sanitize the victim machine from the malware to stop other possible C&C communications.

Malware	C&C Servers	Detection (%)	Malware	C&C Servers	Detection (%)
Ammyy	8	100	ImminentMonitor	4	75
ChChes	1	100	Magic-Hound	3	100
CobaltStrike	2	100	OnionDuke	6	33.34
Dridex	2	100	PoisonIvy	1	100
Emotet	13	53.84	Ramnit	2	100
Empire	5	100	StrongPity	1	0
FlawedAmmy	4	100	Zeus	3	33.34

Table 5.7: Detection rate on unseen malware over HTTPS.

5.6.6 Features and TTPs Correlation

In Figures 5.9.a-c, we present a heatmap on the third dataset based on the unseen malware classification experiment (Section 5.6.2). We confirm that legitimate traffic uses some APT TTPs, increasing the overlap, which confirms our findings of the measurement study. With the help of other features, detecting the malicious one can be accomplished. For example, many legitimate connections can use *encrypted channel*. It has a major overlap between all classes; therefore, EarlyCrow is not able to detect the malicious use of such TTP. However, the encrypted channel is ineffective in hiding other TTPs against EarlyCrow because several of our features do not rely on deciphering the HTTPs payload. We find that features (#33, #64, #68) have a solid signal to separate APTs from botnets, but not legitimate ones. The number of connected hosts to a destination shows the APT mimic the legitimate, unlike botnets, where the infection spreads throughout several hosts. Feature (#58) does not show correlation to *web protocol* in APTs and legitimate due to the standard use of DNS requests ratio to the response. Still, botnet shows a higher negative correlation due to DGA use. We also identify minor overlaps in further TTPs. For instance, *Targeted and stealthy* TTP has a small difference in linear correlation between APTs and legitimate (+0.73, +0.41). Both APTs and legitimate behave similarly in terms of points below the average of SMA; however, the linear relationship is slightly larger in APTs than in legitimate (+0.87, +0.64).

On the other hand, other features are powerful in identifying a separable correlation of malicious use of TTPs between classes. We find feature (#55) and *web protocol* are negatively correlated with APTs and botnets but almost no correlation with legitimate. Feature (#54) can identify the *protocol impersonation* in APTs and botnets while the legitimate seems to not correlate with it. Non-Application Protocol TTP has a large positive correlation to APTs (+0.82), while the legitimate has weaker ones (+0.35) due to the common use of frequent Raw TCP in APTs.

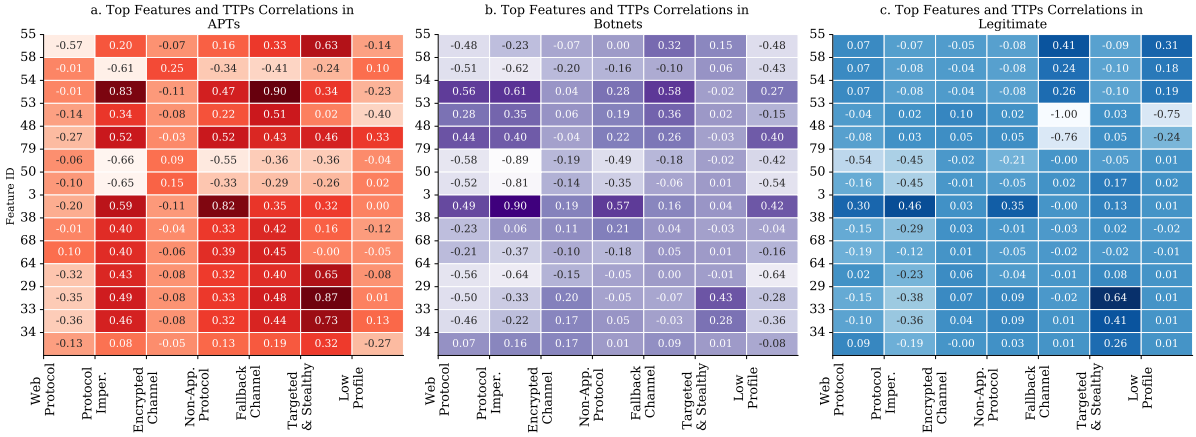


Figure 5.9: Heatmap for EarlyCrow-HTTPS.

Finally, it should be known that none of these correlations is used for detection due to some of the possible spurious correlations. For instance, a spurious correlation can be found between the feature (#55) and *fallback channel*. Thus, we use it for analysis only to explain the differences in relationships between features and TTPs across all classes.

5.6.7 Discussion

As presented previously, this section discusses the performance of all classifiers and highlights the difference between known (Section 5.6.1) and unseen malware (Section 5.6.2). For EarlyCrow, the performance loss ($\Delta mF1$) between known and unseen is marginally low (1.97%) in the third dataset, while the baseline suffers a loss of 9.6%. Likewise, the performance loss ($\Delta mF1$) for EarlyCrow between the two modes of known malware compared to unknown malware is +27.27% from 1.1% to 1.4%. For the baseline, the Δ is dramatically increasing by up to +108.16%, namely, from 17.27% to 35.95%.

As pointed out in Section 5.6.2, we observe a high difference between mP and mR against unseen botnets, which is not the case for unseen APTs. We conclude that the plaintext HTTP features play a center role in detecting unseen botnets. This is also the case for both APT and botnet detection for FPR reduction, where these features can reduce the FPR by nearly %50 for EarlyCrow and the baseline. That suggests more work on HTTPS mode has a high chance to improve the stability on differences between mP and mR, in addition to the FPR reduction close to EarlyCrow-HTTP. We suggest investigating if using time series techniques based, such as DSP, on the interarrival time of packets as a time series problem can improve

the HTTPS-mode classifier. Using suggested techniques can also help cover Cases III and IV discussed in Section 5.1. We will investigate such approaches in the next chapter.

5.6.8 Limitations

In the early stages of an infection, it is difficult to tell how many suspicious activities are linked to more advanced attacks and how many are mainstream malware variants because EarlyCrow is geared toward detecting the first stages of infection. We recommend tracking the malware activities over a longer period of time with various APT campaign usage. This could be done by placing EarlyCrow in a targeted entity such as a sovereign entity or large financial institution network over months and reevaluating EarlyCrow reports by security experts for further improvement. There are also other limitations which may overlap with Chapters 4 and 6, which are grouped together as the limitations of this thesis in Chapter 7.

5.7 Related Work

There is very limited previous work on detecting APTs at the network level. Detecting C&C in general is the closest area. In our approach we test several features from the literature which can be relevant for APTs, including URLs and UA features [78, 20, 234, 21], traffic exchange bytes [225, 17, 233, 78], HTTP content types [78, 20], and GET and POST ratio [21, 78]. Besides directly using such features, EarlyCrow pivots them into host, destination and URL profiles, and combines them in contextual summaries.

Some previous works focus on detecting APTs in addition to other kinds of malicious communications [20, 78]. Oprea et al. [20] propose a belief propagation (BP) algorithm to detect early-stage infection of APTs. They model enterprise communication using a bipartite graph with two vertices, hosts, and domains based on simulated attacks. Once the detector identifies a malicious remote host or domain based on several features, BP identifies communities of malicious domains with similar features that are part of the same attack campaign. Domain scores are calculated as a supervised linear regression weighted sum of features. As discussed in Section 5.6, APTs tend to infect a lower number of hosts than botnets. Therefore, EarlyCrow considers other features based on different TTPs discussed in Section 5.1. EarlyCrow

is closer to MADE [78], which instead uses web proxy logs at the edge of an enterprise network to detect malicious C&C communications, including APTs. MADE leverages features related to the communication, HTTP request, response and its content, URL, and UAs. These are used by a random forest classifier to assign a risk score for each connection. As discussed in Section 5.6, MADE is not as effective on HTTPS traffic, which is nowadays harder to intercept and decrypt due to technical and legal requirements. In addition, EARLYCROW considers five other TTPs besides the *Web Application Protocol* TPP at the heart of MADE.

ExeceScent [238] detects C&C domains by clustering incoming requests into five templates, including median URL path, URL query component, User-Agent, other headers, and destination network. These templates are used to estimate similarity scores to predefined Control Protocol Templates (CPT) centroids. However, this is open to evasion if an adversary copies the UA of the victim machine from the Windows Registry [239]. In addition, it is not possible to extract most HTTP header features when HTTPS is in use, which hinders the generalization process and may result in mixing APTs with legitimate in many clusters. A related approach [225] adopts similar features, only using histogram bins which also can be evaded using HTTPS. BAYWATCH [19] is a filtering system to detect the beaconing of infected hosts. Universal and local whitelists are filtered, and then beaconing can be detected using the Discrete Fourier Transform (DFT) and Gaussian Mixture Model (GMM), awarding a high Agglomerative Hierarchical Clustering (ACF) score for strong periodicity. BAYWATCH filters URLs and domains that are likely to be legitimate. Unprocessed connections with all previous features are sent to a random forest for classification. BAYWATCH can be computationally expensive for only beaconing behaviour, and many APTs also have non-beaconing connections. EARLYCROW detects malicious connections regardless of their pattern. Finally, Kitsune [240] adopts an ensemble of autoencoders, proving the efficiency of unsupervised deep learning to detect classic attacks such as ARP poisoning and SYN DoS, which are rarely used by APTs. As discussed in Chapter 2, we avoid using deep learning because of the scarce dataset representing various APTs TTPs, which is essential for deep learning models.

5.8 Conclusions

In this chapter, we presented a threat model for APTs informed on the TTPs used by adversaries to avoid existing NIDS. Informed by that, we designed and implemented EarlyCrow, a random-forest-based classifier which can detect APT malware network activities that are missed by currently deployed defence mechanisms. We recommend using EarlyCrow as an additional layer of defence, besides SIEM, Host Intrusion detectors (HIDS), and domain detectors. Building PairFlow while considering an APT-specific threat model enabled EarlyCrow to catch TTPs used in APTs with promising performance. However, PairFlow can be used for further tasks beyond malicious HTTP(S) and domain detection, including non-application protocols (Raw TCP, UDP, ICMP, and Socket Secure (SOCKS)). Adversarial attacks against PairFlow fields and EarlyCrow features, their robustness, and deployment issues are left to future work.

In summary, our main contributions are:

- We present an evidence-based analysis of various TTPs used by APTs in order to bypass NIDS. We present a traffic measurement study on popular features for the APT traffic compared to the botnet and legitimate.
- We introduce PairFlow, which summarises a PCAP into contextually relevant fields, including packet behaviour, domain and URL list, TLS/SSL certificates and client and server cipher suits and extensions, User-Agent, status code, and content type for HTTP. PairFlow generates four main output files for consumption by NIDSs.
- We implement EarlyCrow, a system to detect evasive malicious communication, in particular from APTs and botnets. EarlyCrow leverages the context of communication through four perspectives, including PairFlow, host, destination, and URL.
- We evaluate the classification performance of new and existing features for malicious traffic detection under different scenarios, distinguishing ATP, botnet and legitimate traffic.

Chapter 6

APT Malware Command and Control Detection Through Hierarchical Clustering and Wavelet Decomposition

Until now, we have proposed two lines of defences in Chapters 4 and 5 based on our technical analysis in Chapter 3. We discussed in Chapter 5 that the non-HTTP(s) malicious connection detection, Case III and IV, is left for this chapter. We aim to provide a generic detection approach, regardless of the protocols that are being used for malicious communications. This can also tackle privacy issues for NIDS for enterprises with strict standards on exposing DNS or HTTP-based features. Therefore, we build NIGHTVISION which only accesses the time series of network data without diving into the application layers (DNS, HTTP ... etc).

In this chapter, we investigate the usage of digital signal processing (DSP) techniques for anomaly detection on network data. Then, we propose NIGHTVISION, which adopts the discrete Fourier and wavelet transforms (DFT and DWT, respectively) to build templates for several APT behaviours, including stealthiness, payload transfer, dynamic resolution, and malicious web browsing-like. In addition, we propose time-frequency statistics on each band decomposed by DWT, which is a popular approach for ECG to identify heart disease [9]. To enhance NIGHTVISION, we also introduce traffic gauges for a network that can be tailored for a specific organisation. We mainly profile the traffic volume, non-application, and scanning activities

using a combined Agglomerative Hierarchical Clustering (AHC) framework for the training and decision tree at testing time. Various extracted features are introduced to feed the clustering and decision tree models, including those that track TCP flags activities using a Finite State Machine (FSM) that distinguishes normal behaviour from SYN, ACK, and FIN scan attacks. We evaluate NIGHTVISION against APT, botnet, and legitimate traffic under different scenarios, where the classifier predicts known malware families or unknown ones. The $mF1$of our system against unseen malware is 80.09%, indicating its overall performance. It also demonstrates a high accuracy rate of 97.71% and a low FPR of 0.25%. In comparison, the state of the art which performs at 67.61%, 95.82%, and 1.61%, respectively.

6.1 Digital Signal Processing for Network Data

We introduced DSP techniques in 2.6, page 73, including Fourier and Wavelet transforms, which are frequently used in speech and image processing and many other areas. It is rare to find examples of using such techniques in network data. In this section, we provide multiple examples of how we utilise these techniques. In addition, it is important to use only techniques that can enable a reasonable interpretation from the network behaviour perspective.

6.1.1 Frequency Analysis

The frequency analysis is the output of the Fourier transform. In the network data, we can identify if a connection has periodicity or a burst by analysing the frequency domain. Figure 6.1 shows various examples of packet analysis using the Fourier transform. In Figure 6.1.a, we present an example of one packet at index 0. Recall the basis functions of the DFT in eq. 2.23 and eq. 2.24, page 75:

$$X_R[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi ki/N) \tag{6.1}$$

$$X_J[k] = -\sum_{i=0}^{N-1} x[i] \sin(2\pi ki/N) \tag{6.2}$$

The real part of X ($X_R[k]$) and its imaginary one ($X_J[k]$) describe the amplitude of the cosine and sine waves, respectively. If we substitute i with zero for the real part $X_R[k]$, the cosine function will equal one, which means a horizontal line, as shown in Figure 6.1.a. The amplitude of the horizontal line is 12 because only x[i] is left in eq. 6.1, which is the same one as the time domain in bytes. For the imaginary part $X_J[k]$, the sine wave at zero seconds is zero, resulting in a horizontal one with an amplitude equal to zero.

In Figure 6.1.b, we present a single packet that arrives after one second ($i = 1$). $X_R[k]$ presents one cycle of cosine. The sine function in $X_J[k]$ is shifted by $-\pi$.

Figure 6.1.c shows a packet arrives after four seconds ($i = 4$), which produces four cycles in $X_R[k]$ and $X_j[k]$ of cosine and sine waves, respectively, and so on. Figures 6.1.d and 6.1.e show the cases where two packets arrive subsequently, and minor changes occur in their respective $X_R[k]$ and $X_Jk]$. As discussed in Chapter 2, page 75, we need to convert the rectangular notation of DFT to polar ones to interpret the result of the real and imaginary parts as magnitude ($X_M[k]$) and phase ($X_P[k]$):

$$X_M[k] = \sqrt{X_R[k]^2 + X_J[k]^2} \tag{6.3}$$

$$X_P[k] = arctan\left(\frac{X_J[k]}{X_R[k]}\right) \tag{6.4}$$

This chapter focuses on $X_M[k]$ because we use it for the matched filters and produce the periodogram by taking the square of $X_M[k]$. For all previous examples, the magnitude is represented as the horizontal line. We neglect $X_P[k]$ because it does not provide useful information for the network data. Figures 6.1.b and 6.1.c present two sinusoids of $X_R[k]$ and $X_J[k]$. Using eq. 6.3 to calculate $X_M[k]$, they cancel each other, resulting in a horizontal line again. This can be seen in $X_P[k]$, which oscillates between -1 and 1. While the shape of the waves of $X_R[k]$ and $X_J[k]$ is affected by the arrival time, we can see the usefulness of observing $X_M[k]$, which is not vulnerable to the arrival time, and confirm that there is one packet has arrived. Another observation is that $X_R[k]$ and $X_J[k]$ in 6.1.b and 6.1.c almost identical, where the former has one packet and the latter has two packets. We can see their $X_M[k]$ is different, reflecting the packet number difference. In the next sections, we will explain how to use matched filters based

on $X_M[k]$ output.

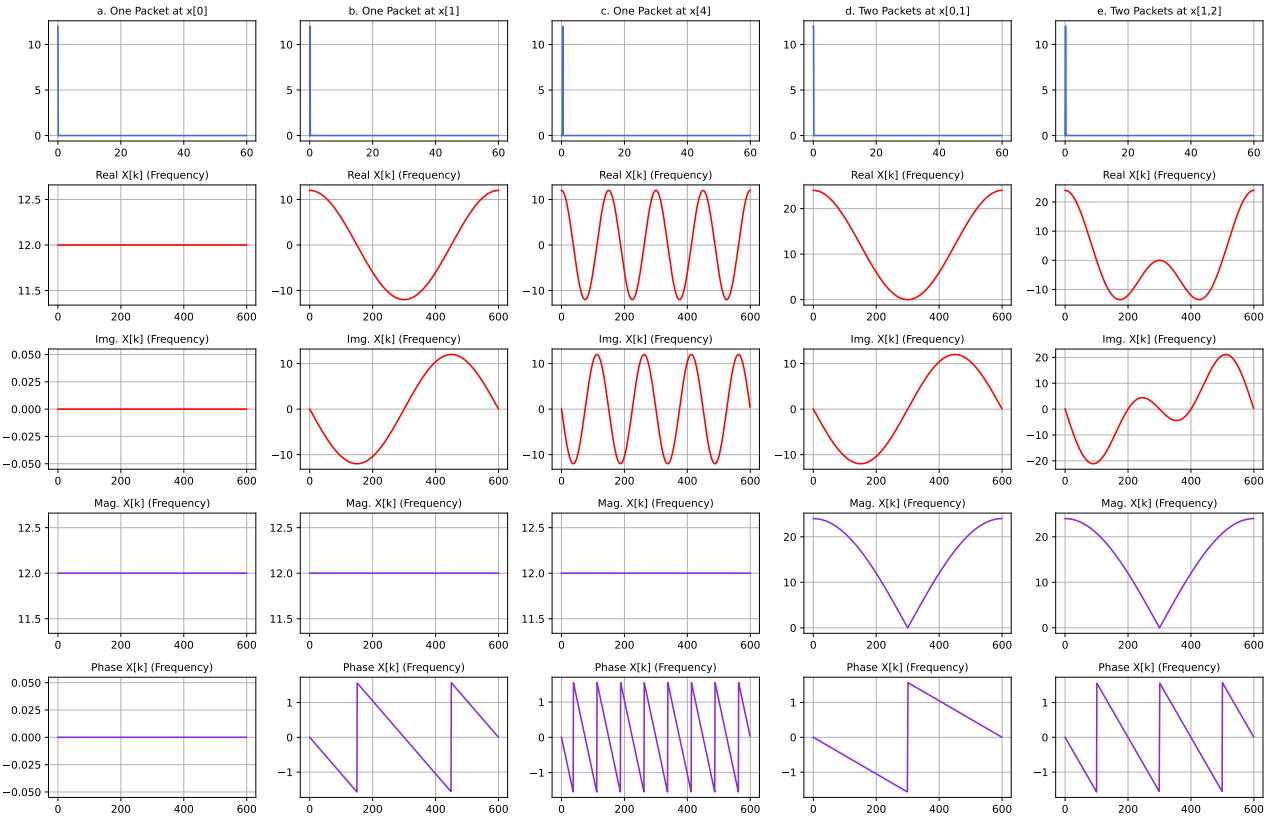


Figure 6.1: Fourier transform for network data.

Figure 6.2 shows the frequency analysis if the packet has beaconing behaviour. In Figure 6.2.a we present a case when a host beacons 21 bytes of packets every 8 seconds. We can view eight impulses of the same magnitude $X_M[k]$. These repetitive impulses are called harmonics, which means a signal in a time domain has periodic behaviour.

Figure 6.2.b shows the traffic beacons every 16 seconds with 100 bytes. $X_M[k]$ shows 16 impulses for this process, which reflect the same period in the time domain. Next, we present Figure 6.2.c, where two processes generate two beacons with different packet sizes every 8 and 16 seconds. We identify that $X_M[k]$ has two harmonics, which reflect two processes with different packet sizes that exchange data periodically. Finally, because of the sensitivity of the start and end of the packet arrival time at the time domain in rectangular notation for the real and imaginary components of the basis function of the Fourier transform, we present the last example in Figure 6.2.e. We show if the same examples exist in Figure 6.2.d, where the start and the end are not the boundaries of the number of samples N in the DFT basis functions. The number of harmonics and their frequency are maintained for $X_M[k]$. Because of the robustness

of $X_M[k]$, compared to $X_R[k]$ and $X_J[k]$ in terms of extracting the information, we limit our frequency analysis on $X_M[k]$ for the rest of this chapter.

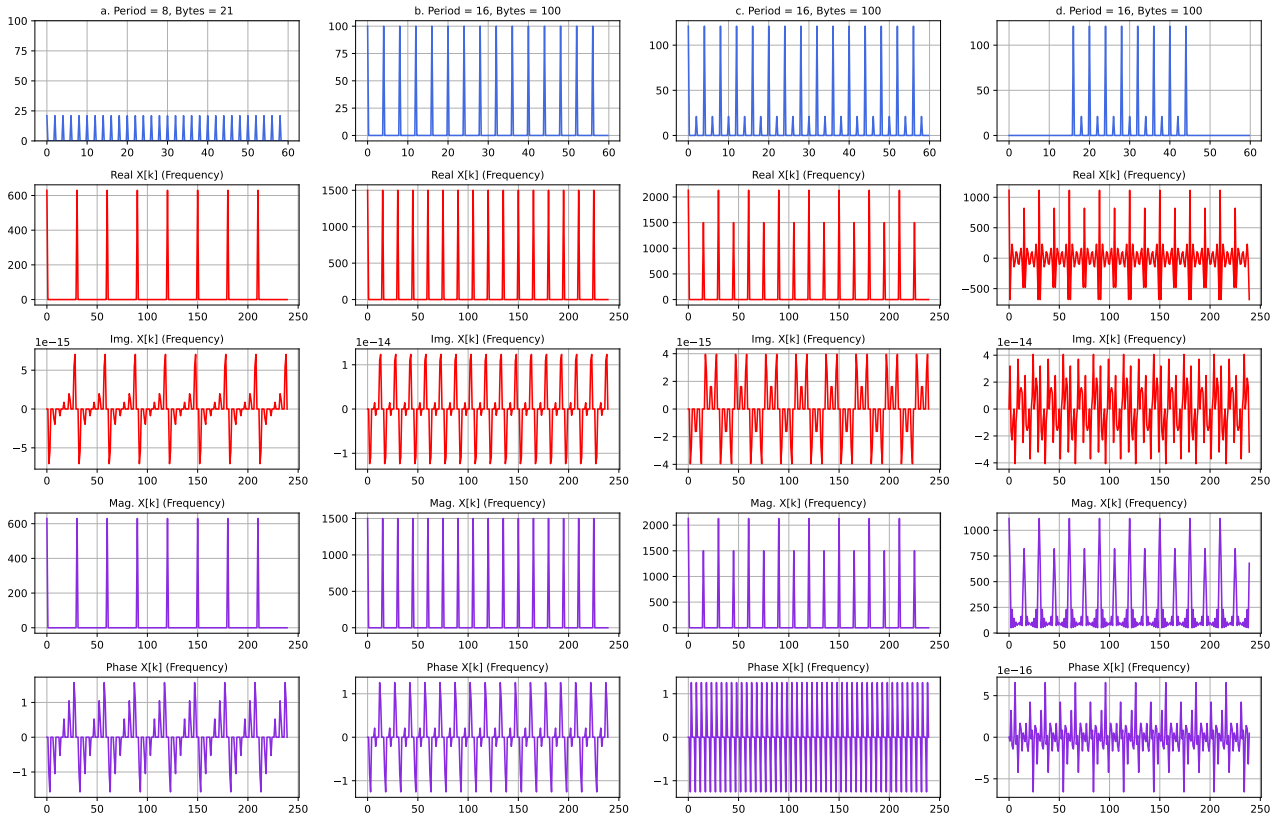


Figure 6.2: Periodic packet analysis.

6.1.2 Wavelet Analysis

Wavelet analysis can decompose network data to identify if there is a specific component involved. In Figure 6.3, we provide an example of the decomposition of traffic that contains a payload transfer. On the first scale, it may not be obvious whether the intense communication in a short period results from exchanging information, web browsing, or payload transfer. Therefore, we need to identify if the low-frequency level has a component that stays active longer at the same data rate. So, for web browsing, it should be a group of repetitive frequency components like spikes in high-pass filters, which frequently happen during this connection. In this example, we find a square function which can be used as a representation for payload transfer. We know that the square function in the time domain results in a Sinc function [85], which is given by $sin(x)/x$. So we can search for the Sinc function to identify the payload component, as we explain more in Section 6.4.2.4.

As expected from the periodogram of high-pass filters, it is difficult to extract useful information for such outputs. For this reason, it is standard to focus on the periodogram of the low-pass filter due to the nature of wavelet analysis, which decomposes the signal into multiple low-pass filters at different scales [9]. There is another method for Wavelet to look into two of them, which is called filter bank. However, its time complexity is exponential because each level produces double the output of the previous levels. For example, on a scale of 12, it will generate 2^{12} periodograms instead of 2 [9]. Therefore, we did not use such a method in this chapter.

6.2 NIGHTVISION

NIGHTVISION detects malicious network connections regardless of the protocol type. We discuss the architecture of NIGHTVISION, and how features are extracted and selected.

6.2.1 Architecture Overview

In Figure 6.4, we present the flow of the main components of NIGHTVISION. At the training time (Figure 6.4,Ⓐ - ❶), training datasets of PCAP (described in Section 6.5.1) are divided into multiple time windows, and the packets are tracked and aggregated similar to Chapter 5. Next, the traffic is decomposed into three planes, i.e. TCP, UDP and ICMP, in step ❷. The control plane is used to build Finite State Machines (FSMs) to track the activity of each connection in step ❸. Several features are extracted from FSM to build three hierarchical clusters to profile each flow. The output of these clusters is used as the final feature for training the classifier.

At the same time, other planes are used as time-series arrays, which are needed to be prepared for the digital signal processing techniques. To prepare these arrays, sampling techniques are used in step ❹. In step ❺, we apply DWT for the time-series array to generate a time-frequency representation for each connection. We consider extracting multiple features at each level. Those features are generated based on two categories. The first category ❻ is the template-based. We aim to monitor such common aspects of APT connections as discussed in Chapter 3. These aspects are the stealthiness behaviour, dynamic resolution, payload transfer and web browsing-like. The second category ❼ takes the statistical measurement of each level of the

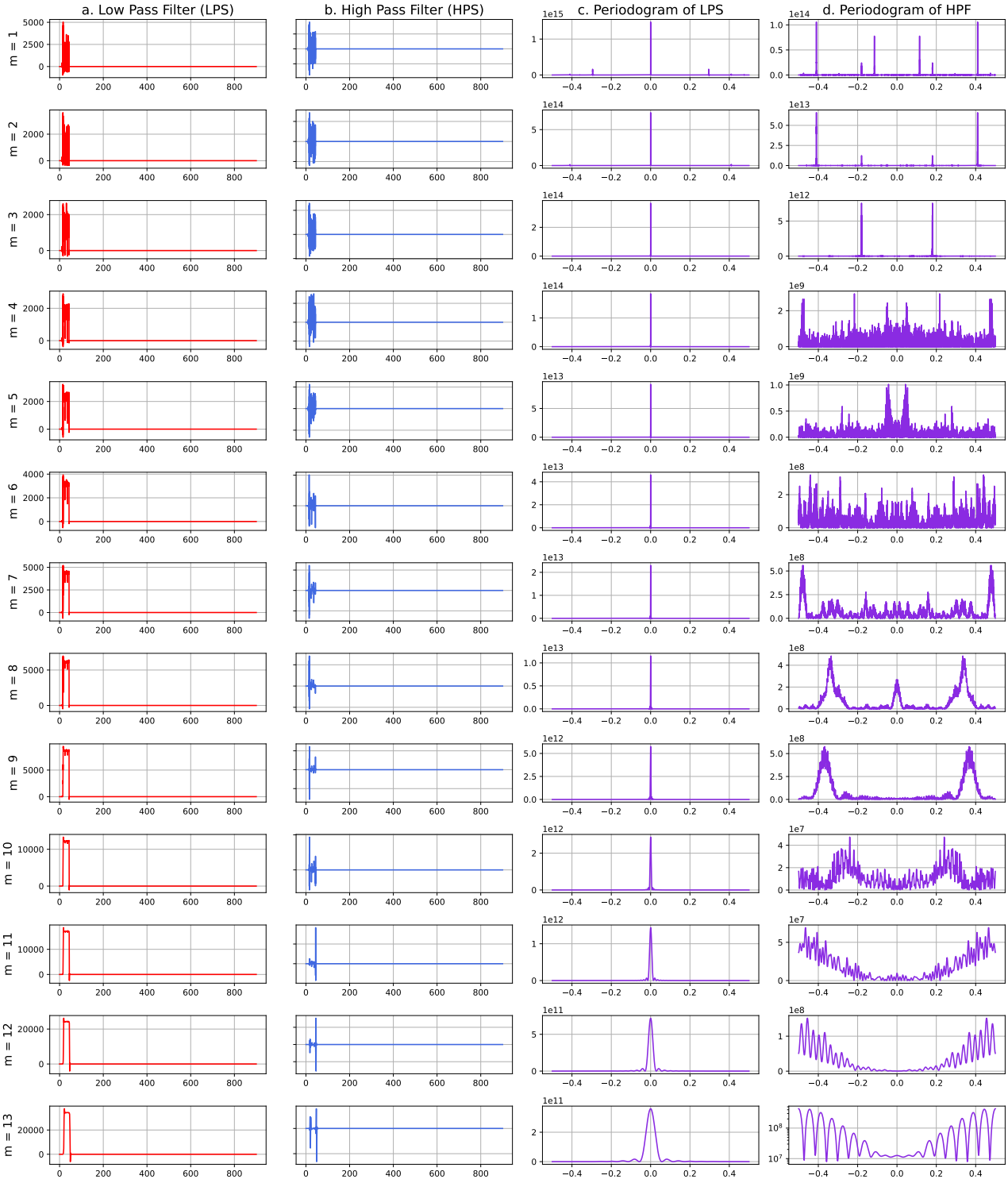


Figure 6.3: Example of file transfer behaviour and its DWT and periodogram.

DWT output at different frequencies. Some of these features of different levels may generate low variance between them, which causes overhead and may impact the performance as discussed in Chapter 2, page 66. For this reason, we use feature selection (step ❽) for those ones that generate the highest gain and push them to the feature space.

The steps in testing time (Figure Ⓑ) are almost similar to the training time. One of the differences is in steps ❶, where we filter the malicious connection in the previous time window. Another difference is to use three decision trees rather than the hierarchical ones in step ❸. Finally, the selected features in step ❻ and ❼ are the only ones to be generated to save the time complexity during the testing time. In the next sections, we discuss each step in detail with some evidence based on our training dataset.

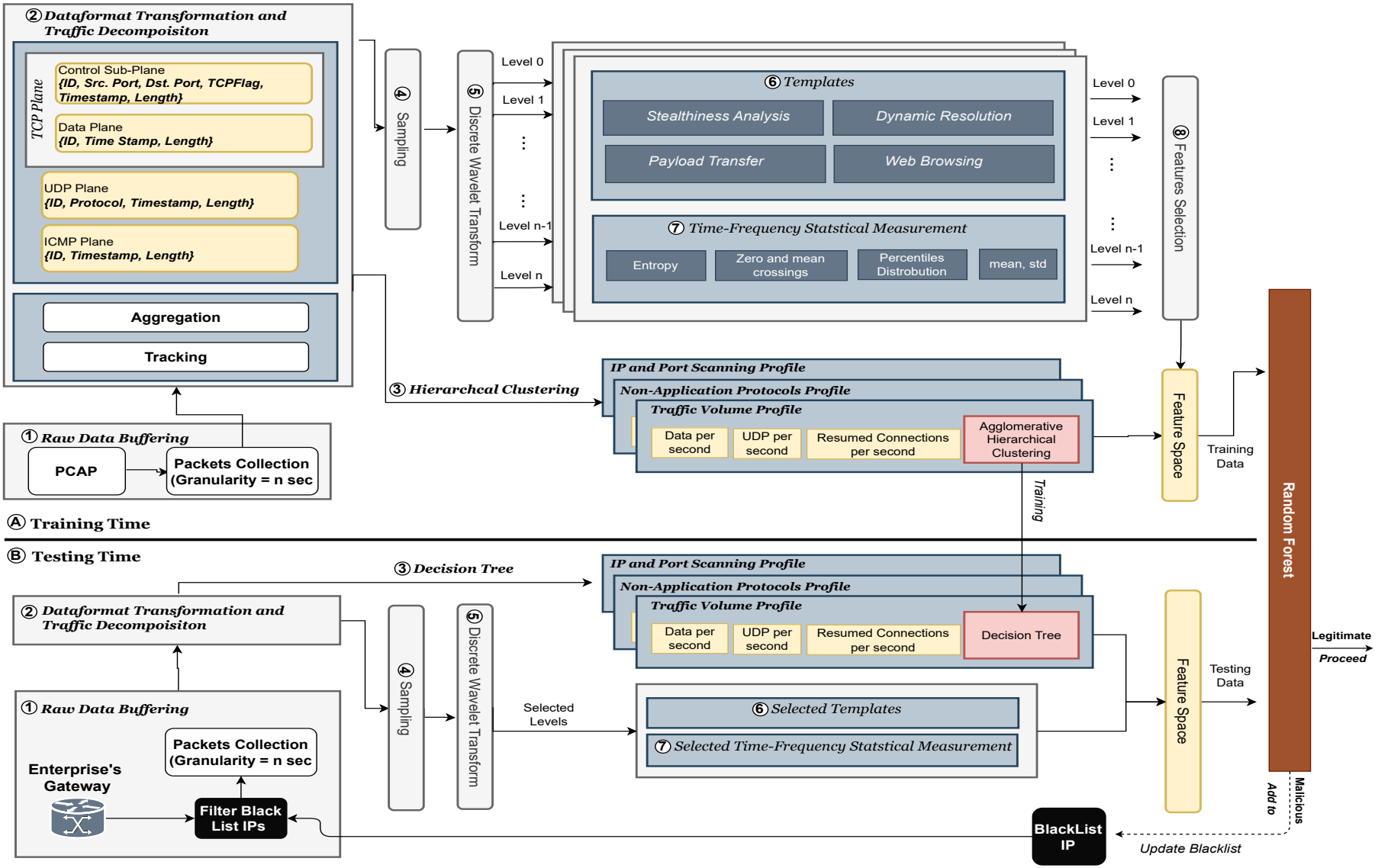


Figure 6.4: Overview of the NIGHTVISION architecture.

6.2.2 Traffic Decomposition

In digital signal processing, digital filters can be used to decompose the signal for further processing later, as discussed in Section 2.6.2, page 75. For the network data, a low or high pass filter may lose the data itself. For this reason, we decompose the traffic at the time domain based on TCP, UDP and ICMP packets. This will allow us to extract useful information when we search for some malicious behaviour used by APT, such as dynamic resolution or payload transfer.

We update PairFlow according to our needs in this chapter. For TCP, traffic is divided into two planes, control and data. The control sub-plane summarises the packets in terms of ID, source and destination port, TCP flag, timestamp and packet length. We retrieve the source and destination ports for each packet for the purpose of profiling in Section 6.3. For the data plane, the packet information is limited to time series data, including the ID, timestamp and packet length. This differs from our works in Chapter 5 due to the privacy-preserving requirement in this chapter. In the UDP plane, we retrieve the ID, source and destination ports, Timestamp and the length of the packet. Finally, the ICMP plane collects the ID, timestamp and packet length.

Figure 6.5 shows an example of decomposing the traffic based on these planes. Separating these signals can only be achieved in the time domain, and we can see it is useful to apply several techniques on a specific signal to extract key information, such as the correlation between the UDP and control planes, or to extract the periodogram of UDP plane to search for dynamic resolution technique.

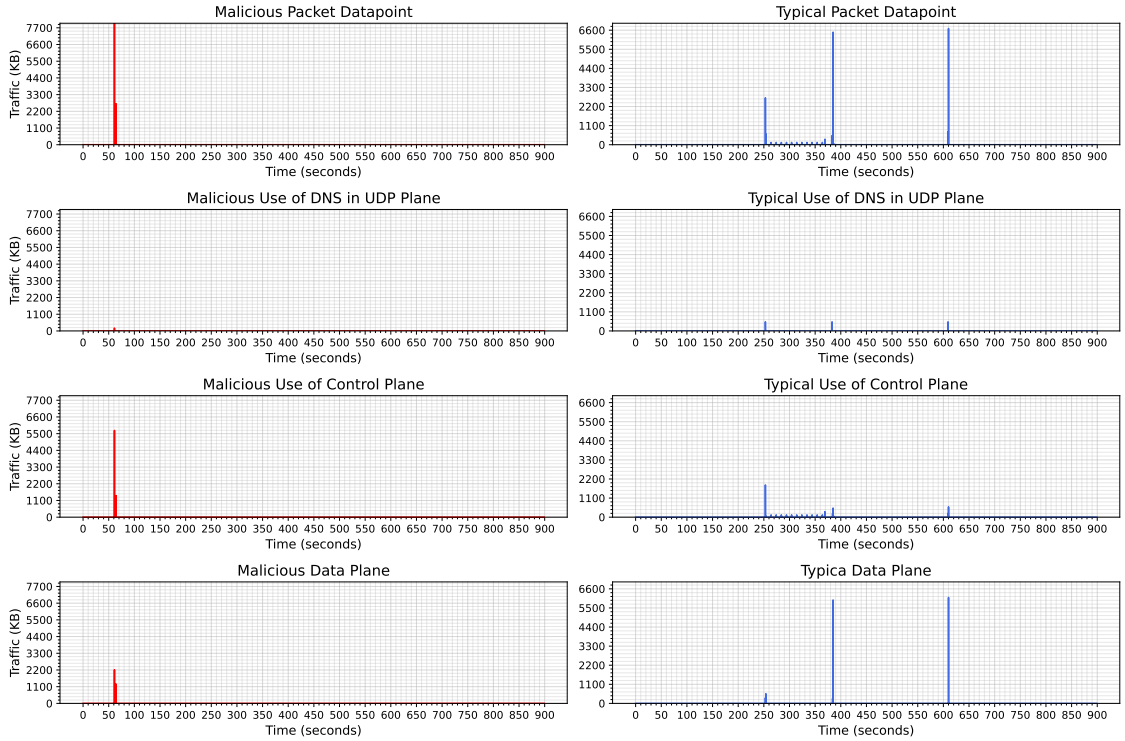


Figure 6.5: Traffic decomposition into different planes.

6.3 Traffic Profiling Clustering

To define low-profile connections, we need to identify a threshold for each network. However, identifying the threshold manually is different from one network to another. This is also analogous to the threshold for non-application protocols and scanning activities. Therefore, we resort to agglomerative hierarchical clustering (AHC). As presented in Chapter 2, AHC is a bottom-up approach where two connections are measured in terms of their similarity. The similarity is quantified by using the Euclidean distance between every pair of PairFlow based on selected features, which will be discussed in the following subsections. Therefore, each connection will be measured against all other connections independently. As a result, AHC will produce $n * (n-1)/2$ clusters.

Next, each pair of PairFlow is grouped into clusters according to their proximity. This process, called linkage, uses the similarity distance in the previous step as an input. Each pair of nodes is recursively used in the linkage process to be merged into the larger cluster until we have one large cluster at the head of the tree. The cutoff tree is recommended for AHC to determine the maximum number of clusters to stop at. Since each AHC is for network profiling, we select 100 for the cutoff parameter so we measure the usage of each profile from 0% to 100%.

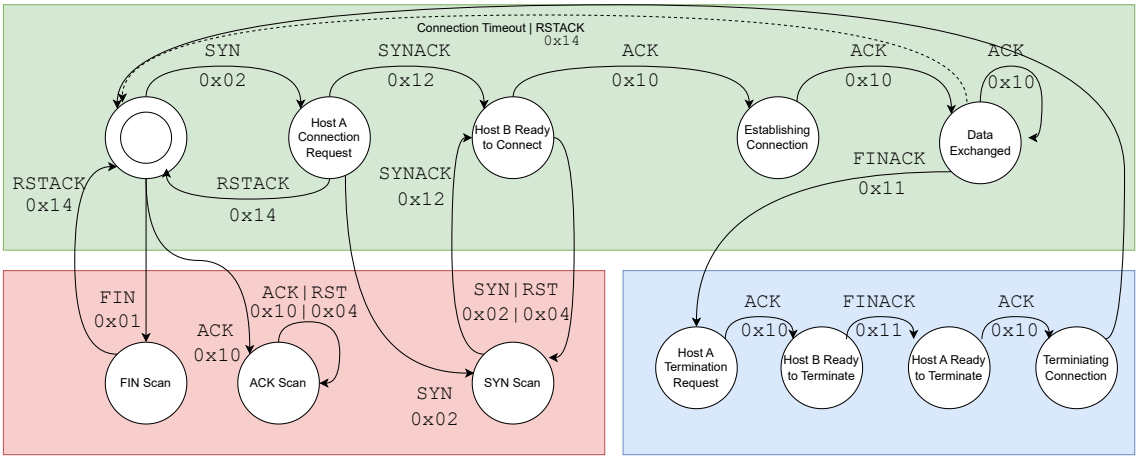


Figure 6.6: Finite state machine to identify scan packets with respect to the source and destination port number. The transitions are illustrated with the input annotated with their TCP flags binary corresponding.

While AHC is used for analysis tasks, any new connection may influence the AHC tree. To overcome this issue, we only used AHC on the training set to fix the AHC for a specific network. The decision tree model is used for the testing set or even a new connection. The decision tree will be trained using similar features used for each AHC presenting a profile. The clustering number generated by AHC will be used as multiple labels during the decision tree training. At the testing time, the decision tree predicts the cluster that belongs to an AHC tree.

6.3.1 IP and Port Scanning Profile

Scanning activities can be malicious as part of APT, botnet or any other attack type. However, it can also be legitimate for network administrators or applications. Therefore, we profile scanning activities to identify legitimate and malicious clusters. While scanning activities can be deployed using raw UDP or ICMP, different techniques can be used based on TCP, such as SYN, ACK and FIN [241]. To track SYN, ACK and FIN scans, we build a Finite State Machine (FSM) as depicted in Figure 6.6.

FSM is a mathematical model to represent a set of states S_i for a machine depicted as circles. To move from one state to another, a set of transitions T_i should be defined. In our FSM, states will represent the state of the TCP connections, whereas the transitions represent the TCP flag for a packet. The machine is a couple of the source and destination ports M_s^d, which means each couple of TCP-based ports used during a communication presents independent FSM. For

example, if a PairFlow between a server and clients starts communicating over ports 443 and 1174, NightVision spawns a new object M_{443}^{1174} from the FSM class to track all states according to the TCP flags for subsequent packets. When the same connection switches to communicate over port 80, another object called M_{80}^{1175} is created starting from the initial state. Next, if the connection receives a termination request for M_{443}^{1174}, the object M_{443}^{1174} is called again with its current state based without affecting the state of M_{80}^{1175}. In Figure 6.6, the red box presents the scan states while the other boxes belong to the legitimate behaviour.

To build AHC for this purpose, we count the number of ICMP packets, raw UDP per second, SYN, ACK, and FIN scans from the FSM. We avoid considering the packet size in bytes because scanning is normally short packets, and we need to focus on scanning activities and neglect activities such as administrative tasks or file transfers. We acknowledge that overlap may occur, but using the number of packets instead of their size will reduce such overlap.

6.3.2 Traffic Volume Profile

We propose three features based on the traffic decomposition to cluster profiles based on the volume. First, we calculate Data bytes per second on the data plane, which represents traffic generated by applications such as HTTP(S) and SMTP. The second feature is the number of resumed connections per second. Resumed connections can be extracted from the FSM when the state returns to the initial one. When the data per second is small compared to the number of resumed connections, it helps reflect low-profile connections. The third feature is the UDP bytes per second, which may also be used for data transfer, DGA and DDoS by an adversary in addition to the typical usage of DNS resolution. AHC, based on these three features, will identify the typical legitimate clusters and separate them from the malicious ones.

6.3.3 Non-Application Protocols Profile

As discussed in Chapter 3, non-application protocols are extensively used by several APT campaigns [213, 24]. To use raw TCP, clients may use HTTP(S) port 80 or 443 to evade the firewall in the infected network [175]. We identify the raw TCP packets by subtracting the data packets in the data plane from ACK packets in the *data exchange state* in Figure 6.6. Those

packets flagged with ACK without their counterpart in the data plane reflect the host uses raw TCP without any application packets. Raw UDP often uses DNS port 53 to evade the firewall [242]. We calculate the application bytes for each UDP packet to distinguish raw UDP from DNS. If the length of the application is zero, then the UDP packet is raw. We avoid using such a method to identify the raw TCP since there are other control packets, such as the three-way handshaking where the application bytes are zero, and it is not considered as raw TCP-based data.

6.4 Discrete Wavelet Transform (DWT)

DWT can be used for network data, as we have seen in Section 6.1. We utilise DWT to detect APT communications, in which many TTPs are exploited. These TTPs can be represented by different functions, such as the Sinc function or exponential random process. However, it is essential in the DSP field to perform sampling techniques before using DSP techniques. This is important to go back and forth from one representation to another, as discussed in Section 2.6, page 73. Then, we described the templates for malicious behaviour using various techniques. Following that, we present the time-frequency analysis for statistical features for each band.

6.4.1 Sampling Rate

DSP techniques require a fixed sample rate per second to apply the decomposition or the reconstruction between the time and frequency domains. In the network data, packet timestamps are only recorded when the packet arrives or when it is sent. For example, when two packets, with 64 bytes for each, are exchanged at timestamp 1 and 15 seconds, the packet data points will only have two tuples (1, 64) and (15, 64). In signal processing, all seconds are required to be presented starting from 0 until 60 if the time window is 60 seconds. Then those tuples are padded with zero bytes as there are no packets arrived at these points except at 1 and 15 sec. When we have other packets, let us say 5000 packets, in the bin of 15th sec with only μs difference, the method required to be improved to represent these packets between the bin of 15 and 16, which leads to selecting the sampling rate properly. The sampling rate or frequency is the number of samples recorded per second. The sample rate should also be applied to all

bins on a time scale. To determine the proper sampling rate, the threshold must adhere to the Nyquist theorem. The condition is to select a sampling rate *at least twice* the highest frequency of the packet data points for a connection [85].

For example, if the highest number of samples in a connection is at a second 15, e.g. 5000 samples (15.001, 15.020 .. etc.), that would be the highest frequency, which is equal to 5k samples per second or 5kHz. Next, the twice of this is 10kHz. In practice, it is recommended to sample slightly above the twice (i.e. 12kHz) to give more space for quality assurance [85]. Therefore, if we sample at 12kHz, we allow frequencies between zero and 6kHz to be properly presented in the frequency domain generated by Fourier or Wavelet transforms. The one-half sampling rate is 6kHz, which is the Nyquist frequency. Nyquist confirms that any frequencies above the one-half sampling rate cannot be presented. If the data has a frequency above the Nyquist frequency, it will cause aliasing, which adds to other frequency bins and distorts the signal, which prevents the reconstruction later for wavelet or Fourier transform. NIGHTVISION determine a sample rate for each connection independent from other connections to retain all its frequency and prevent aliasing.

6.4.2 DWT-based Templates

In this section, we describe the four templates using DWT to detect malicious behaviour. The stealthiness analysis helps in identifying the stealthy connections used by APT, which is a quite common technique. The second template is to investigate if the web protocol behaves as legitimate or malicious by analysing the correlation between UDP (i.e. DNS) and control or UDP and data planes. The third template extracts the autocorrelation of the UDP plane to identify if the dynamic resolution is being used for a given connection. The last template is to extract if there is a payload transfer during a connection by tracking the square signal.

6.4.2.1 Stealthiness Analysis.

Stealthy communication means it stays idle for a long time and pushes the packets in bursts within a short period. We aim to propose a probability model for stealthy behaviour. The probability distribution varied depending on the random variable. Exponential random variables

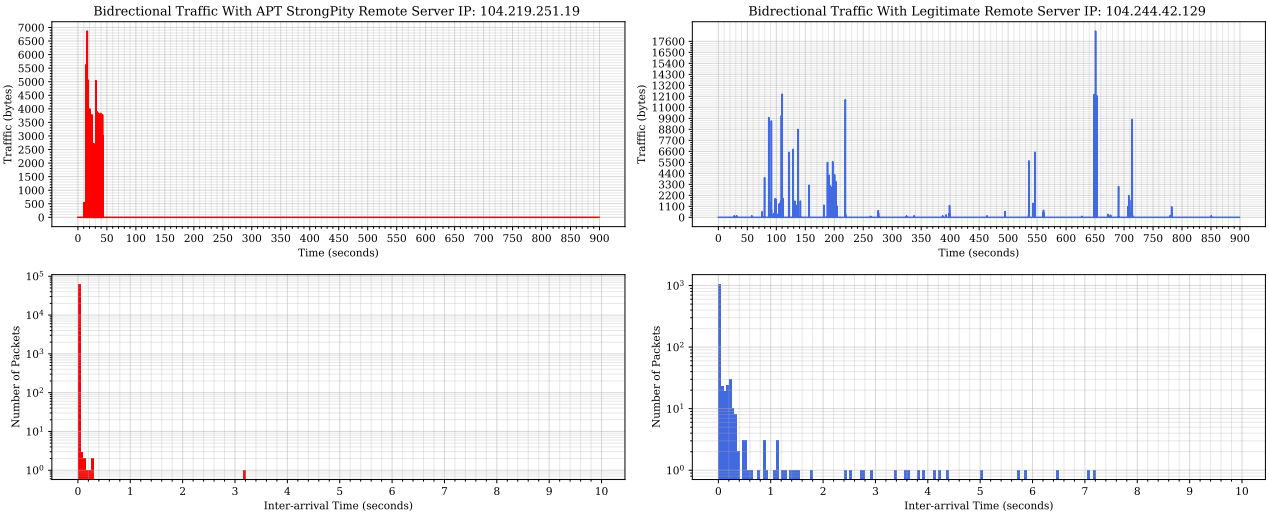


Figure 6.7: Examples of the random variable distribution based on the inter-arrival time between packets.

are often used to model time [243]. The probability density function (PDF) of an exponential random variable is given by:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

The parameter λ controls the speed of the decay. For network data, we first need to calculate the histogram of the inter-arrival time. A histogram is generated by dividing the data into groups of the same size and then counting how many outcomes fit into each group. The resolution may be changed by adjusting the bin width.

Therefore, we calculate the histogram of the inter-arrival time of stealthy packets, which fits the exponential distribution as presented in Figure 6.7. Then, we calculate the expected value $E(X)$, given by $E(X) = 1/\lambda$. The higher value of the $E(X)$ indicates the quicker packets are exchanged, meaning the decay factor is larger; therefore, no more packets are presented at a lower rate. We also consider kurtosis, which reflects the sharpness of the left tail. For example, the higher value of bin[0], which presents the minimum inter-arrival time, the more significant the kurtosis value. Also, kurtosis can clarify if the distribution is not exponential (e.g., Poisson distribution), which will converge to zero. Measuring the skewness can also reveal if the random variable is exponential (right-skewed) or Poisson (skewness equals zero). Entropy is measured for more complicated cases, such as if some of the packets are stealthy (e.g., burst), followed by

frequent wide-spread packets. For instance, If the entropy is large, the time sequence includes other packets in a larger inter-arrival time located in the right tail (e.g., bin$[n-1]$).

6.4.2.2 Malicious Web Protocol TTP

Typical benign web-based protocols, such as browsing activity, initiate the connection using DNS packets to retrieve the destination IP. Another click for another page request will also require DNS packets to resolve the new URL. Some malicious connections do not follow the same pattern, such as connections that use only raw TCP, which lacks the application layer bytes. Another example is when the connection communicates over HTTP(S) without DNS packet.

Therefore, we measure the cross-correlation between the UDP and the control packets' time sequences (ρ_c). The cross-correlation function rises up to 1 for benign connection when the DNS packet is sent every time. The frequent times that the cross-correlation reaches one will indicate the connection is benign. Otherwise, it is indicated as suspicious. Therefore, calculating the mean of the correlation and the standard deviation is useful to determine the threshold between legitimate and malicious connections. To enhance this feature, we also measure the correlation between the UDP and data planes (ρ_d) and compute the ratio of mean $\frac{\mu(\rho_d)}{\mu(\rho_c)}$ and standard decision $\frac{\sigma(\rho_d)}{\sigma(\rho_c)}$ values of the two correlations. Figure 6.8, shows the two correlations are almost identical, which means the ratio may reach nearly 1. This is the typical use of the web protocol. In case the correlation converges to zero, it indicates the web protocol is used suspiciously.

We consider the case when the connection suffers from delay or any failed packets. Therefore, we use a moving average filter to smooth the time for each plane to be robust against such cases, as presented in each row in Figure 6.8. Then, We used Pearson correlation (ρ) for an interval (t) as follows:

$$\rho_t(X,Y) = \frac{\sum(x-m_x)(y-m_y)}{\sqrt{\sum(x-m_x)^2 \sum(y-m_y)^2}} \tag{6.6}$$

Where x and y are the two inputs of planes, such as the moving average of UDP and control plane, m_x and m_y are the mean of the vectors x and y, respectively.

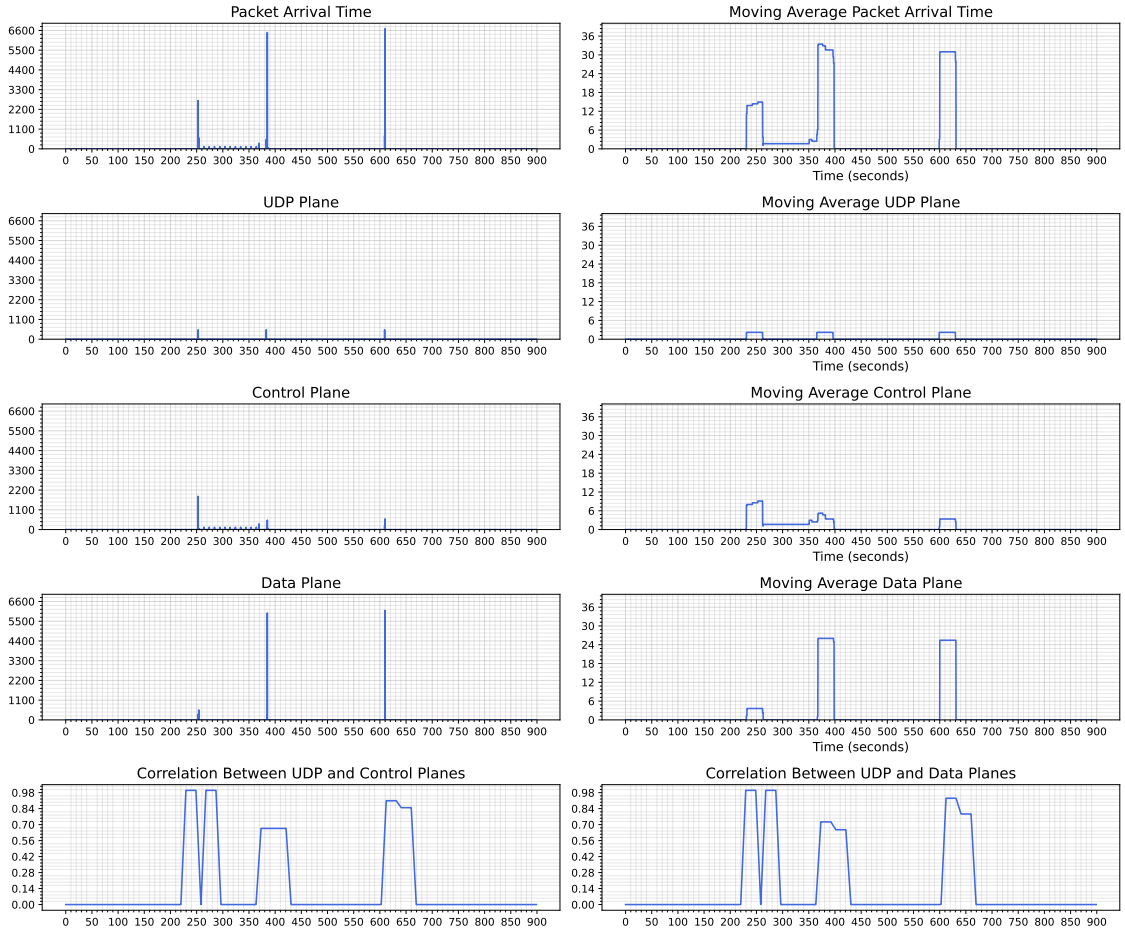


Figure 6.8: Cross-correlation between UDP, control, data planes.

6.4.2.3 Dynamic Resolution

A single use of dynamic DNS can be detected using DFT. When an adversary uses dynamic DNS one time and continues using TCP-based protocols, we decompose DNS packets as a time sequence. One spike in the time domain will produce a sinusoid wave in the frequency domain and a horizontal line for its periodogram, as we have presented in Figure 6.1, page 203. Therefore, use the Fourier transform to convert the DNS time sequence to obtain the periodigram. Then, we use the autocorrelation function, which performs the correlation between the input with itself but with lags. The autocorrelation function of a discrete-time signal is given by:

$$ACF_{yy}(l) = \sum_{n \in Z} y(n)\overline{y(n-l)} \tag{6.7}$$

where y is the periodogram, l is the lag, and $\overline{y(n-1)}$ is the complex conjugate of $y(n-1)$.

Figure 6.9 shows OnionDuke malware that uses dynamic resolution in the UDP plane and its periodogram, which is a straight line. So, the correlation with its delayed version will show a straight line, meaning the wave repeats itself and matches dynamic resolution behaviour. In the legitimate example, we can see the periodogram presents a noise signal. The noise signal has a weak correlation with itself until it reaches zero.

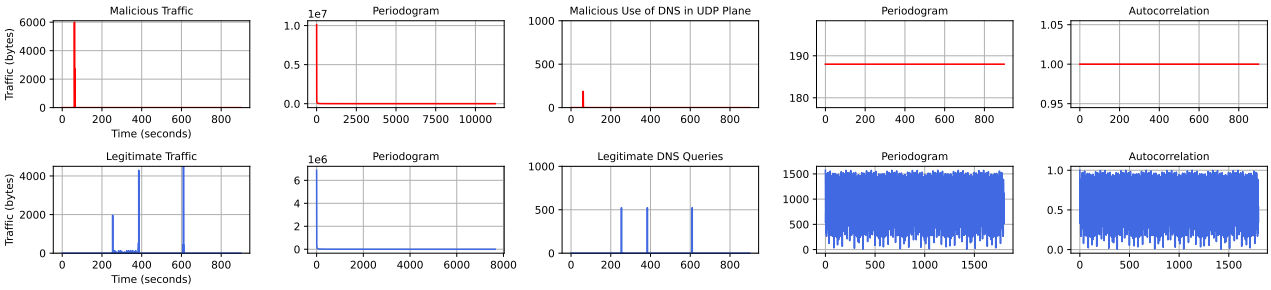


Figure 6.9: OnionDuke example of UDP plane which includes dynamic DNS resolution.

6.4.2.4 Payload Transfer

Payload transfer can be modelled as a burst that includes a continuous square wave that starts from t_0 until t_n. Wavelet decomposition can be used to identify if a burst incorporates a square signal. For instance, at scale m, time-frequency traffic will approach to be square wave-like. Therefore, checking that when it is transformed into a periodogram is more convenient. The typical periodogram of a square wave generates a Sinc function. Inspired by radar applications for particular wave detection [244], the matched filter can be used. The matched filter is the correlation between the original and investigated waves. When the correlation is high at the same time slots as the investigated wave, that indicates the investigated wave is an ingredient of the original signal. Our investigated signal for the payload transfer is the Sinc function in the frequency domain, which is a square wave in the time domain, as we discussed earlier.

Since the square wave in the time domain will be an approximation to the square wave-like of the payload transfer, such an approximation may raise a false positive because it may not be fully identical. Therefore, we used the coefficient of variations (CV) to measure the dispersion of the correlation and the investigated wave. CV is the standard deviation divided by the mean. The two outputs are calculated as a ratio CV_i/CV_c, where CV_i refer to the dispersion

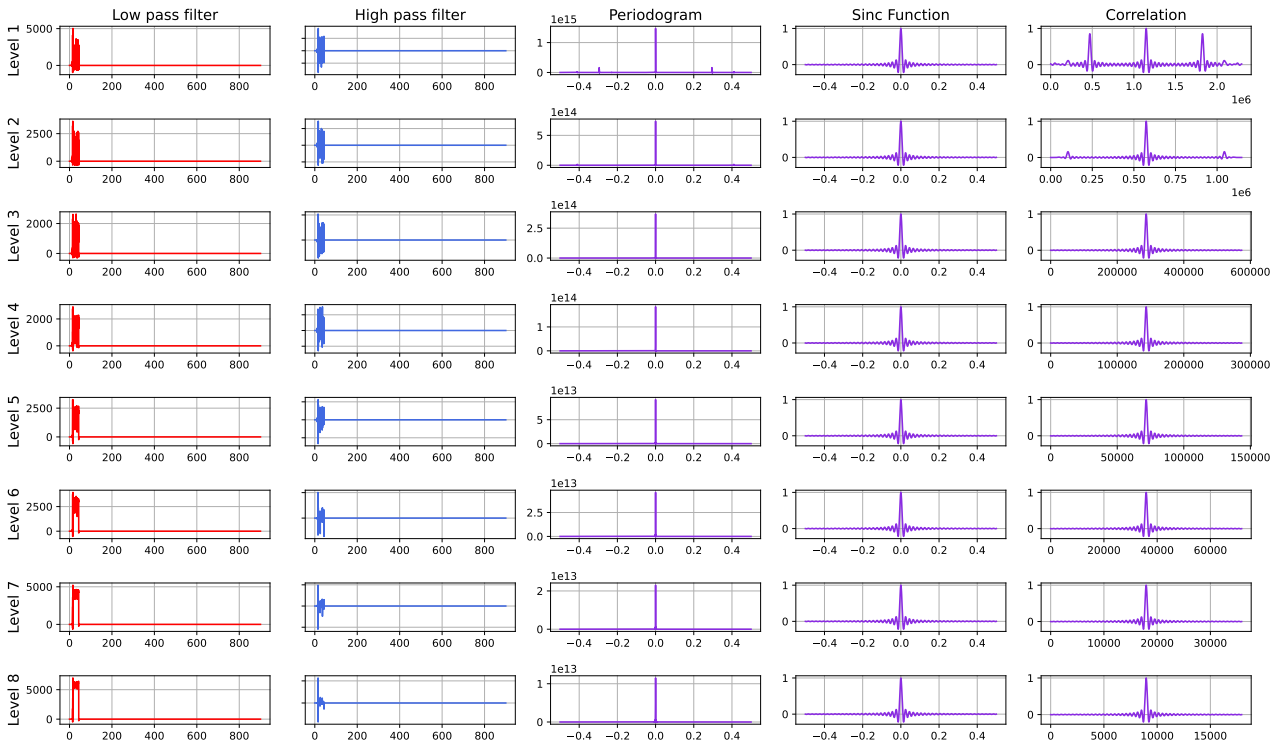


Figure 6.10: Example of the payload correlation with Sinc.

of the investigated wave, i.e. Sinc function in this case, and CV_c represents the dispersion of the correlation. If the ratio approaches 1, the correlation is fully matched to the Sinc function, whereas a larger number indicates that the input does not include any Sinc component.

Figures 6.10 and 6.11 present two signals; the first figure shows a signal with a payload. The square wave can be seen at the level 8. The periodogram here is for the low pass filter only. We can see if we calculate the correlation between the Sinc function and the periodogram, it generates a Sinc-like wave. On the other hand, the second figure shows arbitrary results of correlation, which is far from the Sinc function. This can tell us that the Sinc function does not exist.

6.4.3 Time-Frequency Statistical Measurement

We leveraged the main usage of wavelet decomposition in ECG applications. Wavelet decomposition statistical analysis on ECG and EEG signals is used frequently to identify heart diseases [245, 246, 247, 248]. By reporting the statistical measurements for each frequency band, Medical experts or ML models can predict and diagnose a disease based on wave shape in a particular

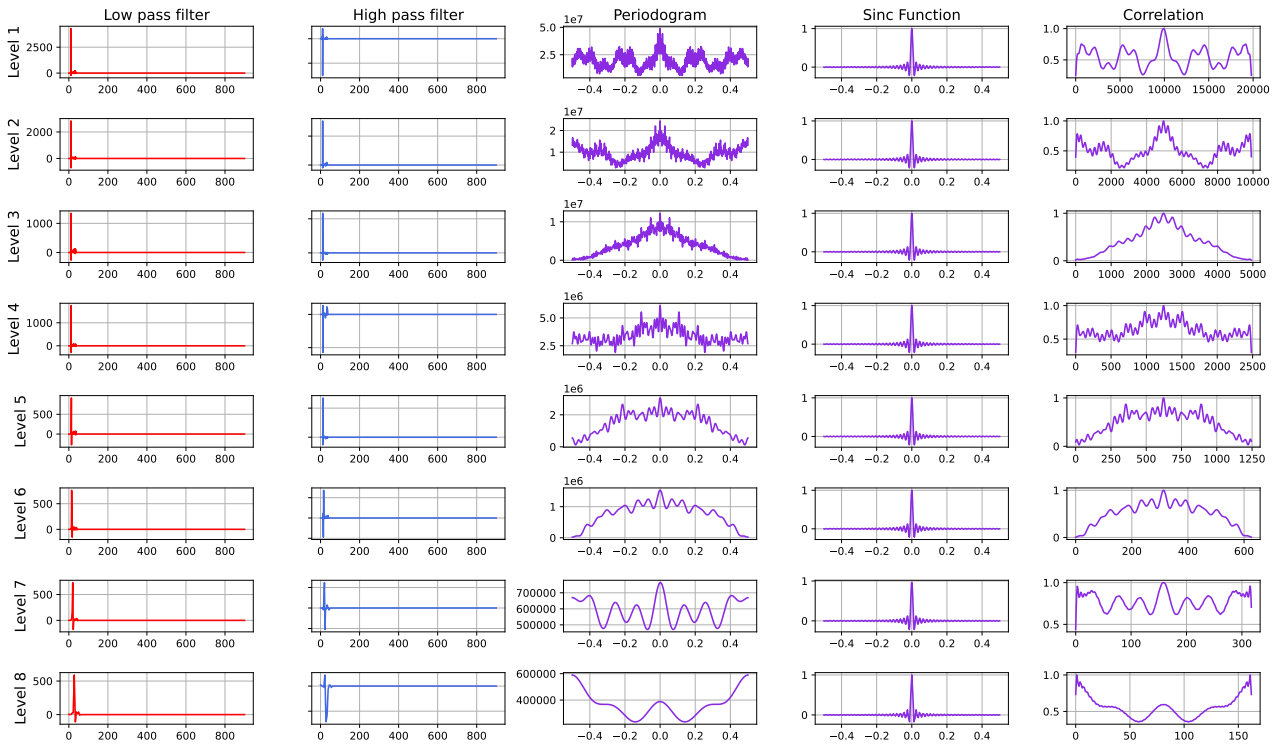


Figure 6.11: Example of the non-payload correlation with Sinc.

band [9].

For each band, we sort the wavelet transform based on the volume (bytes) as a histogram. Then, we calculate the 5, 25, 50, 75 and 95 percentiles to identify the spread of bytes. We also calculate the entropy, which is useful for highlighting the bursts for each band. For instance, if the entropy is low, it will indicate the data is concentrated in adjacent time slots, which may represent a burst. We also consider the mean and the standard deviation. In addition, we compute the number of zero and mean crossings, which appear to be useful in computer vision applications. For the security domain, such features can also participate in identifying the low profile in a particular band.

6.4.4 Features Selection

The wavelet function for the templates and the time-frequency analysis generate a large number of features for each level. For this reason, we need to limit the number of features to the most useful one. There are many techniques to be used for feature selection, such as Gini index, chi-squared, k highest scores based on p-value, dimensionality reduction, wrapper and filter

ID	Feature (Group)	Description	New?
		I. AHC-based Profiles	
1	Traffic volume profile	Clustering the traffic consumption for a particular network to identify the connection profile from the lowest to the highest. Data and TCP bytes per second are input to the AHC	✓
2	Non-Application profile	Clustering the usage of non-application protocols usage with raw TCP and UDP bytes as input to the AHC	✓
3	IP and port scanning profile	Counting SYN, ACK, FIN and UDP scans	✓
		II. DWT-based Templates	
4	Histogram of interarrival time	Kurtosis, skewness, entropy, mean and std of the histogram of interarrival time as exponential random variable	✓
5	Web protocols cross-correlation	Cross-correlation of DNS and control packets	Similar to [230]
6	Dynamic resolution periodogram autocorrelation	correlation of the periodogram of DNS time sequence with itself	✓
7	Payload transfer wave variation	The ratio of the coefficient of variation of the correlation of a Sinc function with the input wave CV_i and the variation of a Sinc function itself CV_c	✓
		III. Time-Frequency Statistical Measurements	
8-13	5, 25, 50, 75, 95 percentile	Description of the data distribution if the volume is sorted regardless of its time index	[245, 246, 248]
14-17	Entropy, mean, and sd.	Entropy, mean, and standard deviation of each band.	[247]
18-19	Zero and mean crossing	Number of zeros and mean crossings at each band	[9]

Table 6.1: NightVision features.

methods. For tree-based models, it is more convenient to use the Gini index to extract those features with the highest information gain because this is how it works later for random forest to build its trees, as discussed in Section 2.5.1.2, page 66.

In Table 6.1, we summarise the selected feature groups categorized into three sets: AHC-based profiles, DWT-based templates and time-frequency statistical measurements.

Label	Set	Malware Families
Malicious 981,447 packets	Training	AutoIt (0.68%), Bitsadmin (0.75%), Carbank (0.04%), Conficker (22.62%), FinFisher (3.45%), GRIFFON (4.78%), Mivast&Sakula (0.76%), NanoCore (0.48%), njRAT (23.37%), PlugX (0.09%) , Regin (0.62%), Remcos (0.71%), Sogou (2.99%), Virut (7.86%), Zebrocy (1.12%),
	Testing (Unseen)	Ammyy (0.83%), ChChes (0.11%), CobaltStrike (0.31%), CrossRAT (1.10%), Dridex (0.18%), Emotet (0.01%), Empire (3.13%), FlawedAmmy (0.20%), ImminentMonitor (9.76%), DarkComet (2.32%), MagicHound (0.32%), MiniDuke (1.76%), OnionDuke (0.11%), PoisonIvy (0.20%), Ramnit (0.17%), StrongPity (9.47%), Zeus (0.03%)
Legitimate 1,776,720 packets		Training: 70%, Testing: 30%.

Table 6.2: Dataset for unseen malware evaluation.

6.5 Results

In this section, we evaluate NIGHTVISION performance on the three datasets described below in Section 6.5.1, each with the same configuration as the previous chapter, which is a split of 70% for training and 30% for testing. The same experiments are performed on a baseline inspired by BOTection [79], which is a state-of-the-art detecting botnet using Markov Chain on the TCP flags. BOTection trains the random forest using NetFlow data. In our setting, we train BOTection using PAIRFLOW similar to NIGHTVISION. We note that all classifiers perform well in terms of weighted metrics such as $wF1$, wP, wR and accuracy due to the high volume of flows for the legitimate compared to the malicious. Therefore, we focus on the macro metrics to reflect the performance for such unbalanced datasets as discussed in Section 2.5.3, page 70.

6.5.1 Datasets

We used the same dataset (APTrace) and followed the causal controls used in Chapter 5, but we consider all connections beyond HTTP(S) made by a host. In addition, we capture more malware used by APTs such as DarkCommet, FinFisher, GRIFFON, Regin, MiniDuke, and CrossRat. We also add public botnet datasets from Malware Capture Facility Project (MCFP)[1] to verify if NIGHTVISION can generalise to detect other malware families that use C&C. We

[1]https://www.stratosphereips.org/datasets-overview

manually label these connections using the available IoCs and intelligence platforms, such as *IBM X-Force Exchange*[2] and *AlienVault Open Threat Exchange*[3] The malware families and how we separate the unseen ones are reported in Table 6.2.

6.5.2 Known Malware Classification Performance

Similar to Chapter 4 and 5, we start our evaluation against known malware to measure the performance if the classifier is trained and tested for the same malware. We shuffled the two sets of data ten times. In the next step, we average the results obtained while limiting ourselves to two parameters. At first, the same malware is presented in both sets. Second, no flows from the training set matched the testing phase in terms of the infected hosts or the C&C server. For the three datasets, NIGHTVISION obtained the best $mF1$ with 86.67%, 89.52%, and 85.03%, while the baseline achieves 76.38%, 82.32% and 77.16%, which results $\Delta mF1$, +10.29%, +7.2% and 7.87%. For the other metrics, the performance of NIGHTVISION also achieves higher than baseline for mP and $mR$and lower for FPR. For the APT detection, NIGHTVISION obtains 95.64%, 80.79% and 0.18 compared to 79.94%, 73.62%, and 0.81% for the baseline, respectively. In botnet detection, 96.71%, 84.32% and 0.12 for NIGHTVISION while it is 88.53%, 77.91% and 0.43% for the same evaluation metrics.

We note that the baseline works better against botnet than APT (5.94 of $\Delta mF1$), while the gap performance in NIGHTVISION between them is lower (2.85 $\Delta mF1$) with higher performance than the baseline. This confirms our finding in the previous chapter, which is improving APT detection can help detect another type of malware, such as botnets. Also, we identify the FPRs of NIGHTVISION are lower than the baseline by at least two-thirds. In this case, the detection performance of NIGHTVISION does not result in a higher FPR similar to our works in the previous chapters. For the third dataset, NIGHTVISION sustain at 95.07%, 78.90% and 0.26 of mP, mR, and FPR, whereas the baseline at 84.44%, 72.63% and 0.97%. This yields an improvement of 12.59%, 8.63% and 73.19%, respectively.

[2]https://exchange.xforce.ibmcloud.com
[3]https://otx.alienvault.com

	Macro			*Weighted*				
Classifier Name	*mP*	*mR*	*mF*1	*wP*	*wR*	*wF*1	*FPR*	*A*
I. Dataset: APTs vs. Legitimate								
NIGHTVISION	**95.64**	**80.79**	**86.67**	**98.43**	**98.50**	**98.35**	**0.18**	**98.50**
Baseline	79.94	73.62	76.38	97.66	97.86	97.74	0.81	97.86
II. Dataset: Botnets vs. Legitimate								
NIGHTVISION	**96.71**	**84.32**	**89.52**	**99.01**	**99.05**	**98.98**	**0.12**	**99.05**
Baseline	88.53	77.91	82.32	98.25	98.40	98.28	0.43	98.40
III. Malicious vs. Legitimate								
NIGHTVISION	**95.07**	**78.90**	**85.03**	**97.55**	**97.65**	**97.39**	**0.26**	**97.65**
Baseline	84.44	72.63	77.16	95.92	96.38	96.00	0.97	96.38

Table 6.3: Known malware classification performance.

6.5.3 Unseen Malware Classification Performance

After we measure the performance against known malware, we aim to evaluate against unseen malware to investigate the ability of all classifiers to predict future malware. In this experiment, we train our classifiers on the training set of a set of malware. Then, we evaluate the performance against the unseen malware described in Table 6.2. NIGHTVISION obtains the best performance with *mF*1 of 79.69%, 83.63%, and 80.09% for the three datasets, while the baseline perform at 59.86%, 75.65% and 67.61%.

For the APT, NIGHTVISION achieves *mP* 92.88% compared to 59.92% for the baseline, which results in a difference of 32.96%. The gap between the two classifiers is lower regarding *mR*. NIGHTVISION obtains 72.88% compared to 59.79% for the baseline which yield Δ 13.09. The large gap in *mP* and *mR* reflects the high FPR for the baseline (2.04%) compared to NIGHTVISION (0.18%).

In the second dataset, we observe that the gap between the NIGHTVISION and the baseline is lower. The *mP* and *mR* of NIGHTVISION are 93.9% and 77.33% compared to 78.18% and 73.54%. This is due to the baseline aim to detect botnets exclusively. We also find the FPR for NIGHTVISION is 0.11%, which is lower than the baseline of 0.53%. However, all evaluation metrics for NIGHTVISION record higher than the baseline. Therefore, our hypothesis continues to be valid even for unseen malware, which defending against APT can improve the performance of botnet detection. In the third dataset, where both malicious types are presented, the *mP*

Classifier Name	*Macro*			*Weighted*				
	mP	*mR*	*mF*1	*wP*	*wR*	*wF*1	*FPR*	*A*
I. Dataset: APTs vs. Legitimate								
NIGHTVISION	**92.88**	**72.88**	**79.69**	**98.29**	**98.43**	**98.19**	**0.18**	**98.43**
Baseline	59.92	59.79	59.86	95.97	95.99	95.98	2.04	95.99
II. Dataset: Botnets vs. Legitimate								
NIGHTVISION	**93.90**	**77.33**	**83.63**	**99.17**	**99.23**	**99.14**	**0.11**	**99.23**
Baseline	78.18	73.54	75.65	98.61	98.71	98.65	0.53	98.71
III. Malicious vs. Legitimate								
NIGHTVISION	**93.21**	**73.36**	**80.09**	**97.54**	**97.71**	**97.37**	**0.25**	**97.71**
Baseline	70.94	65.28	67.61	95.25	95.82	95.49	1.61	95.82

Table 6.4: Unseen malware classification performance.

and *mR* of NIGHTVISION are 93.21% and 73.36% in comparison to 70.94% and 65.28% for the baseline. The FPR for the NIGHTVISION is up to 0.25% while the baseline performs at 1.61%.

6.5.4 Discussion

In this section, we discuss the performance difference between known (Section 6.5.2) and unseen malware (Section 6.5.3). The performance loss ($\Delta mF1$) for NIGHTVISION between known and unseen is 6.98%, 5.89% and 4.94% in the three datasets, while the baseline suffers a loss of 16.52%, 6.67% and 9.55%. We also observe that NIGHTVISION sustainability in terms of FPR between the two settings. The difference in FPRs for NIGHTVISION are 0%, 0.01%, 0.01%, while the baseline FPRs are increased up to 1.23%, 0.1% and 0.64%.

In the comparison of this work with the results of the previous chapter, where we focus on HTTP(S) connections, the performance decreased by 10.03% and 13.02% for known and unseen settings. The performance loss is at the expense of protocol-independent analysis and privacy-preserving requirements. NIGHTVISION can help in the detection regardless of the protocol, including HTTP(S), TCP/UDP-based, raw TCP and raw UDP protocols. We also identify the motivation for APTs using Raw TCP, where fewer features can be extracted, and the chances of escaping from NIDS are quite higher than in some protocols.

In this chapter, we attempt to explain the wavelet-based features with some examples. However, we confirm that the wavelet and Fourier transform are less explainable than clustering features

or those features presented in Chapter 4 and Chapter 5. For these reasons, we avoid using features based on the convolution of the input with the impulse function and extract information from its impulse response. It is quite useful to do that in many applications in the DSP field, but it is hard to explain the performance of these features in the NIDS area. However, one advantage of NIGHTVISION compared to the previous chapters is that the features are hard to guess for adversarial attacks.

6.6 Related Work

There is quite limited work on detecting APT beyond the HTTP(S) protocol. Fewer works on using DSP techniques for anomaly detection. Tracking the arrival time of packets is one of the traditional approaches for many NIDS [249, 225, 24]. However, it is more effective if such information is considered as a time-series problem to extract key information in the context of an end-to-end connection over a time window. One of the earlier works on utilising DSP techniques on network data was proposed by AsSadhan and Moura [231]. They investigate periodic components for botnets by applying Walker's large sample test to the periodogram. Also, BayWatch uses FFT to detect the beaconing behaviour for network data [19].

Other works are proposed for time-series data for other computer applications. Wen et al. [250] propose a dissimilarity measure for outlier and periodicity detection. The approach excludes outliers and noise to check if such periodicity exists by calculating the distance between the impulses. However, in the network data, it is challenging to decide which packets are noisy based on their arrival time. In addition, the internet lag can confuse the calculation of the distance, which may increase false negatives. Fan et al. [251] propose a replacement strategy to identify the high frequency in the data stream, which can be used to find the periodic items in data streams. Song et al. [252] present a technique based on DWT to identify the periodicity in database management systems. They use Fisher's test on the Huber Autocorrelation Function on the output of DWT. We investigated several techniques to identify the periodic components of APTs, but we did not identify any periodic components except DNS packets. The method of identifying the dynamic DNS, which uses autocorrelation of the periodogram of DNS packets, can also detect the beaconing at the DNS level. We did not design a specific module for detecting the periodicity of DNS because it is rarely found in APTs.

AsSadhan and Moura propose an approach based on the correlation between the control plane and actual ACK to identify SYN flood [230]. We follow the same approach for feature group No. 5 in Tabel 6.1, but we include the correlation analysis after the wavelet decomposition. Identifying a threshold in the correlation output automatically to separate malicious or legitimate is difficult. NIGHTVISION utilise the coefficient of variation ratios of two correlation functions to investigate if the connection behaves similarly in TCP, UDP and data planes.

Few works utilise the TCP flags for detection instead of the arrival time of packets. BOTection is a privacy-preserving bot detection system that uses a Markov Chain to model the network flow behaviour [79]. By using high-level flow features as states, the Markov Chains are able to capture the bots' network behaviour and provide behavioural features that are both content-agnostic and decryption-independent. These attributes are used to train a classifier that can recognise bot-generated flows and then determine which bot families generated those flows. NIGHTVISION uses TCP flags for profiling the connections and identifying the low profile, scan attacks, and the excessive use of non-application protocols used frequently by APTs.

Several works use AHC to measure the similarity between hosts [14, 19]. Botminer [14] uses AHC for classification after being trained by a couple of statistical features, such as flow packets and flows per hour. BayWatch [19] relies on AHC to identify the beaconing connections after training on FFT output. NIGHTVISION also uses AHC only for the training to generate the cluster labels for the decision tree during the testing time. Features used for AHCs and decision trees are extracted by FSM, which can accurately track the TCP flags in terms of volume, scanning and non-application protocol usage. Other works can only work for specific protocols, i.e. HTTP [24, 78]. NIGHTVISION is protocol-independent, which means it does not consider the type of protocol to extract useful information for detection.

6.7 Conclusions

In this chapter, we provided examples of DSP applications on network data with a proper sampling rate, low or high frequencies are filtered and transformed to the time domain to view these bands in the time domain. Then, we presented a novel approach to catching TTPs used by APTs. Our approach mainly relies on DSP techniques, including Wavelet and Fourier transform. We explain how to extract the key information on network data using DSP techniques. We also

utilise FSM to track TCP flags and feed three profiles, i.e., scanning, traffic volume, and non-application profiles. These profiles are built by a hybrid of hierarchical clustering and decision trees.

Next, we designed and implemented NIGHTVISION, a random-forest-based classifier which can detect APTs at the network level that are missed by previous chapters and by current defences. We recommend using NIGHTVISION as the last line of our framework presented in Figure 7.1, page 232. Adversarial attacks against DSP and clustering techniques, their robustness, and possible defences are left to future work.

In summary, our main contributions are:

- We update PAIRFLOW with stricter requirements for preserving the privacy of each user. The information does not include the domain name or the conversation for each protocol.
- We discuss the limitations and opportunities of using DSP for network data analysis, which is quite different from other conventional DSP applications.
- We implement NIGHTVISION, a privacy-preserving system, to detect evasive malicious communication using DSP and clustering techniques.
- We evaluate the classification performance of known and unseen malware, where all cases in the threat model presented in Section 5.1, page 156 are included.

Chapter 7

Conclusion and Future Work

In this final chapter, we conclude this thesis by summarizing the main achievements throughout our research journey. We also acknowledge the limitations encountered during the thesis and discuss potential avenues for future research. Section 7.1 summarise the main outcomes and contributions. Looking ahead, we identify promising directions for future research based on the gaps and unanswered questions that have emerged from our work in Section 7.3. These suggestions aim to inspire further investigations, expand upon our current findings, and explore new avenues for knowledge advancement. By outlining potential future research paths, we hope to encourage other researchers to build upon our work and contribute to the collective understanding of the research community.

7.1 Summary of Thesis Achievements

In this thesis, we began by examining recent global instances of APTs across different sectors. Next, we introduced three attack models and discussed their relevance to our study's scope. We reviewed state-of-the-art research on network-based detection of APT campaigns, which encompasses various protocols like HTTP, HTTPS, DNS, P2P, and custom RAT protocols. Following this, we identified gaps in current research and discussed potential future trends. We then provided background information on the techniques employed in our thesis, including supervised and unsupervised machine learning for cybersecurity and digital signal processing.

We analyzed 33 APT campaigns, focusing particularly on their communication methods across multiple channels and their use of evasion techniques. Our analysis covered the delivery method, vulnerabilities, backdoors, protocols, obfuscation, payloads, as well as the number of FQDNs and resolved IP addresses. Additionally, we described the TTPs utilized by APT campaigns from a network perspective, drawing from industry reports and our own technical analysis. Specifically, we explored the TTPs related to DNS, traffic, and alternative channels.

We presented an extensive study on the usage of domains within the context of APT C&C infrastructure. This study encompasses 63 APT campaigns spanning the last 13 years until August 2020, and draws from 125 public reports and 146 threat intelligence pulses from 35 leading security organizations. Based on this analysis, we proposed a detailed threat model for APT C&C usage, focusing on evasion techniques. Leveraging this insight, we examined the effectiveness of existing and novel features of domain names and their DNS infrastructure for detecting APT C&C domains. To facilitate this, we developed HawkEye, a system for classifying domain names requested in PCAP files. HawkEye includes modules for parsing and retrieving DNS data, as well as extracting features and classifying domains.

We Introduced PairFlow, a novel network flow format, which captures essential connection fields between specific host pairs over a customizable time frame. This is achieved through three main components: Tracking, Aggregation, and Encapsulation. Subsequently, we created EarlyCrow, a feature-based classifier that utilizes the contextual fields provided by PairFlow. EarlyCrow generates four sets of data focused on connections, hosts, destinations, and URLs, each with its own set of features grouped into a ContextualSummary. These multidimensional features contribute to the construction of more informative random forest trees for classification.

We presented an approach applicable to various protocol types, including raw TCP, which can be beneficial for APT operators or enterprises with strict standards on exposing DNS or HTTP-based features. To this end, we developed NightVision, which operates solely on the time series of network data without delving into the application layer headers (such as DNS, HTTP, TLS, etc.). NightVision utilizes the DWT to build a matching filter template, commonly used in radar applications for identifying specific malicious TTPs. We also drew inspiration from the use of ECG in identifying heart disease through DWT, leveraging statistical features extracted from each frequency band and selecting relevant bands for detecting malicious behaviour.

To enhance NIGHTVISION, we introduced enterprise-based measurements that can be customized for specific organizations. We primarily profile traffic volume, non-application activities, and scanning activities using a combination of AHC for training and a decision tree model for testing. Various features, including those tracking TCP flags activities using FSM, are utilized to distinguish normal behaviour from SYN, ACK, and FIN scan attacks.

7.2 Potential Deployment

The tools described in this thesis are intended to be utilized collectively, as depicted in Figure 7.1. Initially, the DNS requests are examined by HAWKEYE before establishing a connection. Once successful connections are established, they are analyzed, and contextual information is gathered through PAIRFLOW. The connection is then investigated and classified using EARLYCROW and NIGHTVISION, depending on the protocol type employed for the traffic. Our objective is to enhance security defences. Consequently, existing Host Intrusion Detection Systems (HIDS) play a crucial role in safeguarding against APT. We recommend employing our approach in conjunction with the research published in the field of HIDS.

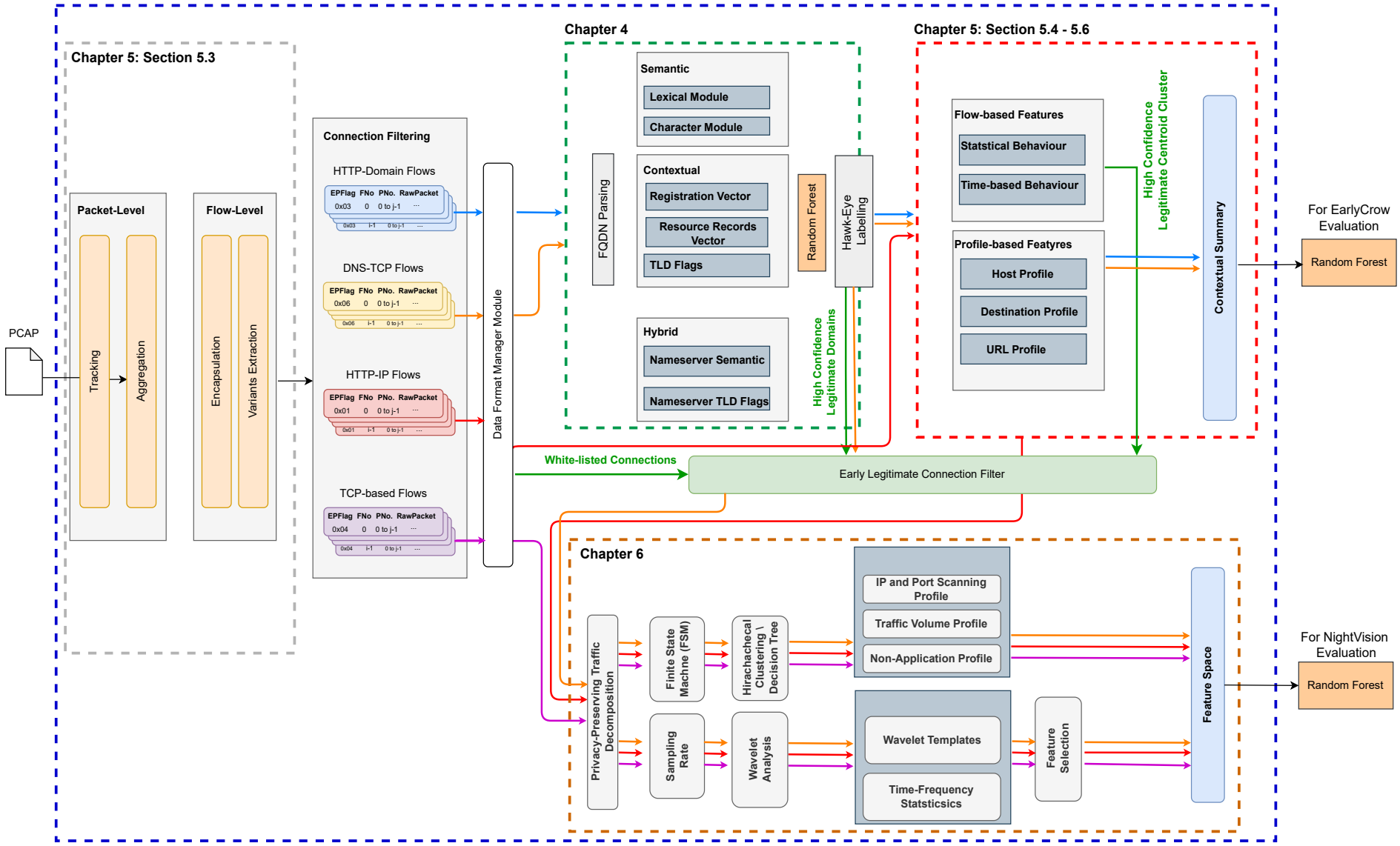


Figure 7.1: The structure of proposed deployment for the main contribution in this thesis.

7.3 Limitation and Future Work

Multiple directions for our future work can enhance the current approaches introduced in this thesis. In this section, we discuss the limitations and future work direction that can improve the future of APT detection.

7.3.1 Adversarial Attacks against Random Forest

We suggest evaluating the robustness of Chapters 4-6 against evasion attacks at the network level. There are some previous works on adversarial attacks against deep learning for NIDS [240, 253, 254, 255]. Fewer works focus on a discussion without implementation on traditional machine learning such as random forest [239, 238, 78]. Other attack vectors inspired by APTs should be considered as reasonable adversarial perturbations against a feature-based random forest. For instance, Random Interval-Time (RIT) [256, 17], is used to evade botnet, and Random Duplication (RD) [257]. The former generates adversarial samples by altering packets' time arrival, which may reuse for *Targeted and Stealthy*. The latter duplicates the number of packets randomly, which may be limited to be deployed for APTs but to measure robustness on such tiny perturbations. We suggest deploying both techniques at packet-level before the capture compiled by PairFlow.

Other techniques can also be considered when evaluating the robustness of APTs classifiers, including Practical Attack (PA) and Feature-Space Attack (FA) [258]. The Practical Black-box Attack (PBA) relies on traffic space but against PairFlow fields presented in Table 5.1. Therefore, the adversary knows what traffic features are selected for most classifiers, including PairFlow, which is used by EarlyCrow. Moreover, we argue that adversaries can access most features published in the past to adapt their traffic according to the targeted feature extraction to evade NIDS. Therefore, we refer to such an assumption as a Practical Gray-Box Attack (PGA) for those features used in the literature presented in Table 5.2. Another two attack configurations can be considered [259, 260], including Feature-space Grey-box Attack (FGA), and Feature-space Black-box Attack (FBA). FGA may attack all features produced by EarlyCrow, while the FBA is produced by the state-of-the-art baseline, i.e., reproducible Made and non-novel features in Table 5.2. However, there are several variations in finding the

optimization of evasion attacks. We suggest to adopt variants of Euclidean norm [261, 262] (l_p) for black-box configuration and free-range [263] for Gray-box.

7.3.2 Defences against Adversarial Attacks

Defending against adversarial examples is common for Deep Neural Networks (DNN). To defend against adversarial examples, a heuristic approach for training DNN is proposed by [264]. The distillation approach has been an effective defence against FGSM and JSMA attacks, and it recommends using one-hot encoding for features and then replacing the softmax function with:

$$\mathrm{F_i(x)} = \frac{e^{Z_i(x)/T}}{\sum_{j=1}^{Y} e^{Z_j(x)/T}} \tag{7.1}$$

Where $F_i(x)$ is the softmax layer, $Z_i(x)$ is its precedence layer, and T is the temperature parameter. The objective of this step is to create a new dataset D_n that can be used later with the original DNN with a typical softmax function.

The retraining approach is another defence method recommended by [261]. The basic idea is to add generated η against $F(x_i, \theta)$ into the training data D_T. Although the iterative retraining is intuitive, it can provide better generality over which the learned model is used to generate attacks or to defend against to [265]. However, [266] present a comparison of iterative retraining and distillation on malware detection, the latter outperforms the former by nearly 43% improvement. The same study presents two simple feature reduction approaches that degrade the misclassification ratio even more than the baseline [266]. Another improvement of the retraining approach is presented by Goodfellow et al. [267]. In addition to the above discussion, they add an adversarial loss function on $\eta + D_T$ in the direction of the network's gradient during the training in order to generalise adversarial inputs to linear models.

Other research directions focus on robust optimisation. Madry et al [268] use Danskin's classic in such a way that the gradient descent of adversarial example added to a data set is a descent direction for the loss function for the same data but without noise η. Furthermore, [269] argues that the Danskin-based robust optimisation approach which cannot be guaranteed and proposes a method to gain a malleable upper bound of the cost function on worst-case loss for DNN that uses ReLU activation functions. The optimisation is solved by the duality of linear

programming to produce the upper-bound [269].

For a random forest model, we suggest using Generative adversarial networks (GAN) [270] for defence. GAN is adopted in multiple works in anomaly detection. The rise of GAN came from the powerful idea of adopting binary classification problems with the presence of a two-player game, i.e. adversary and defender. The Generative framework creates a competition between a generator G and discriminator D models. A generator model G learns the distribution of the training dataset and, subsequently, constructs a model distribution that could be used to generate an indistinguishable fake sample. On the other hand, A discriminator D classifies a sample, whether it is coming from data distribution or from the learned model distribution - e.g. true MNIST images or fake ones generated by G.

One of the GAN applications has been demonstrated in [271] in the form of anomaly detection using a novel GAN architecture in order to generate additional samples to make a classifier more robust against adversarial attacks [272]. While the Generator model has been used for generating more adversarial attacks, an autoencoder D_2 has been used for anomaly detection. The generator G model and the real dataset both feed D_1 and D_2 to discriminate between the real and fake samples, and to minimise the loss function of $D_2/$. Several datasets have been evaluated, including NSL-KDD with 0.9% improvement [271].

Although GAN can be used for defence, it can also be used for attacks. Rigalki and Garcia [273] use GAN to mimic a Facebook chat traffic to hide malware's C&C traffic in order to evade the next generation of intrusion prevention systems (IPS) that use machine learning and 63.42 % of the C&C NetFlow traffic is successfully avoid being detected. We suggest using GAN to generate adversarial samples so we can attach them to the training set to cover these small variances that may confuse the Gini index used by the random forest. Also, the autoencoder may improve the latent for the defence, which may cover future attacks. There are multiple ways to improve the robustness, which is beyond the scope of this thesis.

7.3.3 HIDS-NIDS Integration

In this thesis, we encountered difficulties in detecting some attacks. To improve our detection framework, we suggest integrating some important information from the HIDS perspective into the NIDS framework. Several works in HIDS for APT detection rely on provenance graph

[274, 275, 276]. The provenance graph can summarise the information flow at the host level as discussed in Chapter 2. It also helps in detecting lateral movement in the enterprise network [277]. These works vary based on various improvements. First, an approach may focus on the reduction techniques to summarise the only important information, which saves memory space for reliable deployment. Second, other works propose different techniques for automated detection based on the audit data. Others explore the methods of visualisation to help forensics investigators to track the attacks.

In this thesis, we focus on the detection problem. Therefore, we suggest including summarised information in PairFlow to enhance the Contextual Summaries. Then, we can update all three lines of defence to include host-level features. Another potential direction is to propose a fourth line of defence which can focus on the host level only so it can filter the evident malicious processes from the beginning. One of the possible datasets that can be used for evaluating HIDS against APTs is DARPA Transparent Computing (TC) [73]. The dataset includes traces of a professional red team that simulates APT attacks against targets. The captures include the host-related logs such as memory, system calls, and events. For the network, they recently added NetFlow data. we discuss the limitation of the current data format in Section 5.1.1, 157. Therefore, we cannot use NetFlow data in our line of defences because of their dependency on more information at the packet level (PCAP). One way of utilising such a dataset is to improve NightVision (Chapter 6) to adopt NetFlow and integrate that with another component for host-based logs to achieve and evaluate the proposed approach.

Bibliography

[1] The MITRE Corporation. Command and control. `https://attack.mitre.org/tactics/TA0011/`. Accessed: 2021-12-18.

[2] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[3] FireEye Mandiant Lab. APT 1: Exposing one of china's cyber espionage units, Feb 2013.

[4] Blake E Strom, Joseph A Battaglia, Michael S Kemmerer, William Kupersanin, Douglas P Miller, Craig Wampler, Sean M Whitley, and Ross D Wolf. Finding cyber threats with att&ck™-based analytics. Technical report, The MITRE Corporation, 2017.

[5] Guodong Zhao, Ke Xu, Lei Xu, and Bo Wu. Detecting APT malware infections based on malicious dns and traffic analysis. *IEEE access*, 3:1132–1142, 2015.

[6] SciKit-Learn. Decision trees. `https://scikit-learn.org/stable/modules/tree.html`. 2023-08-10.

[7] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[8] SciPy. Matched filter using cross-correlation. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.correlate.html`. 2023-08-25.

[9] Paul S Addison. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press, 2017.

[10] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies*, 2015(2):46–64, 2015.

[11] Daniel C Castro, Ian Walker, and Ben Glocker. Causality matters in medical imaging. *Nature Communications*, 11(1):1–10, 2020.

[12] Colin Tankard. Advanced persistent threats and how to monitor and deter them. *Network security*, 2011(8):16–19, 2011.

[13] Paul Cichonski, Tom Millar, Tim Grance, and Karen Scarfone. Computer security incident handling guide. *NIST Special Publication*, 800(61):1–147, 2012.

[14] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *Proceedings of the 17th Conference on Security Symposium (USENIX Security)*, pages 139–154, 2008.

[15] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *Network and Distributed System Security Symposium (NDSS 11)*, pages 1–17, 2011.

[16] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From Throw-Away traffic to bots: Detecting the rise of DGA-Based malware. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 491–506, 2012.

[17] Leyla Bilge, Davide Balzarotti, William Robertson, Engin Kirda, and Christopher Kruegel. Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC 12)*, pages 129–138, 2012.

[18] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference (ACSAC 13)*, pages 199–208, 2013.

[19] Xin Hu, Jiyong Jang, Marc Ph Stoecklin, Ting Wang, Douglas L Schales, Dhilung Kirat, and Josyula R Rao. Baywatch: robust beaconing detection to identify infected hosts in large-scale enterprise networks. In *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 16)*, pages 479–490. IEEE, 2016.

[20] Alina Oprea, Zhou Li, Ting-Fang Yen, Sang H Chin, and Sumayah Alrwais. Detection of early-stage enterprise infection by mining large-scale log data. In *Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 15)*, pages 45–56. IEEE, 2015.

[21] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 10)*, volume 10, page 14, 2010.

[22] Jos van Roosmalen, Harald Vranken, and Marko van Eekelen. Applying deep learning on packet flows for botnet detection. In *Proceedings of the 33rd Annual ACM/SIGAPP Symposium on Applied Computing (SAC 18)*, pages 1629–1636. ACM, 2018.

[23] Almuthanna Alageel and Sergio Maffeis. HAWK-EYE: Holistic detection of APT command and control domains. In *In The 36th ACM/SIGAPP Symposium on Applied Computing (SAC 21)*, pages 1664–1673. ACM, 2021.

[24] Almuthanna Alageel and Sergio Maffeis. EARLYCROW: Detecting APT malware command and control over HTTP(S) using contextual summaries. In *25th International Information Security Conference (ISC 22)*, pages 290–316. Springer, 2022.

[25] Zhuoqun Fu, Mingxuan Liu, Yue Qin, Jia Zhang, Yuan Zou, Qilei Yin, Qi Li, and Haixin Duan. Encrypted malware traffic detection via graph-based network analysis. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 495–509, 2022.

[26] Răzvan Benchea, Cristina Vatamanu, Alexandru Maximciuc, and Victor Luncaşu. APT 28 under the scope a journey into exfiltrating intelligence and government information, 2015.

[27] ESET Research. Sednit update: How fancy bear spent the year. `https://www.welivesecurity.com/2017/12/21/sednit-update-fancy-bear-spent-year/`, December 2017. Accessed: 2019-04-10.

[28] 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. Accessed: 2019-04-14.

[29] Cybereason Labs and Assaf Dahan. Operation cobalt kitty cybereason labs analysis, 2017.

[30] F-Secure Labs Threat Intelligence. The dukes 7 years of russian cyberespionage, 2016.

[31] Dmitri Alperovitch. Bears in the midst: Intrusion into the democratic national committee. https://www.crowdstrike.com/blog/bears-midst-intrusion-democratic-national-committee/. Accessed: 2019-04-14.

[32] Symantec Security Response. Buckeye cyberespionage group shifts gaze from us to hong kong. https://www.symantec.com/connect/blogs/buckeye-cyberespionage-group-shifts-gaze-us-hong-kong, September 2016. Accessed: 2019-04-23.

[33] Harlan Carvey. Where you at?: Indicators of lateral movement using at.exe on windows 7 systems. https://www.secureworks.com/blog/where-you-at-indicators-of-lateral-movement-using-at-exe-on-windows-7-systems, September 2014. Accessed: 2019-04-14.

[34] Aaron Shelmire. Evasive maneuvers by the wekby group with custom rop-packing and dns covert channels. https://www.anomali.com/blog/evasive-maneuvers-the-wekby-group-attempts-to-evade-analysis-via-custom-rop, July 2015. Accessed: 2019-04-14.

[35] FireEye Lab. APT 37 (reaper) the overlooked north korean actor, special report, 2018.

[36] Brown Farinholt, Mohammad Rezaeirad, Damon McCoy, and Kirill Levchenko. Dark matter: Uncovering the darkcomet rat ecosystem. In *Proceedings of The Web Conference (WWW 20)*, pages 2109–2120, 2020.

[37] Mohammad Rezaeirad, Brown Farinholt, Hitesh Dharmdasani, Paul Pearce, Kirill Levchenko, and Damon McCoy. Schrödinger's RAT: Profiling the stakeholders in the remote access trojan ecosystem. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1043–1060, 2018.

[38] Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, and Kirill Levchenko. To catch a ratter: Monitoring the behavior of amateur darkcomet rat operators in the wild. In *IEEE Symposium on Security and Privacy (S&P 17)*, pages 770–787. IEEE, 2017.

[39] Max Heinemeyer. Fin7.5: the infamous cybercrime rig "fin7" continues its activities. `https://securelist.com/fin7-5-the-infamous-cybercrime-rig-fin7-continues-its-activities/90703//`. Accessed: 2021-07-18.

[40] Johannes de Vries, Hans Hoogstraaten, Jan van den Berg, and Semir Daskapan. Systems for detecting advanced persistent threats: A development roadmap using intelligent data analysis. In *2012 International Conference on Cyber Security*, pages 54–61. IEEE, 2012.

[41] Paul Giura and Wei Wang. A context-based detection framework for advanced persistent threats. In *International Conference on Cyber Security*, pages 69–74. IEEE, 2012.

[42] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[43] Tarun Yadav and Arvind Mallari Rao. Technical aspects of cyber kill chain. In *International Symposium on Security in Computing and Communication*, pages 438–452. Springer, 2015.

[44] Dennis Kiwia, Ali Dehghantanha, Kim-Kwang Raymond Choo, and Jim Slaughter. A cyber kill chain based taxonomy of banking trojans for evolutionary computational intelligence. *Journal of computational science*, 27:394–409, 2018.

[45] Georgios Ioannou, Panos Louvieris, Natalie Clewley, and Gavin Powell. A markov multiphase transferable belief model: An application for predicting data exfiltration apts. In *Proceedings of the 16th International Conference on Information Fusion*, pages 842–849. IEEE, 2013.

[46] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, R Sekar, and VN Venkatakrishnan. Holmes: real-time APT detection through correlation of suspicious information flows. *arXiv preprint arXiv:1810.01594*, 2018.

[47] Sunu Mathew, Shambhu Upadhyaya, Moisés Sudit, and Adam Stotz. Situation awareness of multistage cyber attacks by semantic event fusion. In *Military Communications Conference (MILCOM 10)*, pages 1286–1291. IEEE, 2010.

[48] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials*, 2019.

[49] Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon, Daesung Moon, and Jong Hyuk Park. A comprehensive study on APT attacks and countermeasures for future networks and communications: challenges and solutions. *The Journal of Supercomputing*, 75:4543–4574, 2019.

[50] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109:127–141, 2016.

[51] Ivo Friedberg, Florian Skopik, Giuseppe Settanni, and Roman Fiedler. Combating advanced persistent threats: From network event correlation to incident detection. *Computers & Security*, 48:35–57, 2015.

[52] Fàtima Barceló-Rico, Anna I Esparcia-Alcázar, and Antonio Villalón-Huerta. Semi-supervised classification system for the detection of advanced persistent threats. In *Recent Advances in Computational Intelligence in Defense and Security*, pages 225–248. Springer, 2016.

[53] Jiazhong Lu, Xiaosong Zhang, Wang Junfeng, and Ying Lingyun. APT traffic detection based on time transform. In *2016 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS 16)*, pages 9–13. IEEE, 2016.

[54] Andrew Vance. Flow based analysis of advanced persistent threats detecting targeted attacks in cloud computing. In *2014 First International Scientific-Practical Conference Problems of Infocommunications Science and Technology*, pages 173–176. IEEE, 2014.

[55] Sana Siddiqui, Muhammad Salman Khan, Ken Ferens, and Witold Kinsner. Detecting advanced persistent threats using fractal dimension based machine learning classifica-

tion. In *Proceedings of the 2016 ACM on international workshop on security and privacy analytics*, pages 64–69. ACM, 2016.

[56] Massimiliano Albanese, Sushil Jajodia, and Sridhar Venkatesan. Defending from stealthy botnets using moving target defenses. *IEEE Security & Privacy*, 16(1):92–97, 2018.

[57] Guillaume Brogi and Valérie Viet Triem Tong. Terminaptor: Highlighting advanced persistent threats through information flow tracking. In *8th IFIP International Conference on New Technologies, Mobility and Security (NTMS 16)*, pages 1–5. IEEE, 2016.

[58] Michael Atighetchi, John Griffith, Ian Emmons, David Mankins, and Richard Guidorizzi. Federated access to cyber observables for detection of targeted attacks. In *IEEE Military Communications Conference (MILCOM 14)*, pages 60–66. IEEE, 2014.

[59] Parth Bhatt, Edgar Toshiro Yano, and Per Gustavsson. Towards a framework to detect multi-stage advanced persistent threats attacks. In *IEEE 8th International Symposium on Service Oriented System Engineering*, pages 390–395. IEEE, 2014.

[60] Jeslin Thomas John. State of the art analysis of defense techniques against advanced persistent threats. *Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM) Focal Topic: Advanced Persistent Threats*, 63, 2017.

[61] Saranya Chandran, P Hrudya, and Prabaharan Poornachandran. An efficient classification model for detecting advanced persistent threat. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI 15)*, pages 2001–2009. IEEE, 2015.

[62] Yuan Wang, Yongjun Wang, Jing Liu, and Zhijian Huang. A network gene-based framework for detecting advanced persistent threats. In *Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 97–102. IEEE, 2014.

[63] John R Koza and John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[64] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement (IMC 03)*, pages 234–247. ACM, 2003.

[65] Mila Parkour. Contagio malware database. https://www.mediafire.com/folder/c2az029ch6cke/TRAFFIC. Accessed: 2019-04-05.

[66] Wireshark-tshark. https://www.wireshark.org/docs/man-pages/tshark.html. Accessed: 2019-04-12.

[67] Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and VN Venkatakrishnan. SLEUTH: Real-time attack scenario reconstruction from COTS audit data. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 487–504, 2017.

[68] Md Nahid Hossain, Junao Wang, Ofir Weisse, R Sekar, Daniel Genkin, Boyuan He, Scott D Stoller, Gan Fang, Frank Piessens, Evan Downing, et al. Dependence-preserving data compaction for scalable forensic analysis. In *27th USENIX Security Symposium USENIX Security 18)*, pages 1723–1740, 2018.

[69] Vulnerability metrics. https://nvd.nist.gov/vuln-metrics/cvss. Accessed: 2019-04-26.

[70] B Schneier. Attack trees-modeling security threats. counterpane internet security, 1999.

[71] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238, 1999.

[72] Florian Skopik, Giuseppe Settanni, Roman Fiedler, and Ivo Friedberg. Semi-synthetic data set generation for security software evaluation. In *12th Annual International Conference on Privacy, Security and Trust*, pages 156–163. IEEE, 2014.

[73] Angelos D. Keromytis. Transparent computing engagement 3 data release. https://github.com/darpa-i2o/Transparent-Computing. Accessed: 2019-04-01.

[74] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. Generic black-box end-to-end attack against state of the art api call based malware classifiers. In *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID 18)*, pages 490–510. Springer, 2018.

[75] Enrico Mariconti, Jeremiah Onaolapo, Gordon Ross, and Gianluca Stringhini. What's your major threat? on the differences between the network behavior of targeted and

commodity malware. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 599–608. IEEE, 2016.

[76] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[77] Luca Boero, Marco Cello, Mario Marchese, Enrico Mariconti, Talha Naqash, and Sandro Zappatore. Statistical fingerprint-based intrusion detection system (sf-ids). *International Journal of Communication Systems*, 30(10):e3225, 2017.

[78] Alina Oprea, Zhou Li, Robin Norris, and Kevin Bowers. Made: Security analytics for enterprise threat detection. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC 18)*, pages 124–136, 2018.

[79] Bushra A Alahmadi, Enrico Mariconti, Riccardo Spolaor, Gianluca Stringhini, and Ivan Martinovic. Botection: Bot detection by building markov chain models of bots network behavior. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIACCS 20)*, pages 652–664, 2020.

[80] SciKit-Learn. The iris dataset. `https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html`. 2023-08-10.

[81] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[82] SciKit-Learn. Random forest classifier. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`. 2023-08-10.

[83] Yun Shen, Enrico Mariconti, Pierre Antoine Vervier, and Gianluca Stringhini. Tiresias: Predicting security events through deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 592–605, 2018.

[84] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. ATLAS: A sequence-based learning approach for attack investigation. In *30th USENIX security symposium (USENIX security 21)*, pages 3005–3022, 2021.

[85] Steven W Smith. The scientist and engineer's guide to digital signal processing, 1997.

[86] 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, 2013.

[87] Gernot Vormayr, Tanja Zseby, and Joachim Fabini. Botnet communication patterns. *IEEE Communications Surveys & Tutorials*, 19(4):2768–2796, 2017.

[88] Martin Ussath, David Jaeger, Feng Cheng, and Christoph Meinel. Advanced persistent threats: Behind the scenes. In *Annual Conference on Information Science and Systems (CISS 16)*, pages 181–186. IEEE, 2016.

[89] Antoine Lemay, Joan Calvet, François Menet, and José M Fernandez. Survey of publicly available reports on advanced persistent threat actors. *Computers & Security*, 72:26–59, 2018.

[90] Crowdstrike Global Intelligence Team. *CrowdStrike Intelligence Report: Putter Panda*. CrowdStrike, June 2014.

[91] Christopher A. Korban, Douglas P. Miller, Adam Pennington, Cody B. Thomas, and The MITRE Corporation. APT 3 adversary emulation plan, September 2017.

[92] Ned Moran, Mike Scott, Mike Oppenheim, and Joshua Homan. Operation double tap. `https://www.fireeye.com/blog/threat-research/2014/11/operation_doubletap.html`, November 2014. Accessed: 2019-04-23.

[93] Micah Yates. *APT 3 Uncovered: The code evolution of Pirpi*. Palo Alto Networks, June 2017.

[94] PwC and BAE Systems. Operation cloud hopper, April 2017.

[95] PwC and BAE Systems. Operation cloud hopper technical annex, April 2017.

[96] Ned Moran and Mike Oppenheim. Darwin's favorite APT group. `https://www.fireeye.com/blog/threat-research/2014/09/darwins-favorite-apt-group-2.html`. Accessed: 2019-04-18.

[97] Nart Villeneuve, James T. Bennett, Ned Moran, Thoufique Haq, Mike Scott, and Kenneth Geers. Operation "ke3chang": Targeted attacks against ministries of foreign affairs, 2014.

[98] NCC Group. Apt15 is alive and strong: An analysis of royalcli and royaldns. https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2018/march/apt15-is-alive-and-strong-an-analysis-of-royalcli-and-royaldns/. Accessed: 2019-04-18.

[99] Ryann Winters. The eps awakens - part 2. https://www.fireeye.com/blog/threat-research/2015/12/the-eps-awakens-part-two.html, December 2015. Accessed: 2019-04-18.

[100] Genwei Jiang, Dan Caselden, and Ryann Winters. The eps awakens. https://www.fireeye.com/blog/threat-research/2015/12/the_eps_awakens.html, December 2015. Accessed: 2019-04-18.

[101] FireEye Labs / FireEye Threat Intelligence. Hiding in plain sight: Fireeye and microsoft expose, May 2015.

[102] Josh Grunzweig and Bryan Lee. New attacks linked to c0d0so0 group. https://unit42.paloaltonetworks.com/new-attacks-linked-to-c0d0s0-group/. Accessed: 2019-04-10.

[103] Ian Ahl. Privileges and credentials: Phished at the request of counsel. https://www.fireeye.com/blog/threat-research/2017/06/phished-at-the-request-of-counsel.html. Accessed: 2019-04-10.

[104] Dell SecureWorks Counter Threat Unit Threat Intelligence. Threat group 3390 cyberespionage. https://www.secureworks.com/research/threat-group-3390-targets-organizations-for-cyberespionage, August 2015. Accessed: 2019-04-23.

[105] Counter Threat Unit Research Team. Bronze union cyberespionage persists despite disclosures. https://www.secureworks.com/research/bronze-union, June 2017. Accessed: 2019-04-23.

[106] Denis Legezo. Luckymouse hits national data center to organize country-level waterholing campaign. https://securelist.com/luckymouse-hits-national-data-center/86083/, June 2018. Accessed: 2019-04-23.

[107] FireEye Lab. APT 28: A window into russia's cyber espionage operations and a special report, 2014.

[108] Charlie Anthe, Patti Chrzan, Elia Florio, Chad Foster, Paul Henry, Jeff Jones, Nam Ng, Niall O'Sullivan, Daryl Pecelj, Anthony Penta, Ina Ragragio, Tim Rains, and Paul Rebriy. *Microsoft Security Intelligence Report*, volume 19. Microsoft, June 2015.

[109] Security Response Attack Investigation Team. APT 28: New espionage operations target military and government organizations. `https://www.symantec.com/blogs/election-security/apt28-espionage-military-government`, October 2018. Accessed: 2019-04-10.

[110] ESET Lab. *En Route with Sednit Part 2: Observing the Comings and Goings*, volume 1. ESET, October 2016.

[111] Department of Homeland Security and Federal Bureau of Investigation. Grizzly steppe – russian malicious cyber activity, December 2016.

[112] Matthew Dunwoody, Andrew Thompson, Ben Withnell, Jonathan Leathery, Michael Matonis, and Nick Carr. Not so cozy: An uncomfortable examination of a suspected apt29 phishing campaign. `https://www.fireeye.com/blog/threat-research/2018/11/not-so-cozy-an-uncomfortable-examination-of-a-suspected-apt29-phishing-campaign.html`, November 2018. Accessed: 2019-04-14.

[113] FireEye Labs. *APT 30 and the mechanics of a long-running cyber espionage operation*. FireEye, April 2015.

[114] Romain Dumont. Fake or fake: Keeping up with oceanlotus decoys. `https://www.welivesecurity.com/2019/03/20/fake-or-fake-keeping-up-with-oceanlotus-decoys/`. Accessed: 2019-04-14.

[115] Nick Carr. Cyber espionage is alive and well: APT 32 and the threat to global corporations. `https://www.fireeye.com/blog/threat-research/2017/05/cyber-espionage-apt32.html`, May 2017. Accessed: 2019-06-10.

[116] Jacqueline O'Leary, Josiah Kimble, Kelli Vanderlee, and Nalani Fraser. Insights into iranian cyber espionage: APT 33 targets aerospace and energy sectors and has ties

to destructive malware. https://www.fireeye.com/blog/threat-research/2017/09/apt33-insights-into-iranian-cyber-espionage.html. Accessed: 2019-04-20.

[117] Security Response Attack Investigation Team at Symantec. Elfin: Relentless espionage group targets multiple organizations in saudi arabia and u.s. https://www.symantec.com/blogs/threat-intelligence/elfin-apt33-espionage. Accessed: 2019-04-20.

[118] Geoff Ackerman, Rick Cole, Andrew Thompson, Alex Orleans, and Nick Carr. Overruled: Containing a potentially destructive adversary. https://www.fireeye.com/blog/threat-research/2018/12/overruled-containing-a-potentially-destructive-adversary.html. Accessed: 2019-04-20.

[119] Manish Sardiwal, Vincent Cannon, Nalani Fraser, Yogesh Londhe, Nick Richard, and Jacqueline O'Leary. New targeted attack in the middle east by apt34, a suspected iranian threat group, using cve-2017-11882 exploit. https://www.fireeye.com/blog/threat-research/2017/12/targeted-attack-in-middle-east-by-apt34.html. Accessed: 2019-04-21.

[120] Robert Falcone and Bryan Lee. Oilrig uses ismdoor variant; possibly linked to greenbug threat group. https://unit42.paloaltonetworks.com/unit42-oilrig-uses-ismdoor-variant-possibly-linked-greenbug-threat-group/. Accessed: 2019-04-21.

[121] Threat Intelligence and Check Point Research. Rocket kitten: A campaign with 9 lives, 2015.

[122] FireEye Threat Intelligence. China-based cyber threat group uses dropbox for malware communications and targets hong kong media outlets. https://www.fireeye.com/blog/threat-research/2015/11/china-based-threat.html. Accessed: 2019-04-20.

[123] Novetta. Operation blockbuster: Unraveling the long thread of the sony attack, August 2017.

[124] Novetta. Operation blockbuster: Remote administration tools and content staging malware report, August 2017.

[125] Positive Technologies. Colbat snatch, December 2016.

[126] Vanja Svajcer. Multiple cobalt personality disorder. https://blog.talosintelligence.com/2018/07/multiple-cobalt-personality-disorder.html, July 2018. Accessed: 2019-04-05.

[127] Symantec Security Response. *Dragonfly: Cyberespionage AttacksAgainst Energy Suppliers*. Symantec, July 2014.

[128] Kaspersky Lab. *The Duqu 2.0 technical details*. Kaspersky, June 2015.

[129] Gavin O'Gorman and Geoff McDonald. The elderwood project, September 2012.

[130] Nick Carr, Kimberly Goody, Steve Miller, and Barry Vengerik. On the hunt for fin7: Pursuing an enigmatic and evasive global criminal operation. https://www.fireeye.com/blog/threat-research/2018/08/fin7-pursuing-an-enigmatic-and-evasive-global-criminal-operation.html, August 2018. Accessed: 2019-05-01.

[131] Fred Plan, Nalani Fraser, Jacqueline O'Leary, Vincent Cannon, and Ben Read. Apt40: Examining a china-nexus espionage actor. https://www.fireeye.com/blog/threat-research/2019/03/apt40-examining-a-china-nexus-espionage-actor.html, March 2019. Accessed: 2019-05-04.

[132] FireEye Lab. Suspected chinese cyber espionage group (temp.periscope) targeting u.s. engineering and maritime industries. https://www.fireeye.com/blog/threat-research/2018/03/suspected-chinese-espionage-group-targeting-maritime-and-engineering-industries.html, March 2018. Accessed: 2019-05-04.

[133] Kurt Baumgartner and Maxim Golovkin. The msnmm campaigns the earliest naikon APT campaigns, May 2015.

[134] ThreatConnect Inc. and Defense Group Inc. Camerashy closing the aperture on china's unit 78020, 2015. Accessed: 2019-01-25.

[135] Daniel Lunghi, Jaromir Horejsi, and Cedric Pernet. Untangling the patchwork cyberespionage group, October 2018.

[136] Matthew Meltzer, Sean Koessel, and Steven Adair. Patchwork APT group targets us think tanks. https://www.volexity.com/blog/2018/06/07/patchwork-apt-group-targets-us-think-tanks/, June 2018. Accessed: 2019-04-02.

[137] Nicholas Griffin Andy Settle and Abel Toro. *Monsoon – analysis of an APT campaign espionage and data loss under the cover of current affairs*, volume 1. Raytheon - Forcepoint Security Labs, September 2016.

[138] F-Secure Labs Security Response. *Blackenergy and Quedagh*. F-Secure, 2014.

[139] Global Research and Analysis Team. *The ProjectSauron APT. Technical analysis*. Kaspersky, August 2016.

[140] Semantec Security Response. Backdoor.remsec indicators of compromise, August 2016.

[141] Symantec Security Response. *Regin: Top-tier espionage tool enables stealthy surveillance*. Symantec, Auguest 2014.

[142] Global Research and Analysis Team. *Cloud Atlas: RedOctober APT is back in style*. Kaspersky, December 2014. Accessed: 2019-05-04.

[143] Global Research and Analysis Team. *Red October" Diplomatic Cyber Attacks Investigation*. Kaspersky, January 2013. Accessed: 2019-05-04.

[144] Global Research and Analysis Team. *Red October" – Part Two, the Modules*. Kaspersky, January 2013. Accessed: 2019-05-04.

[145] Jon Gross and Jim Walter. Puttering into the future.... `https://threatvector.cylance.com/en_us/home/puttering-into-the-future.html`, January 2016. Accessed: 2019-04-23.

[146] Nikolaos Pantazopoulos. Decoding network data from a gh0st rat variant. `https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2018/april/decoding-network-data-from-a-gh0st-rat-variant/`, April 2018. Accessed: 2019-04-18.

[147] Josh Grunzweig, Mike Scott, and Bryan Lee. 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/`, September 2014. Accessed: 2019-04-14.

[148] NCC Group. Emissary panda – a potential new malicious tool. `https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2018/may/`

emissary-panda-a-potential-new-malicious-tool/, May 2018. Accessed: 2019-04-23.

[149] Matthew Dunwoody. Apt29 domain fronting with tor. https://www.fireeye.com/blog/threat-research/2017/03/apt29_domain_frontin.html. Accessed: 2019-04-14.

[150] Dave Lassalle, Sean Koessel, and Steven Adair. Oceanlotus blossoms: Mass digital surveillance and attacks targeting asean, asian nations, the media, human rights groups, and civil society. https://www.volexity.com/blog/2017/11/06/oceanlotus-blossoms-mass-digital-surveillance-and-exploitation-of-asean-nations-the-media-human-rights-and-civil-society/, November 2017. Accessed: 2019-06-10.

[151] ESET. Oceanlotus old techniques, new backdoor, March 2018.

[152] Palo Alto - Unit 42. OilReg. https://pan-unit42.github.io/playbook_viewer/. Accessed: April 2019.

[153] Josh Grunzweig and Robert Falcone. Oilrig malware campaign updates toolset and expands targets. https://unit42.paloaltonetworks.com/unit42-oilrig-malware-campaign-updates-toolset-and-expands-targets/. Accessed: 2019-04-21.

[154] Bryan Lee and Robert Falcone. Magic hound campaign attacks saudi targets. https://unit42.paloaltonetworks.com/unit42-magic-hound-campaign-attacks-saudi-targets/, February 2017. Accessed: 2019-04-23.

[155] Novetta. Loaders and installers and uninstallers report, August 2017.

[156] Ryan Sherstobitoff. Lazarus resurfaces, targets global banks and bitcoin users. https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/lazarus-resurfaces-targets-global-banks-bitcoin-users/, Feb 2018. Accessed: 2019-04-08.

[157] Vesta Matveena. Secrets of cobalt: How cobalt hackers bypass your defenses. https://www.group-ib.com/blog/cobalt, August 2017. Accessed: 2019-04-05.

[158] Security Response Attack Investigation Team. *Dragonfly: Western energy sector targeted by sophisticated attack group*. Symantec, October 2017. Accessed: 2019-04-09.

[159] US-CERT at Department of Homeland Security. Russian government cyber activity targeting energy and other critical infrastructure sectors. `https://www.us-cert.gov/ncas/alerts/TA18-074A`, March 2018. Accessed: 2019-04-09.

[160] James T. Bennett and Barry Vengerik. Behind the carbanak backdoor. `https://www.fireeye.com/blog/threat-research/2017/06/behind-the-carbanak-backdoor.html`, June 2017. Accessed: 2019-05-01.

[161] F-Secure Labs Threat Intelligence. Nanhaishu rating the south china sea, July 2016.

[162] Cymmetria Inc. Unveiling patchwork – the copy-paste apt: A targeted attack caught with cyber deception. `https://cymmetria.com/research/patchwork-targeted-attack/`, 2016. Accessed: 2019-04-02.

[163] Kaspersky Lab. *The regin platform nation-state ownage of gsm networks*. Kaspersky, November 2014.

[164] Kenton Born and David Gustafson. Detecting DNS tunnels using character frequency analysis. *arXiv preprint arXiv:1004.4358*, 2010.

[165] Cheng Qi, Xiaojun Chen, Cui Xu, Jinqiao Shi, and Peipeng Liu. A bigram based real time dns tunnel detection approach. *Procedia Computer Science*, 17:852–860, 2013.

[166] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. A comprehensive measurement study of domain generating malware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 263–278, 2016.

[167] Aditya K Sood and Sherali Zeadally. A taxonomy of domain-generation algorithms. *IEEE Security & Privacy*, 14(4):46–53, 2016.

[168] Yu Fu, Lu Yu, Oluwakemi Hambolu, Ilker Ozcelik, Benafsh Husain, Jingxuan Sun, Karan Sapra, Dan Du, Christopher Tate Beasley, and Richard R Brooks. Stealthy domain generation algorithms. *IEEE Transactions on Information Forensics and Security*, 12(6):1430–1443, 2017.

[169] Shahar Tavor. Brazking android malware upgraded and targeting brazilian banks. https://securityintelligence.com/posts/brazking-android-malware-upgraded-targeting-brazilian-banks/. 2021-07-17.

[170] FortiGuard Threat Intelligence. Weekly threat briefs. https://www.fortiguard.com/resources/threat-brief/2018/06/08/fortiguard-threat-intelligence-brief-june-08-2018. 2018-06-08.

[171] US-CERT. Malware analysis report- 10135536-b. https://www.cisa.gov/sites/default/files/publications/MAR-10135536-B_WHITE.PDF. 2017-11-13.

[172] ESET Research. Okrum and ketrican: an overview of recent ke3chang group activity, December 2017. Accessed: 2019-07-01.

[173] Mandiant. Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor. https://support.solarwinds.com/SuccessCenter/s/article/Orion-Improvement-Program?language=en_US. 2022-07-01.

[174] Mandiant. Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor. https://www.mandiant.com/resources/blog/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor/. 2022-05-10.

[175] Douglas Bienstock Geoff Ackerman Rufus Brown, Van Ta and John Wolfram. Does this look infected? a summary of apt41 targeting u.s. state governments. https://www.mandiant.com/resources/blog/apt41-us-state-governments. Accessed: 2023-06-29.

[176] Graham Holmes. Evolution of attacks on cisco ios devices. https://blogs.cisco.com/security/evolution-of-attacks-on-cisco-ios-devices. 2015-10-08.

[177] TrendMicro. Remcos malware information. https://success.trendmicro.com/dcx/s/solution/1123281-remcos-malware-information?language=en_US&sfdcIFrameOrigin=null. 2019-12-30.

[178] Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock. Character level based detection of dga domain names. In *International Joint Conference on Neural Networks (IJCNN 18)*, pages 1–8. IEEE, 2018.

[179] Xiaoqing Sun, Mingkai Tong, Jiahai Yang, Liu Xinran, and Liu Heng. Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 19)*, pages 399–412, 2019.

[180] Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike Meyer. FANCI: Feature-based automated NXDomain classification and intelligence. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1165–1181, 2018.

[181] Ke Tian, Steve TK Jan, Hang Hu, Danfeng Yao, and Gang Wang. Needle in a haystack: Tracking down elite phishing domains in the wild. In *Proceedings of the Internet Measurement Conference (IMC 18)*, pages 429–442, 2018.

[182] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *Network and Distributed System Security Symposium (NDSS 10)*, 2010.

[183] Robert Falcone, Bryan Lee, and Tom Lancaster. New threat actor group darkhydrus targets middle east government. `https://unit42.paloaltonetworks.com/unit42-new-threat-actor-group-darkhydrus-targets-middle-east-government/`, July 2018. Accessed: 2019-12-13.

[184] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. The long "taile" of typosquatting domain names. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 191–206, 2014.

[185] Securitytrails. `https://securitytrails.com/corp/api`. Accessed: 2019-01-22.

[186] M. Majkowski J. Abley, O. Gudmundsson and E. Hunt. Providing minimal-sized responses to dns queries that have qtype=any, 1987.

[187] Publicsuffix. `https://publicsuffix.org/list/public_suffix_list.dat`. Accessed: 2019-10-18.

[188] Weina Niu, Xiaosong Zhang, GuoWu Yang, Jianan Zhu, and Zhongwei Ren. Identifying APT malware domain based on mobile dns logging. *Mathematical Problems in Engineering*, 2017, 2017.

[189] Jan Spooren, Thomas Vissers, Peter Janssen, Wouter Joosen, and Lieven Desmet. Premadoma: an operational solution for dns registries to prevent malicious domain registrations. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC 19)*, pages 557–567, 2019.

[190] Natural language toolkit. `https://www.nltk.org/`. Accessed: 2019-09-01.

[191] Wolf Garbe. Fast word segmentation with a triangular matrix. `https://gist.github.com/wolfgarbe`, April 2018. Accessed: 2019-09-24.

[192] Grant Jenks. Python word segmentation. `http://www.grantjenks.com/docs/wordsegment/`, 2018. Accessed: 2019-09-24.

[193] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *The Network and Distributed System Security Symposium (NDSS 19)*, 2019.

[194] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for DNS. In *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[195] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. *LEET*, 10:6–6, 2010.

[196] Zhou Li, Sumayah Alrwais, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *IEEE S&P*, pages 112–126. IEEE, 2013.

[197] Dynamic DNS: Data Exfiltration. `https://www.rsa.com/content/dam/en/solution-brief/asoc-dynamic-dns-data-exfiltration.pdf`. September 202020.

[198] Splunk Security Essentials Docs. `https://docs.splunksecurityessentials.com/content-detail/sse_dyndns/`. September 2020.

[199] Calypso APT: new group attacking state institutions. https://www.ptsecurity.com/ww-en/analytics/calypso-apt-2019/. Sepetember 2020.

[200] Suphannee Sivakorn, Kangkook Jee, Yixin Sun, Lauri Korts-Pärn, Zhichun Li, Cristian Lumezanu, Zhenyu Wu, Lu-An Tang, and Ding Li. Countering malicious processes with process-dns association. In *The Network and Distributed System Security Symposium (NDSS 19)*, 2019.

[201] Join the fight against phishing. https://www.phishtank.com/. July 2020.

[202] Eihal Alowaisheq, Peng Wang, Sumayah A Alrwais, Xiaojing Liao, XiaoFeng Wang, Tasneem Alowaisheq, Xianghang Mi, Siyuan Tang, and Baojun Liu. Cracking the wall of confinement: Understanding and analyzing malicious domain take-downs. In *The Network and Distributed System Security Symposium (NDSS 19)*, 2019.

[203] ESET. Machete just got sharper venezuelan government institutions under attack, July 2019.

[204] Record low for number of .uk domains suspended by law enforcement. https://www.nominet.uk/record-low-for-number-of-uk-domains-suspended-by-law-enforcement. December 2021.

[205] Daniel Burrus. Converting information into knowledge-based assets. https://www.burrus.com/2018/08/converting-information-into-knowledge-based-assets/. August 2018.

[206] Vern Paxson. Bro: A system for detecting network intruders in real-time. volume 31, pages 2435–2463. Elsevier, 1999.

[207] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy (S&P 10)*, pages 305–316. IEEE, 2010.

[208] The MITRE Corporation. Fallback channel ttp. https://attack.mitre.org/techniques/T1008/. Accessed: 2021-12-18.

[209] The MITRE Corporation. Protocol tunneling. https://attack.mitre.org/techniques/T1572/. Accessed: 2021-12-18.

[210] Wouter Jansen. Abusing cloud services to fly under the radar. https://research.nccgroup.com/2021/01/12/abusing-cloud-services-to-fly-under-the-radar/. Accessed: 2021-12-18.

[211] FireEye. Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor. https://www.mandiant.com/resources/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor. 2020-12-13.

[212] The MITRE Corporation. Application layer protocol: Web protocols. https://attack.mitre.org/techniques/T1071/001/. Accessed: 2021-12-18.

[213] The MITRE Corporation. Non-application layer protocol. https://attack.mitre.org/techniques/T1095/. Accessed: 2021-12-18.

[214] The MITRE Corporation. Encrypted channel. https://attack.mitre.org/techniques/T1573/. Accessed: 2021-12-18.

[215] The MITRE Corporation. Dynamic resolution: Fast flux dns. https://attack.mitre.org/techniques/T1568/001/. Accessed: 2021-12-18.

[216] The MITRE Corporation. Data obfuscation: Protocol impersonation. https://attack.mitre.org/techniques/T1001/003/. Accessed: 2021-12-18.

[217] Benoit Claise, Ganesh Sadasivan, Vamsi Valluri, and Martin Djernaes. Cisco systems netflow services export version 9. RFC 3954, October, 2004.

[218] Stefan Burschka and Benoît Dupasquier. Tranalyzer: Versatile high performance network traffic analyser. In *IEEE symposium series on computational intelligence (SSCI 16)*, pages 1–8. IEEE, 2016.

[219] Blake Anderson and David McGrew. Identifying encrypted malware traffic with contextual flow data. In *Proceedings of the 2016 ACM workshop on artificial intelligence and security (AISec 16)*, pages 35–46, 2016.

[220] Florian Weimer. Passive dns replication. In *FIRST conference on computer security incident*, volume 98, 2005.

[221] Microsoft. Forefront tmg web proxy. https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ff827434(v=vs.85). Accessed: 2021-05-17.

[222] Squid. Squid: Optimising web delivery. http://www.squid-cache.org/. Accessed: 2021-08-20.

[223] The MITRE Corporation. Dynamic dns. https://attack.mitre.org/techniques/T1568/. Accessed: 2021-12-18.

[224] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254, 2009.

[225] Karel Bartos, Michal Sofka, and Vojtech Franc. Optimized invariant representation of network traffic for detecting unseen malware variants. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 807–822, 2016.

[226] Nizar Kheir. Behavioral classification and detection of malware through http user agent anomalies. *Journal of Information Security and Applications*, 18(1):2–13, 2013.

[227] Shanshan Wang, Zhenxiang Chen, Lei Zhang, Qiben Yan, Bo Yang, Lizhi Peng, and Zhongtian Jia. Trafficav: An effective and explainable detection of mobile malware behavior using network traffic. In *IEEE/ACM 24th International Symposium on Quality of Service (IWQoS 16)*, pages 1–6. IEEE, 2016.

[228] Martin Grill and Martin Rehák. Malware detection using http user-agent discrepancy identification. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 221–226. IEEE, 2014.

[229] Hans Christian Rudolph and Nils Grundmann. Cipher suite info. https://ciphersuite.info/cs/. Accessed: 2021-06-03.

[230] Basil AsSadhan, Hyong Kim, José MF Moura, and Xiaohui Wang. Network traffic behavior analysis by decomposition into control and data planes. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS 08)*, pages 1–8. IEEE, 2008.

[231] Basil AsSadhan, José MF Moura, David Lapsley, Christine Jones, and W Timothy Strayer. Detecting botnets using command and control traffic. In *IEEE 8th International Symposium on Network Computing and Applications (NCA 09)*, pages 156–162. IEEE, 2009.

[232] 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 *EC2N*, pages 9–16. IEEE, 2011.

[233] Florian Tegeler, Xiaoming Fu, Giovanni Vigna, and Christopher Kruegel. Botfinder: Finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 349–360, 2012.

[234] Luca Invernizzi, Stanislav Miskovic, Ruben Torres, Christopher Kruegel, Sabyasachi Saha, Giovanni Vigna, Sung-Ju Lee, and Marco Mellia. Nazca: Detecting malware distribution in large-scale networks. In *The Network and Distributed System Security Symposium (NDSS 14)*, volume 14, pages 23–26, 2014.

[235] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[236] Gints Engelen, Vera Rimmer, and Wouter Joosen. Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *IEEE Security and Privacy Workshops (SPW 21)*, pages 7–12. IEEE, 2021.

[237] Junnan Wang, Liu Qixu, Wu Di, Ying Dong, and Xiang Cui. Crafting adversarial example to bypass flow-&ml-based botnet detector via rl. In *RAID 20*, pages 193–204, 2021.

[238] Terry Nelms, Roberto Perdisci, and Mustaque Ahamad. Execscent: Mining for new c&c domains in live networks with adaptive control protocol templates. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 589–604, 2013.

[239] Riccardo Bortolameotti, Thijs van Ede, Marco Caselli, Maarten H Everts, Pieter Hartel, Rick Hofstede, Willem Jonker, and Andreas Peter. Decanter: Detection of anomalous

outbound http traffic by passive application fingerprinting. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC 17)*, pages 373–386, 2017.

[240] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. In *The Network and Distributed System Security Symposium (NDSS 18)*, 2018.

[241] Gordon Lyon. Converting information into knowledge-based assets. `https://nmap.org/book/toc.html`. March 2023.

[242] Matthieu Faou Thomas Dupuy. Gelsemium, Jun 2021.

[243] Alberto Leon-Garcia. *Probability, Statistics, and Random Processes for Electrical Engineering*. Pearson Education, 2008.

[244] Charles Cook. *Radar signals: An introduction to theory and application*. Elsevier, 2012.

[245] Nicoletta Nicolaou and Julius Georgiou. Detection of epileptic electroencephalogram based on permutation entropy and support vector machines. *Expert Systems with Applications*, 39(1):202–209, 2012.

[246] Ling Guo, Daniel Rivero, Julián Dorado, Juan R Rabunal, and Alejandro Pazos. Automatic epileptic seizure detection in eegs based on line length feature and artificial neural networks. *Journal of neuroscience methods*, 191(1):101–109, 2010.

[247] Ling Guo, Daniel Rivero, and Alejandro Pazos. Epileptic seizure detection using multiwavelet transform based approximate entropy and artificial neural networks. *Journal of neuroscience methods*, 193(1):156–163, 2010.

[248] Elif Derya Übeyli. Combined neural network model employing wavelet coefficients for eeg signals classification. *Digital Signal Processing*, 19(2):297–308, 2009.

[249] Guofei Gu, Junjie Zhang, and Wenke Lee. Botsniffer: Detecting botnet command and control channels in network traffic.

[250] Qingsong Wen, Kai He, Liang Sun, Yingying Zhang, Min Ke, and Huan Xu. Robustperiod: Robust time-frequency mining for multiple periodicity detection. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD 21)*, pages 2328–2337, 2021.

[251] Zhuochen Fan, Yinda Zhang, Tong Yang, Mingyi Yan, Gang Wen, Yuhan Wu, Hongze Li, and Bin Cui. Periodicsketch: Finding periodic items in data streams. In *IEEE 38th International Conference on Data Engineering (ICDE 22)*, pages 96–109. IEEE, 2022.

[252] Xiaomin Song, Qingsong Wen, Yan Li, and Liang Sun. Robust time series dissimilarity measure for outlier detection and periodicity detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM 22)*, pages 4510–4514, 2022.

[253] Joseph Clements, Yuzhe Yang, Ankur A Sharma, Hongxin Hu, and Yingjie Lao. Rallying adversarial techniques against deep learning for network security. In *IEEE Symposium Series on Computational Intelligence (SSCI 21)*, pages 01–08. IEEE, 2021.

[254] Mohammad J Hashemi, Greg Cusack, and Eric Keller. Towards evaluation of nidss in adversarial setting. In *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, pages 14–21, 2019.

[255] Mohammad J Hashemi and Eric Keller. Enhancing robustness against adversarial examples in network intrusion detection systems. In *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN 20)*, pages 37–43. IEEE, 2020.

[256] Elizabeth Stinson and John C Mitchell. Towards systematic evaluation of the evadability of bot/botnet detection methods. *WOOT*, 8:1–9, 2008.

[257] Ivan Homoliak, Martin Teknøs, Martîn Ochoa, Dominik Breitenbacher, Saeid Hosseini, and Petr Hanacek. Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach. eai endorsed transactions on security and safety 5, 17 (12 2018), 2018.

[258] Dongqi Han, Zhiliang Wang, Ying Zhong, Wenqi Chen, Jiahai Yang, Shuqiang Lu, Xingang Shi, and Xia Yin. Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors. *IEEE Journal on Selected Areas in Communications*, 2021.

[259] Zheng Wang. Deep learning-based intrusion detection with adversaries. *IEEE Access*, 6:38367–38384, 2018.

[260] Joseph Clements, Yuzhe Yang, Ankur Sharma, Hongxin Hu, and Yingjie Lao. Rallying adversarial techniques against deep learning for network security. *arXiv preprint arXiv:1903.11688*, 2019.

[261] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

[262] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P 17)*, pages 39–57. IEEE, 2017.

[263] Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Bowei Xi. Adversarial support vector machine learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 12)*, pages 1059–1067. ACM, 2012.

[264] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (S&P 16)*, pages 582–597. IEEE, 2016.

[265] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence (AISec 11)*, pages 43–58. ACM, 2011.

[266] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. Adversarial perturbations against deep neural networks for malware classification. *CoRR*, abs/1606.04435, 2016.

[267] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[268] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[269] Eric Wong and J Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

[270] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[271] Yotam Intrator, Gilad Katz, and Asaf Shabtai. Mdgan: Boosting anomaly detection using multi-discriminator generative adversarial networks. *arXiv preprint arXiv:1810.05221*, 2018.

[272] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. Generative adversarial trainer: Defense to adversarial perturbations with gan. *CoRR*, abs/1705.03387, 2017.

[273] Maria Rigaki and Sebastian Garcia. Bringing a gan to a knife-fight: adapting malware communication to avoid detection. In *IEEE Security and Privacy Workshops (SPW 18)*, pages 70–75. IEEE, 2018.

[274] Jun Zeng, Zheng Leong Chua, Yinfang Chen, Kaihang Ji, Zhenkai Liang, and Jian Mao. Watson: Abstracting behaviors from audit logs via aggregation of contextual semantics. In *The Network and Distributed System Security Symposium (NDSS 21)*, 2021.

[275] Hailun Ding, Juan Zhai, Yuhong Nan, and Shiqing Ma. AIRTAG: Towards automated attack investigation by unsupervised learning with log texts. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 373–390, 2023.

[276] Thijs Van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. Deepcase: Semi-supervised contextual analysis of security events. In *IEEE Symposium on Security and Privacy (S&P 22)*, pages 522–539. IEEE, 2022.

[277] Benjamin Bowman, Craig Laprade, Yuede Ji, and H Howie Huang. Detecting lateral movement in enterprise computer networks with unsupervised graph AI. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 20)*, pages 257–268, 2020.

Appendix A

Vita

Almuthanna A. Alageel is a Ph.D. candidate at Imperial College London (2018 - 2023) in cybersecurity under the supervision of Dr. Sergio Maffeis. He received his MSc in computer science from the University of Colorado, Denver, USA (2012 - 2014). Before that, he obtained his BSc in computer engineering from King Saud University, Riyadh, KSA (2004 - 2009).

He joined KACST, Computer Research Institute (CRI)-Security Group, in 2009, then affiliated with The National Center for Artificial Intelligence (NCAI) in 2015 and The National Center for Cybersecurity (C4C) in 2016. He holds many professional certifications, including CISSP®, CISM®, CRISC®, CISA®, PMP®, CEH®, TOGAF® 9 Foundation and Certified Levels, COBIT® 5 Foundation, and ITIL® 4 Foundation.

PC Membership and Reviewer:

- Safe and Trustworthy AI (STAI)@ICLP, 2023 (PC Member)
- International Journal of Information Security, Springer, 2022, 2023 (Reviewer)
- Scientific Reports, Nature, 2023 (Reviewer)