

Technische Hochschule Deggendorf
Fakultät Angewandte Informatik
Studiengang Master Angewandte Informatik

ERKENNUNG VON BÖSARTIGEM
NETZWERKVERHALTEN IN
ALLIANZ-UNTERNEHMENSNETZWERKDATEN

DETECTION OF MALICIOUS NETWORK BEHAVIOR
IN ALLIANZ COMPANY NETWORK DATA

Masterarbeit zur Erlangung des akademischen Grades:

Master of Engineering (M.Eng.)

an der Technischen Hochschule Deggendorf

Vorgelegt von:
Aida Nikkhah Nasab
Matrikelnummer: 22208964

Am: March 2025

Prüfungsleitung:
Prof. Dr. Fischer

Ergänzende Prüfende:
Zineddine Bettouche

Erklärung

Name des Studierenden: Aida Nikkhah Nasab

Name des Betreuenden: Prof. Dr. Fischer

Thema der Abschlussarbeit:

Erkennung von böartigem Netzwerkverhalten in Allianz-Unternehmensnetzwerkdaten...

.....
.....
.....

1. Ich erkläre hiermit, dass ich die Abschlussarbeit gemäß § 35 Abs. 7 RaPO (Rahmenprüfungsordnung für die Fachhochschulen in Bayern, BayRS 2210-4-1-4-1-WFK) selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Deggendorf,
Datum

.....
Unterschrift des Studierenden

2. Ich bin damit einverstanden, dass die von mir angefertigte Abschlussarbeit über die Bibliothek der Hochschule einer breiteren Öffentlichkeit zugänglich gemacht wird:

- ☐ Nein
☐ Ja, nach Abschluss des Prüfungsverfahrens
☐ Ja, nach Ablauf einer Sperrfrist von ...Jahren.

Deggendorf,
Datum

.....
Unterschrift des Studierenden

Bei Einverständnis des Verfassenden vom Betreuenden auszufüllen:

Eine Aufnahme eines Exemplars der Abschlussarbeit in den Bestand der Bibliothek und die Ausleihe des Exemplars wird:

- ☐ Befürwortet
☐ Nicht befürwortet

Deggendorf,
Datum

.....
Unterschrift des Betreuenden

Abstract

This thesis offers a comprehensive examination of the BAYWATCH framework, an advanced system designed for monitoring, detecting, and analyzing data patterns, applied to both real-world and synthetic datasets. The research delves into the theoretical underpinnings of BAYWATCH, outlining its algorithmic architecture, essential components, and the innovative methods it utilizes for real-time anomaly detection and data pattern recognition. Through a systematic evaluation, the study assesses the framework's performance in controlled experimental settings and its effectiveness in complex, real-world scenarios. The findings indicate that BAYWATCH not only exhibits robust performance and adaptability across diverse data environments but also reveals significant sensitivity to parameter configurations and noise factors present in live datasets. Furthermore, a comparative analysis with existing methodologies highlights BAYWATCH's strengths and identifies areas for optimization. The insights gained from this research contribute to a deeper understanding of data monitoring systems and offer practical recommendations for future improvements, thereby advancing the application of intelligent data analysis techniques in both academic research and industry practice.

Contents

Abstract	v
1. Topical Overview	1
1.1. Problem Statement	1
1.2. Methodology Overview	1
1.2.1. Data Extraction and Prepration	1
1.2.2. Whitelist Creation and Filtering	2
1.2.3. Time Interval Analysis	2
1.2.4. Bandpass Filtering	2
1.2.5. Power Calculation and Normalization	2
1.2.6. Behavior Detection and Threshold Analysis	2
1.2.7. Validation and Continuous Improvement	2
1.3. Research Objectives	3
1.3.1. Research Questions	3
1.4. Structure of Thesis	3
2. Background	5
2.1. Cybersecurity Landscape	5
2.1.1. Emerging Trends and Challenges	5
2.2. Advanced Persistent Threats (APTs) and Covert Tactics	6
2.2.1. Case Studies of APT Attacks	8
2.3. Enterprise Networks	8
2.3.1. Key Aspects of Enterprise Networks	9
2.3.2. Vulnerabilities in Enterprise Networks	9
2.4. Band-Pass Filtering	9
2.5. Periodicity in Network Communication	11
2.5.1. Importance in Cybersecurity	11
2.6. Time Series Databases	11
2.6.1. Characteristics of Time Series Databases	11
2.6.2. InfluxDB	12
2.7. Summary	12
3. Related Work	15
4. Methodology	19
4.1. Overview of the BAYWATCH Framework	19
4.2. Whitelist Analysis	20
4.2.1. Universal Whitelisting	20

Contents

4.2.2.	Local Whitelisting	21
4.3.	Time Series Analysis	21
4.3.1.	Algorithm Overview	21
4.3.2.	Candidate Discovery Using FFT	21
4.3.3.	Pruning Using Bandpass Filtering	22
4.3.4.	Verification Using Autocorrelation	22
4.3.5.	Handling Multiple Periodicities	23
4.4.	Suspicious Indicator Analysis	23
4.5.	Investigation and Verification	23
4.5.1.	Feature Set	23
4.5.2.	Classifier	24
4.5.3.	Bootstrapping Process	24
4.6.	Real Data Source	24
4.6.1.	Data Structure and Schema	24
4.6.2.	Data Collection and Scale	25
4.6.3.	Data Management and Preprocessing	26
4.6.4.	Challenges with Real-World Data	26
4.7.	Artificial Data Source	26
4.7.1.	Design of Artificial Data	27
4.7.2.	Jitter Ranges	27
4.7.3.	Scenarios Tested	28
4.7.4.	Integration with Real-World Data	28
4.8.	Summary	28
5.	Data Analysis	29
5.1.	Visualization of URL Request Counts	29
5.2.	24-Hour URL Visit Analysis	31
5.3.	Time Interval Analysis of URL Requests	33
5.4.	Distribution of Hosts Based on Unique URLs Contacted	35
6.	Implementation	37
6.1.	Experimental Setup	37
6.2.	Whitelisting Mechanism for URL Filtering	38
6.3.	Average Power Calculation	39
6.4.	Band-Pass Filtering	40
6.5.	Beaconing Data Generation	41
6.6.	Fast Fourier Transform (FFT)	42
6.7.	Autocorrelation	42
6.8.	Behavior Detection	43
6.9.	Algorithm Output	43
6.10.	Summary	44
6.11.	Next Steps	45

7. Experiments	47
7.1. Validation and Testing	47
8. Results and Discussions	53
8.1. Detection of Beaconsing Behavior	53
8.1.1. Algorithm Development and Implementation	53
8.1.2. Data Collection and Preprocessing	53
8.1.3. Validation and Testing	54
8.2. Impact of Periodicity in Network Communication	54
8.2.1. Identification of Regular Intervals	54
8.2.2. Differentiation Between Benign and Malicious Periodicity	54
8.2.3. Impact on False Positives and Negatives	54
8.2.4. Case Studies and Empirical Evidence	55
9. Conclusion and Future Work	57
9.1. Conclusion	57
9.2. Future Work	57
A. Appendix	59
A.1. Algorithm Implementation	59
A.2. Data Analysis Implementation	62

1. Topical Overview

1.1. Problem Statement

In the modern world it is important to protect the information and to ensure the security of the network systems as much as it is important for organizations. Like so many others, companies create a large number of user log data every day. It is closely watched for any signs of potential security threats and it is rich with potential insights. However, as cyber attacks become more sophisticated and complex especially APTs, there is the need to have strong and preventive cybersecurity measures in place. The major challenge is how to effectively sort through this huge amount of log data to determine the malicious events and respond to them before they lead to any damage. This research is aimed at enhancing the cybersecurity postures of an organization with a view to detecting APT and other threats early enough in order to protect the network infrastructure.

1.2. Methodology Overview

The methodology followed for this research systematically detects and analyzes the beaconing behavior of network traffic. Several steps are involved in handling this, starting with data collection where both real and artificial datasets have been gathered to ensure that the analysis is substantial.

Advanced data preprocessing on the collected datasets has been carried out in cleaning and preparing the datasets for analysis. It's an important step in which noise and irrelevant information that might distort results can be removed. First, the pre-processing step will ensure that the format of data is appropriate to the analysis step. It would enable better detectability of the beaconing pattern. Thus, the methodology refines the data in order to give more precision to the algorithm in detecting the subtle signs of beaconing behavior that would have been impossible if the approach is relaxed.

1.2.1. Data Extraction and Prepration

These involve gathering large-scale amounts of data across network traffic; it shall further include the variation of attribute varieties such as time stamps, source and destination IP addresses, and URLs visited. The raw data then has an extensive preprocessing step, filtering out irrelevant information, normalizing the data format, and filling in missing values; thus, after the cleaning is performed, structured data will come forth for advanced analytics. The various analytical techniques used to become more familiar with the data will explain how these URLs would behave on a certain day. The different analyses possible on the data can reveal the pat-

1. Topical Overview

tern, anomalies, and trends hidden in this dataset, and that gives interesting insights into the network activity or user behavior.

1.2.2. Whitelist Creation and Filtering

To enhance the efficiency of the analysis, a whitelist of trusted URLs is created. This whitelist is based on known safe domains and frequently accessed URLs within the organization. By filtering out these trusted URLs, the methodology focuses on potentially suspicious activities, reducing the noise and improving the accuracy of the detection process.

1.2.3. Time Interval Analysis

A critical aspect of the methodology is the analysis of time intervals between successive network requests. By calculating the time differences between consecutive requests to the same URL, the methodology identifies patterns that may indicate beaconing behavior. This analysis helps in distinguishing regular, benign activities from irregular, potentially malicious ones.

1.2.4. Bandpass Filtering

To further refine the analysis, bandpass filtering is applied to the time interval data. This technique isolates specific frequency components within a defined range, effectively filtering out noise and irrelevant fluctuations. The filtered data highlights significant patterns and periodicities, which are crucial for detecting beaconing behavior.

1.2.5. Power Calculation and Normalization

The concept of "power" is introduced to quantify the frequency of network requests. By calculating the power for each URL and normalizing it against the average power, the methodology identifies URLs with unusually high or low request frequencies. This step helps in pinpointing URLs that exhibit abnormal behavior, warranting further investigation.

1.2.6. Behavior Detection and Threshold Analysis

The final stage involves the detection of suspicious behavior based on predefined thresholds. URLs with power values exceeding the threshold are flagged for further scrutiny. This step ensures that only the most significant anomalies are investigated, optimizing the use of resources and enhancing the overall effectiveness of the detection process.

1.2.7. Validation and Continuous Improvement

The methodology is validated using both synthetic and real-world datasets, ensuring its robustness and reliability. Continuous feedback and refinement are integral to the process, allowing the methodology to adapt to evolving network traffic patterns and emerging threats. This iterative approach ensures that the detection system remains effective and up-to-date.

In summary, the methodology combines advanced data analysis techniques with practical filtering and detection strategies to identify beaconing behavior within network traffic. By systematically addressing each stage of the process, the methodology provides a comprehensive framework for enhancing network security and mitigating potential threats.

1.3. Research Objectives

The main goal of this research is to improve the network security of Allianz Company by finding and using better ways to spot and respond to possible cyber threats. The focus is mainly on advanced persistent threats (APTs), which are known for being sneaky and long-lasting. To do this, the research aims to:

1. **Develop Advanced Detection Techniques:** Create methods to identify potential threats early, focusing on the unique behaviors of APTs.
2. **Implement Proactive Security Measures:** Establish protocols that enable quicker responses to detected threats, reducing the risk of data breaches.
3. **Educate and Train Staff:** Provide training for employees to recognize and respond to potential security threats effectively.
4. **Evaluate and Update Security Policies:** Continuously assess and refine the company's security policies to adapt to new and evolving cyber threats.

1.3.1. Research Questions

The research is guided by the following key questions:

- How can beaconing behavior be effectively detected within Allianz Company's network?
- What is the impact of periodicity in network communication on the detection of malicious behavior?

1.4. Structure of Thesis

This section outlines the organization of the chapters in this thesis. The structure is designed to systematically address the research objectives and questions outlined above, ensuring a comprehensive understanding of the problem and the proposed solutions.

Chapter 2 provides the background necessary for understanding the context of this research. It begins with an introduction that sets the stage for the subsequent discussions. This chapter delves into the cybersecurity landscape, highlighting the current state of cybersecurity and the challenges enterprises face. It then explores advanced persistent threats (APTs) and their covert tactics, providing a detailed examination of how these threats operate and the sophisticated methods they employ. Additionally, this chapter discusses enterprise networks, focusing on their structure, functionality, and the inherent vulnerabilities that make them targets for

1. Topical Overview

cyberattacks. Finally, it introduces the concept of periodicity in network communication, explaining its relevance to detecting malicious activities.

Chapter 3 is dedicated to a review of related work. It begins with an overview of the BAY-WATCH framework, then an exploration of various methods for APT beaconing detection. The chapter then discusses peer-based tracking techniques and presents a systematic review of the literature on APT beaconing detection. Further, it examines the use of DNS logs for malware beaconing detection and the application of AI-driven approaches to identify malicious beaconing. The chapter concludes with a discussion on local periodic communication behavior, providing a comprehensive overview of existing research and highlighting gaps that this thesis aims to address.

Chapter 4 focuses on the methodology adopted for this research. It begins with the design of the proposed method, outlining the theoretical foundation and the rationale behind the chosen approach. This is followed by a detailed description of data extraction and preparation processes, ensuring the data used is both relevant and reliable. The chapter also covers data preprocessing techniques, time interval analysis, and data enhancement methods. Band-pass filtering is introduced as an important step in the methodology, followed by a discussion on the evaluation criteria used to assess the effectiveness of the proposed solution.

Chapter 5 details the implementation of the proposed method. It starts with an explanation of the experimental setup, describing the environment and tools used for the experiments. This chapter also introduces a whitelisting mechanism for URL filtering, aimed at reducing false positives. It then describes the process of average power calculation and the application of band-pass filtering to the data. The chapter concludes with a discussion on behavior detection, explaining how the proposed method identifies malicious activities.

Chapter 6 presents the experiments conducted to validate the proposed method. It includes sections on validation and testing, detailing the procedures and metrics used to assess the performance of the method. An in-depth analysis of the algorithm's output is provided, focusing on the detection of malicious behavior. This chapter aims to demonstrate the efficacy of the proposed method through empirical evidence.

Chapter 7 covers the results and discussions, summarizing the key findings of the research. It provides a critical analysis of the results, discussing their implications and relevance to the field of cybersecurity. This chapter also highlights the contributions of the research, emphasizing how it advances the current state of knowledge.

Chapter 8 concludes the thesis by summarizing the main findings and providing insights into future work. It outlines potential avenues for further research, suggesting how the proposed method can be refined and extended. This chapter aims to provide a comprehensive conclusion to the thesis, tying together the various elements and emphasizing the significance of the research. In summary, the structure of this thesis is designed to provide a logical and coherent progression from background information and related work to methodology, implementation, experiments, and results, culminating in a comprehensive conclusion and suggestions for future research.

2. Background

This chapter provides the background necessary for understanding the context of this research. It begins with an overview of the cybersecurity landscape, emphasizing the current state, emerging trends, and persistent challenges faced by organizations. It then explores Advanced Persistent Threats (APTs) and their sophisticated, covert tactics that pose significant risks to enterprise networks. The discussion also covers the concept of periodicity in network communication, crucial for detecting anomalies in cybersecurity contexts. Finally, the chapter delves into the role of time series databases, with a specific focus on InfluxDB, in managing and analyzing the vast amounts of data generated in cybersecurity operations.

The field of cybersecurity is continually evolving, with new threats emerging as technology advances. Understanding these threats and the strategies to counter them is crucial for protecting sensitive information, ensuring the continuity of operations, and maintaining the integrity of enterprise networks. This chapter lays the foundation for the research by discussing key concepts and technologies relevant to cybersecurity, setting the stage for the detailed analysis and solutions proposed in subsequent chapters.

2.1. Cybersecurity Landscape

The cybersecurity landscape is characterized by a dynamic and increasingly complex environment where various types of cyber threats continually evolve. Organizations across the globe face numerous challenges in protecting their networks, data, and systems from these threats, which range from malware and ransomware to sophisticated nation-state attacks.

Cybersecurity encompasses a wide range of practices, technologies, and strategies aimed at safeguarding information and systems from unauthorized access, damage, or disruption. It involves both proactive measures, such as implementing robust security architectures and practices, and reactive measures, such as incident response and recovery strategies.

Figure 2.1 presents a global map of cybersecurity threats, illustrating the widespread nature of these challenges. This visualization highlights regions most affected by various types of cyber attacks, underscoring the global reach and impact of cyber threats.

2.1.1. Emerging Trends and Challenges

The rapid digitization of industries, the increasing reliance on cloud services, and the proliferation of Internet of Things (IoT) devices have significantly expanded the attack surface for cyber threats. These developments, while beneficial, have introduced new vulnerabilities that attackers are quick to exploit. Additionally, the rise of ransomware as a service (RaaS) and the growing sophistication of phishing attacks reflect the evolving threat landscape.

2. Background

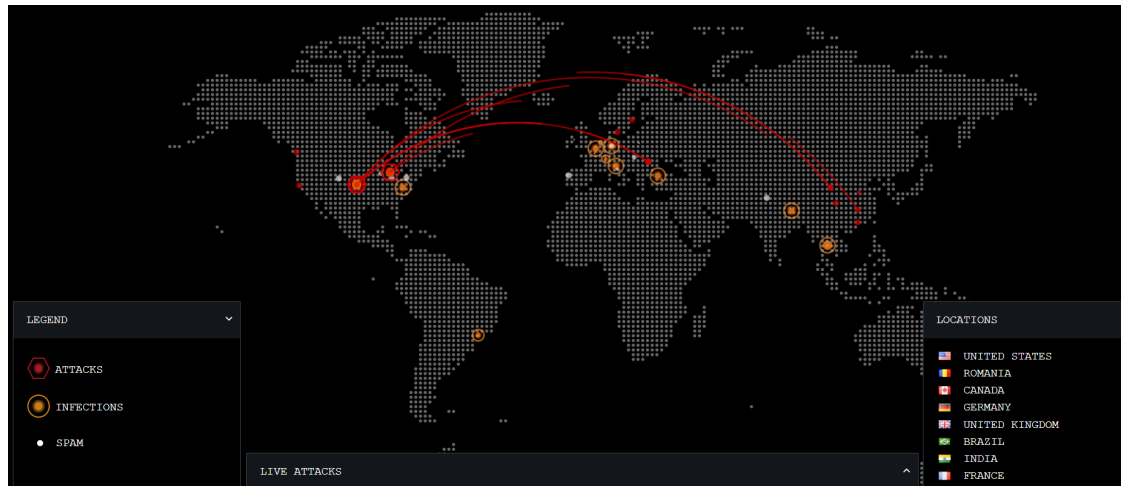


Figure 2.1.: Global cybersecurity threat map [1]

Another significant challenge is the shortage of skilled cybersecurity professionals, which hampers the ability of organizations to effectively defend against these threats. This gap is exacerbated by the complexity of modern networks and the need for advanced tools and techniques to detect and mitigate sophisticated attacks.

2.2. Advanced Persistent Threats (APTs) and Covert Tactics

Advanced Persistent Threats (APTs) represent one of the most sophisticated and dangerous forms of cyber attacks. APTs involve prolonged, targeted efforts by attackers, typically state-sponsored or highly organized criminal groups, aimed at stealing sensitive information, disrupting operations, or compromising critical infrastructure. Unlike traditional cyber attacks, which may be opportunistic and short-lived, APTs are characterized by their stealth, persistence, and the significant resources devoted to them.

Figure 2.2 illustrates the lifecycle of an APT attack, highlighting the various stages involved, from initial reconnaissance to exfiltration of data. Understanding these stages is crucial for developing effective detection and mitigation strategies.

APT actors employ various covert tactics to remain undetected and achieve their objectives. Some of these tactics include:

- **Spear Phishing:** Crafting highly personalized email messages that appear legitimate to the recipient. These emails are designed to trick recipients into clicking on malicious links or attachments, leading to the compromise of their credentials or systems.
- **Zero-Day Exploits:** Exploiting previously unknown vulnerabilities in software or hardware, which have not yet been patched by the vendor. This allows attackers to gain unauthorized access to systems without triggering existing security defenses.

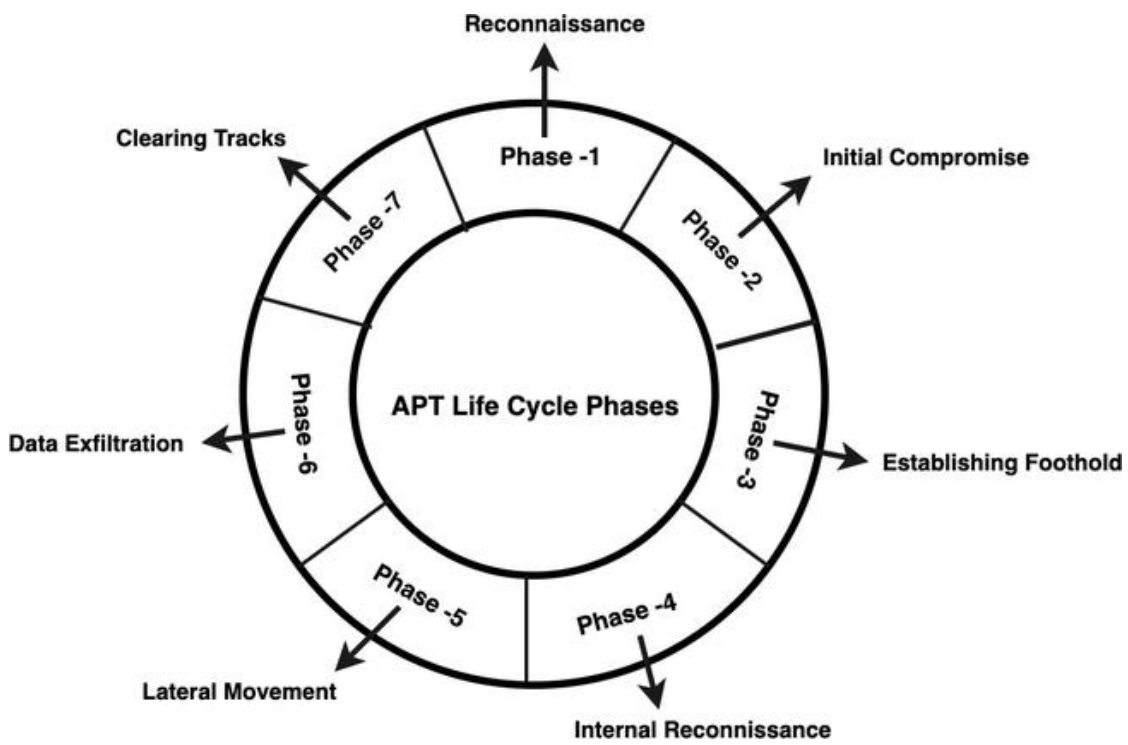


Figure 2.2.: APT attack lifecycle [?]

2. Background

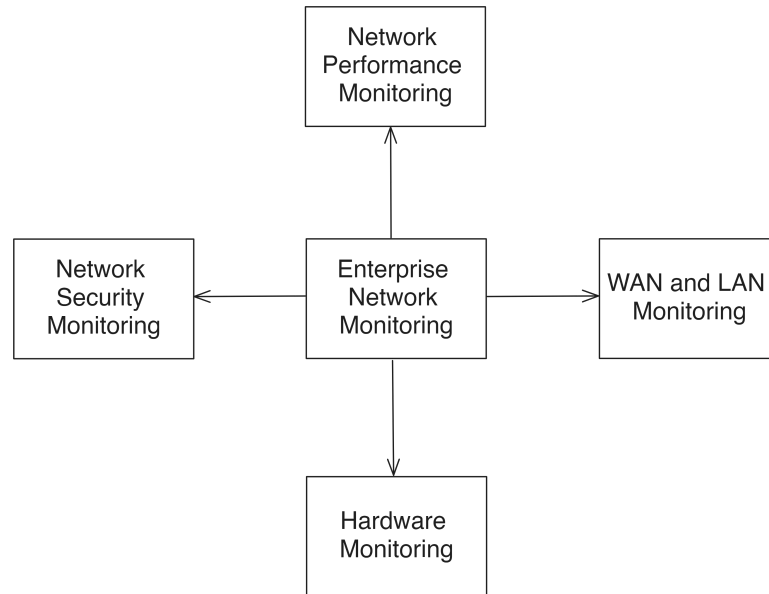


Figure 2.3.: Enterprise network diagram

- **Lateral Movement:** After gaining initial access, attackers move within the compromised network, exploring and compromising additional systems to find and exfiltrate valuable data. This tactic often involves the use of legitimate administrative tools to avoid detection.
- **Command and Control (C2):** Establishing a secure communication channel with the compromised systems to remotely control them, issue commands, and exfiltrate data.

2.2.1. Case Studies of APT Attacks

Prominent examples of APT attacks include the Stuxnet worm, which targeted Iran's nuclear program, and the SolarWinds breach, which compromised numerous U.S. government agencies and corporations. These cases underscore the potential impact of APTs on national security and global business operations.

2.3. Enterprise Networks

Enterprise networks are the backbone of modern organizations, providing the necessary infrastructure for communication, data sharing, and operational efficiency. However, their complexity and scale make them attractive targets for cyber attackers. Understanding the architecture, components, and vulnerabilities of enterprise networks is for developing effective cybersecurity strategies.

Figure 2.3 provides a visual representation of an enterprise network, illustrating the various components such as servers, workstations, routers, and communication links, as well as

potential points of vulnerability.

2.3.1. Key Aspects of Enterprise Networks

Enterprise networks typically consist of multiple interconnected subsystems, including:

- **Network Architecture:** The physical and logical design of the network, including the layout and interconnection of routers, switches, firewalls, and other network devices. A well-designed architecture enhances security by segmenting the network and controlling traffic flow.
- **Security Protocols:** Protocols such as TLS (Transport Layer Security) and IPSec (Internet Protocol Security) protect data in transit. Additionally, firewalls, intrusion detection/prevention systems (IDS/IPS), and encryption mechanisms are employed to safeguard data and systems.
- **Access Controls:** Policies and technologies that regulate who can access specific data and resources within the network. This includes user authentication, role-based access control (RBAC), and multi-factor authentication (MFA) to ensure that only authorized personnel can access sensitive information.
- **Network Monitoring and Management:** Tools and practices for monitoring network traffic, identifying anomalies, and managing network resources to maintain performance and security.

2.3.2. Vulnerabilities in Enterprise Networks

Despite the implementation of robust security measures, enterprise networks remain vulnerable to a variety of threats, including:

- **Insider Threats:** Employees or contractors with legitimate access who misuse their privileges, either maliciously or negligently.
- **Advanced Malware:** Malware is designed to bypass traditional security measures, often delivered through phishing attacks or drive-by downloads.
- **Misconfigurations:** Incorrectly configured devices or systems that leave the network open to exploitation.
- **Supply Chain Attacks:** Attacks that target the software or hardware supply chain, introducing vulnerabilities that can be exploited after deployment.

2.4. Band-Pass Filtering

In network processing, bandpass filtering is a technique employed to dissect time-series data, allowing the extraction of specific frequency components within a predefined range. This technique is particularly useful in analyzing patterns in HTTP requests. Bandpass filtering involves

2. Background

the application of a filter that selectively passes signals whose frequencies fall within a certain range, known as the "bandpass" range. By isolating these specific frequencies, the technique enables a focused examination of data that is most relevant to the analysis, effectively filtering out noise and irrelevant information. This selective process enhances the clarity and precision of the data, making it easier to identify significant patterns and trends in HTTP requests. For instance, in a dataset containing web traffic data, bandpass filtering can help highlight the intervals and frequencies at which certain URLs are accessed, providing insights into user behavior and potential security threats. The ability to concentrate on a specific frequency range allows analysts to zero in on the most pertinent signals, thereby improving the accuracy and effectiveness of the analysis.

Furthermore, bandpass filtering aids in detecting anomalies and irregularities within the network. By focusing on the relevant frequency components, it becomes easier to spot deviations from the norm, which could indicate unusual or suspicious activity. This method is instrumental in the context of network security, where identifying and understanding these anomalies is key for protecting against potential threats. In addition to its application in security, bandpass filtering is also valuable for optimizing network performance. By understanding the regular patterns of data flow and identifying any irregular spikes or drops, network administrators can make informed decisions to enhance the efficiency and reliability of the network. This comprehensive approach ensures that only the most significant data is analyzed, leading to more accurate and actionable insights. Overall, bandpass filtering is a powerful technique in-network processing, enabling the extraction of meaningful information from large datasets. By focusing on specific frequency components, it facilitates a detailed and precise analysis of network interactions, helping to uncover important patterns and trends. This technique not only improves the understanding of user behavior and network performance but also plays a vital role in enhancing security by detecting potential threats and anomalies.

The bandpass filter formula can be expressed as:

$$H(\omega) = \frac{1}{1 + \frac{j(\omega - \omega_{\text{low}})}{\omega_c}} \cdot \frac{1}{1 + \frac{j(\omega_{\text{high}} - \omega)}{\omega_c}}$$

Where:

- $H(\omega)$ is the frequency response of the bandpass filter,
- ω is the angular frequency,
- ω_{low} is the low cut-off frequency,
- ω_{high} is the high cut-off frequency,
- ω_c is the critical frequency.

The bandpass filter selectively passes frequencies between the low cut-off frequency (ω_{low}) and the high cut-off frequency (ω_{high}), while attenuating frequencies outside this range. This formula provides a mathematical representation of how the bandpass filter operates to isolate specific frequency components within the defined range.

2.5. Periodicity in Network Communication

Periodicity in network communication refers to the recurring patterns observed in network traffic over time. Detecting and analyzing these patterns can provide valuable insights into normal and anomalous behavior within the network. In cybersecurity, periodicity analysis is particularly useful for identifying stealthy activities, such as those conducted by APTs, which may generate periodic communication to maintain control over compromised systems.

2.5.1. Importance in Cybersecurity

Understanding periodicity is crucial for the following reasons:

- **Anomaly Detection:** Deviations from established periodic patterns can indicate the presence of malware or other malicious activities.
- **Traffic Analysis:** Analyzing periodic traffic can help in identifying command and control (C2) communications used by attackers.
- **Resource Optimization:** Periodicity analysis can be used to optimize network resources by predicting traffic loads and adjusting resources accordingly.

2.6. Time Series Databases

Time series databases are specialized databases designed to handle time-stamped or time-series data efficiently. This type of data is common in network activity logs, sensor readings, financial transactions, and many other applications where the sequence and timing of data points are critical. Time series databases are optimized for high-frequency data writes and efficient queries over time intervals, making them ideal for use in monitoring, alerting, and anomaly detection in cybersecurity contexts.

2.6.1. Characteristics of Time Series Databases

Time series databases differ from traditional relational databases in several key ways:

- **Time-Optimized Storage:** Data is stored in a way that optimizes retrieval by time, enabling fast queries across large datasets.
- **Efficient Data Compression:** Given the often high volume of data, time series databases employ advanced compression techniques to reduce storage requirements.
- **High Throughput:** They are optimized to handle high-frequency data writes and queries, ensuring efficient data handling even under heavy load.
- **Querying Capabilities:** Time series databases support complex querying over time intervals, which is for trend analysis and anomaly detection.

2. Background

2.6.2. InfluxDB

InfluxDB is a popular time series database known for its high performance and ease of use. It is optimized for handling large-scale time-series data, providing powerful querying capabilities and efficient storage.

Key Features of InfluxDB

- **Time-Optimized Storage:** InfluxDB uses a custom storage engine that efficiently writes and reads time-series data.
- **High Throughput:** It can handle high write and query loads, making it suitable for large-scale monitoring applications.
- **SQL-like Query Language (Flux):** InfluxDB offers a powerful query language that is both easy to learn and capable of complex data manipulations.
- **Retention Policies:** Users can define retention policies to manage data lifecycle, automatically deleting old data to save storage.
- **Integrations:** InfluxDB integrates well with other tools and platforms, supporting various data inputs and outputs.

Applications in Cybersecurity

InfluxDB can be employed in cybersecurity for:

- **Real-Time Monitoring:** Capturing and analyzing live data to detect anomalies and potential threats.
- **Historical Analysis:** Storing historical data for trend analysis and forensic investigations.
- **Alerting:** Setting up alerts based on specific criteria to notify administrators of suspicious activities.
- **Visualization:** Integrating with visualization tools like Grafana to create dashboards that display network metrics and security insights.

Figure 2.4 illustrates the architecture of InfluxDB and how data flows through the system, from ingestion to querying and visualization.

2.7. Summary

This chapter has provided a comprehensive overview of the cybersecurity landscape, APTs and their covert tactics, enterprise networks, periodicity in network communication, and time series databases, with a detailed focus on InfluxDB. These foundational topics are for understanding the subsequent chapters, which will delve deeper into related work, methodology,

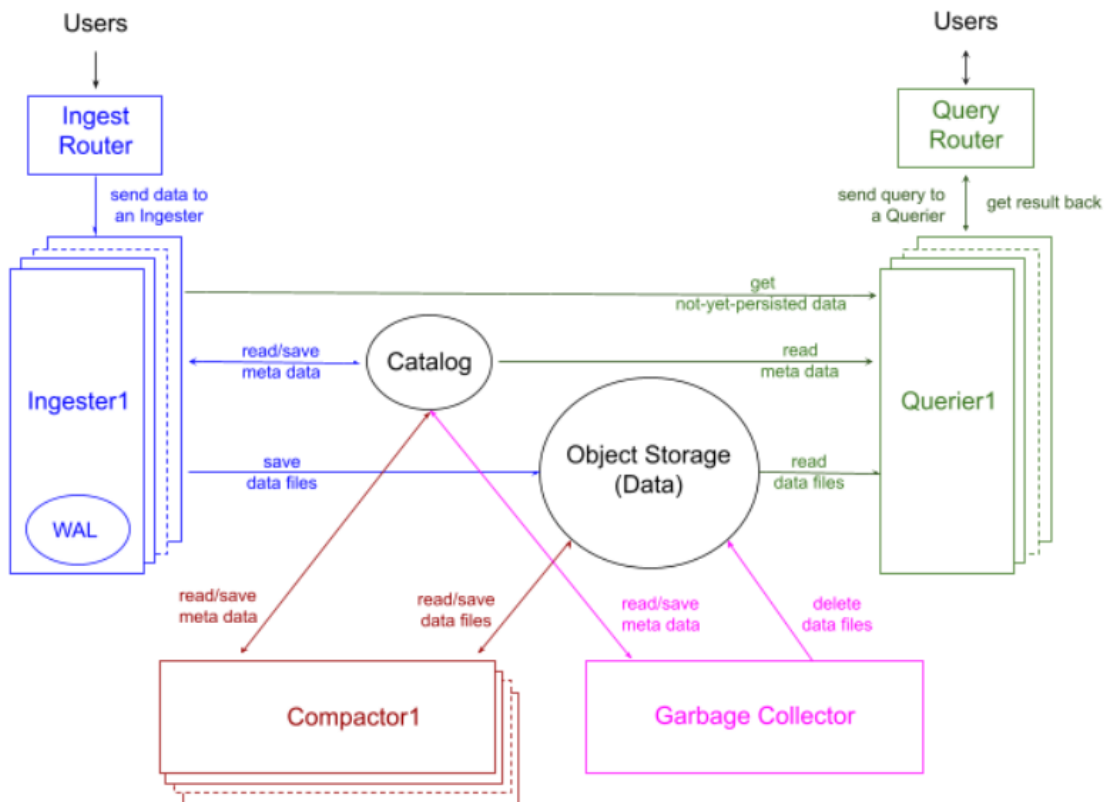


Figure 2.4.: InfluxDB Architecture [2]

2. Background

implementation, experiments, and results. The knowledge gained from this background will inform the development and evaluation of advanced techniques for detecting and mitigating cyber threats in enterprise networks.

3. Related Work

Hu et al. (2016) proposed BAYWATCH, a robust beaconing detection method designed to identify infected hosts in large-scale enterprise networks [3]. The method focuses on detecting beaconing behavior, which is commonly exhibited by compromised hosts communicating with external command and control servers. By analyzing network traffic patterns, BAYWATCH can efficiently detect infected devices while minimizing false positives. The system is specifically designed to scale in large enterprise environments, making it suitable for real-world deployment. The authors validate BAYWATCH through extensive evaluation using real-world network traffic, demonstrating its effectiveness in identifying infected hosts and improving network security.

Zhang et al. (2023) introduced a global analysis approach for aggregation-based beaconing detection across large campus networks [4]. Their method focuses on detecting beaconing behavior in network traffic by aggregating data from multiple sources within a campus network, which enhances detection accuracy. The approach is designed to scale across large networks and aims to minimize false positives by leveraging aggregation techniques. The authors validate their method through extensive experiments on real-world campus networks, demonstrating its effectiveness in identifying compromised hosts and improving network security in large-scale environments.

Apruzzese et al. (2017) proposed a method for identifying malicious hosts involved in periodic communications [5]. Their approach focuses on detecting abnormal periodic communication patterns in network traffic, which are often indicative of compromised hosts communicating with external servers. The authors introduce a novel technique for identifying such hosts by analyzing the timing and frequency of communication sessions. The proposed method is evaluated through experiments, demonstrating its effectiveness in identifying malicious hosts and enhancing network security by targeting irregular communication patterns.

Seo and Lee (2018) proposed an abnormal behavior detection method to identify infected systems using the APChain algorithm and behavioral profiling [6]. Their approach focuses on analyzing system behavior to detect deviations from normal activity, which may indicate the presence of malware or compromised systems. The APChain algorithm is used to model and track the behavior of systems, allowing for the identification of anomalous patterns associated with infected hosts. The authors validate their method through experiments, demonstrating its effectiveness in detecting abnormal behaviors and enhancing system security in real-world environments.

Huynh et al. (2016) focused on uncovering periodic network signals of cyber attacks [7]. The paper explores how periodic network traffic patterns can indicate cyber attacks, particularly in the context of detecting covert channels used by attackers for command and control. The authors propose a method to identify such periodic signals by analyzing network traffic over time. Their approach highlights the importance of periodicity in revealing malicious ac-

3. Related Work

tivity and introduces a visualization technique to facilitate the detection of these patterns. The study contributes to improving network security by enabling better detection of stealthy attack signals.

Jang et al. (2021) proposed a method for detecting malicious beaconing communities using lockstep detection and co-occurrence graphs [8]. The paper introduces an innovative approach to identifying groups of compromised hosts involved in coordinated beaconing behavior, a common indicator of malicious activity. By using lockstep detection and analyzing the co-occurrence of network events, the method can effectively pinpoint these malicious communities. The authors present this approach as part of a patent (US Patent 10,887,323), contributing to the detection of advanced persistent threats (APTs) and enhancing network security by identifying coordinated attacks.

Talib et al. (2022) conducted a systematic review on APT beaconing detection techniques [9]. The paper provides an extensive analysis of various methods used to detect Advanced Persistent Threats (APT) based on beaconing behavior, which is a common communication pattern in APT attacks. The authors review different detection techniques, including signature-based, anomaly-based, and machine learning methods, highlighting their strengths and limitations in identifying beaconing activities in network traffic. This review serves as a valuable resource for researchers and practitioners aiming to enhance APT detection and improve network security against sophisticated cyber threats.

Charan et al. (2021) explored the use of data mining and machine learning techniques for Advanced Persistent Threat (APT) attribution and detection in their study on DMAPT [10]. The paper focuses on the application of various data mining and machine learning methods to improve the identification and attribution of APTs, which are often difficult to detect due to their stealthy nature. The authors discuss the effectiveness of different approaches in detecting APTs and their potential for enhancing threat detection capabilities in network security. The study provides valuable insights into the role of advanced analytics in tackling sophisticated cyber threats.

Hagan et al. (2018) proposed a peer-based tracking method using multi-tuple indexing for network traffic analysis and malware detection [11]. The approach aims to improve malware detection by analyzing network traffic patterns using multi-tuple indexing, which allows for more efficient tracking of peer interactions in the network. By examining the flow of traffic between different peers, the method identifies suspicious activities that may indicate the presence of malware. The authors validate their technique through experiments, demonstrating its effectiveness in detecting malicious traffic and enhancing network security by providing a more granular analysis of peer behavior.

Shalaginov et al. (2016) focused on malware beaconing detection by mining large-scale DNS logs for targeted attack identification [12]. The paper explores the use of DNS logs to detect beaconing behavior, which is commonly associated with malware communicating with external command and control servers. By analyzing large-scale DNS traffic data, the authors propose a method to identify targeted attacks based on the periodic patterns of beaconing. Their approach highlights the importance of leveraging DNS traffic for identifying malware infections, contributing to enhanced detection capabilities in large-scale network environments.

Yeh et al. (2018) investigated a malware beacon of botnet by analyzing local periodic communication behavior [13]. The paper focuses on identifying malware beaconing behavior in

botnets by studying the periodic communication patterns between infected hosts and their command and control servers. The authors propose a method to detect these periodic behaviors, which are typically used by botnets to maintain control over compromised systems. Their approach highlights the importance of analyzing local traffic patterns for detecting botnet infections and contributes to improving malware detection techniques through the identification of communication anomalies.

Borchani (2020) proposed an advanced approach to malicious beaconing detection using Artificial Intelligence (AI) [14]. The paper explores the application of AI techniques, particularly machine learning algorithms, to enhance the detection of beaconing behavior associated with malicious activity. By leveraging AI, the author aims to improve the accuracy and efficiency of detecting beaconing patterns that indicate compromised hosts within a network. The study demonstrates the potential of AI to significantly improve the detection and mitigation of threats posed by beaconing malware, contributing to more effective network security solutions.

Enright et al. (2022) introduced a learning-based zero-trust architecture for 6G and future networks [15]. The paper explores the integration of machine learning with zero-trust security models to address the evolving security challenges in next-generation networks, particularly 6G. The authors propose a framework that combines learning-based techniques with zero-trust principles to enhance the detection of malicious activity and improve overall network security. The study contributes to the development of more adaptive and robust security architectures for future networks, offering a promising solution to the emerging threats in 6G environments.

Van Ede et al. (2022) introduced Deepcase, a semi-supervised contextual analysis method for security events [16]. The paper presents a novel approach that combines semi-supervised learning techniques with contextual analysis to enhance the detection of security events. By leveraging contextual information, Deepcase can identify complex patterns and relationships in security data, improving the accuracy of event classification and anomaly detection. The authors demonstrate the effectiveness of their approach in real-world security environments, showing its potential to enhance the detection and response capabilities of security systems in large-scale networks.

Ongun et al. (2021) introduced PORTFILER, a port-level network profiling approach for detecting self-propagating malware [17]. The paper presents a novel method that profiles network traffic at the port level to identify self-propagating malware, which often uses specific ports for communication and propagation. PORTFILER analyzes network behavior to detect irregularities and patterns associated with malware activity. By focusing on port-level communication, the approach improves malware detection, providing more accurate identification of self-propagating threats in real-time. The authors demonstrate the effectiveness of their method through experiments, showing its potential to enhance network security against rapidly spreading malware.

Niu et al. (2020) proposed a method for detecting malware on the Internet of Unmanned Aerial Vehicles (IoUAVs) by combining string matching and Fourier transformation techniques [18]. The paper addresses the growing concern of malware targeting UAV networks and introduces a hybrid approach that leverages string matching for identifying suspicious patterns in network traffic and Fourier transformation for analyzing periodic behaviors associated with malware. By combining these techniques, the authors enhance the detection accuracy of malicious activities, offering a more robust solution to securing UAV-based networks. The study

3. Related Work

contributes to the advancement of IoT security, particularly in the context of UAV systems, which are increasingly vulnerable to cyberattacks.

Duan et al. (2018) presented an approach for the automated generation and selection of interpretable features for enterprise security [19]. The paper focuses on improving enterprise security by developing methods for automatically generating and selecting meaningful, interpretable features from large datasets. These features can be used in security models to detect anomalies and potential threats more efficiently. The authors propose a framework that integrates automated feature engineering with machine learning techniques to enhance security monitoring systems. Their work contributes to the field by improving the interpretability and performance of security analytics, making it easier for security teams to understand and respond to potential threats.

Haffey et al. (2018) focused on modeling, analyzing, and characterizing periodic traffic on a campus edge network [20]. The paper explores the behavior of periodic traffic patterns in campus networks, which are often indicative of scheduled communications, including those used by malware. The authors propose models to better understand and quantify these traffic patterns, helping to distinguish between legitimate and potentially malicious activities. Their work provides insights into how periodic traffic can be leveraged to enhance network security, particularly in the detection of botnets and other forms of malware that use regular communication intervals.

Recent research has focused on various aspects of enterprise security and malicious activity detection. Oprea et al. (2018) introduced MADE, a security analytics framework designed to enhance threat detection in enterprise environments [21]. The framework leverages advanced analytics to detect potential threats by analyzing large volumes of security data, enabling organizations to respond more effectively to cyber incidents. Ukrop et al. (2019) investigated the perception of IT professionals regarding the trustworthiness of TLS certificates, highlighting challenges in assessing certificate legitimacy and its implications for secure communications [22]. In a related study, Vissers et al. (2017) explored the ecosystem of malicious domain registrations within the .eu top-level domain (TLD), providing insights into the strategies used by attackers to exploit domain registration systems for malicious purposes [23]. Together, these works contribute to the broader understanding of security challenges in modern networks and propose solutions to improve detection and mitigation strategies.

4. Methodology

The BAYWATCH framework is a comprehensive methodology designed to identify stealthy beaconing behavior in large-scale enterprise networks. Beaconing, a common behavior in malware-infected hosts, involves periodic communication with a command and control (C&C) infrastructure. Detecting such behavior is challenging due to the presence of legitimate periodic traffic (e.g., software updates, email polling) and the various strategies employed by malware authors to evade detection. The BAYWATCH framework addresses these challenges through an 8-step filtering approach, which iteratively refines and eliminates legitimate traffic to pinpoint malicious beaconing cases. This chapter provides a detailed explanation of each step in the BAYWATCH methodology.

4.1. Overview of the BAYWATCH Framework

The BAYWATCH framework consists of four main phases, each involving one or more filtering steps. These phases are:

1. **Whitelist Analysis:** This phase eliminates known legitimate beaconing traffic using universal and local whitelists.
2. **Time Series Analysis:** This phase identifies periodic communication patterns in the remaining traffic using a robust periodicity detection algorithm.
3. **Suspicious Indicator Analysis:** This phase further filters out legitimate beaconing behavior by analyzing domain-specific indicators of malicious activity.
4. **Investigation and Verification:** The final phase involves manual investigation and verification of the remaining suspicious cases.

Figure 4.1 provides a detailed overview of the algorithm's processing steps, which occur in four distinct phases. In Phase 1, the input data undergoes whitelist analysis, where it is categorized into two separate whitelists. The Universal Whitelist contains common, globally trusted URLs such as major search engines (e.g., Google, Yahoo) and other widely recognized platforms. The Local Whitelist, on the other hand, includes URLs that are specifically trusted within the organization, such as internal Allianz resolution URLs. This step helps to filter out trusted sources, ensuring that only potentially suspicious or unknown URLs are subjected to further analysis in subsequent phases.

Phase 2 focuses on time series analysis, where the algorithm processes the data to detect beaconing activity. Beaconing refers to the repeated communication of data to external servers, which can indicate malicious activity or unauthorized data exfiltration.

BAYWATCH Algorithm Steps

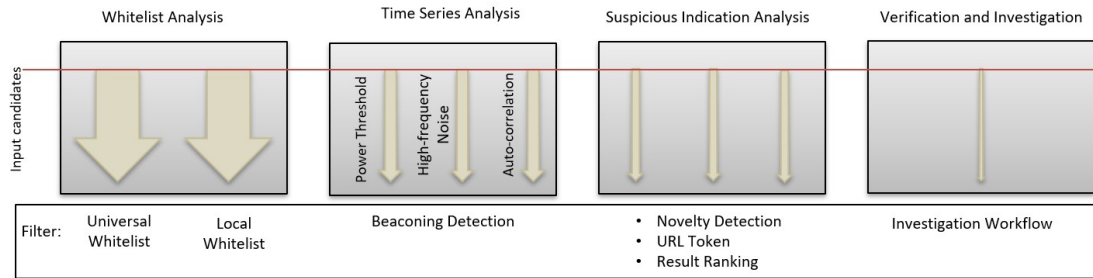


Figure 4.1.: Algorithm steps

In Phase 3, the Suspicious Indication Analysis takes place. This phase is composed of several sub-processes aimed at identifying and ranking suspicious URLs. It includes Novelty Detection, which focuses on recognizing previously unseen or unusual patterns in the data; URL Token analysis, which breaks down and examines specific elements of URLs for signs of malicious intent; and Result Ranking, where URLs are ranked based on their likelihood of being malicious, helping to prioritize further investigation. This step refines the list of suspicious indicators to ensure that only the most relevant ones are brought forward for verification.

Finally, in Phase 4, the algorithm enters the Verification and Investigation phase. This is the concluding step where the flagged URLs and potential threats are thoroughly investigated and verified. This phase ensures that any suspicious activities are properly validated, confirming whether they are legitimate threats or false alarms. The outcome of this phase directly informs decision-making processes regarding the necessary actions to mitigate or resolve any identified risks.

The following sections provide a detailed explanation of each phase and its corresponding steps.

4.2. Whitelist Analysis

The first phase of the BAYWATCH framework focuses on reducing the workload of subsequent phases by eliminating known legitimate beaconsing traffic. This is achieved through two whitelisting mechanisms: universal whitelisting and local whitelisting.

4.2.1. Universal Whitelisting

The universal whitelist contains globally trusted destinations, such as popular search engines, software update servers, and news feeds. These destinations are unlikely to be involved in malicious beaconsing unless they are compromised. The whitelist is dynamic and can be updated with new trusted destinations. In the BAYWATCH framework, the universal whitelist

is constructed using publicly available lists of popular domain names. These lists are curated based on the popularity and trustworthiness of the domains, ensuring that only well-known and widely used destinations are whitelisted.

4.2.2. Local Whitelisting

In addition to the universal whitelist, the BAYWATCH framework employs a local whitelist that is specific to the environment being monitored. This whitelist is constructed by measuring the popularity of destinations within the network. A destination is considered locally popular if it is accessed by a significant portion of the hosts in the network. These locally popular destinations are unlikely to be involved in malicious beaconing and are therefore whitelisted.

4.3. Time Series Analysis

The second phase of the BAYWATCH framework focuses on identifying periodic communication patterns in the network traffic. This phase is for detecting beaconing behavior, as it analyzes the temporal regularity of communication between hosts and external destinations. The time series analysis is performed using a combination of Fast Fourier Transform (FFT), autocorrelation, and bandpass filtering to robustly detect periodic signals even in the presence of noise and interruptions.

4.3.1. Algorithm Overview

The time series analysis algorithm is designed to detect periodic patterns in the communication data. It operates on a sequence of timestamps representing the connections between a source and destination pair. The algorithm consists of three main steps:

1. **Candidate Discovery:** This step identifies potential periodic components in the time series using the Fast Fourier Transform (FFT). The FFT converts the time series from the time domain to the frequency domain, allowing the algorithm to identify dominant frequencies that may correspond to periodic behavior.
2. **Pruning:** This step filters out high-frequency noise and less feasible candidates using statistical methods and bandpass filtering. The algorithm applies hypothesis testing to determine the statistical significance of the candidate periods and removes those that do not meet the significance threshold.
3. **Verification:** The final step verifies the validity of the remaining candidate periods using the circular autocorrelation function (ACF). The ACF measures the similarity between the time series and a shifted version of itself, providing a more fine-grained detection of periodic behavior.

4.3.2. Candidate Discovery Using FFT

The candidate discovery step begins by transforming the sequence of connection timestamps into a discrete time series. Let $T = \{t_1, t_2, \dots\}$ be the timestamps of connections between

4. Methodology

a communication pair. These timestamps are converted into an integer sequence $x(n) = \{x_0, x_1, \dots, x_{N-1}\}$, where $x_i > 0$ indicates that a connection occurred at time interval t_i , and $x_i = 0$ indicates no connection.

The Fast Fourier Transform (FFT) is then applied to the time series to identify dominant frequencies. The FFT decomposes the time series into its frequency components, producing a periodogram that shows the power of each frequency component. The periodogram is computed as:

$$P(k) = ||X(k)||^2,$$

where $X(k)$ is the FFT of the time series $x(n)$, and k represents the frequency index. The dominant frequencies in the periodogram correspond to potential periodic behaviors in the time series.

However, the FFT alone is not sufficient for accurate periodicity detection due to several limitations:

- The FFT can produce false positives for non-periodic signals, as it decomposes any signal into sinusoidal components.
- The FFT has limited resolution for low-frequency components, making it difficult to detect long-period beaconing.
- The FFT is sensitive to noise and interruptions in the time series, which can distort the periodogram.

To address these limitations, the BAYWATCH framework uses a permutation-based filtering approach to determine a power threshold for identifying significant frequencies. This threshold is calculated by shuffling the time series and computing the maximum power in the permuted signal. Only frequencies with power above this threshold are considered as potential candidates for periodic behavior.

4.3.3. Pruning Using Bandpass Filtering

After identifying candidate frequencies, the algorithm prunes the set of candidates to remove high-frequency noise and less feasible periods. This is achieved using a combination of bandpass filtering and hypothesis testing.

The bandpass filtering step removes high-frequency noise by focusing on the frequency range of interest. In the context of beaconing detection, the frequency range is determined by the expected beaconing intervals (e.g., from seconds to hours). The algorithm applies a bandpass filter to the time series to isolate the frequency components within this range, effectively removing noise outside the range of interest.

4.3.4. Verification Using Autocorrelation

The final step in the time series analysis is the verification of the remaining candidate periods using the circular autocorrelation function (ACF). The ACF measures the similarity between

the time series and a shifted version of itself, providing a more accurate detection of periodic behavior.

The ACF is calculated as:

$$ACF(\tau) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \cdot x(n + \tau),$$

where τ is the time shift. If τ corresponds to the true period, the shifted time series will overlap with the original time series, resulting in a high correlation value. The algorithm verifies each candidate period by calculating the ACF and retaining only those periods that exhibit strong autocorrelation.

4.3.5. Handling Multiple Periodicities

In some cases, the time series may exhibit multiple periodic behaviors at different time scales (e.g., a botnet that beacons every 5 seconds and every 2 hours). To handle such cases, the BAYWATCH framework uses a Gaussian Mixture Model (GMM) to identify multiple underlying periodic components. The GMM clusters the observed intervals into distinct groups, each corresponding to a different periodicity. The algorithm then applies hypothesis testing to each group separately to verify the validity of the candidate periods.

4.4. Suspicious Indicator Analysis

The third phase of the BAYWATCH framework focuses on distinguishing legitimate beaconing behavior from suspicious behavior by analyzing domain-specific indicators of malicious activity. This phase involves several sub-steps, including novelty detection, URL token analysis, and result ranking.

4.5. Investigation and Verification

The final phase of the BAYWATCH framework involves the manual investigation and verification of the remaining suspicious cases. This phase is for reducing false positives and ensuring that only truly malicious beaconing cases are reported.

4.5.1. Feature Set

Each candidate case is represented by a set of features, including the source, destination, and a series of time intervals. The BAYWATCH framework generates additional features, such as the entropy of the time intervals, the n-gram histogram, and the compressibility of the symbolized series. These features are used to train a classifier for automated classification of the candidate cases.

4. Methodology

4.5.2. Classifier

The BAYWATCH framework employs a random forest classifier to classify the candidate cases as either benign or malicious. The random forest classifier is trained using a small set of manually investigated cases and their corresponding labels. The trained classifier is then applied to the remaining cases to automate the classification process.

4.5.3. Bootstrapping Process

To minimize the manual investigation workload, the BAYWATCH framework employs a bootstrapping process. A small set of candidate cases is manually investigated and used as a training set for the classifier. The trained classifier is then applied to the remaining cases, significantly reducing the number of cases that require manual investigation.

4.6. Real Data Source

The real-world data used in this study was collected from a large-scale enterprise network, capturing user activities as they navigate various URLs throughout the workday. This dataset provides a detailed perspective on user interactions, enabling an in-depth analysis of browsing patterns and behaviors. The data is stored in JSON format, which offers flexibility and readability, making it easier to manage and manipulate large volumes of information. Each entry in the dataset records a specific user interaction, including precise timestamps and the URLs visited, allowing for a chronological reconstruction of user activities. This level of detail is crucial for identifying patterns and trends over time, such as peak usage periods or frequent transitions between specific URLs.

4.6.1. Data Structure and Schema

The dataset is structured as a collection of JSON files, with each file containing detailed logs of user interactions. Each entry in the JSON files includes the following fields:

- **IP_Address:** The IP_Address of the user's device, providing a unique identifier for each host
- **logdate:** The date and time of the user interaction, recorded in a standardized date-time format.
- **url_hostname:** The hostname of the URL visited by the user.
- **user:** An optional field denoting the user identifier. For security reasons, usernames are deliberately omitted during the import process.

The structure of the JSON files is defined by a Document Type Definition (DTD), which ensures consistency and reliability across all entries. Below is an example of the JSON schema used for the dataset:

```

1 {
2   "$schema": "http://json-schema.org/draft-07/schema#",
3   "type": "object",
4   "properties": {
5     "logdate": {
6       "type": "string",
7       "format": "date-time"
8     },
9     "urlhostname": {
10      "type": "string"
11    },
12    "user": {
13      "type": "string"
14    }
15  },
16  "required": ["logdate", "urlhostname"]
17 }

```

The structured format of the JSON files ensures that each entry is consistent and comprehensive, providing a reliable record of user activities for analysis.

4.6.2. Data Collection and Scale

The dataset was collected over the course of a single day, specifically a typical Tuesday workday, generating nearly 73 gigabytes of information. This large-scale data collection captures the following details:

- **Host Information:** The IP addresses of the user devices, enabling the tracking of individual hosts and their activities.
- **Timestamps:** Precise date and time of each user interaction, enabling temporal analysis of browsing patterns.
- **URL Hostnames:** The hostnames of the URLs visited, providing insights into the destinations of user traffic.
- **User Interactions:** A chronological record of user activities, facilitating the identification of trends and anomalies.

The dataset's scale and granularity make it an ideal resource for analyzing user behavior, identifying significant patterns, and supporting the development of effective beaconing detection strategies.

4.6.3. Data Management and Preprocessing

To manage and analyze the dataset effectively, a sophisticated data management system was implemented. The system leverages **InfluxDB**, a time-series database optimized for handling high volumes of temporal data. The data management process involves the following steps:

1. **Data Import:** The dataset is imported into InfluxDB using custom Python scripts. These scripts automate the creation of a dedicated "bucket" within InfluxDB, ensuring that the data is organized and stored efficiently.
2. **Schema Implementation:** A predefined schema is applied to enforce data integrity and consistency. This schema ensures that all entries adhere to the same format and standards, facilitating smoother data processing and analysis.
3. **Initial Data Analysis:** The dataset is analyzed to understand its behavior, including:
 - Observing overall data trends throughout the day.
 - Identifying the most frequently accessed URLs and calculating their averages.
 - Analyzing the time intervals between requests.
 - Examining the distribution of hosts and their activity patterns.

4.6.4. Challenges with Real-World Data

The real-world dataset presents several challenges that must be addressed to ensure accurate and reliable analysis:

- **Noise and Variability:** Real-world network traffic is inherently noisy, with random variations in connection timing due to network delays, retransmissions, and other factors. This noise can obscure periodic patterns and complicate the detection of beaconing behavior.
- **Missing Data:** Devices may go offline or move out of the observation range, resulting in gaps in the data. These gaps can disrupt the detection of periodic behavior and require careful handling during analysis.
- **Legitimate Periodic Traffic:** Many legitimate applications (e.g., software updates, email polling) exhibit periodic behavior that resembles beaconing. Distinguishing between legitimate and malicious periodic traffic is a key challenge in real-world data analysis.

4.7. Artificial Data Source

In addition to analyzing real-world network traffic, the BAYWATCH framework was evaluated using artificial data to test its robustness and accuracy under controlled conditions. The artificial data was designed to simulate various types of beaconing behavior, including different periodicities, noise levels, and evasion techniques commonly employed by malware authors. A key feature of the artificial data is the introduction of jitter, which simulates random variations

in the timing of beaconing events. This section describes the process of generating the artificial data, the specific jitter ranges used, and the structure of the data.

4.7.1. Design of Artificial Data

The artificial data was generated to mimic the structure of real-world network traffic, while allowing for precise control over the parameters of the beaconing behavior. Each artificial data set consists of the following fields:

- **Host Information:** The IP addresses of the user devices, enabling the tracking of individual hosts and their activities.
- **Timestamps:** Precise date and time of each user interaction, enabling temporal analysis of browsing patterns.
- **URL Hostnames:** The hostnames of the URLs visited, providing insights into the destinations of user traffic.
- **User Interactions:** A chronological record of user activities, facilitating the identification of trends and anomalies.
- **Is Artificial:** A tag (labeled as "yes") was added to distinguish the artificial data from real-world data. This tag ensures that the artificial data can be easily identified and separated during analysis.

4.7.2. Jitter Ranges

Jitter is a critical parameter in simulating real-world beaconing behavior, as it introduces randomness into the timing of beaconing events. To evaluate the robustness of the BAYWATCH framework, the following jitter ranges were used:

Jitter ranges: [2, 5, 10, 30, 60] seconds

Each jitter range represents a different level of perturbation in the beaconing behavior:

- **2 seconds:** Minimal jitter, simulating near-ideal conditions with very little variation in beacon timing.
- **5 seconds:** Low jitter, simulating slight variations in beacon timing due to minor network delays.
- **10 seconds:** Moderate jitter, simulating more noticeable variations in beacon timing.
- **30 seconds:** High jitter, simulating significant variations in beacon timing, potentially due to network congestion or intentional evasion techniques.
- **60 seconds:** Very high jitter, simulating extreme variations in beacon timing, which may occur in highly unstable network conditions.

4. Methodology

4.7.3. Scenarios Tested

The artificial data was used to test the BAYWATCH framework under the following scenarios:

- **Low Jitter:** Beaconsing behavior with minimal jitter (e.g., 2 seconds). This scenario tests the framework's ability to detect periodic behavior in near-ideal conditions.
- **Moderate Jitter:** Beaconsing behavior with moderate jitter (e.g., 10 seconds). This scenario tests the framework's robustness to typical real-world perturbations.
- **High Jitter:** Beaconsing behavior with significant jitter (e.g., 30 seconds). This scenario tests the framework's ability to handle more extreme variations in beacon timing.
- **Very High Jitter:** Beaconsing behavior with very high jitter (e.g., 60 seconds). This scenario tests the framework's performance under highly unstable network conditions.

4.7.4. Integration with Real-World Data

The artificial data was used in conjunction with real-world network traffic to provide a comprehensive evaluation of the BAYWATCH framework. While the real-world data provides insights into the framework's performance in a production environment, the artificial data allows for controlled testing of specific scenarios and edge cases. The `is_Artificial` tag ensures that the artificial data can be easily distinguished from real-world data during analysis. This combination ensures that the framework is both robust to real-world perturbations and accurate in detecting malicious beaconsing behavior.

4.8. Summary

The BAYWATCH framework is a robust and scalable methodology designed to detect stealthy beaconsing behavior in large-scale enterprise networks. It operates in four main phases: **Whitelist Analysis**, which eliminates known legitimate traffic using universal and local whitelists; **Time Series Analysis**, which identifies periodic communication patterns using advanced signal processing techniques such as Fast Fourier Transform (FFT), autocorrelation, and bandpass filtering; **Suspicious Indicator Analysis**, which further filters out legitimate behavior by analyzing domain-specific indicators like URL tokens and novelty; and **Investigation and Verification**, where remaining suspicious cases are manually reviewed using a bootstrapping process to minimize workload. The framework was evaluated using both **real-world data**, collected from a large-scale enterprise network, and **artificial data**, which simulated various beaconsing scenarios with controlled jitter ranges (2, 5, 10, 30, and 60 seconds) and noise levels. The integration of real-world and artificial data ensures a comprehensive evaluation, demonstrating the framework's ability to reliably detect malicious beaconsing behavior while remaining robust to real-world perturbations and noise. This makes BAYWATCH a valuable tool for securing enterprise networks against advanced cyber threats.

5. Data Analysis

This chapter delves into the heart of the research methodology, exploring the data analysis process in detail. By examining the various analytical techniques and algorithms employed, the chapter aims to provide a comprehensive understanding of how the dataset is processed and interpreted to extract meaningful insights. The analysis encompasses a range of advanced methods, including anomaly detection, pattern recognition, and machine learning algorithms, all of which contribute to a robust framework for identifying and mitigating malicious beaconing activities.

5.1. Visualization of URL Request Counts

To better understand the distribution and frequency of URL requests within the dataset, visual representations are utilized. These visualizations help in identifying patterns and anomalies in user behavior and resource utilization. The following figures provide insights into the request counts for different URLs, using both logarithmic and linear scales for comparison.

Figure 5.1 provides a visual representation of the request counts for different URLs within the dataset. The logarithmic scale on the Y-axis allows for a clearer comparison of the visit frequencies across URLs with varying levels of activity. This visualization highlights the distribution of request counts, showcasing the range of visit frequencies observed within the dataset. By examining this distribution, it is possible to identify URLs with high visit counts, which may indicate critical resources or frequently accessed services. Conversely, URLs with lower visit counts may represent less frequently accessed or less critical components of the network. This analysis provides valuable insights into user behavior and resource utilization, enabling organizations to optimize their network infrastructure and prioritize security measures effectively.

Figure 5.2 provides a linear scale representation of the request counts for different URLs within the dataset. This visualization offers a more detailed view of the visit frequencies across URLs, highlighting the distribution of request counts with greater granularity. By examining this distribution, it is possible to identify URLs with varying levels of activity, ranging from high-visit counts to low-visit counts. This analysis enables organizations to gain insights into user behavior and resource utilization, facilitating informed decision-making and strategic planning.

The logarithmic scale in Figure 5.1 allows for a clearer comparison of URLs with varying request counts by compressing the scale for higher values and expanding it for lower values. This makes it easier to identify both high and low-frequency URLs, providing a balanced view of the data.

In contrast, the linear scale in Figure 5.2 presents the data with equal distances on the Y-axis representing equal differences in request counts. This offers a more detailed view of URLs with similar request counts but may obscure relative differences when the range of request counts

5. Data Analysis

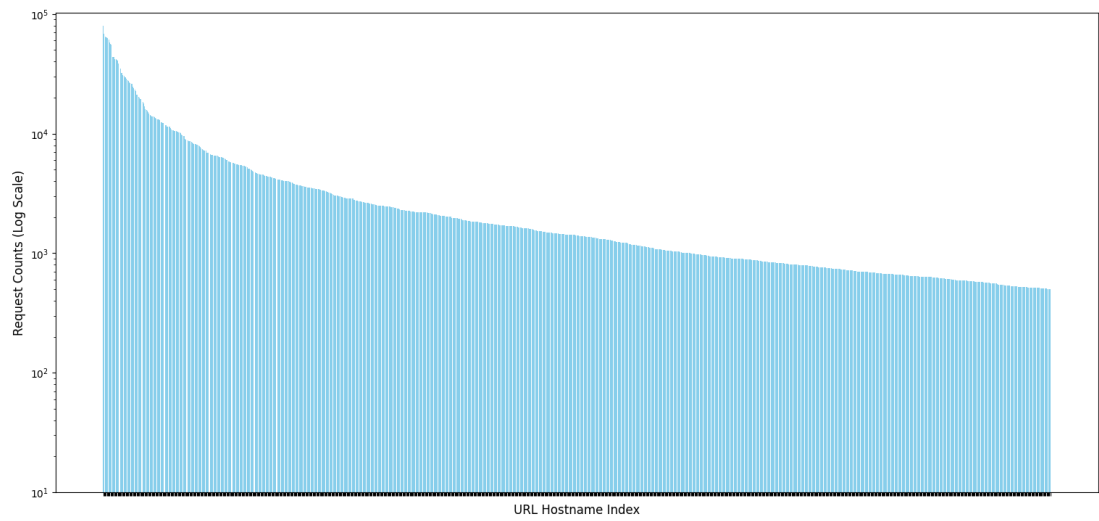


Figure 5.1.: Request counts of URLs (log scale)

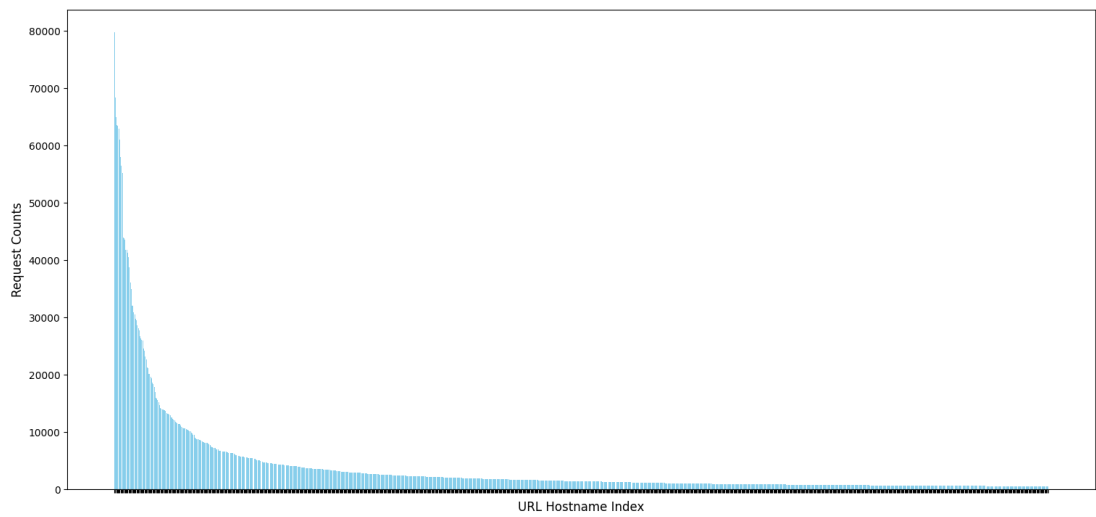


Figure 5.2.: Request counts of URLs (linear scale)

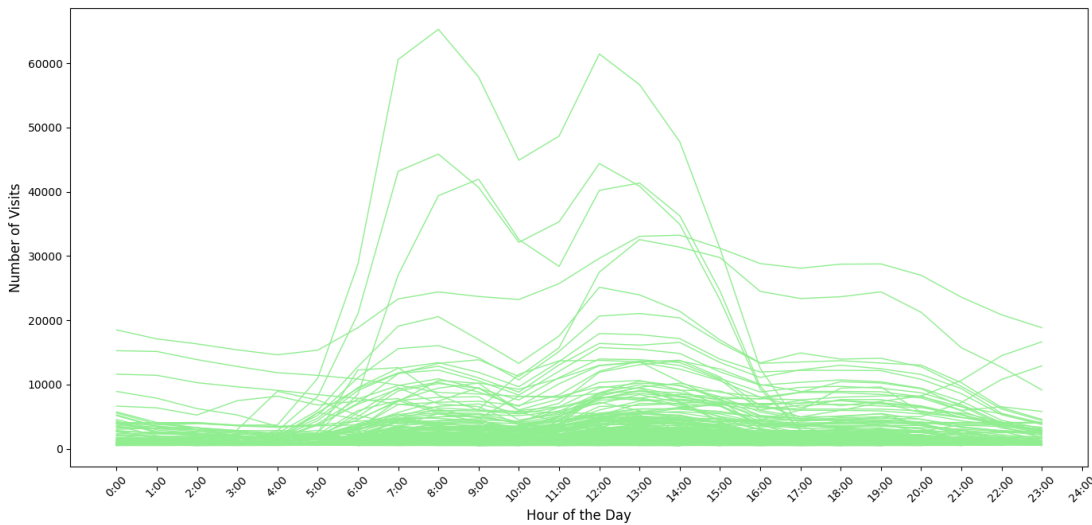


Figure 5.3.: Number of visit by hour (24 hours)

is large. The choice between logarithmic and linear scales depends on the specific aspects of the data that need to be emphasized.

5.2. 24-Hour URL Visit Analysis

To gain insights into the temporal patterns of URL visits, a 24-hour analysis is conducted. This analysis helps in understanding the distribution of user activity throughout the day, identifying peak usage times, and detecting periods of lower activity. Such temporal analysis is for recognizing trends and potential anomalies in user behavior.

Figure 5.3 illustrates the number of visits to different URLs over a 24-hour period. The x-axis represents the hours of the day, while the y-axis indicates the number of visits to each URL. This visualization provides a clear overview of the distribution of visits throughout the day, highlighting peak usage times and periods of lower activity. By examining this data, it is possible to identify trends and patterns in user behavior, which can be instrumental in detecting anomalies or suspicious activities. This analysis serves as a foundational step in understanding the dataset's behavior and establishing a baseline for further investigations. As shown, the distribution of visits predominantly falls within the range of 0–500, which is significantly higher compared to the rest.

From the figure, it is evident that certain URLs exhibit high activity levels during the initial hours but experience a sharp decline, with their visit counts approaching zero around 04:00. This observation led to the categorization of URL activity into two distinct periods: day activity, which begins at 00:00 and ends at 04:00, and night activity, which spans from 04:00 to 24:00. To better understand these terms and analyze the patterns, the average number of visits during each period was calculated. This analysis provides valuable insights into how URL activity

5. Data Analysis

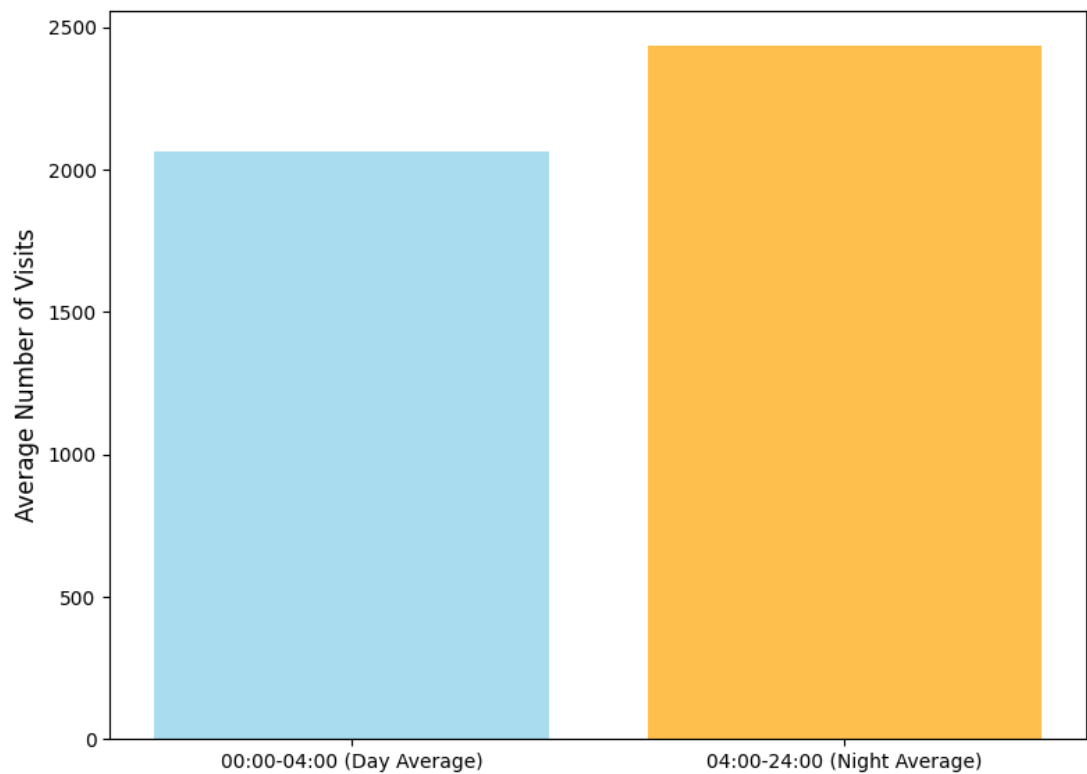


Figure 5.4.: Average number of visits during day and night

fluctuates throughout the day and highlights significant differences in usage between the two time frames, enabling more effective resource allocation and decision-making based on user behavior.

Figure 5.4 compares the average number of visits during two distinct time ranges: from 00:00 to 04:00 ("Day Average") and from 04:00 to 24:00 ("Night Average"). The "Day Average" is represented by a light blue bar, which shows an average number of visits slightly exceeding 2,000. On the other hand, the "Night Average" is depicted with an orange bar, which is taller and indicates a higher average of approximately 2,500 visits. The chart effectively highlights that the visitation rate is higher during the later time period, with the bars labeled clearly and a descriptive title ("Day vs. Night Average Visits") at the top. Additionally, the y-axis represents the average number of visits, ranging from 0 to 2,500, with consistent scaling and proper spacing for visual clarity. This analysis provides valuable insights into user behavior patterns and resource utilization, enabling organizations to optimize their network infrastructure and enhance security measures effectively.

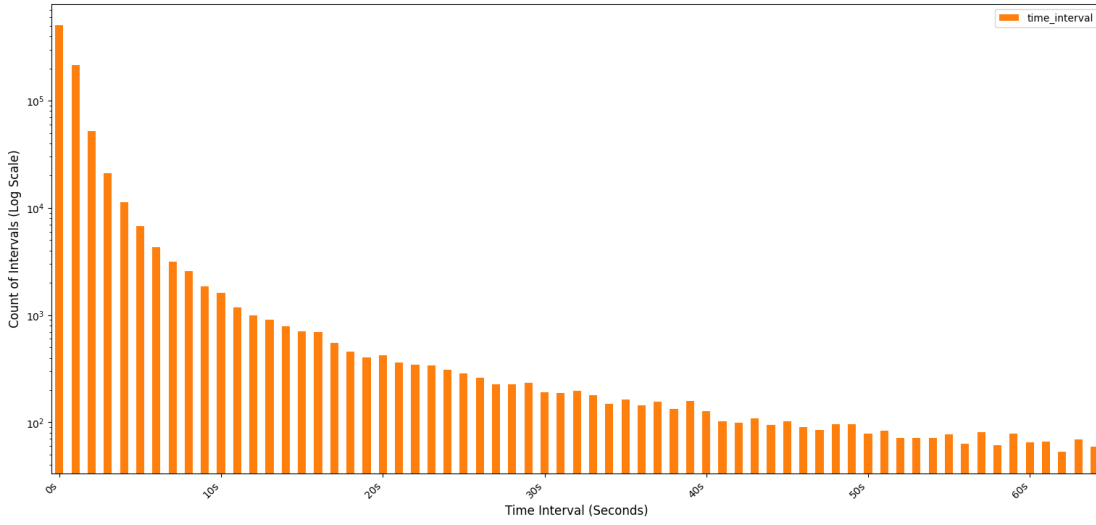


Figure 5.5.: Time interval 0-65s (log scale)

5.3. Time Interval Analysis of URL Requests

To understand the temporal dynamics of user interactions, an analysis of the time intervals between requests for different URLs is conducted. This analysis helps in identifying patterns and trends in the timing of user activities, which can be crucial for optimizing network resources and enhancing security measures.

Figure 5.5 illustrates the distribution of time intervals between requests for different URLs within the dataset. The logarithmic scale on the Y-axis allows for a clearer comparison of the time intervals across URLs with varying patterns of activity. The X-axis demonstrates the bins, which are divided into 90 bins. These bins are structured as follows: from 0 to 65 seconds, each second has its own bin. This visualization highlights the variability in time intervals between requests, showcasing the range of durations observed within the dataset. By examining this distribution, it is possible to identify URLs with distinct time interval patterns, which may indicate specific usage behaviors or interaction trends. This analysis provides valuable insights into the frequency and timing of user interactions, enabling organizations to optimize their network resources and enhance security measures effectively. As can be seen in the figure, the requests are decreasing across the entire time range, but every 10 seconds, the value is slightly higher than the previous second. This pattern is consistent across all URLs, indicating a common behavior in the dataset. This observation suggests that the time intervals between requests follow a specific pattern, which may be indicative of regular user activity or system behavior. By analyzing these patterns, organizations can gain valuable insights into user behavior and resource utilization, enabling them to optimize their network infrastructure and enhance security measures effectively.

Figure 5.6 illustrates the distribution of time intervals between requests for different URLs within the dataset, with the Y-axis represented on a logarithmic scale. The X-axis demonstrates

5. Data Analysis

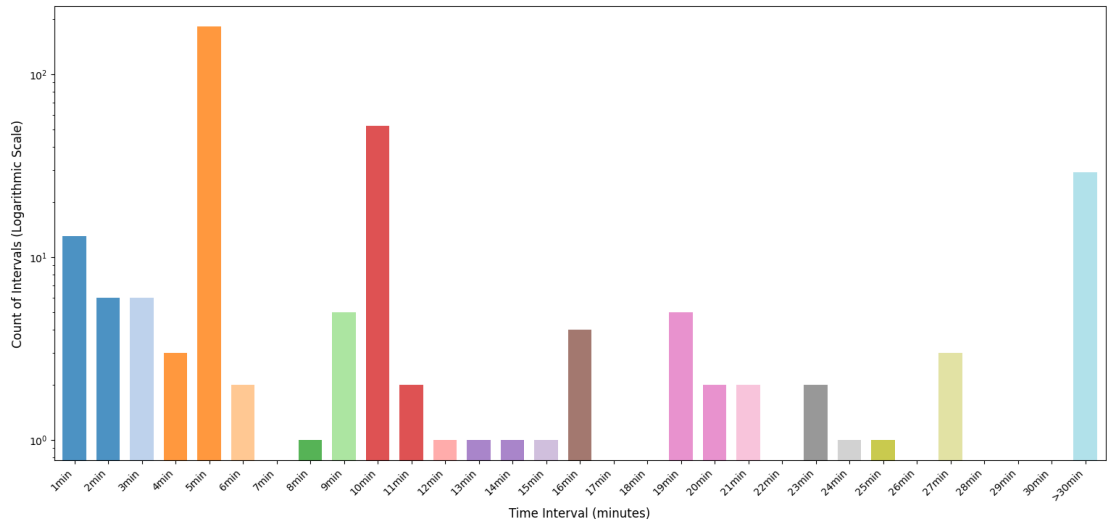


Figure 5.6.: Time interval in minutes (log scale)

the bins, which are divided into 31 bins. These bins are structured as follows: Each bin shows the requests in minutes. To ensure that beaconing behavior at the edge of each minute is not missed, the number is shown with ± 30 seconds. This visualization provides a detailed view of the time intervals between requests, highlighting the variability in durations observed within the dataset. By examining this distribution, it is possible to identify URLs with distinct time interval patterns, which may indicate specific usage behaviors or interaction trends. This analysis offers valuable insights into the frequency and timing of user interactions, enabling organizations to optimize their network resources and enhance security measures effectively. As can be seen in the figure, the requests are decreasing across the entire time range, but every minute, the value is slightly higher than the previous minute. This pattern is consistent across all URLs, indicating a common behavior in the dataset. This observation suggests that the time intervals between requests follow a specific pattern, which may be indicative of regular user activity or system behavior. By analyzing these patterns, organizations can gain valuable insights into user behavior and resource utilization, enabling them to optimize their network infrastructure and enhance security measures effectively.

both figures 5.5 and 5.6 provide a detailed view of the time intervals between requests for different URLs within the dataset. The logarithmic scale on the Y-axis allows for a clearer comparison of the time intervals across URLs with varying patterns of activity. The scale is in logarithmic because, in the linear scale, the differences in time intervals were not shown as clearly. The differences between the bins were too large. The X-axis demonstrates the bins, which are divided into 90 bins for seconds and 31 bins for minutes. These bins are structured to capture the variability in time intervals between requests, showcasing the range of durations observed within the dataset. By examining this distribution, it is possible to identify URLs with distinct time interval patterns, which may indicate specific usage behaviors or interaction trends. This analysis provides valuable insights into the frequency and timing of user interactions, enabling

5.4. Distribution of Hosts Based on Unique URLs Contacted

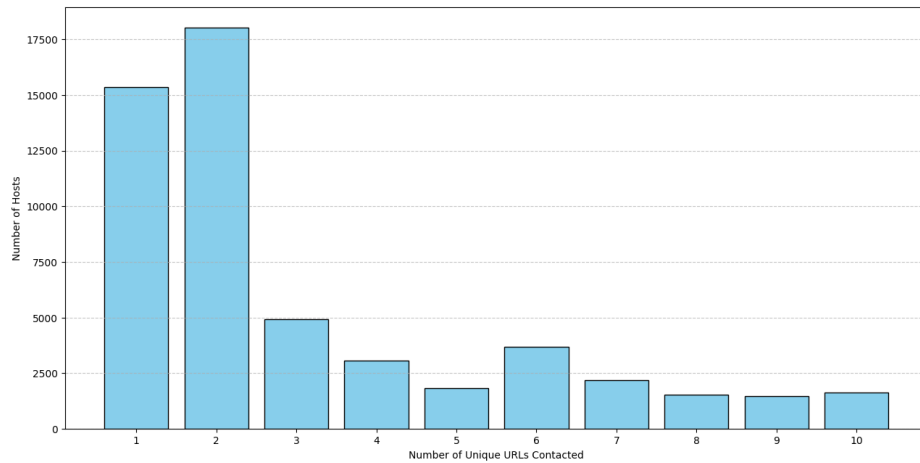


Figure 5.7.: Distribution of hosts

organizations to optimize their network resources and enhance security measures effectively. The consistent patterns observed in the time intervals between requests suggest a regularity in user behavior or system interactions, which may be indicative of normal network activity. By analyzing these patterns, organizations can gain valuable insights into user behavior and resource utilization, enabling them to optimize their network infrastructure and enhance security measures effectively.

5.4. Distribution of Hosts Based on Unique URLs Contacted

To analyze the interaction patterns of hosts within the network, a bar chart is used to illustrate the distribution of hosts (IP addresses) based on the number of unique URLs they contacted. This analysis helps in understanding the concentration of network activity and identifying key services or domains being accessed.

Figure 5.7 The bar chart provides an analysis of the distribution of hosts (IP addresses) based on the number of unique URLs they contacted. The X-axis represents the number of unique URLs contacted by each host, ranging from 1 to 10, while the Y-axis depicts the total count of hosts for each corresponding category. Each bar in the chart indicates how many hosts interacted with a specific number of unique URLs. The data reveals a clear trend: the majority of hosts engaged with only a small number of unique URLs. Specifically, the highest number of hosts, approximately 17,500, connected to exactly two unique URLs. This is followed closely by the category of hosts that connected to only one unique URL, which includes around 15,000 hosts. As the number of unique URLs increases beyond two, there is a steady decline in the number of hosts. For instance, significantly fewer hosts contacted 4 or more unique URLs, and the numbers continue to diminish as the variety of URLs increases. This pattern suggests that most network activity is concentrated on a limited set of destinations, with relatively few

5. Data Analysis

hosts interacting with a wide range of URLs. Such insights could be valuable for understanding traffic patterns, identifying key services or domains being accessed, and optimizing network performance or security measures. The visualization highlights the sparsity of hosts interacting with a larger variety of destinations, emphasizing the importance of focusing on the most commonly accessed URLs in any further analysis or intervention.

6. Implementation

This chapter introduces the implementation of the proposed methodology within Allianz Company's network infrastructure, delving into the intricate process of adapting the system to integrate seamlessly with the company's extensive log data. It begins by exploring the necessary adjustments made to align the methodology with Allianz's specific log data formats and structures, highlighting the critical decisions in parameter selection and the strategic use of various analytical tools. The chapter aims to provide a comprehensive evaluation of the performance metrics derived from this implementation, offering a detailed analysis of the methodology's efficacy. Supported by visualizations such as graphs and charts, this analysis facilitates a clearer understanding of complex data and key findings.

Furthermore, the chapter rigorously assesses the methodology's effectiveness in detecting malicious behavior, providing an in-depth examination of detected anomalies, their correlation with potential security threats, and the system's responsiveness. This assessment underscores the practical value of the methodology, demonstrating its significant impact on enhancing network security. By presenting concrete evidence of the methodology's success in identifying and mitigating threats, the chapter establishes a foundation for discussing advanced security strategies.

The insights gained from this implementation are important for Allianz, as they pave the way for continuous improvement initiatives and the development of more robust security measures. This chapter sets the stage for broader discussions on enhancing network security, offering a clear pathway for future chapters to explore advanced techniques and strategies aimed at fortifying Allianz's network infrastructure against the ever-evolving landscape of cyber threats. Through this comprehensive examination, the chapter not only highlights the immediate benefits of the methodology but also its long-term potential to significantly bolster Allianz's cybersecurity framework.

6.1. Experimental Setup

This section details the adaptation process to integrate the methodology within Allianz Company's network infrastructure. Initially, adjustments were made to ensure compatibility with the company's log data, involving comprehensive data mapping and transformation to align with the methodology's requirements. This step was key to guarantee that the raw data could be accurately interpreted and utilized by the detection algorithms. Stringent measures were taken to address any discrepancies or inconsistencies encountered during the integration process. These measures included validating data integrity, standardizing log formats, and resolving any anomalies to ensure seamless integration.

For testing purposes, an experimental framework was established. This framework was designed to cover a wide range of scenarios, from routine network operations to sophisticated

6. Implementation

simulated cyber-attacks. These scenarios were crafted to assess the methodology's performance under diverse conditions, providing a realistic and thorough evaluation. The scenarios mimicked real-world network behaviors, encompassing various types of user activities, network loads, and potential threat vectors. This comprehensive testing ensured that the methodology was robust and applicable in practical settings, capable of handling the dynamic nature of real-world network environments.

Additionally, real data sourced from Allianz Company was utilized to validate the methodology's efficacy under authentic operational conditions. This real-world validation was a critical component of the adaptation process, as it provided invaluable insights into the methodology's performance in a live environment. The use of genuine operational data bolstered the credibility and relevance of the methodology, demonstrating its practical utility in addressing real-world cybersecurity challenges. By evaluating the methodology against actual network traffic and user behavior, the team could identify and address any limitations, fine-tuning the system to enhance its effectiveness.

Throughout the testing phase, data collection was conducted rigorously, capturing a comprehensive array of network activities and events. This extensive data collection ensured a rich dataset for analysis, encompassing a wide variety of normal and abnormal behaviors. The collected data served as the foundation for subsequent analyses, enabling a thorough and detailed evaluation of the methodology's effectiveness in detecting and mitigating malicious behavior within Allianz Company's network infrastructure. The analysis focused on identifying patterns and anomalies indicative of malicious activity and assessing the accuracy and reliability of the detection algorithms.

The comprehensive approach to adaptation and testing detailed in this section underscores the methodology's readiness for real-world deployment. By ensuring data compatibility, rigorously testing under diverse scenarios, and validating with real-world data, the methodology is demonstrated to be not only theoretically sound but also practically effective. This robust process lays a solid foundation for enhancing Allianz's network security, providing a reliable tool to tackle the complex cybersecurity issues faced by the organization. The insights and results garnered from this extensive testing phase set the stage for further refinement and optimization, ensuring that the methodology remains effective against evolving cyber threats.

6.2. Whitelisting Mechanism for URL Filtering

In the heart of Allianz Company's expansive network infrastructure, a pivotal decision was made to bolster its security framework: the establishment of a robust URL whitelist. This action was driven by the need to strengthen monitoring and enhance the safety of the company's online activities. The idea was simple yet profound—by creating a safe space within the system where only trusted URLs would be allowed to flourish, potential risks could be minimized, and network security maximized.

The process of curating this whitelist involved a meticulous selection of URLs. These were determined by several criteria, each designed to ensure that only the most legitimate and useful links were included. URLs that employees visited regularly, those associated with company resolutions, and other trusted websites made it onto the list. On the other hand, URLs that did

not meet these predefined conditions were excluded from the whitelist. These exclusionary decisions were not arbitrary; instead, they stemmed from a deep understanding of what constituted a potential threat to the network. Suspicious IP addresses, known malicious domains, and unauthorized access points all triggered exclusion, thereby preserving the integrity of the company's digital infrastructure.

The function at the core of this process was designed to filter and process URLs with precision. It carefully examined each one against the exclusion criteria and sifted out those that posed a threat. What remained was a curated list of URLs that were deemed safe and, more importantly, relevant. This refined subset of URLs, free from the noise of irrelevant or dangerous data, allowed for more focused analysis, enabling security teams to zoom in on genuine threats. It also made the data easier to interpret, fostering clearer and more actionable insights.

By ensuring that only safe URLs entered the system, the whitelist function served as a key tool in the network's defense. It reduced the amount of data to be analyzed, cutting through the clutter and allowing for faster and more accurate detection of potential security breaches. This streamlined approach, which prioritized clarity and relevance, empowered analysts to identify anomalies in the network more effectively, making it easier to pinpoint suspicious activity and prevent breaches before they escalated.

As the team at Allianz worked tirelessly to perfect this methodology, it became clear that the whitelist was more than just a protective measure—it was a vital cog in the larger machine of network security. Its careful design and implementation underscored a strategic approach to safeguarding digital assets, ensuring that every step taken was one toward a more secure and resilient system. The precision with which the whitelist function was crafted, and the clarity it brought to network analysis, was a testament to the team's commitment to protecting the company's online environment. Through this careful curation of data, the organization could focus on what mattered most—securing the network while minimizing the risk of potential cyber threats.

6.3. Average Power Calculation

The process of calculating the power of requests is a critical component of analyzing network activity and discerning meaningful patterns within large datasets. This methodology unfolds as a sequential progression, beginning with the preliminary task of filtering URLs based on a predefined whitelist. Once this filtration is complete, the focus shifts to the intricate process of determining the power associated with each URL.

At the heart of this approach lies the notion of request power, a concept that quantitatively captures the frequency dynamics of network interactions. Specifically, this metric is derived by analyzing the temporal intervals between successive requests made to the same URL hostname. Each interval represents the duration elapsed between two consecutive requests, effectively capturing the rhythm and regularity of access patterns. By measuring these intervals across the dataset, the power of each URL is computed, providing valuable insights into the underlying behavioral trends.

The process begins with the creation of a dictionary, often referred to as the power dictionary, which serves as the repository for storing these calculated values. For every request

6. Implementation

encountered in the dataset, the corresponding time interval is computed by subtracting the timestamp of the last occurrence from that of the current request. These intervals, measured in seconds, form the basis for assessing the frequency of access. Each time interval is then mapped to its occurrence count within the dictionary, where the count represents the number of times a specific interval has been observed. As this iterative process unfolds, the dictionary gradually accumulates a comprehensive record of interval frequencies, encapsulating the temporal dynamics of the dataset.

Upon completion of this calculation phase, the next step involves computing the average power across all URLs. This average serves as a benchmark against which individual URL powers are evaluated. The normalization process entails subtracting this average from the power values of each URL. This critical step ensures that the analysis focuses on relative deviations rather than absolute values, thereby highlighting URLs that exhibit unusual activity. Notably, URLs with normalized power values that fall below zero are deemed indicative of non-malicious behavior and are excluded from further analysis. Conversely, those with positive values proceed to subsequent stages of scrutiny, marking them as potential candidates for deeper investigation.

This analytical framework offers a robust mechanism for distinguishing normal network behavior from anomalies. By systematically identifying URLs with significant deviations in access frequency, the methodology enhances the detection of patterns that may signal malicious intent. The exclusion of URLs with negative power values serves to streamline the analysis, enabling a sharper focus on high-priority cases. Moreover, the iterative nature of this calculation process facilitates continuous refinement, allowing the methodology to adapt dynamically to the evolving landscape of network activity.

Beyond its technical utility, the calculation of request power provides profound insights into the behavioral dynamics of users and systems. By revealing the cadence of interactions and the distribution of access intervals, it paints a detailed picture of network usage patterns. These insights are invaluable not only for identifying potential security threats but also for understanding the broader context of network operations. The power dictionary, as the tangible output of this process, serves as a foundational tool for exploring these patterns. Its structured representation of time intervals and their corresponding power values offers a precise lens through which the intricacies of network activity can be examined.

In essence, the request power calculation methodology exemplifies the fusion of quantitative rigor and analytical depth. It transforms raw data into actionable intelligence, equipping analysts with the tools needed to navigate the complexities of modern network environments. Through its emphasis on precision, scalability, and adaptability, this approach underscores its pivotal role in fortifying network security and advancing the frontier of behavioral analytics.

6.4. Band-Pass Filtering

Bandpass filtering is an advanced signal processing technique used to refine time-series data by isolating specific frequency components within a defined range. The method operates by allowing only those components of a signal whose frequencies lie within a certain interval to pass through, while suppressing or attenuating those outside this range. This selective filter-

ing approach is instrumental in reducing noise, enhancing clarity, and extracting meaningful patterns from complex datasets. In the context of network traffic analysis, this technique is particularly valuable for identifying periodic behaviors or anomalies that occur within a specific frequency spectrum.

The process begins by setting lower and upper frequency boundaries, defined in seconds, to determine the target frequency range. These boundaries are normalized using the Nyquist frequency, which is half the sampling rate of the data, ensuring that the filtering criteria align with the temporal resolution of the dataset. Validation checks are performed to ensure that the normalized thresholds fall within an acceptable range, thereby avoiding computational errors. The Butterworth filter, known for its smooth frequency response and minimal distortions, is employed for this purpose. Unlike other filters, the Butterworth design maintains a balanced trade-off between precision and computational efficiency, making it well-suited for processing large or sensitive datasets.

To ensure accuracy and minimize phase distortions, the filtering process applies a forward and backward pass on the data, effectively refining the signal. This dual-pass approach produces a dataset containing only the frequency components that meet the specified criteria. For network traffic analysis, the bandpass filter can, for instance, evaluate the time intervals associated with URLs. By retaining only those URLs with temporal patterns falling within the target frequency range, the filter ensures that the analysis focuses on the most relevant components.

The practical implementation in your research involves defining specific lowcut and highcut frequencies, such as 1 second and 1 hour, respectively. This range captures patterns of interest while excluding noise and irrelevant fluctuations. The filtering process not only reduces the dataset's complexity and volume but also amplifies the significance of the retained information. By systematically excluding URLs with unimportant temporal dynamics, the technique supports a more focused and effective analysis of network behaviors.

This methodology has proven critical for cybersecurity applications, where detecting subtle variations in traffic patterns can reveal potential threats or anomalies. For example, identifying periodic spikes in requests to specific URLs within a defined frequency band can indicate malicious activity or abnormal network behavior. The refined dataset resulting from bandpass filtering forms a robust foundation for further analysis, enabling researchers to derive accurate, actionable insights.

In addition to improving analytical precision, the computational efficiency of the bandpass filtering process makes it suitable for processing datasets of varying sizes. While the method is efficient for moderately sized datasets, optimizations such as parallel processing or alternative algorithms can further enhance performance for very large datasets. Overall, this technique plays a pivotal role in the comprehensive study of network traffic dynamics, contributing to a deeper understanding of meaningful patterns and supporting the overarching goals of the research.

6.5. Beaconsing Data Generation

To simulate beaconsing behavior, synthetic data was generated to replicate periodic URL access patterns. The logic behind this data generation involved creating intervals of uniform duration

6. Implementation

between successive accesses for each URL while varying the start times and values for individual URLs. Specifically, each URL's activity was designed to start at different time offsets, such as 5 minutes, 10 minutes, and so on, ensuring that their patterns did not overlap perfectly on the timeline. Additionally, unique y-axis values were assigned to each URL to simulate real-world variations in behavior.

This approach ensured that the generated dataset demonstrated beaconing-like periodicity without perfect alignment among different URLs. The resulting data served as a realistic foundation for subsequent analysis using FFT and autocorrelation.

6.6. Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) was applied to analyze the periodicity of the generated beaconing data. FFT transforms time-domain data into the frequency domain, allowing the identification of dominant frequencies within the dataset. The logic of FFT involves decomposing the time-series data into its constituent sinusoidal components, thereby revealing the frequency characteristics of the periodic intervals.

When applied to the generated beaconing data, the FFT was expected to highlight a spike at low frequencies. This spike corresponds to the periodic nature of the beaconing behavior, reflecting the presence of long-duration periodic patterns in the dataset. The FFT analysis provided critical insights into the underlying periodicity of the data and validated the uniform interval design implemented during data generation.

The analysis using FFT confirmed the successful implementation of beaconing data generation. The FFT revealed the periodic components. The FFT results were instrumental in identifying the dominant frequencies and periodic patterns within the dataset, providing a quantitative basis for understanding the beaconing behavior. By leveraging the FFT function, organizations can gain deeper insights into the temporal dynamics of network traffic and identify potential security threats more effectively.

6.7. Autocorrelation

The autocorrelation function was used to measure the self-similarity of the generated data over varying time lags. This method calculates the degree of correlation between the data and a shifted version of itself, quantifying how well the intervals align at different lags.

In this implementation, the autocorrelation of the beaconing data exhibited a high correlation at lag 0, representing perfect similarity. However, due to the intentional offset in the start times of the intervals for each URL, the autocorrelation values decreased rapidly as the lag increased. This ensured that no consistent repetitive patterns appeared, validating the staggered start times and the designed variability in the dataset.

The combined analysis using autocorrelation confirmed the successful implementation of beaconing data generation. The autocorrelation highlighted the lack of consistent repetition among URLs, reflecting the staggered and realistic nature of the generated data. This analysis provided valuable insights into the temporal dependencies and patterns within the dataset, enhancing the understanding of beaconing behavior.

6.8. Behavior Detection

In the final stage of the algorithm, behavior detection is performed to determine the relevance and significance of the URLs retained after the filtering process. This stage is critical for identifying potentially malicious or anomalous URLs. The process begins with establishing a threshold value, which is determined through a combination of extensive experimentation and leveraging past experiences.

The experimentation phase involves testing various threshold levels against historical data to evaluate their effectiveness in flagging suspicious activities without generating excessive false positives. Insights from previous network security incidents and the specific operational context of the network further refine this threshold. By integrating empirical data with historical knowledge, the threshold is calibrated to balance sensitivity and specificity. Once defined, this threshold becomes the standard against which URL behavior is measured. URLs exhibiting characteristics that surpass this threshold are flagged for further investigation, as they may indicate potential security threats or deviations from normal network behavior.

This behavior detection step transforms filtered data into actionable intelligence, enabling network administrators and security professionals to focus on the most critical and relevant threats. By filtering out noise and highlighting significant anomalies, this stage enhances the cybersecurity framework's overall effectiveness, ensuring the network remains secure against evolving threats.

6.9. Algorithm Output

The algorithm's output provides a targeted overview of URLs requiring detailed examination. Each URL is compared against a predefined threshold of 500, and those exceeding this threshold are flagged for potential concerns. Such URLs are prioritized for closer investigation due to their deviation from normal behavior, which could indicate possible security threats or unusual activity.

This alert system is for prioritizing high-risk URLs, allowing security analysts to concentrate on the most significant issues. By efficiently filtering out less critical data, the system improves threat detection accuracy and minimizes false positives. Analyzing flagged URLs can uncover hidden patterns or attack methods that might otherwise be missed. This proactive alert mechanism is integral to effective threat management, facilitating early detection and response to mitigate risks and safeguard the network against potential breaches.

The algorithm's ability to promptly identify and address potential threats demonstrates its effectiveness in reinforcing network security. By streamlining the detection process and focusing on high-priority URLs, the system enhances the organization's cybersecurity posture and resilience. The algorithm's output serves as a valuable tool for security analysts, providing actionable insights and enabling informed decision-making to protect the network against emerging threats.

6.10. Summary

In this chapter, we delved into the intricacies of preprocessing network activity data, focusing on the creation and utilization of a whitelist and the application of bandpass filtering to enhance data analysis. The primary learnings from this chapter include:

- **Whitelist Creation:**

- We defined a function to create a whitelist by filtering out URLs based on specified exclusion criteria. This function plays a crucial role in curating a subset of URLs for further analysis, ensuring that only relevant and trustworthy URLs are retained.
- We discussed the importance of excluding irrelevant or untrustworthy URLs to improve the accuracy and interpretability of subsequent analyses.
- Performance considerations were examined, highlighting the efficiency of the whitelist creation process and its scalability for larger datasets.

- **Bandpass Filtering:**

- A function for applying bandpass filtering to the power values of network activity data was introduced. This function isolates significant frequency components within a specified range, reducing noise and enhancing the clarity of the dataset.
- The role of bandpass filtering in refining the dataset and focusing on the most relevant data was emphasized, contributing to a more robust understanding of temporal patterns and behaviors within the network.
- We explored the performance implications of the filtering process and discussed potential optimizations for handling large datasets.

- **Functional Analysis:**

- Both functions were defined with detailed explanations of their parameters, functionality, and performance considerations. This structured approach ensures that the functions are well-documented and easy to understand for future use and modification.

- **Function to Calculate Autocorrelation:**

- Computes the autocorrelation of a time series of power values.
- Measures similarity between a signal and its lagged version over time.
- Useful for identifying patterns, periodicity, and trends in the data.
- Uses the `acf` method from the `statsmodels` library to compute autocorrelation values up to a specified lag.
- Helps detect time-dependent structures, periodicities, noise levels, and potential anomalies in the dataset.

- **Function to Calculate Fourier Transform:**

- Performs a Fourier Transform on the time series of power values.
- Decomposes the time-domain signal into its frequency components.
- Useful for analyzing periodic behaviors or oscillations in power data.
- Uses the `fft` method from the `scipy.fft` library to compute the discrete Fourier transform (DFT).
- Provides frequencies and corresponding amplitudes, highlighting dominant frequencies and their correlation with system events or behaviors.

6.11. Next Steps

Building on the foundational work presented in this chapter, the next steps involve:

- **Implementing Additional Preprocessing Techniques:**
 - Investigate and implement additional preprocessing techniques to further enhance data quality and relevance. This may include methods such as data normalization, anomaly detection, and more sophisticated filtering techniques.
- **Integrating the Preprocessed Data into Analytical Models:**
 - Utilize the preprocessed data in advanced analytical models to uncover deeper insights into network behavior. This could involve machine learning algorithms, statistical analyses, and other data mining techniques.
- **Evaluating and Validating the Methods:**
 - Perform rigorous evaluation and validation of the preprocessing methods to ensure their effectiveness and reliability. This includes testing the methods on different datasets and scenarios to assess their generalizability and robustness.
- **Automating the Preprocessing Pipeline:**
 - Develop an automated preprocessing pipeline that seamlessly integrates the whitelist creation and bandpass filtering functions. This will streamline the data preparation process, making it more efficient and scalable for real-time applications.
- **Documenting and Sharing Findings:**
 - Document the findings and methodologies in detail to facilitate knowledge sharing and reproducibility. This includes creating comprehensive reports, code documentation, and potentially publishing the results in academic journals or conferences.

By following these next steps, we can build upon the foundation established in this chapter, advancing our understanding and capabilities in network activity data analysis. This progression will not only enhance the accuracy and effectiveness of our analytical models but also contribute to the broader field of network security and behavior analysis.

7. Experiments

In this experimental chapter, the algorithm's capability to detect malicious data undergoes a rigorous inspection. Data from various days is collected to encompass various scenarios, ensuring a comprehensive evaluation of the algorithm's performance across different conditions. Once the data is gathered, it is processed through the algorithm to assess its ability to identify potentially harmful content. The primary focus of this chapter lies in evaluating the algorithm's effectiveness in detecting malicious content and its consistency over time. Special attention is given to the URLs flagged as suspicious by the algorithm, which are closely examined to gain deeper insights into their functionality and potential areas for enhancement. By scrutinizing these flagged URLs, the chapter aims to uncover patterns and behaviors that might indicate malicious activity, providing valuable feedback for refining the algorithm. This chapter comprehensively evaluates the algorithm's performance, utilizing real-world data to gauge its effectiveness and explore avenues for improvement. The findings from this analysis not only demonstrate the algorithm's current capabilities but also highlight opportunities for further development, ensuring its continued relevance and robustness in detecting evolving cyber threats. Through this detailed examination, the chapter aims to bolster the algorithm's ability to safeguard the network, contributing to the overall security infrastructure of the system.

7.1. Validation and Testing

To affirm the efficacy of beaconing detection, the methodology undergoes rigorous testing using diverse datasets, simulating a range of scenarios that reflect various web traffic patterns. This comprehensive validation process is undertaken to ensure that the algorithm operates reliably across different frequency ranges and adapts seamlessly to the dynamic nature of HTTP requests. By employing datasets that encompass a wide array of traffic behaviors—from normal browsing activities to more erratic patterns indicative of potential security threats—the testing aims to demonstrate the filter's robustness and versatility. Each dataset is crafted to mimic real-world conditions, providing a realistic context for evaluating the bandpass filter's performance. The results from these tests offer critical insights into the filter's ability to isolate relevant frequency components while effectively minimizing noise and irrelevant data.

Furthermore, this validation process helps in identifying any potential weaknesses or limitations of the bandpass filter, guiding subsequent refinements and optimizations. The ultimate goal is to ensure that the beaconing detection consistently enhances the accuracy and reliability of the data analysis, regardless of the variability in web traffic patterns. By confirming its adaptability and precision, this rigorous testing phase substantiates the filter's integral role in the overall methodology, cementing its contribution to the accurate detection and analysis of network behaviors.

7. Experiments

Validation Steps:

1. **Diverse Datasets:** The beaconing detection is subjected to rigorous testing using datasets that exhibit varying frequencies of HTTP requests, each embodying distinct traffic patterns. These datasets are carefully curated to represent a broad range of real-world web traffic scenarios, from sporadic and unpredictable requests to highly regular and predictable beaconing activity. By employing such a diverse set of datasets, the aim is to thoroughly evaluate the filter's adaptability and effectiveness. This comprehensive approach ensures that the filter can robustly identify beaconing activity amidst different traffic environments, including those with fluctuating request intervals, mixed legitimate traffic, and potential noise. Ultimately, this testing strategy is designed to refine the beaconing detection mechanism, enhancing its accuracy and reliability across a wide array of web traffic conditions, thereby improving its practical applicability in detecting malicious or anomalous behavior in varied network contexts.
2. **Performance Metrics:** To evaluate the method's performance, the methodology employs metrics and the preservation of relevant frequency components. These metrics serve as quantitative indicators, allowing for a thorough assessment of the filter's ability to discern and retain meaningful signal components while minimizing noise.
3. **Real-world Scenarios:** The beaconing technique is rigorously evaluated on historical datasets containing documented instances of diverse HTTP request patterns within Allianz Company's network. This real-world testing ensures that the filter can effectively handle the complexities and nuances inherent in actual network traffic scenarios, further validating its practical utility.

By subjecting the method to these comprehensive validation steps, the methodology aims to establish its reliability and robustness in handling a wide array of web traffic patterns. The results obtained from this testing process contribute to the confidence in the filter's performance, reinforcing its role as a valuable tool in the analysis of HTTP request patterns over time.

Figure 7.1 illustrates the results of the Fast Fourier Transform (FFT) analysis conducted on the beaconing data. The graph displays the frequency spectrum of the data, highlighting the dominant frequencies present in the dataset. The spike at low frequencies indicates the presence of periodic patterns within the data, confirming the successful generation of beaconing-like behavior. The FFT analysis provides valuable insights into the temporal dynamics of the dataset, revealing the underlying frequency components that characterize beaconing activity. By leveraging the FFT function, organizations can gain deeper insights into the periodic behaviors of network traffic, enabling more effective detection of malicious or anomalous activities.

Figure 7.2 illustrates the results of the autocorrelation analysis conducted on the beaconing data. The graph depicts the autocorrelation values at different lag intervals, showcasing the self-similarity of the dataset over time. The high correlation at lag 0 indicates perfect similarity, while the rapid decrease in correlation values at subsequent lags reflects the intentional variability in the start times of the intervals for each URL. This analysis validates the staggered start times and realistic nature of the generated data, providing valuable insights into the temporal dependencies and patterns within the dataset. The autocorrelation function serves as a

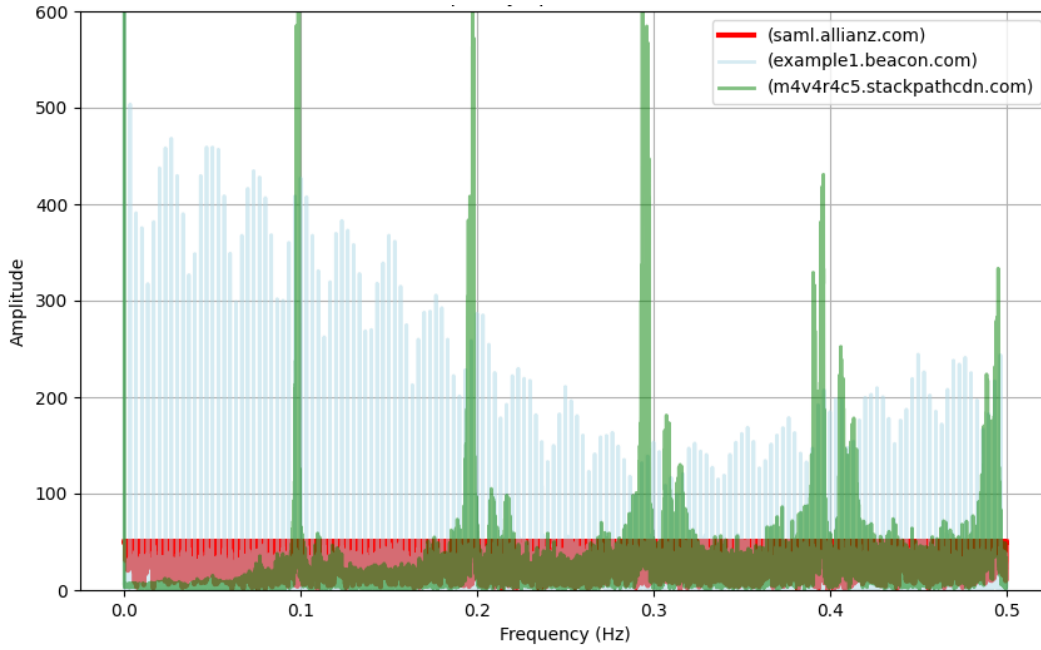


Figure 7.1.: FFT

powerful tool for detecting time-dependent structures, periodicities, noise levels, and potential anomalies in the data, enhancing the understanding of beaconing behavior and network dynamics. In this broad-lag plot, the orange line (`example1.beacon.com`) stands out with a much higher autocorrelation than the other two signals over a very large range of lags. This indicates that the orange dataset has a long-lasting or slowly decaying correlation structure, remaining significantly correlated with itself at high lag values. Meanwhile, the blue (`saml.allianz.com`) and green (`m4v4r4c5.stackpathcdn.com`) lines are relatively small in comparison, suggesting that they lose their correlation more quickly or do not exhibit strong long-range patterns. Because of the large scale, details of the blue and green signals are overshadowed, but you can still observe their initial peaks near lag 0 and a subsequent decay in autocorrelation.

Figure 7.3 offers a mid-level look at the autocorrelation patterns. The green line continues to show strong, evenly spaced spikes, confirming a periodic structure that extends beyond very short lags. The orange line exhibits a few notable peaks, but they are neither as large nor as regularly spaced, implying a less consistent periodicity. The blue line remains relatively smooth, suggesting a mild or more complex pattern—perhaps an overlap of smaller periodic components or sporadic events. Overall, this figure helps clarify how the signals behave at intermediate time scales: one remains strongly periodic (green), another has weaker or irregular peaks (orange), and the third shows moderate oscillations (blue).

Figure 7.4 the green line (`m4v4r4c5.stackpathcdn.com`) becomes the most prominent, displaying strong, regularly spaced spikes. This behavior is typical of a highly periodic signal—suggesting repeated events at specific intervals. The blue line (`saml.allianz.com`) shows a moderate, wavelike pattern that oscillates and gradually diminishes, indicating some peri-

7. Experiments

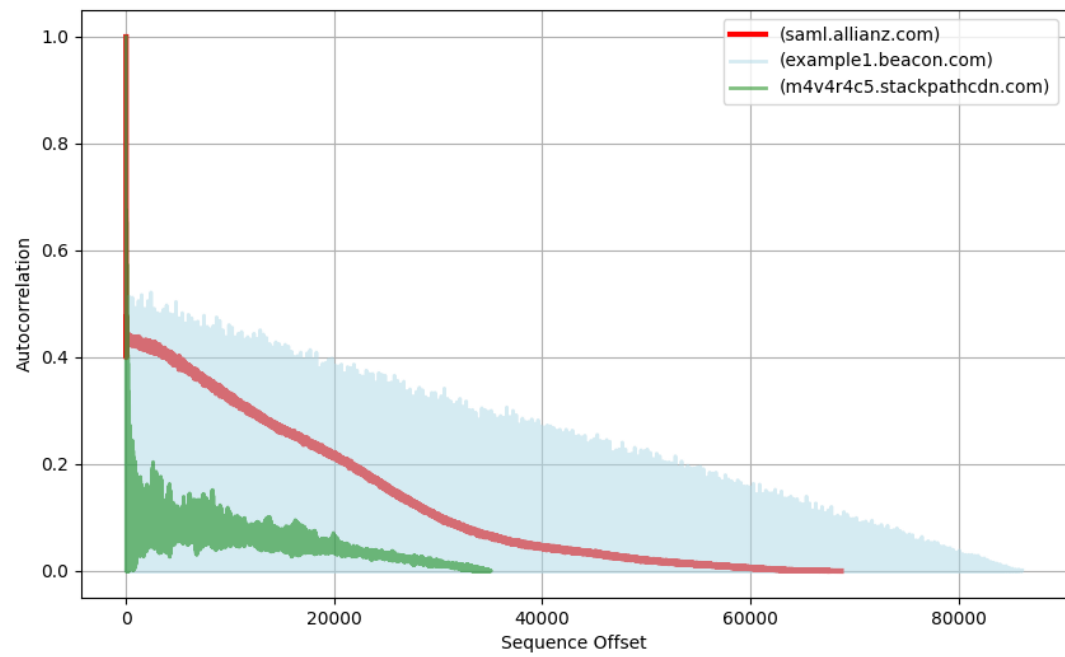


Figure 7.2.: Autocorrelation

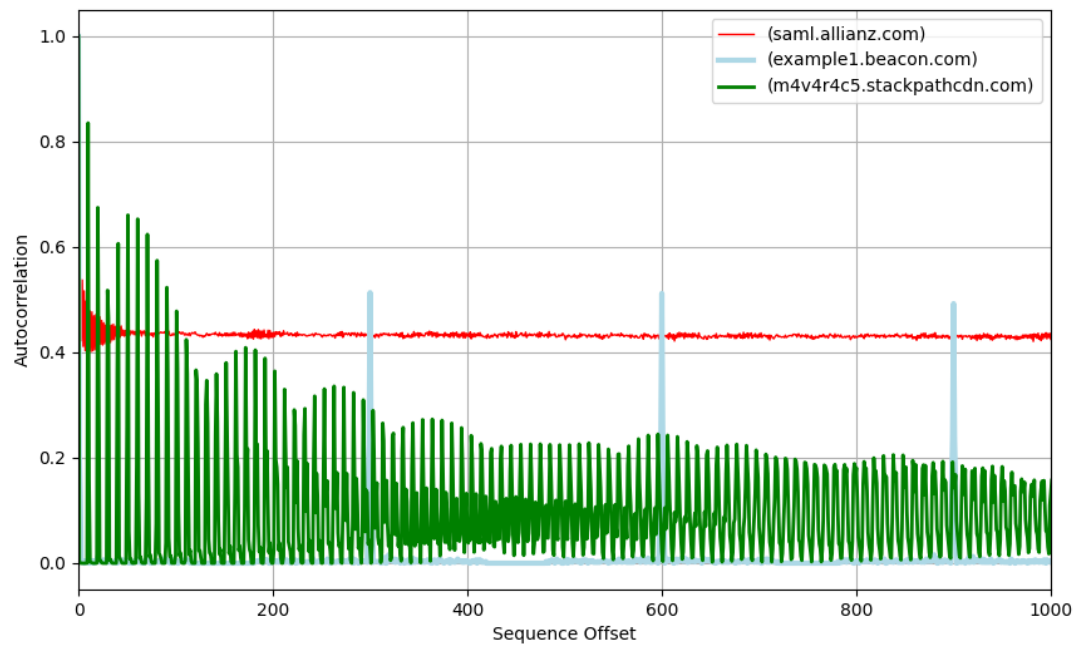


Figure 7.3.: Autocorrelation

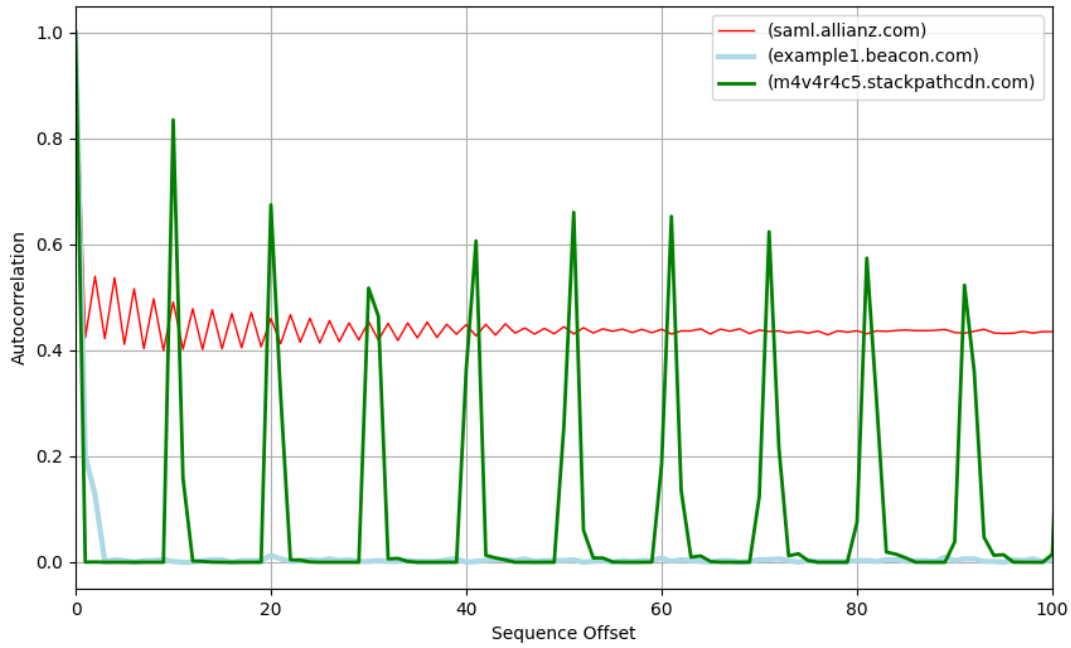


Figure 7.4.: Autocorrelation

odic or quasi-periodic behavior but not as pronounced. By contrast, the orange line (example1.beacon.com) remains relatively low in this short-lag region, suggesting fewer short-term repeated events or a more random short-range pattern. This zoomed-in view clearly exposes short-term periodicities that were hidden on the larger scale.

8. Results and Discussions

The implementation of the methodology within Allianz Company's network infrastructure represents a pivotal advancement in enhancing network security and resilience. This chapter provides a detailed discussion of how beaconing behavior can be effectively detected and the impact of periodicity in network communication on the detection of malicious behavior. The methodology's application involved several key steps, each contributing to the robustness of the network monitoring and security measures.

8.1. Detection of Beaconing Behavior

To address the question of how beaconing behavior can be effectively detected within Allianz Company's network, several strategies and methodologies were employed and evaluated:

8.1.1. Algorithm Development and Implementation

The core of the detection process involved the development and implementation of advanced algorithms tailored to identify beaconing behavior. These algorithms were designed to analyze network traffic for recurring patterns indicative of beaconing. Key methods included:

- **Pattern Recognition Algorithms:** These algorithms scan for regular intervals in network communication, a hallmark of beaconing activity often used by malware to maintain contact with a command-and-control server.
- **Threshold Analysis:** A critical component of the detection system involved setting thresholds for communication frequencies. URLs with communication intervals exceeding these thresholds were flagged for further investigation.

8.1.2. Data Collection and Preprocessing

Effective detection required comprehensive data collection and preprocessing:

- **Network Monitoring:** Continuous monitoring captured a wide array of network activities, including data packets, source and destination addresses, timestamps, and communication frequencies.
- **Filtering and Aggregation:** Known benign traffic was filtered out, and similar types of communication were aggregated to reduce noise and focus on potentially malicious activities.

8.1.3. Validation and Testing

To ensure the effectiveness of the detection methods:

- **Synthetic and Real-World Data:** The algorithm was tested on both synthetic datasets and real-world traffic from Allianz's network.
- **Integration with Existing Systems:** The detection mechanisms were integrated with Security Information and Event Management (SIEM) systems to enable automated alerts and responses, and incident response teams were notified for further investigation.

8.2. Impact of Periodicity in Network Communication

The second research question addresses the impact of periodicity in network communication on the detection of malicious behavior. Periodicity significantly affects detection capabilities, as detailed below:

8.2.1. Identification of Regular Intervals

- **Time-Series Analysis:** Network traffic was analyzed as time-series data to detect regular communication intervals. Techniques such as bandpass filtering was employed to identify periodic patterns.
- **Baseline Establishment:** A baseline of normal network behavior was established to identify deviations that might indicate malicious activity. Communication frequencies that deviated from this baseline were flagged as suspicious.

8.2.2. Differentiation Between Benign and Malicious Periodicity

- **Contextual Analysis:** Not all periodic communications are indicative of malicious behavior. Contextual analysis helped distinguish between normal periodic activities (e.g., scheduled updates) and potentially harmful beaconing.
- **Anomaly Detection Algorithms:** Algorithms trained on periodicity patterns of normal traffic helped identify anomalies. Techniques such as clustering and classification were used to differentiate benign from malicious behavior.

8.2.3. Impact on False Positives and Negatives

- **Reduction of False Positives:** Accurate modeling of normal periodic patterns helped reduce the number of false positives, ensuring that alerts were actionable.
- **Handling False Negatives:** Sensitivity adjustments in detection algorithms ensured that subtle periodic patterns associated with stealthy beaconing were not missed, minimizing false negatives.

8.2.4. Case Studies and Empirical Evidence

- **Real-World Examples:** Analysis of real-world cases of beaconing behavior provided empirical evidence on the effectiveness of periodicity-based detection methods.
- **Continuous Learning and Adaptation:** The detection system was designed to adapt to evolving patterns of network traffic, ensuring ongoing effectiveness in identifying new and emerging threats.

The comprehensive evaluation of the methodology demonstrated its efficacy in handling diverse network traffic scenarios and real-world cybersecurity challenges. The methodology, which included advanced algorithms for detecting beaconing behavior, robust data preprocessing techniques, and the use of periodicity in network communication, proved effective in identifying and mitigating malicious activities. The integration of these methods with Allianz Company's existing security infrastructure highlighted the importance of continuous monitoring and proactive response strategies in maintaining network security. The promising results from this implementation provide a strong foundation for future research and further enhancement of network security measures.

9. Conclusion and Future Work

9.1. Conclusion

The thorough study of the dataset provided valuable insights into network security, particularly in the detection of beaconing activities that may indicate potential threats. This examination involved a deep dive into the data, encompassing its various aspects and collection methods, which laid a solid foundation for understanding how networks can detect and respond to suspicious signals effectively.

Systematic data collection, cleaning, and processing were important to ensuring the dataset's accuracy and reliability. The gathering of relevant information, removal of inconsistencies or errors, and careful preparation for analysis were steps that validated the conclusions drawn from the data. These steps ensured that the findings were both accurate and actionable.

A significant component of the methodology was the implementation of a "whitelist" mechanism alongside specialized filtering and analysis techniques. The whitelist, which consists of trusted entities or activities within the network, plays an important role in focusing attention on potentially harmful signals while minimizing the impact of irrelevant noise. This strategic filtering enhanced the network's ability to detect threats more effectively by isolating and addressing only the potentially malicious activities.

The evaluation of time intervals between actions, combined with data enhancement techniques and specialized filtering methods, yielded valuable insights into potential malicious beaconing activity. These analyses illuminated underlying patterns and behaviors within the network, aiding in the identification of anomalies that could signify suspicious or harmful actions. The study highlighted the importance of proactive monitoring and response strategies in mitigating cybersecurity risks, demonstrating the broader significance of data-driven methodologies in fortifying network security.

In an era where organizations face increasingly sophisticated cyber threats, the insights from this research can guide strategic decision-making and resource allocation. By leveraging these findings, organizations can enhance their protection of critical digital assets and bolster their defenses against evolving threats.

9.2. Future Work

Building on the findings and methodology presented in this study, several promising avenues for future research and development can be explored:

1. **Enhanced Detection Algorithms:** Refining and optimizing beaconing detection algorithms can significantly improve accuracy and reduce false positives. Future research could explore novel approaches, such as deep learning techniques, which may provide

9. Conclusion and Future Work

new insights into anomaly detection in network behavior. These advancements could lead to more effective threat mitigation strategies and enhanced detection capabilities.

2. **Real-Time Monitoring Solutions:** Investigating real-time monitoring solutions can enable prompt detection and response to emerging threats, thereby minimizing potential damages caused by malicious activities. The development of automated response mechanisms could streamline incident response procedures, reducing the burden on cybersecurity personnel and enhancing the overall efficiency of threat management.
3. **Behavioral Analysis Across Diverse Networks:** Extending the study to analyze network behavior across a variety of organizational networks can reveal commonalities and variations in malicious activities. Understanding the unique challenges faced by different industries and sectors could lead to the development of tailored security measures that address specific threats more effectively. This cross-network analysis could provide valuable insights into how different environments respond to and manage cybersecurity risks.
4. **Integration with Machine Learning Techniques:** Exploring the integration of advanced machine learning techniques for anomaly detection can augment the capabilities of beaconing detection systems. Leveraging historical data and learning from past incidents can help these systems adapt to evolving cybersecurity threats, enhancing proactive defense mechanisms. Machine learning models can be trained to recognize subtle patterns and adapt to new types of attacks, improving overall detection and response.
5. **Collaborative Research Initiatives:** Engaging in collaborative research with industry partners and cybersecurity experts can foster innovation and the development of proactive security solutions. Joint research initiatives can facilitate the exchange of knowledge and resources, addressing complex cybersecurity challenges more effectively. Collaboration can lead to the creation of comprehensive solutions that benefit from diverse expertise and perspectives, driving progress in the field of network security.

Pursuing these avenues for future research and development will help advance the field of cybersecurity, strengthening defenses against emerging threats and safeguarding the integrity of digital infrastructures. By continuing to innovate and adapt, the cybersecurity community can better protect critical assets and respond to the dynamic landscape of cyber threats.

A. Appendix

A.1. Algorithm Implementation

```
1 from influxdb.client import InfluxDBClient
2 from datetime import datetime
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 from scipy.signal import butter, filtfilt
6
7 # Function to calculate request power
8 def calculate_request_power(request_list):
9     power_dictionary = {}
10    last_date_time = request_list[0]["_time"]
11
12    for request_dict in request_list:
13        current_date_time = request_dict["_time"]
14
15        # Check if current_date_time is a string, and convert it to a
16        # datetime object
17        if isinstance(current_date_time, str):
18            current_date_time = datetime.strptime(current_date_time, "%Y-%m-%
19            dT%H:%M:%S.%fZ")
20
21        time_delta = int((current_date_time - last_date_time).total_seconds()
22        )
23
24        # Add the power to the power dictionary
25        power_dictionary[time_delta] = power_dictionary.get(time_delta, 0) +
26        1
27        last_date_time = current_date_time
28
29        # Sort the dictionary for better visualization
30        power_dictionary = dict(sorted(power_dictionary.items()))
31
32    return power_dictionary
33
34 # Function to apply bandpass filtering in terms of time
35 def bandpass_filter(data, lowcut_time, highcut_time, sampling_rate, order=4):
36     nyquist = 0.5 * sampling_rate
37     lowcut = lowcut_time / nyquist
38     highcut = highcut_time / nyquist
39
40     if lowcut >= 1 or highcut >= 1:
41         raise ValueError("Digital filter critical frequencies must be 0 < Wn
42         < 1")
```

A. Appendix

```
39     b, a = butter(order, [lowcut, highcut], btype='band')
40     filtered_data = filtfilt(b, a, data)
41     return filtered_data
42
43     # InfluxDB connection details
44     url = "http://localhost:8086"
45     token = "*****"
46     org = "Student"
47     bucket = "Net"
48     influx_username = '*****'
49     influx_password = '*****'
50
51     try:
52         # Create an InfluxDB client
53         client = InfluxDBClient(url=url, token=token, org=org)
54
55         # Query data from InfluxDB
56         query = f'from(bucket:"{bucket}") -> range(start: 2023-08-01T00:00:00Z,
57             stop: 2023-08-02T00:00:00Z)'
58         tables = client.query_api().query(query, org=org)
59
60         # Extract points from the result
61         points = [record.values for table in tables for record in table.records]
62
63         # Process and organize the InfluxDB data
64         print("Processing InfluxDB Data:")
65         extracted_influx_objects = {}
66
67         for point in points:
68             url_hostname = point.get("url_hostname")
69
70             # Check if url_hostname is already in the dictionary
71             if url_hostname not in extracted_influx_objects:
72                 extracted_influx_objects[url_hostname] = []
73
74             # Append under the corresponding url_hostname
75             extracted_influx_objects[url_hostname].append(point)
76
77             # Print the extracted InfluxDB data for debugging
78             print(point)
79
80         # Create a whitelist to filter out unwanted URLs
81         def create_whitelist(data, exclusion_criteria):
82             whitelist = {url: requests for url, requests in data.items() if not
83                 any(exclusion in url for exclusion in exclusion_criteria)}
84             return whitelist
85
86         # Define exclusion criteria for URLs
87         exclusion_criteria = ['allianz', 'res']
88
89         # Create a whitelist based on the exclusion criteria
90         whitelist = create_whitelist(extracted_influx_objects, exclusion_criteria
91             )
```

```

90 print("Filtered URLs based on whitelist:")
91 for url_hostname in whitelist:
92     print(url_hostname)
93
94 # Create a table of power for each URL hostname with bandpass filtering
   in terms of time
95 print("nPower Table with Bandpass Filtering in Terms of Time:")
96 for url_hostname, requests in whitelist.items():
97     print(f"URL Hostname: {url_hostname}")
98     power_dictionary = calculate_request_power(requests)
99
100 # Print the power dictionary for debugging
101 print("Power Dictionary:", power_dictionary)
102
103 # Extract keys and values from the power dictionary
104 time_intervals = list(power_dictionary.keys())
105 power_values = list(power_dictionary.values())
106
107 # Apply bandpass filtering in terms of time
108 lowcut_time = 5 # 5 seconds
109 highcut_time = 1000 # 1000 seconds
110
111 # Check if there are enough elements to calculate sampling rate
112 if len(time_intervals) >= 2:
113     sampling_rate = 1.0 / (time_intervals[1] - time_intervals[0]) #
        Sampling frequency
114
115     try:
116         filtered_power_values = bandpass_filter(power_values, lowcut_
            time, highcut_time, sampling_rate)
117     except ValueError as e:
118         print(f"Error: {e}")
119         filtered_power_values = power_values
120
121 # Print the filtered power values for debugging
122 print("Filtered Power Values:", filtered_power_values)
123
124 # Calculate the average power
125 average_power = sum(filtered_power_values) / len(filtered_power_
    values)
126
127 # Print the average power for debugging
128 print("Average Power:", average_power)
129
130 # Subtract average power from all power values
131 adjusted_power_values = [power - average_power for power in
    filtered_power_values]
132
133 # Print the adjusted power values for debugging
134 print("Adjusted Power Values:", adjusted_power_values)
135
136 # Remove negative powers
137 non_negative_power_values = [max(0, power) for power in adjusted_
    power_values]

```

A. Appendix

```
138
139     # Get indices for the time range of interest (5 to 1000 seconds)
140     time_range_indices = [i for i, t in enumerate(time_intervals) if
141                           5 <= t <= 1000]
142
143     # Print the time range indices for debugging
144     print("Time Range Indices:", time_range_indices)
145
146     # Plot the adjusted data within the specified time range
147     plt.plot([time_intervals[i] for i in time_range_indices], [non_
148                   negative_power_values[i] for i in time_range_indices], label=
149                   url_hostname)
150
151     # Check if there are multiple URLs in the whitelist before creating
152     # legend
153     if len(whitelist) > 1:
154         plt.legend()
155
156     plt.xlabel("Time Interval (seconds)") # Change x-axis label
157     plt.ylabel("Adjusted Power") # Change y-axis label
158     plt.title("Adjusted Power over Time") # Change the chart title
159     plt.show()
160
161 except Exception as e:
162     print(f"An error occurred: {e}")
```

A.2. Data Analysis Implementation

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 from scipy.signal import butter, filtfilt
4
5 # Function to calculate request occurrence
6 def calculate_request_occurrence(request_list):
7     occurrence_dictionary = {}
8     last_date_time = None
9
10    for _, request_row in request_list.iterrows():
11        try:
12            current_date_time = pd.to_datetime(request_row["time"])
13
14            if last_date_time is not None:
15                time_delta = int((current_date_time - last_date_time).total_
16                                seconds())
17
18                # Add the occurrence to the occurrence dictionary
19                occurrence_dictionary[time_delta] = occurrence_dictionary.get
20                (time_delta, 0) + 1
21
22            last_date_time = current_date_time
```



```

22     except (ValueError, TypeError) as e:
23         print(f"Error in row: {-}, Timestamp value: {request_row['_time']}
24             }, Error message: {e}")
25
26     # Sort the dictionary for better visualization
27     occurrence_dictionary = dict(sorted(occurrence_dictionary.items()))
28
29     return occurrence_dictionary
30
31 # Function to apply bandpass filtering in terms of time
32 def bandpass_filter(data, lowcut_time, highcut_time, sampling_rate, order=4):
33     nyquist = 0.5 * sampling_rate
34     lowcut = lowcut_time / nyquist
35     highcut = highcut_time / nyquist
36
37     if lowcut >= 1 or highcut >= 1:
38         raise ValueError("Digital filter critical frequencies must be 0 < Wn
39             < 1")
40
41     b, a = butter(order, [lowcut, highcut], btype='band')
42     filtered_data = filtfilt(b, a, data)
43     return filtered_data
44
45 # CSV file path for the cleaned and modified data
46 csv_file_path = r'C:\Allianz\4\1125\Modified_Beaconing.csv'
47
48 try:
49     # Read data from CSV file and explicitly convert "_time" to datetime
50     df = pd.read_csv(csv_file_path)
51     df["_time"] = pd.to_datetime(df["_time"], format='%H:%M:%S.%f', errors='
52         coerce')
53
54     # Process and organize the CSV data
55     print("Processing CSV Data:")
56     extracted_csv_objects = {}
57
58     for _, row in df.iterrows():
59         url_hostname = row.get("url_hostname")
60
61         # Check if url_hostname is already in the dictionary
62         if url_hostname not in extracted_csv_objects:
63             extracted_csv_objects[url_hostname] = []
64
65         # Append under the corresponding url_hostname
66         extracted_csv_objects[url_hostname].append(row)
67
68         # Print the extracted CSV data for debugging
69         print(row)
70
71     # Create a whitelist to filter out unwanted URLs
72     def create_whitelist(data, exclusion_criteria):
73         whitelist = {url: requests for url, requests in data.items() if not
74             any(exclusion in url for exclusion in exclusion_criteria)}
75         return whitelist

```

A. Appendix

```
72
73 # Define exclusion criteria for URLs
74 exclusion_criteria = ['allianz', 'res']
75
76 # Create a whitelist based on the exclusion criteria
77 whitelist = create_whitelist(extracted_csv_objects, exclusion_criteria)
78
79 print("Filtered URLs based on whitelist:")
80 for url_hostname in whitelist:
81     print(url_hostname)
82
83 # Create a table of occurrence for each URL hostname with bandpass
84   filtering in terms of time
85 print("nOccurrence Table with Bandpass Filtering in Terms of Time:")
86 peak_url_hostname = None
87 peak_occurrence_value = 0
88 for url_hostname, requests in whitelist.items():
89     print(f"URL Hostname: {url_hostname}")
90     occurrence_dictionary = calculate_request_occurrence(pd.DataFrame(
91         requests))
92
93     # Print the occurrence dictionary for debugging
94     print("Occurrence Dictionary:", occurrence_dictionary)
95
96     # Identify peak occurrence value and corresponding URL hostname
97     if occurrence_dictionary:
98         max_occurrence_value = max(occurrence_dictionary.values())
99         if max_occurrence_value > peak_occurrence_value:
100             peak_occurrence_value = max_occurrence_value
101             peak_url_hostname = url_hostname
102
103     # Extract keys and values from the occurrence dictionary
104     time_intervals = list(occurrence_dictionary.keys())
105     occurrence_values = list(occurrence_dictionary.values())
106
107     # Apply bandpass filtering in terms of time
108     lowcut_time = 5 # 5 seconds
109     highcut_time = 1000 # 1000 seconds
110
111     # Check if there are enough elements to calculate sampling rate
112     if len(time_intervals) >= 2:
113         sampling_rate = 1.0 / (time_intervals[1] - time_intervals[0]) #
114           Sampling frequency
115
116         try:
117             filtered_occurrence_values = bandpass_filter(occurrence_
118                 values, lowcut_time, highcut_time, sampling_rate)
119         except ValueError as e:
120             print(f"Error: {e}")
121             filtered_occurrence_values = occurrence_values
122
123     # Print the filtered occurrence values for debugging
124     print("Filtered Occurrence Values:", filtered_occurrence_values)
```

```

122     # Calculate the average occurrence
123     average_occurrence = sum(filtered_occurrence_values) / len(
        filtered_occurrence_values)
124
125     # Print the average occurrence for debugging
126     print("Average Occurrence:", average_occurrence)
127
128     # Subtract average occurrence from all occurrence values
129     adjusted_occurrence_values = [occurrence - average_occurrence for
        occurrence in filtered_occurrence_values]
130
131     # Print the adjusted occurrence values for debugging
132     print("Adjusted Occurrence Values:", adjusted_occurrence_values)
133
134     # Remove negative occurrences
135     non_negative_occurrence_values = [max(0, occurrence) for
        occurrence in adjusted_occurrence_values]
136
137     # Get indices for the time range of interest (5 to 1000 seconds)
138     time_range_indices = [i for i, t in enumerate(time_intervals) if
        5 <= t <= 1000]
139
140     # Print the time range indices for debugging
141     print("Time Range Indices:", time_range_indices)
142
143     # Plot the adjusted data within the specified time range
144     plt.plot([time_intervals[i] for i in time_range_indices], [non-
        negative_occurrence_values[i] for i in time_range_indices],
        label=url_hostname)
145
146     # Print the URL hostname with the peak occurrence
147     if peak_url_hostname:
148         print(f"nURL Hostname with Peak Occurrence: {peak_url_hostname}")
149         print(f"Peak Occurrence Value: {peak_occurrence_value}")
150
151     # Check if there are multiple URLs in the whitelist before creating
        legend
152     if len(whitelist) > 1:
153         plt.legend()
154
155     plt.xlabel("Time Interval (seconds)") # Change x-axis label
156     plt.ylabel("Adjusted Occurrence") # Change y-axis label
157     plt.title("Adjusted Occurrence over Time") # Change the chart title
158     plt.show()
159
160 except Exception as e:
161     print(f"An error occurred: {e}")

```


Bibliography

- [1] Bitdefender, “Global cybersecurity threat map,” accessed: 2024-08-13. [Online]. Available: <https://threatmap.bitdefender.com/>
- [2] InfluxData, “Influxdb 3.0 system architecture,” accessed: 2024-08-13. [Online]. Available: <https://www.influxdata.com/blog/influxdb-3-0-system-architecture/>
- [3] X. Hu, J. Jang, M. P. Stoecklin, T. Wang, D. L. Schales, D. Kirat, and J. R. Rao, “Baywatch: robust beaconing detection to identify infected hosts in large-scale enterprise networks,” in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 479–490.
- [4] Y. Zhang, H. Dong, A. Nottingham, M. Buchanan, D. E. Brown, and Y. Sun, “Global analysis with aggregation-based beaconing detection across large campus networks,” in *ACSAC '23: Proceedings of the 39th Annual Computer Security Applications Conference*, 2023, pp. 565–579.
- [5] G. Apruzzese, M. Marchetti, M. Colajanni, G. G. Zoccoli, and A. Guido, “Identifying malicious hosts involved in periodic communications,” in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2017, pp. 1–8.
- [6] J. Seo and S. Lee, “Abnormal behavior detection to identify infected systems using the apchain algorithm and behavioral profiling,” *Security and Communication Networks*, vol. 2018, no. 1, p. 9706706, 2018.
- [7] N. A. Huynh, W. K. Ng, A. Ulmer, and J. Kohlhammer, “Uncovering periodic network signals of cyber attacks,” in *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*. IEEE, 2016, pp. 1–8.
- [8] J. Jang, D. H. Kirat, B. J. Kwon, D. L. Schales, and M. P. Stoecklin, “Detecting malicious beaconing communities using lockstep detection and co-occurrence graph,” Jan. 5 2021, uS Patent 10,887,323.
- [9] M. A. Talib, Q. Nasir, A. B. Nassif, T. Mokhamed, N. Ahmed, and B. Mahfood, “Apt beaconing detection: A systematic review,” *Computers & Security*, vol. 122, p. 102875, 2022.
- [10] P. S. Charan, P. M. Anand, and S. K. Shukla, “Dmapt: Study of data mining and machine learning techniques in advanced persistent threat attribution and detection,” *Data Mining-Concepts and Applications*, p. 63, 2021.

Bibliography

- [11] M. Hagan, B. Kang, K. McLaughlin, and S. Sezer, "Peer based tracking using multi-tuple indexing for network traffic analysis and malware detection," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018, pp. 1–5.
- [12] A. Shalaginov, K. Franke, and X. Huang, "Malware beaconing detection by mining large-scale dns logs for targeted attack identification," *International Journal of Computer and Systems Engineering*, vol. 10, no. 4, pp. 743–755, 2016.
- [13] Y.-R. Yeh, T. C. Tu, M.-K. Sun, S. M. Pi, and C.-Y. Huang, "A malware beacon of botnet by local periodic communication behavior," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2. IEEE, 2018, pp. 653–657.
- [14] Y. Borchani, "Advanced malicious beaconing detection through ai," *Network Security*, vol. 2020, no. 3, pp. 8–14, 2020.
- [15] M. A. Enright, E. Hammad, and A. Dutta, "A learning-based zero-trust architecture for 6g and future networks," in *2022 IEEE Future Networks World Forum (FNWF)*. IEEE, 2022, pp. 64–71.
- [16] T. Van Ede, H. Aghakhani, N. Spahn, R. Bortolameotti, M. Cova, A. Continella, M. van Steen, A. Peter, C. Kruegel, and G. Vigna, "Deepcase: Semi-supervised contextual analysis of security events," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 522–539.
- [17] T. Ongun, O. Spohngellert, B. Miller, S. Boboila, A. Oprea, T. Eliassi-Rad, J. Hiser, A. Nottingham, J. Davidson, and M. Veeraraghavan, "Portfiler: port-level network profiling for self-propagating malware detection," in *2021 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2021, pp. 182–190.
- [18] W. Niu, X. Zhang, X. Zhang, X. Du, X. Huang, M. Guizani *et al.*, "Malware on internet of uavs detection combining string matching and fourier transformation," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9905–9919, 2020.
- [19] J. Duan, Z. Zeng, A. Oprea, and S. Vasudevan, "Automated generation and selection of interpretable features for enterprise security," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1258–1265.
- [20] M. Haffey, M. Arlitt, and C. Williamson, "Modeling, analysis, and characterization of periodic traffic on a campus edge network," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MAS-COTS)*. IEEE, 2018, pp. 170–182.
- [21] A. Oprea, Z. Li, R. Norris, and K. Bowers, "Made: Security analytics for enterprise threat detection," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 124–136.
- [22] M. Ukrop, L. Kraus, V. Matyas, and H. A. M. Wahsheh, "Will you trust this tls certificate? perceptions of people working in it," in *Proceedings of the 35th annual computer security applications conference*, 2019, pp. 718–731.

- [23] T. Vissers, J. Spooren, P. Agten, D. Jumpertz, P. Janssen, M. Van Wesemael, F. Piessens, W. Joosen, and L. Desmet, “Exploring the ecosystem of malicious domain registrations in the. eu tld,” in *Research in Attacks, Intrusions, and Defenses: 20th International Symposium, RAID 2017, Atlanta, GA, USA, September 18–20, 2017, Proceedings*. Springer, 2017, pp. 472–493.