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: 01. Sep 2024

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 bösartigem 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

In today's rapidly evolving cybersecurity landscape, Allianz Company faces significant threats from advanced persistent threats (APTs), which are sophisticated, stealthy attacks that target large organizations over extended periods. This master thesis focuses on the critical need to strengthen Allianz Company's network security by developing a proactive defense mechanism against these persistent threats. Central to this research is the analysis of the Allianz company user log dataset, which includes essential data such as URLs, hostnames, timestamps, and IP addresses. Currently, threat detection occurs once daily, highlighting the urgency for a more adaptive and responsive defense strategy to keep pace with the evolving nature of APTs.

The research builds on key references that delve into APT characteristics, particularly beaconing behavior, a crucial aspect of APT operations that allows attackers to maintain covert communication. The thesis addresses the challenges faced by large-scale enterprise networks in detecting and mitigating APTs, emphasizing the need for periodicity detection and behavior analysis. These strategies are vital for identifying patterns that indicate APT activity, which often blends in with normal network traffic.

The proposed methodology involves several critical steps: data extraction and preparation, time interval analysis, average power calculation, band-pass filtering, and behavior detection. These components are designed to work together to identify and respond effectively to potential threats within Allianz Company's network. To ensure the accuracy and reliability of this approach, the methodology is validated using simulated datasets and historical data with known security incidents.

This research draws upon six key references that provide a strong foundation for detecting malicious network behavior. These references cover beaconing detection, peer-based tracking, systematic reviews, large-scale DNS log mining, and advanced techniques involving artificial intelligence. The integration of these insights into the proposed framework aims to enhance network security by offering a real-time defense mechanism capable of countering persistent and evolving cybersecurity threats.

In summary, this master thesis presents a comprehensive and systematic exploration of methods to detect and mitigate APTs within Allianz Company's network. By developing a structured and validated methodology informed by foundational research and diverse perspectives, the thesis seeks to significantly improve the organization's ability to defend against sophisticated and persistent cyber threats. This work contributes not only to the academic understanding of APT detection but also offers practical solutions to enhance the security of large enterprises in an increasingly complex threat environment.

Contents

1. Topical Overview

1.1. Problem Statement

In today's digital age, protecting sensitive data and ensuring the integrity of network systems are top priorities for organizations. Like many others, Allianz Company produces vast user log data daily. This data contains critical information and is continuously monitored to detect any potential security threats. The increasing complexity and sophistication of cyber-attacks, especially advanced persistent threats (APTs), highlight the need for strong and proactive cybersecurity measures. The main challenge is effectively sifting through this extensive log data to identify signs of malicious behavior and ensure that threats are detected and mitigated quickly. This research aims to improve the company's cybersecurity framework, focusing on early detection of APTs and other potential threats to protect the network infrastructure.

1.2. 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.

1.2.1. 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?

1.3. 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 essential 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 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 BAYWATCH 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 essential 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.

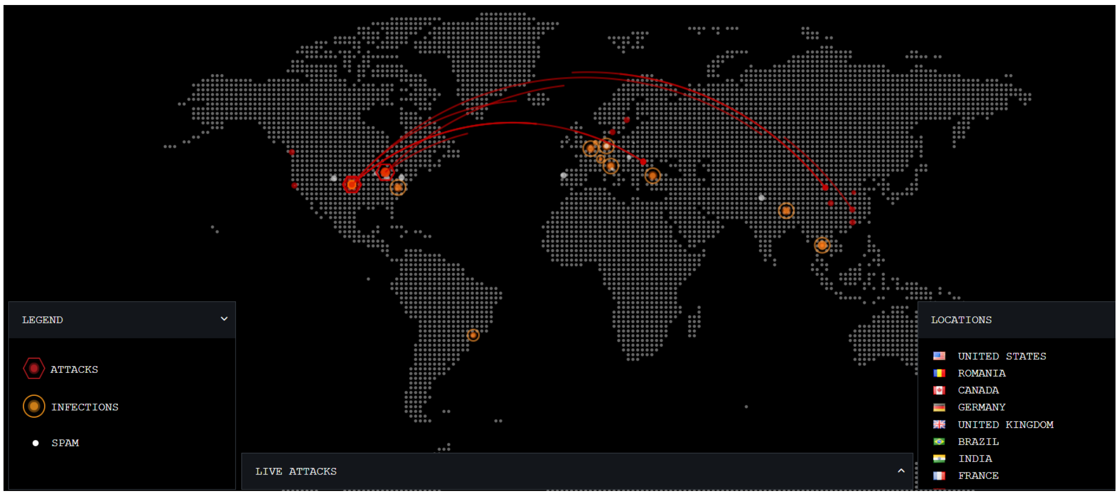


Figure 2.1.: Global cybersecurity threat map [1]

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.

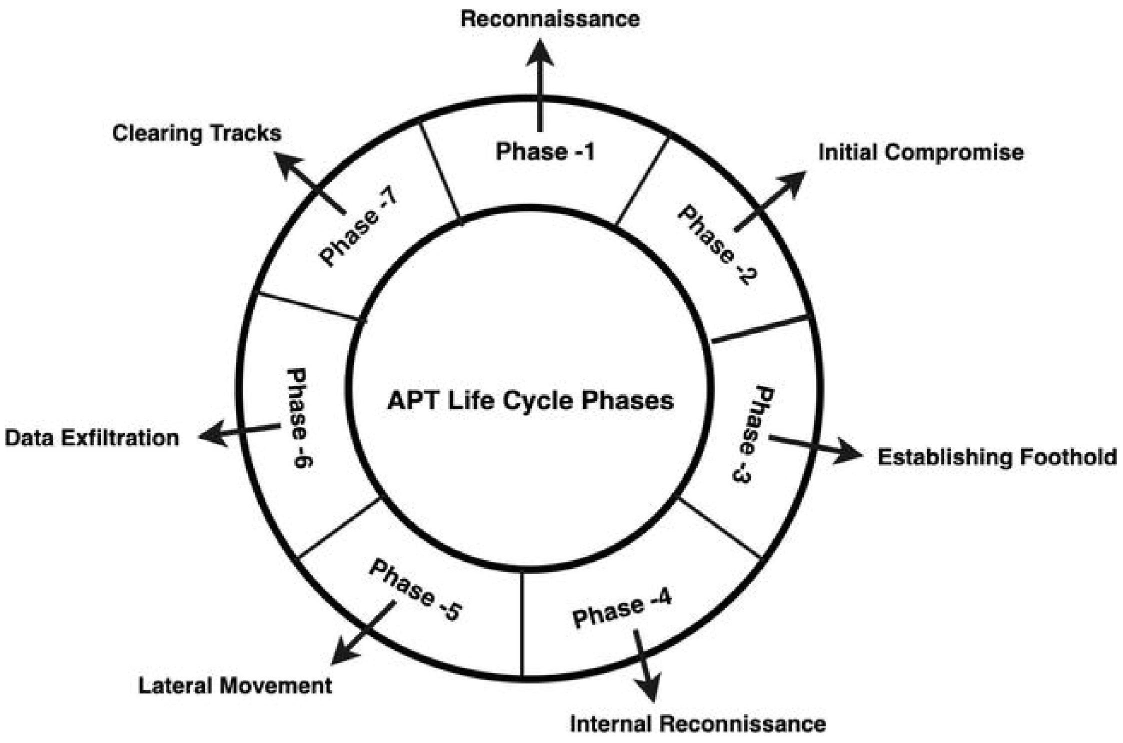


Figure 2.2.: APT attack lifecycle [2]

- **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 essential for developing effective cybersecurity strategies.

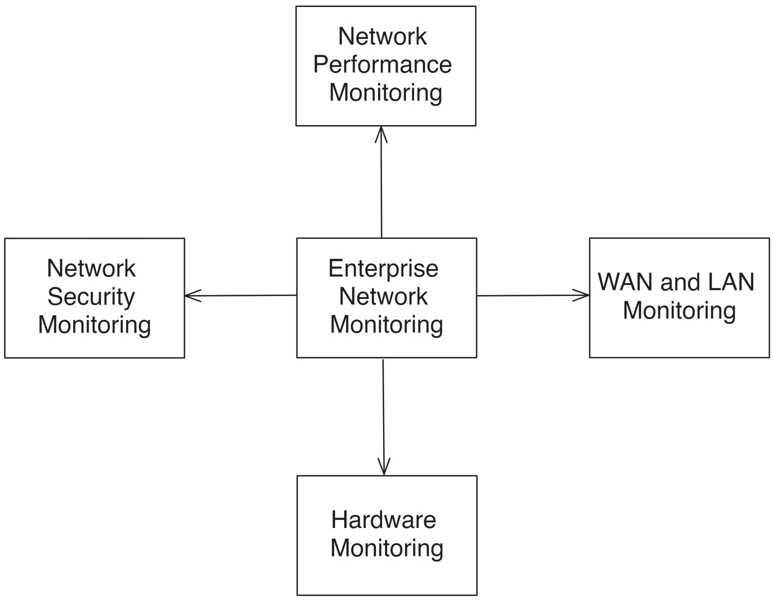

Network Performance Monitoring
Network Security Monitoring
Enterprise Network Monitoring
WAN and LAN Monitoring
Hardware Monitoring

Figure 2.3.: Enterprise network diagram

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. 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.

2.4.1. 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.

2.5. 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.

2.5.1. 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 essential for trend analysis and anomaly detection.

2.5.2. 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.

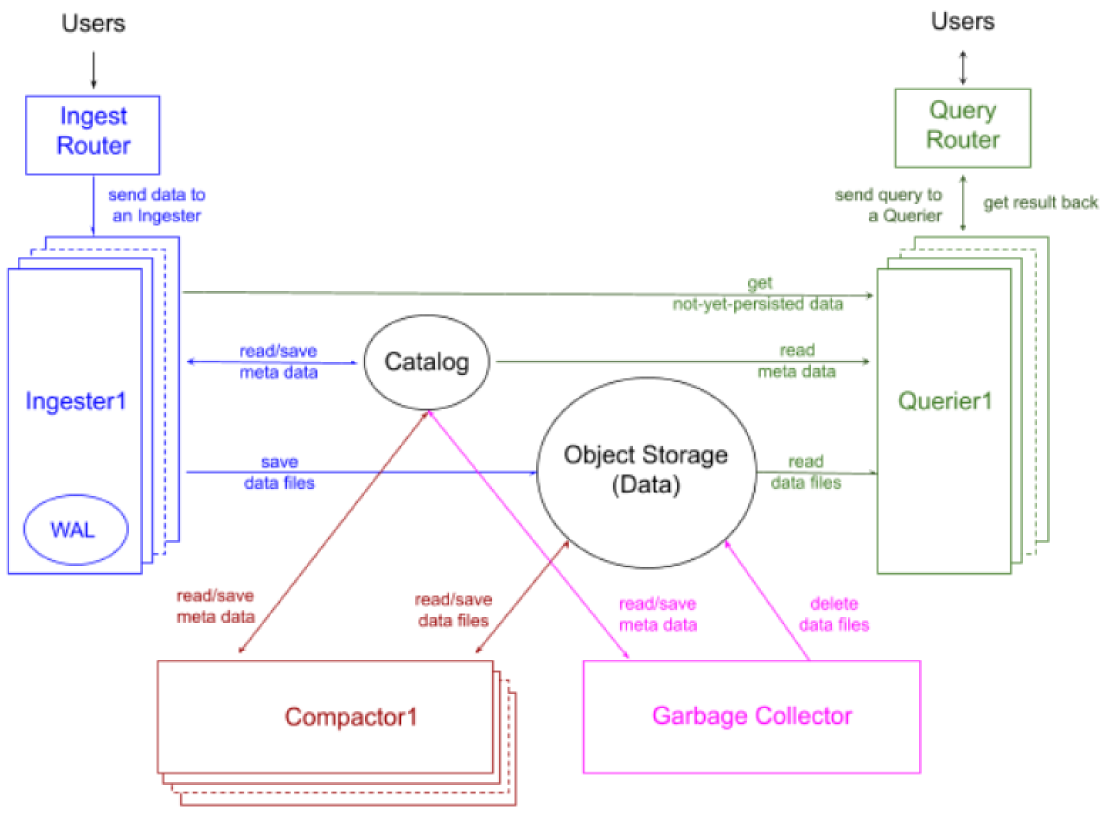


Figure 2.4.: InfluxDB Architecture [3]

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

2.6. 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 essential for understanding the subsequent chapters, which will delve deeper into related work, methodology, 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

In the field of network security, detecting beaconing behavior is a critical task for identifying compromised hosts, especially in large-scale enterprise environments. The study by Hu et al. (2016), *BAYWATCH: Robust Beaconing Detection to Identify Infected Hosts in Large-Scale Enterprise Networks*, addresses this challenge by proposing a robust detection methodology for identifying stealthy beaconing activity. Beaconing is commonly used by malware to establish communication with command and control servers, but it often resembles legitimate network traffic, making detection difficult. The authors introduce an 8-step filtering process that refines and isolates malicious beaconing traffic from legitimate sources. This method is evaluated using a large-scale dataset of over 30 billion events collected over five months from a corporate network with more than 130,000 devices. The results demonstrate that BAYWATCH significantly outperforms traditional security mechanisms in identifying malicious beaconing, which often goes undetected by conventional methods. This work contributes to improving network traffic analysis and provides a foundation for more effective detection of advanced persistent threats (APT) in large, complex networks [4].

The paper *Global Analysis with Aggregation-based Beaconing Detection across Large Campus Networks* presents an advanced approach to detecting beaconing activities in large-scale campus networks. The system introduced aggregates signals across multiple traffic protocols and networks to uncover hidden periodicity, which is often a characteristic of beaconing behaviors. This method utilizes a time-series analysis algorithm to identify periodicity and a ranking-based detection pipeline enhanced by self-training and active-learning techniques. Evaluation of the system using data from two large campus networks, spanning over 75 billion connections over ten months, showed significant improvements. By aggregating signals across multiple networks, the system detected 43% more periodic domains than single-network analysis. Furthermore, it identified 1,387 unique malicious domains, 56% of which were previously unknown to major threat intelligence platforms like VirusTotal. This approach highlights the potential of leveraging aggregated network data to improve the detection of previously undetected threats and is particularly relevant for large organizations and academic environments that deal with massive network traffic [5].

The paper *Identifying Malicious Hosts Involved in Periodic Communications* proposes a novel method for detecting malicious network activities in large-scale networks by focusing on periodic communications. Traditional Network Intrusion Detection Systems (NIDS) often struggle to detect malware-related traffic, as it may not always trigger alerts. This research highlights that periodic communication patterns, even when they occur at varying intervals, are often linked to malicious behavior. The proposed method examines network traffic over extended time windows and identifies external hosts that are likely involved in malicious operations, even if these activities do not trigger immediate alerts. The authors demonstrate that this approach can filter out normal traffic and identify a small subset of suspicious hosts from large amounts of network data. The method integrates well with existing NIDS, helping security teams focus on the most probable threats by providing a manageable list of suspicious hosts [6].

The research presented in *Abnormal Behavior Detection to Identify Infected Systems Using the APChain Algorithm and Behavioral Profiling* focuses on detecting infected hosts in network environments by analyzing abnormal behavior patterns. The APChain algorithm is utilized to track network traffic attributes over time, forming a chain that links these attributes to unusual communication patterns. This chain is used to identify behaviors associated with hosts infected by malicious software, such as establishing communication with Command and Control (C2) servers. The system monitors real-time network traffic, extracts relevant data such as IP addresses, ports, protocols, and timestamps, and profiles the behavior of networked systems based on this information. Suspicious patterns are flagged as potential indicators of infection, facilitating early detection and response. This methodology is further enhanced by eliminating false positives through whitelist-based filtering. The APChain algorithm's approach to behavioral profiling and its detailed traffic attribute analysis provides a robust means of identifying infected systems and preventing widespread network compromise [7].

The study titled *Periodic Behavior in Botnet Command and Control Channels Traffic* explores the detection of periodic patterns in botnet command-and-control (CC2) communications, which are crucial for identifying infected systems within a network. By analyzing the traffic data from botnets, this work demonstrates how to leverage the periodic nature of such communication to distinguish between benign and malicious behavior. The authors employ periodogram-based analysis techniques to detect significant periodic components in traffic patterns. The results show that the periodicity in botnet communications often exhibits distinct statistical properties that can be used as a signature for detection. This work emphasizes the importance of understanding periodic behavior in network traffic to enhance botnet detection strategies, particularly within large-scale systems like enterprise networks. The method also highlights the challenge of distinguishing periodic signals from random noise in network traffic, which requires careful analysis and thresholding techniques [8].

The paper *Uncovering Periodic Network Signals of Cyber Attacks* addresses the detection of malware through the periodic traces it leaves in network traffic. This research identifies that malware behavior often follows periodic patterns, which can be leveraged to detect malicious activity. The authors propose a visual analytics solution that automates detection while also supporting manual inspection of periodic signals. Their approach uses circular graphs and stacked histograms for visual verification, combined with deep packet inspection for detailed analysis. The method demonstrates effectiveness in identifying complex periodic behaviors, enhancing network security by recognizing hidden attack signals [9].

The study *Detecting Malicious Beaconing Communities Using Lockstep Detection and Co-occurrence Graph* introduces a two-phase process for identifying malicious beaconing signals. The first phase uses lockstep detection to recognize synchronized beaconing activities across multiple devices, indicating a coordinated attack. The second phase employs community detection using a co-occurrence graph, which analyzes the relationships between destination servers involved in beaconing to uncover anomalous patterns. This method helps reduce false positives and more accurately identifies malicious activities by detecting communities of compromised devices. The use of community detection based on graph analytics is crucial for distinguishing malicious from benign behaviors [10].

The paper *APT Beaconing Detection: A Systematic Review* provides a comprehensive analysis of advanced persistent threat (APT) detection techniques, with a particular focus on beaconing detection methods. The review emphasizes the increasing need for effective solutions to identify beaconing behavior, a key characteristic of APT attacks, which often serves as a sign of ongoing communication between the infected hosts and their command and control (C2) servers. The paper surveys a variety of detection techniques and classifies them based on their detection methods, strengths, and weaknesses. It also highlights the use of Artificial Intelligence (AI) algorithms, which have been increasingly integrated into APT detection systems, and assesses the datasets commonly used for training and testing these models. The study aims to provide an understanding of current APT detection strategies, while also identifying gaps and potential areas for future research, including more efficient and adaptive solutions for detecting beaconing in real-time environments [11].

Several studies have explored different techniques for detecting beaconing behavior, particularly in the context of network traffic analysis and malware detection. Hagan et al. (2020) propose a peer-based tracking method using multi-tuple indexing to enhance the detection of network traffic anomalies, such as malware beaconing. Their approach focuses on identifying patterns in network traffic that are indicative of malicious activities, making it easier to detect and mitigate potential threats in large-scale systems [12].

Shalaginov et al. (2019) focus on malware beaconing detection by mining large-scale DNS logs. By analyzing these logs, the authors identify targeted attack patterns, providing insights into how malware communicates with command and control servers through periodic signals [13].

Yeh et al. (2015) present a method for detecting botnet malware using local periodic communication behavior, which is a key feature of botnet beaconing. Their work highlights the importance of understanding periodic traffic to improve detection systems in network environments [14].

More recently, Borchani (2021) investigates the use of Artificial Intelligence (AI) techniques to detect malicious beaconing behaviors. This work shows that AI-based methods can significantly enhance the accuracy and efficiency of beaconing detection systems by analyzing network traffic in real time, thereby offering promising solutions for identifying sophisticated attack patterns like those used in advanced persistent threats (APT) [15].

4. Methodology

This chapter initiates a comprehensive exploration of the dataset, providing a detailed introduction to its various aspects and components. By thoroughly examining the dataset, the chapter aims to establish a solid foundation for the subsequent analyses. It begins by describing the origins and nature of the dataset, including its structure, the types of data included, and the context in which it was collected. This includes an overview of the dataset's dimensions, variables, and any relevant metadata that is critical for understanding its scope and limitations.

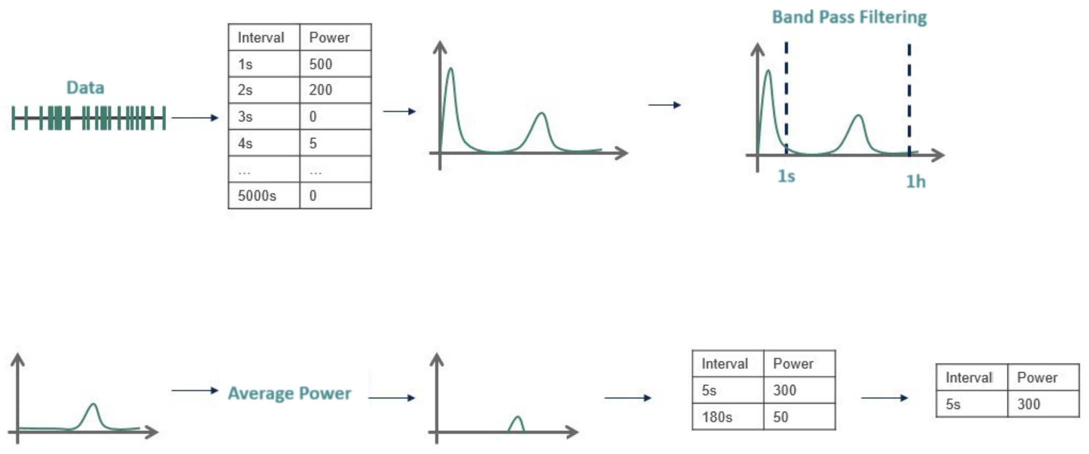
Following this introduction, the process of data generation is elaborated upon. The data generation process is central to the research, ensuring the reliability and validity of the findings. Each step involved in generating the data is outlined, starting from the initial conceptualization to the final implementation. This section covers the design choices made, the tools and technologies used, and the protocols followed to collect and process the data. By providing a step-by-step explanation, the methodology is made transparent and reproducible.

Additionally, the specific functions and algorithms employed at each stage of the data generation process are detailed. This includes a discussion of the rationale behind selecting certain methods over others, as well as the practical considerations that influenced these decisions. Insights are offered into how these functions contribute to the overall data generation process and how they impact the quality and integrity of the dataset. Examples of code snippets, flowcharts, and diagrams may be included to illustrate the implementation process more clearly.

This thorough examination of both the dataset and the data generation process sets the stage for the subsequent analyses and findings presented in the following chapters. By laying this groundwork, the chapter ensures that readers have a comprehensive understanding of the data and the methodology, which is key for interpreting the results and conclusions of the research. This foundational knowledge will facilitate a deeper appreciation of the analyses conducted and the insights derived from them, ultimately contributing to the overall robustness and credibility of the thesis.

4.1. Design of the proposed method

The proposed methodology for beaconing detection encompasses a multifaceted approach designed to effectively identify and mitigate malicious beaconing activities within behavior detection frameworks. This strategy integrates a variety of advanced techniques and algorithms to bolster the system's detection capabilities. Among these techniques are sophisticated anomaly detection methods, and pattern recognition tools, all of which contribute to a robust framework for identifying suspicious activities. By leveraging these advanced technologies, the system is equipped to adapt and respond to new and evolving threats, ensuring that detection remains effective even as malicious actors continuously modify their tactics. A key component of this methodology is its emphasis on real-time detection and response. Continuous monitoring of network traffic and behavior patterns enables the system to quickly identify and flag potential beaconing activities, allowing for rapid threat mitigation before significant harm can occur. This real-time capability is complemented by automated response mechanisms that swiftly neutralize detected threats, thus enhancing the overall security posture of the network. In addition to real-time capabilities, the methodology incorporates a layer of resilience against false positives and negatives. By refining detection algorithms and employing sophisticated filtering techniques, the system aims to minimize incorrect detections that could lead to unnecessary alerts or overlooked threats. This balance is key for maintaining both the efficiency and reliability of the detection system, ensuring that resources are focused on genuine threats. Furthermore, the methodology integrates advanced analytics and continuous learning processes, which are pivotal for maintaining and enhancing detection accuracy over time. By leveraging data analytics and feedback loops, the system can learn from past detections and iteratively improve its performance. This continuous learning process ensures that the detection capabilities are consistently refined and enhanced, keeping pace with the dynamic and ever-evolving nature of cyber threats. Overall, this robust and adaptive methodology significantly strengthens network defenses against beaconing attacks. Adopting a comprehensive and multifaceted approach, enhances both the detection and mitigation processes, providing a resilient and robust defense mechanism against sophisticated and evolving threats. This strategy is poised to play a critical role in safeguarding network security, ensuring the integrity and reliability of the system in the face of malicious beaconing activities.



Interval	Power
1s	500
2s	200
3s	0
4s	5
...	...
5000s	0

Interval	Power
5s	300
180s	50

Interval	Power
5s	300

Figure 4.1.: Steps of the proposed method

Figure 4.1 visually represents the methodology for beaconing detection as proposed in the master thesis. The process begins with the collection of data, a first step that will be elaborated upon in detail later in the thesis. Following data collection, the methodology involves exact data cleaning and processing to ensure the accuracy and quality of the dataset. This stage addresses any inconsistencies or errors in the data, preparing it for more sophisticated analysis.

To enhance security and simplify data management for large organizations with diverse

URL hostnames, a whitelist is implemented. This whitelist restricts user activity to domains containing ”Allianz resolution.” By comparing this whitelist against all user data, entries associated with whitelisted domains are removed. This focused approach significantly reduces the volume of data to be processed, which is given that the dataset exceeds 500,000 entries per day. URLs included in the whitelist are deemed non-malicious and therefore do not require further behavior checks, streamlining the analysis.

Subsequently, the methodology calculates the time intervals for each domain and their occurrences within these intervals. This step is essential for identifying patterns and trends in beaconing activities, providing insight into the frequency and timing of these activities.

The next phase involves bandpass filtering to eliminate noise, retaining only the data within the time range of 1 second to 1 hour for further analysis. This filtering step focuses on relevant signals while minimizing interference from irrelevant data, ensuring that the analysis is both efficient and accurate.

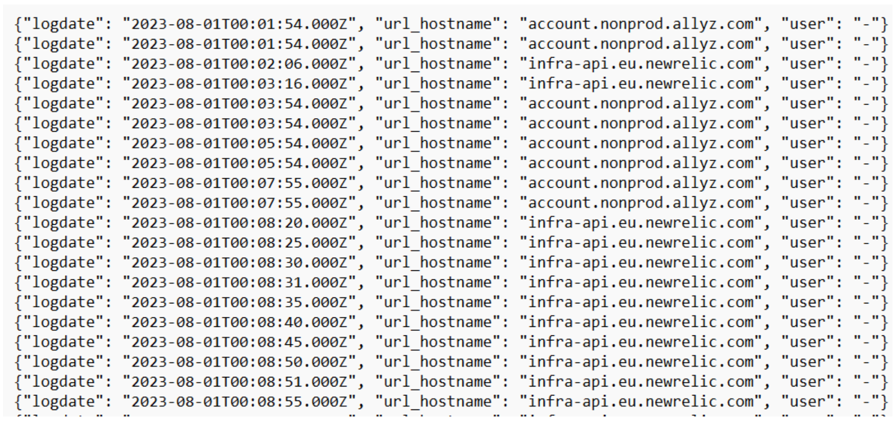
Following filtering, the methodology includes the calculation of average power for each domain and subsequent normalization by adjusting all powers accordingly. In this context, the ”power” metric indicates how frequently a particular website is visited within a given day. This metric relies on the assumption that each access to the website represents a distinct user interaction, with multiple accesses from the same user counted separately. The accuracy of the ”power” measure depends on the consistency and reliability of the data collection methods used to track website visits. Domains exhibiting a negative decrease in power are disregarded, as they do not conform to expected patterns and are unlikely to be indicative of malicious activity.

Finally, the analysis phase involves a thorough examination of the data over time. Significant peaks observed during this analysis are indicative of potential malicious beaconing activity, marking the culmination of the beaconing detection process. This comprehensive methodology ensures that the detection system is robust, accurate, and capable of adapting to evolving threats, thereby enhancing the overall security and integrity of the network.

4.2. Data Extraction and Preparation

The dataset documents the activities of users as they navigate through different URLs during each workday, encompassing a variety of information such as the URLs visited, the date and time of each visit, and potentially other relevant details regarding user behavior or system interactions. This comprehensive collection of data provides a detailed view of user activities and interactions, allowing for an in-depth analysis of browsing patterns and behaviors. Organized within a JSON file, the dataset facilitates efficient storage and retrieval of data, thanks to JSON’s flexibility and readability, which make it easier to manage and manipulate large volumes of information. Each entry logs a specific user interaction, including the precise time and date, allowing for a chronological reconstruction of user activities essential for identifying patterns and trends over time, such as peak usage periods or frequent transitions between specific URLs. The dataset may also include other relevant details that provide further context to user behavior, such as system interactions like login times or error messages, offering additional insights into the user experience and system performance. Overall, the dataset serves as a foundational resource for analyzing user behavior, enabling the identification of significant patterns and trends, and supporting efficient data processing and analysis, which is key for developing effective strategies for beaconing detection and enhancing overall network security.

Figure 4.2 displays a visual representation of the JSON file, providing a comprehensive overview of the specific user's browsing activity.



```
{"logdate": "2023-08-01T00:01:54.000Z", "url_hostname": "account.nonprod.allyz.com", "user": "-"}
{"logdate": "2023-08-01T00:01:54.000Z", "url_hostname": "account.nonprod.allyz.com", "user": "-"}
{"logdate": "2023-08-01T00:02:06.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:03:16.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:03:54.000Z", "url_hostname": "account.nonprod.allyz.com", "user": "-"}
{"logdate": "2023-08-01T00:03:54.000Z", "url_hostname": "account.nonprod.allyz.com", "user": "-"}
{"logdate": "2023-08-01T00:05:54.000Z", "url_hostname": "account.nonprod.allyz.com", "user": "-"}
{"logdate": "2023-08-01T00:05:54.000Z", "url_hostname": "account.nonprod.allyz.com", "user": "-"}
{"logdate": "2023-08-01T00:07:55.000Z", "url_hostname": "account.nonprod.allyz.com", "user": "-"}
{"logdate": "2023-08-01T00:07:55.000Z", "url_hostname": "account.nonprod.allyz.com", "user": "-"}
{"logdate": "2023-08-01T00:08:20.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:25.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:30.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:31.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:35.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:40.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:45.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:50.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:51.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
{"logdate": "2023-08-01T00:08:55.000Z", "url_hostname": "infra-api.eu.newrelic.com", "user": "-"}
```

Figure 4.2.: Steps of the proposed method

Each JSON file acts as a comprehensive record, including important information such as ”logdate” (date and time) and ”url_hostname”. The company prioritizes security; usernames are deliberately omitted during the import process for added protection. The following section outlines the Document Type Definition (DTD) for the JSON files.

```
{
  ”$schema”: ”http://json-schema.org/draft-07/schema#”,
  ”type”: ”object”,
  ”properties”: {
    ”logdate”: {
      ”type”: ”string”,
      ”format”: ”date-time”
    },
    ”urlhostname”: {
      ”type”: ”string”
    },
    ”user”: {
      ”type”: ”string”
    }
  },
  ”required”: [”logdate”, ”urlhostname”]
}
```

The provided DTD serves as a specification for the structure and constraints of the dataset entries. It defines the expected format and mandatory fields for each log entry. Specifically:

- `"logdate"`: Specifies the date and time format for the log entry.
- `"url_hostname"`: Identifies the hostname of the accessed URLs.
- `"user"`: Optional field denoting the user identifier.

This thesis proposes a sophisticated system for managing website connection data, designed to handle and analyze extensive logs of IP address connections to various domains. The system begins by utilizing InfluxDB, a highly specialized database optimized for managing time-series data. InfluxDB's architecture is particularly well-suited for this application because it is engineered to efficiently handle high volumes of data with temporal components, such as timestamps associated with user interactions.

The methodology for managing the data involves importing the dataset into InfluxDB, starting with a strategic focus on a single day's worth of data. This approach allows for an initial, manageable dataset to be processed, which is key for validating the system's functionality and performance before scaling up to larger volumes of data. The dataset itself is composed of JSON files, each containing detailed logs of connections made from various IP addresses to specific domains. These logs include timestamps, IP addresses, and domain names, forming a comprehensive record of user activity.

The success of this methodology relies on establishing a well-structured data environment within InfluxDB. To achieve this, a predefined schema is implemented, which dictates how the data is organized and stored within the database. This schema is designed to ensure data integrity and consistency by enforcing a uniform set of rules across all entries. Consistency in data structure is important as it facilitates smoother data processing and analysis, reducing the likelihood of errors and discrepancies.

The implementation of this schema supports reliable data management by ensuring that each piece of information adheres to the same format and standards. This uniformity is important for maintaining the quality of the data, which in turn impacts the reliability of subsequent analyses and research findings. Clean and reliable data is the cornerstone of trustworthy and effective research, and thus, establishing a well-defined schema is another step in the proposed method. By prioritizing data integrity and consistency, the system lays a solid foundation for accurate and insightful analysis, ultimately contributing to the overall success of the proposed research methodology.

After establishing a well-structured data environment, the next step involves importing the data itself. This process follows a defined sequence, facilitated by custom Python scripts. These scripts automate the creation of a dedicated "bucket" within InfluxDB.

This method establishes a solid foundation for in-depth analysis by transforming raw, unstructured data into a well-organized format that is highly conducive to detailed examination. The strategic decision to import data for a specific day allows for a focused analysis of temporal patterns and variations, which are essential for uncovering trends and anomalies within a defined timeframe. This approach simplifies the analysis, making the data more manageable and representative of specific time-based behaviors. The JSON file structure, with its predefined key criteria and integrated security measures. It provides a clear and consistent framework for storing and accessing data, ensuring that each piece of information adheres to the same standards and is safeguarded against unauthorized access. This structured format supports efficient data processing and retrieval, which is vital for conducting thorough and accurate analyses of network interactions. By utilizing this organized data structure, the methodology enables a comprehensive exploration of user behavior, connection patterns, and potential security threats. This detailed examination is essential for identifying vulnerabilities, detecting suspicious activities, and enhancing overall network security. The systematic approach not only facilitates a deeper understanding of the dataset but also sets the stage for generating actionable insights and implementing effective security measures.

The data is collected over exactly one day, which represents a typical working day on Tuesday. The total volume of data generated during this single day amounts to almost 73 gigabytes. To gain meaningful insights from this data, it is crucial to understand its behavior. Therefore, as an initial step, the data's behavior was thoroughly examined. This analysis was conducted in several stages: By observing the overall data behavior within one day, identifying which data sets were most frequently accessed or visited during the day, and analyzing the time intervals between each request, this approach enables a deeper understanding of usage patterns and helps in identifying any anomalies or trends that may be present in the data.

Figure 4.3 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.

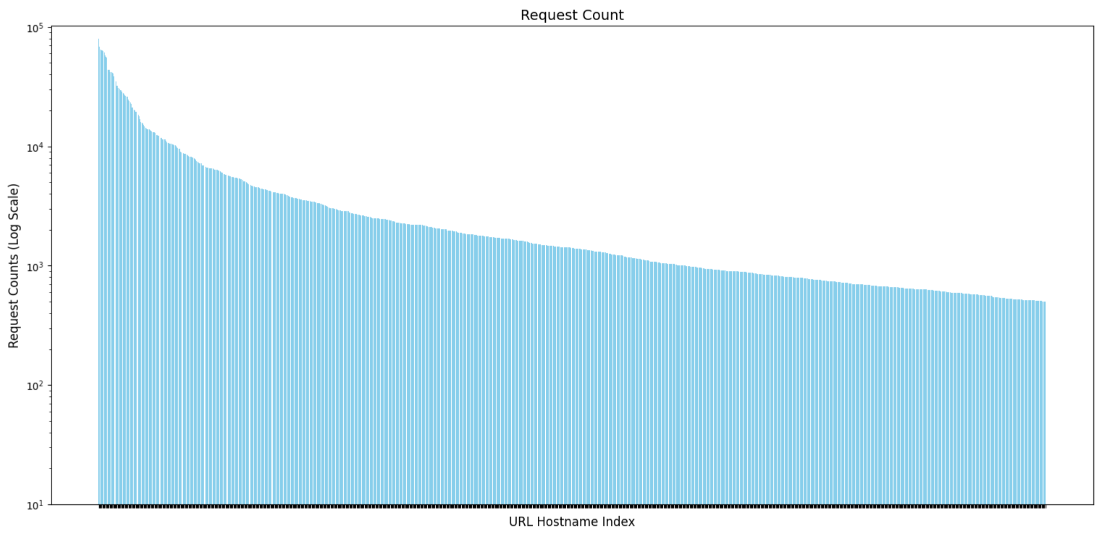


Figure 4.3.: Request counts of URLs (log scale)

Figure 4.4 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.

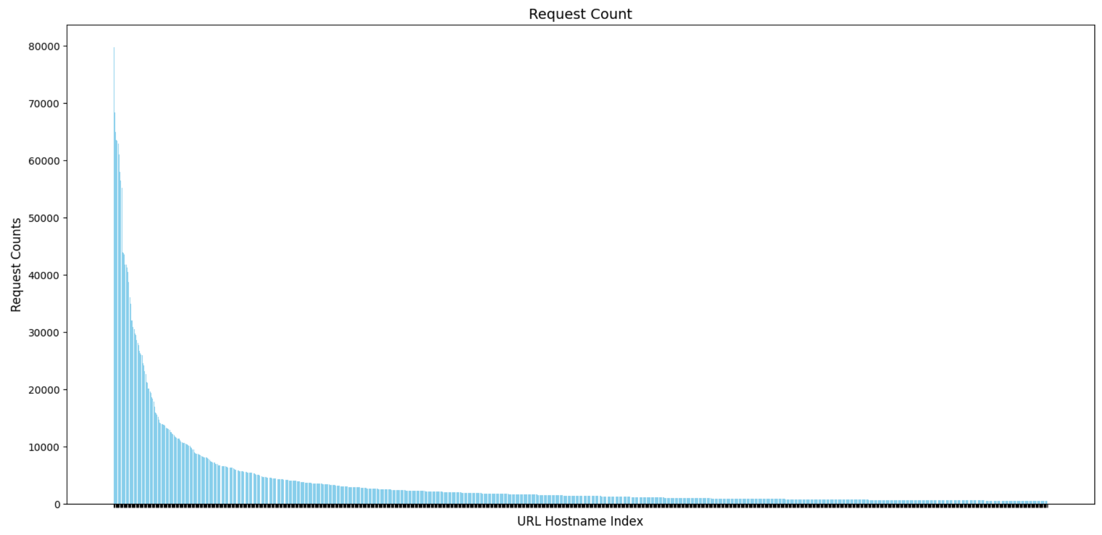


Figure 4.4.: Request counts of URLs (linear scale)

Figure 4.5 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 range of 500–3,500.

From the figure, it is evident that some URLs exhibit high activity levels initially but experience a steep decline, with their visit counts approaching zero around 04:00. Based on this observation, the URL activity was categorized into two distinct periods: day activity and night activity, each represented by separate charts for better analysis.

Figure 4.6 and Figure 4.7 provide a more detailed breakdown of visit patterns during night and day times, respectively. These visualizations offer insights into how user behavior varies between different times of the day, shedding light on potential differences in activity levels and usage patterns. By examining these variations, it is possible to gain a more nuanced understanding of network interactions and identify any irregularities that may require further investigation. This detailed analysis of visit patterns is essential for detecting anomalies and

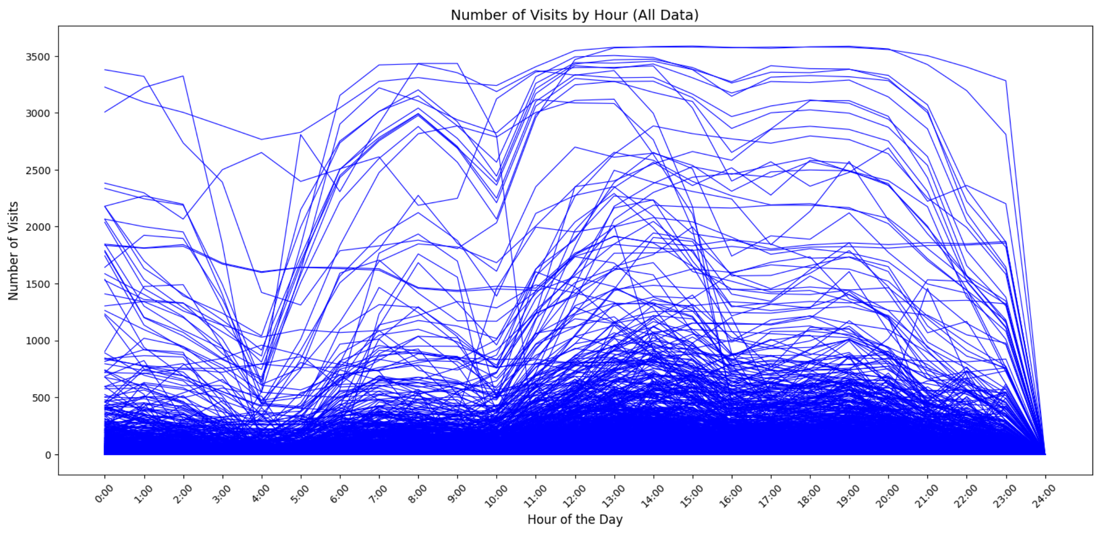


Figure 4.5.: Number of visit by hour (24 hours)

potential security threats, as it provides a comprehensive view of user behavior and network activity.

Figure 4.6 highlights the night activity of URLs, spanning the period from 00:00 to 04:00. A closer examination of the URLs during this time reveals that the URLs with the highest visit counts, which subsequently drop off rapidly near 04:00, are associated with automated software updates occurring over the network. This behavior aligns with typical network maintenance or update schedules, which are often performed during off-peak hours to minimize disruptions.

Conversely, Figure 4.7 depicts the day activity of URLs, covering the period from 04:00 to 24:00. During this time, the distribution of URLs with a visit count in the range of 0–100 is significantly more frequent compared to higher ranges. This indicates that a large portion of URL activity during the day comprises low-visit events, likely reflecting regular user interactions or routine operations.

By separating the data into night and day activity, the analysis provides a clearer understanding of how URL usage patterns vary throughout the day. This division helps identify critical periods of high network activity and aids in distinguishing between automated processes and user-driven events.

A time interval refers to the duration or amount of time between two events or points in time. In this context, it means the period between each request made to the URLs.

Figure 4.8 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 60 seconds, each second has its own bin; from 1 to 30 minutes, each minute has its own bin. The calculation within this period is such that if the bin is 1 minute, the interval is ±30 seconds, meaning it calculates from 30 seconds to 90 seconds. This approach helps ensure that the analysis does

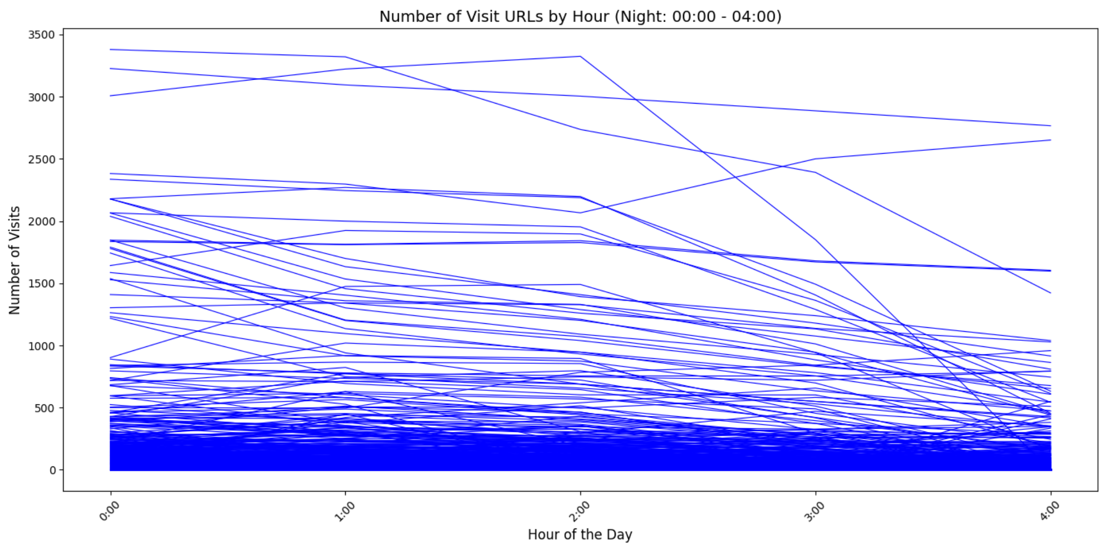


Figure 4.6.: Number of visit by hour (Night time)

not miss any potential malicious behavior within the specified time limits. 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.

Figure 4.9 provides a linear scale representation of the time intervals between requests for different URLs within the dataset. This visualization offers a more detailed view of the time intervals across URLs, highlighting the distribution of durations with greater granularity. By examining this distribution, it is possible to identify URLs with varying patterns of activity, ranging from short intervals to longer durations. This analysis enables organizations to gain insights into user behavior and interaction patterns, facilitating informed decision-making and strategic planning.

To better understand the output, given the large amount of data, the criteria that were evaluated were narrowed down to the top 20 for display.

Figure 4.10 displays the top 20 URLs with the highest visit counts during the day, providing a focused view of the most frequently accessed domains. This visualization highlights the distribution of visits across these top URLs, showcasing the range of activity levels observed within the dataset. By examining the visit patterns of these top URLs, it is possible to identify critical resources or frequently accessed services, which may require additional security measures or monitoring. This analysis offers valuable insights into user behavior and resource utilization, enabling organizations to optimize their network infrastructure effectively.

Figure 4.11 provides a detailed view of the request counts for the top 20 URLs within the dataset, highlighting the distribution of visit frequencies across these high-activity domains.

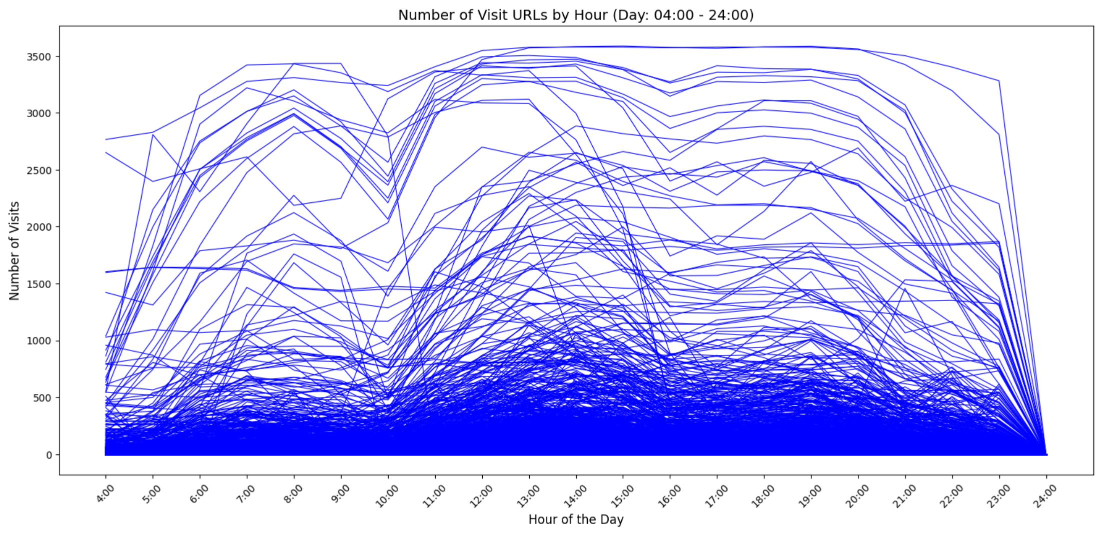


Figure 4.7.: Number of visit by hour (Day time)

This visualization offers insights into the visit patterns of these top URLs, showcasing the range of activity levels observed within the dataset. By examining the request counts of these top URLs, it is possible to identify critical resources or frequently accessed services, which may require additional security measures or monitoring. This analysis enables organizations to gain a deeper understanding of user behavior and resource utilization, facilitating informed decision-making and strategic planning.

Figure 4.12 provides a detailed view of the time intervals between requests for the top 20 URLs within the dataset. This visualization offers insights into the frequency and timing of user interactions with these high-activity domains, shedding light on potential patterns and trends in behavior. By examining the time intervals of these top URLs, it is possible to identify distinct usage patterns and interaction trends, which may be indicative of specific user behaviors or system activities. This analysis enables organizations to gain a deeper understanding of network interactions and user behavior, facilitating informed decision-making and strategic planning.

4.3. Data Preprocessing

The initial step in the data cleaning process involves separating all URL hostnames from one another. Once these hostnames are isolated, they are systematically sorted by date. This sorting step serves several important purposes. Firstly, it organizes the data chronologically, which facilitates the identification of trends or patterns that emerge over time. For instance, if there is a need to investigate a specific security incident that occurred on a certain day, sorting the data by date allows for a more focused and efficient search, thereby reducing the time and effort required to locate the relevant information. Secondly, sorting by date prepares the data for subsequent stages in the data management workflow. Many data analysis techniques depend

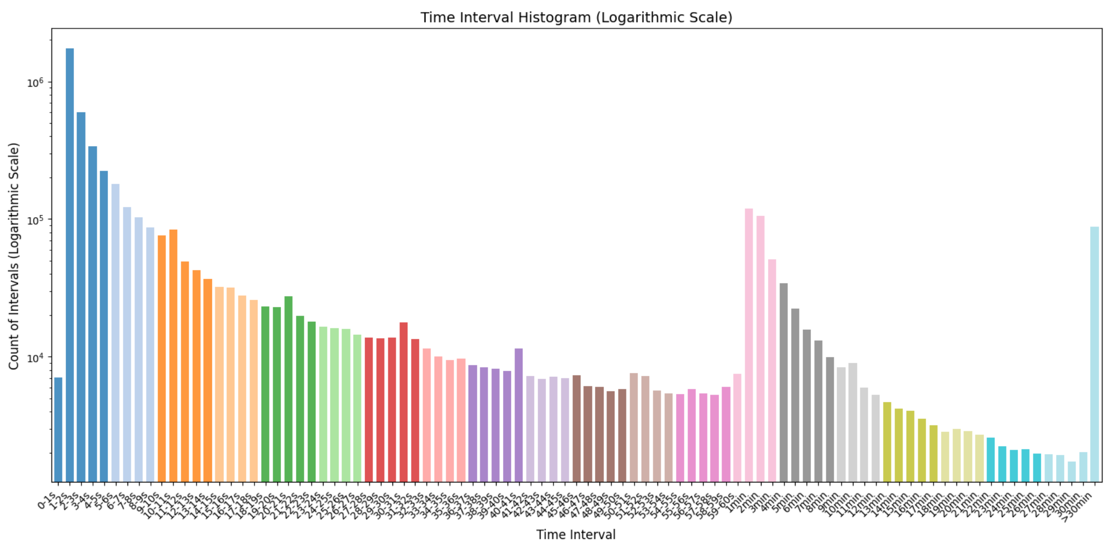


Figure 4.8.: Time interval (log scale)

on the data being arranged in a specific sequence, and organizing the data chronologically ensures that it is ready for these advanced analytical processes. This stage of the cleaning process effectively transforms raw data into a structured and coherent format, which is necessary for performing in-depth analyses of network interactions and identifying potential security threats. Additionally, this preprocessing step includes the creation of a whitelist from all URL hostnames. Since the company frequently accesses a variety of trusted URLs multiple times throughout the day, establishing a whitelist helps to streamline the algorithm's operations. By compiling a list of these consistently trusted URLs, the algorithm's efficiency is enhanced, allowing it to operate more quickly and effectively. This improved efficiency supports the algorithm's ability to detect potentially suspicious activities within network interactions more accurately. Through this cleaning process, raw data is organized into a format that is both structured and easily analyzable, setting the stage for comprehensive insights into network behaviors and security risks.

In the figure 4.13 the various steps undertaken to process the data are illustrated, beginning with the initial stages of data collection and extending through to the final preparation required before the algorithmic process can commence. This visual representation outlines how the raw data is first gathered and then cleaned to ensure accuracy and consistency. The process includes several key stages, Each step is designed to transform the raw, unprocessed data into a well-organized dataset that is ready for detailed algorithmic examination.

4.4. Time Interval Analysis

The concept of a "time interval" in this context pertains to the duration between occurrences of a specific URL hostname within a given day. This analysis focuses on the time elapsed between visits to each unique website (URL hostname) during a single day. Visualize this as tracking the

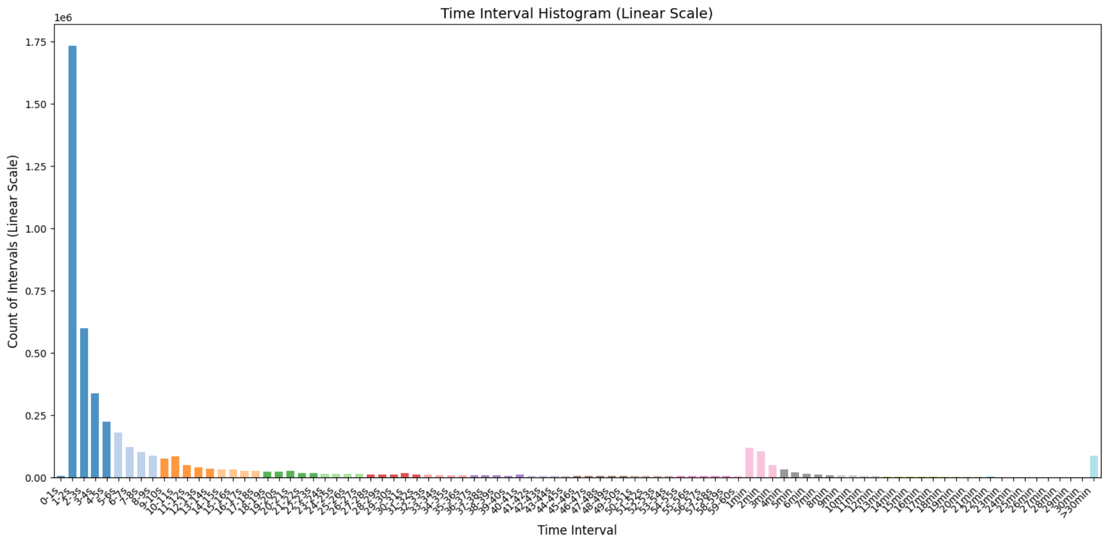


Figure 4.9.: Time interval (linear scale)

gaps between entries in a log for a particular website; each visit to the site is recorded as a new entry. By examining these time intervals, the method aims to decipher how frequently and consistently users access each website throughout the day. This approach effectively captures the ”rhythm” of website activity, revealing patterns such as peak usage times and periods of lower activity.

Calculating these time intervals produces a numerical value that represents the time gaps between visits to the same website. This value serves as a foundational metric for understanding typical website usage patterns. By focusing on daily data, the method ensures that the time intervals reflect a representative snapshot of user behavior, allowing for a nuanced exploration of how web traffic fluctuates throughout the day. This in-depth analysis helps to identify the typical patterns of website access, creating a baseline for what is considered normal activity.

This focus on time intervals is key for setting a robust baseline of regular website traffic. Understanding these regular patterns establishes a benchmark against which unusual or atypical activities can be measured. For instance, if a website that typically has long intervals between visits suddenly experiences a surge in frequent accesses, this deviation from the established norm may indicate suspicious behavior or potential security threats. Therefore, the analysis of time intervals facilitates the identification of normal traffic patterns and enhances the ability to detect anomalies effectively.

Moreover, analyzing time intervals provides a framework for differentiating between routine user behavior and potential malicious activities. By establishing clear patterns of typical website usage, the methodology can more accurately pinpoint deviations that may warrant further investigation. This thorough examination lays the groundwork for subsequent analytical steps, where patterns that deviate from the norm can be scrutinized for signs of anomalous or harmful activities. Ultimately, this focus on time intervals strengthens the overall detection process, ensuring that the system can distinguish between expected and unexpected behavior

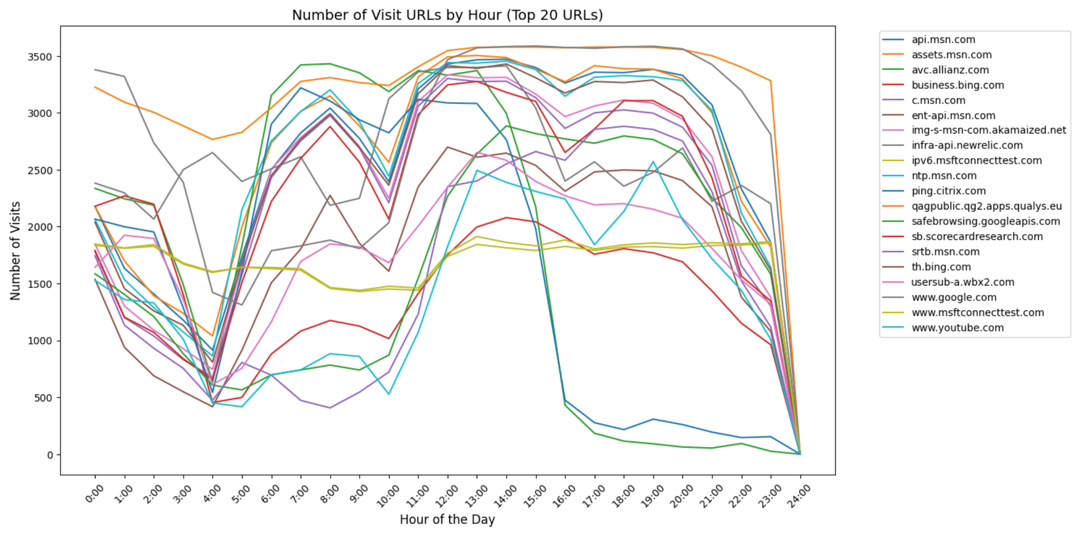


Figure 4.10.: Number of visit by hour (top 20 URLs)

with greater precision, leading to more reliable and actionable insights into network security.

Figure 4.14 is a series of histograms depicting the distribution of time intervals between successive network requests for different URLs within a specified timeframe. Each histogram represents a unique URL, with the x-axis consistently ranging from 0 to 400 seconds. This range captures the variability in time intervals between requests.

The histograms are constructed using data that records the duration between consecutive requests. Each bar in the histogram illustrates the frequency, or "power," (that refers to the frequency of network requests that occur within specific time interval bins) of requests that fall within specific time interval bins. This visualization helps to understand how often requests occur within different time gaps for each URL.

For each URL, the histogram reveals the frequency of requests within various time intervals. Taller bars indicate higher request frequencies within those intervals, while shorter bars suggest less frequent requests. The uniform x-axis range across all plots enables straightforward comparison between different URLs, highlighting differences in request timing patterns.

URLs with concentrated request bursts will show taller bars in certain intervals, reflecting periods of higher request activity. Conversely, URLs with more sporadic or irregular request patterns will exhibit shorter bars, indicating lower or more inconsistent request frequencies. By analyzing these histograms, one can discern patterns such as periodic bursts, irregular intervals, or consistent frequencies. These insights can be indicative of the operational behavior, performance, and load of the respective URLs.

Overall, the histograms provide a visual summary of how request timings are distributed over the observed period, facilitating an understanding of request dynamics and potentially revealing insights into the performance and usage patterns of the monitored web services.

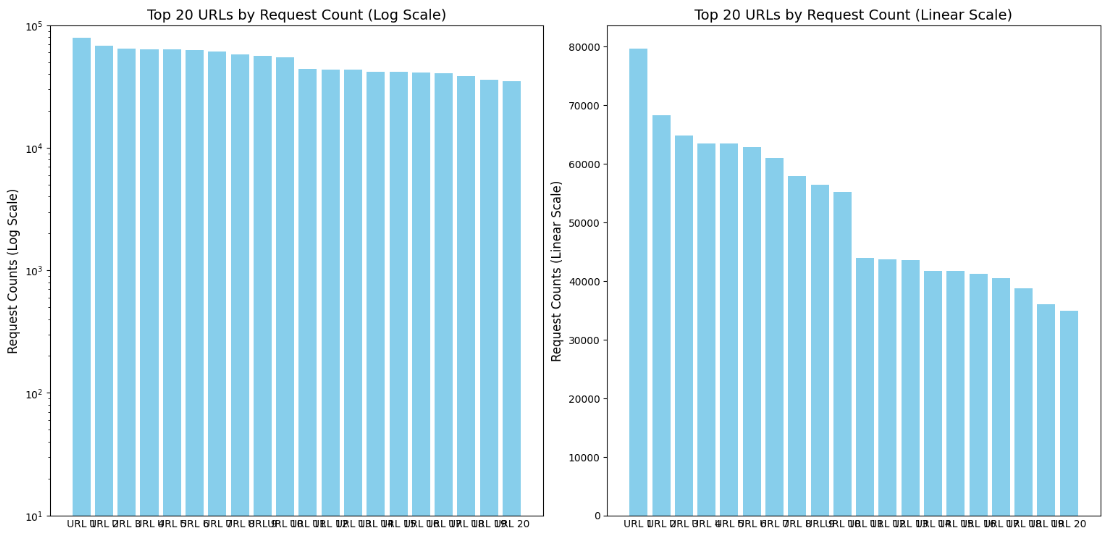


Figure 4.11.: Request count (top 20 URLs)

4.5. Data Enhancement

Once the ”power” for all URLs in the dataset is calculated, representing the frequency of visits to each unique website, the next step involves computing the average power. The ”power” metric serves as a quantitative measure of how often each website is accessed within a given timeframe, providing valuable insight into user behavior patterns. The average power acts as a reference point or baseline against which the individual powers are compared, allowing for the identification of significant deviations from typical visit frequencies. To determine the average power, the total power across all URLs is summed and then divided by the number of URLs in the dataset. This average power serves as a benchmark, providing a context for evaluating the frequency of visits to each specific URL. By establishing this baseline, the methodology can differentiate between normal and abnormal activity levels, thereby facilitating the identification of URLs that exhibit unusual patterns of access. After determining the average power, each calculated power is assessed relative to this baseline. The evaluation process involves comparing the power of each URL to the average power to identify significant deviations. Specifically, the power of each URL is subtracted from the average power. If the resulting value is negative, indicating that the URL’s visit frequency is below the baseline, it is omitted from further analysis. This step ensures that the focus remains on URLs with visit frequencies that significantly exceed the norm, which are more likely to represent meaningful patterns or anomalies. The rationale behind omitting negative deviations is based on the premise that these values do not provide meaningful insights into unusual behavior or potential security threats. By filtering out these less significant data points, the methodology enhances the quality and relevance of the dataset. This focused approach allows for a more efficient examination of the data, concentrating on positive deviations that indicate higher-than-average visit frequencies. Focusing on significant deviations from the average power refines the dataset, making it more manage-

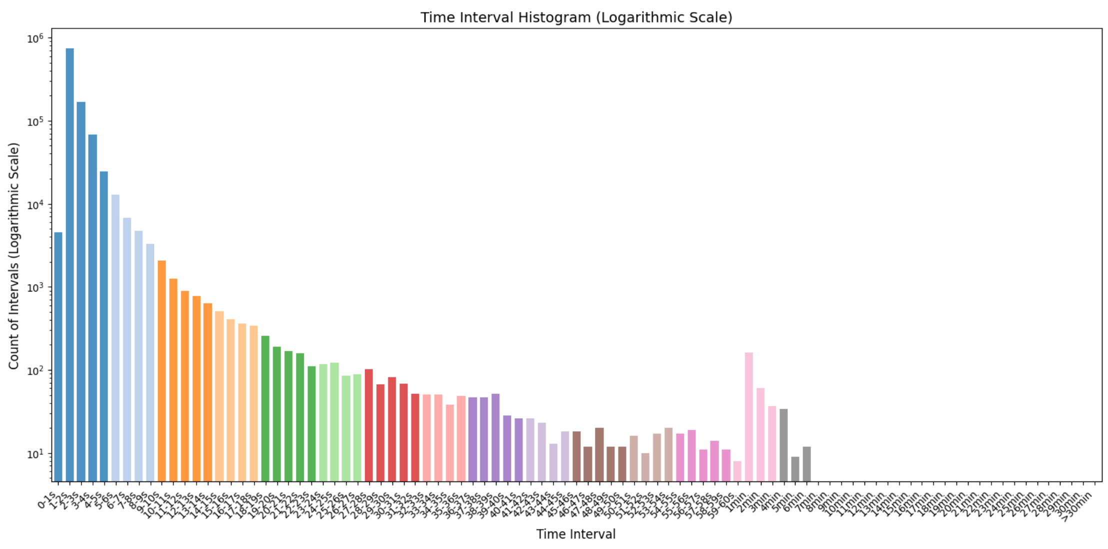


Figure 4.12.: Time interval (top 20 URLs)

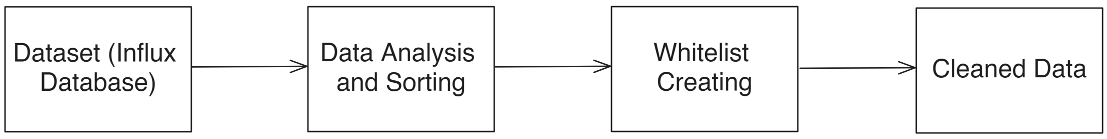


Figure 4.13.: Dataset

able and pertinent for further analysis. This step not only improves the efficiency of the data processing but also enhances the ability to detect anomalies that could signify security threats or other irregularities. By concentrating on URLs with visit frequencies that stand out from the baseline, the methodology increases the likelihood of identifying genuinely noteworthy patterns. In summary, the calculation of average power and the subsequent evaluation of individual powers against this baseline are critical components of the data enhancement process. By omitting powers that result in negative values, the methodology ensures that the dataset is refined to include only significant deviations. This targeted focus on relevant patterns and anomalies lays the groundwork for a more detailed and accurate analysis of network interactions and potential threats, ultimately contributing to a more robust and reliable security framework.

4.6. 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 the application of a filter that selectively passes signals whose frequencies fall within a certain

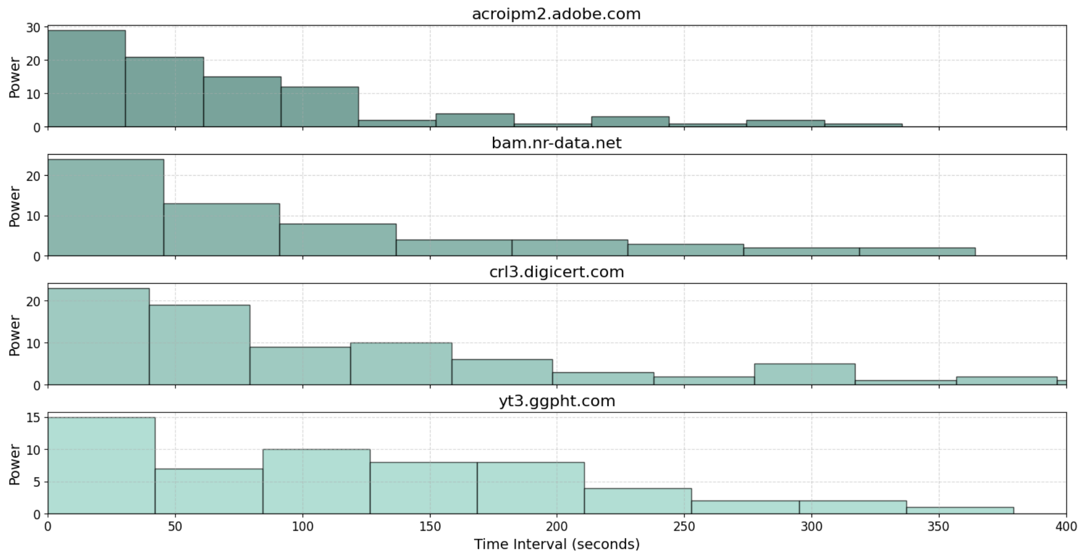


Figure 4.14.: Time Intervals

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:

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

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.

4.7. Evaluation Criteria

The Python-implemented data evaluation process reveals instances of beaconing behavior within the dataset, providing valuable insights for experts to identify potential malicious activity. This task extends beyond simply visualizing the algorithm's output; it relies heavily on the expertise of professionals who navigate vast amounts of data to detect subtle indicators of malicious behavior. The evaluation involves an examination of user behavior patterns to identify any anomalies or suspicious activities that suggest beaconing. Beaconing, a technique used by malicious actors, involves repetitive signal transmissions that allow compromised systems to communicate with external servers. Identifying such behavior requires sophisticated analysis techniques and a deep understanding of network traffic patterns.

Leveraging Python's powerful data analysis and processing capabilities, the evaluation process begins with the systematic analysis of large volumes of network data. Python's libraries and tools enable the handling and manipulation of complex datasets, facilitating the detection of beaconing signals embedded within regular network traffic. The process involves applying various algorithms and analytical methods to sift through the data, identifying patterns that deviate from normal user behavior. This initial algorithmic analysis provides a foundation, but the true strength of the evaluation lies in the subsequent expert review.

Experts play a critical role in this process, as they interpret the algorithmic findings within the broader context of network activity. They scrutinize the identified patterns, cross-referencing them with known threat indicators and leveraging their experience to assess the likelihood of malicious intent. This human expertise is essential for distinguishing between benign anomalies and genuine threats. Upon detection of suspicious behavior, experts are tasked with making informed decisions on how to address the implicated users and domains. This involves a risk assessment to determine the severity and potential impact of the detected activity.

Decisions on handling identified threats may include a range of actions, from monitoring the suspect activity more closely to implementing immediate mitigation measures such as blocking the suspicious domains or isolating affected systems. The evaluation process also involves documenting the findings and actions taken, ensuring a comprehensive record that can be used for future reference and continuous improvement of the detection system.

The integration of Python's technical capabilities with the nuanced understanding of skilled analysts results in a robust and dynamic approach to network security. This comprehensive approach not only enhances the accuracy of detecting beaconing behavior but also ensures that potential threats are thoroughly understood and effectively mitigated. By combining automated data processing with expert analysis, the methodology provides a reliable framework for maintaining network security and proactively addressing emerging threats. The continuous evaluation and refinement of this process are vital for staying ahead of increasingly sophisticated cyber threats, ultimately safeguarding the integrity and security of network environments.

5. 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.

5.1. 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 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.

5.2. Whitelisting Mechanism for URL Filtering

Establishing a robust URL whitelist is paramount in significantly enhancing network security and monitoring capabilities. This section delves into the comprehensive methodology employed for creating and implementing an effective whitelist mechanism within Allianz Company's extensive network infrastructure. The primary purpose of the whitelist is to gather all URLs that are considered safe, based on predefined criteria. These criteria include URLs that are commonly visited by users every day, resolution URLs of the company, and other trusted URLs.

The whitelist creation process involves systematically curating URLs based on these inclusion criteria to ensure that only legitimate and relevant URLs are retained. URLs that do not meet these criteria are excluded from the whitelist. This means that all URLs in the whitelist are trusted and do not need to proceed through the additional steps of the algorithm, while other URLs that do not meet these criteria are subject to further detailed analysis.

Examples of exclusion criteria include known malicious domains, suspicious IP addresses, unauthorized access points, and other indicators of potential threats. These URLs are systematically identified and excluded to maintain the integrity and security of the network environment.

The whitelist creation function, as outlined in the methodology, plays a foundational role in preprocessing network activity data by curating a refined subset of URLs that do not need further in-depth analysis. This ensures that these analyses focus exclusively on potentially harmful communication patterns within the network, thereby greatly enhancing the efficiency and accuracy of the detection process. This essential preprocessing step not only reduces the overall volume of data to be analyzed but also significantly improves the relevance and quality of the dataset, allowing for more targeted and effective monitoring of network activity.

Furthermore, the whitelist substantially enhances the interpretability and clarity of results by providing a refined and high-quality dataset that is specifically tailored to the specific research context. This facilitates more accurate, insightful, and actionable conclusions. By selectively retaining URLs that align precisely with the predefined inclusion criteria, the whitelist mechanism enables the methodology to identify and focus on genuine security threats and critical behavioral patterns that are of primary interest.

This streamlined and optimized dataset ensures that analysts can more easily and effectively identify anomalies and patterns indicative of potential security breaches, ultimately contributing to a more secure, robust, and resilient network infrastructure for Allianz Company. This comprehensive and detailed approach to whitelist creation and implementation not only underscores its importance in the broader context of advanced network security but also highlights the concerted effort invested in ensuring the methodology's robustness, reliability, and overall effectiveness in safeguarding the organization's digital assets.

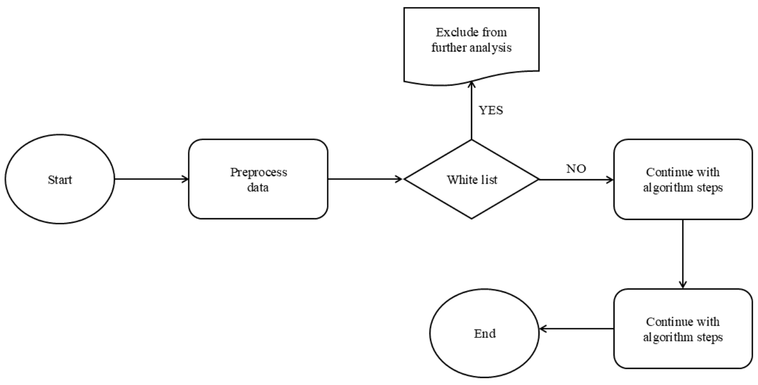

Exclude from further analysis
YES
Start
Preprocess data
White list
NO
Continue with algorithm steps
Continue with algorithm steps
End

Figure 5.1.: Whitelist steps

Figure 5.1 provides a comprehensive visual representation of the whitelist's functionality within the overall algorithm, outlining each stage from the initial reception of raw data to the final determination of which parts to retain based on predefined rules and criteria. Starting with the entry point where raw data, encompassing a wide array of URLs, is collected and fed into the system, the figure details the sophisticated filtration mechanism that systematically evaluates each URL against established whitelist criteria designed to exclude known malicious domains, suspicious IP addresses, and unauthorized access points. As each URL undergoes assessment, those meeting the stringent inclusion criteria are retained, while those that do not are filtered out. This critical stage ensures that only relevant and legitimate URLs proceed to the next phase of analysis, significantly aiding in the isolation of important information. The figure further illustrates how this refined dataset, now devoid of irrelevant and potentially harmful URLs, is seamlessly integrated back into the algorithmic framework for deeper processing. By focusing on retaining only URLs that pass the whitelist criteria, the filtration mechanism not only streamlines the data but also enhances the clarity and utility of the information being analyzed. This refined approach reduces noise and ensures that analysts and automated systems can utilize the cleaner dataset more effectively, leading to more precise detection of patterns, trends, and anomalies indicative of security threats. The figure underscores the critical role of the whitelist in enhancing the overall efficiency and accuracy of the algorithm, making the detection of malicious behavior more reliable and robust. This thorough and detailed visualization emphasizes the importance of each stage in the process, highlighting how the integration of the whitelist fundamentally improves the algorithm's ability to discern and address security challenges within the network infrastructure.

Function Signature:

The whitelist creation function is defined as follows:

Listing 5.1: `Whitelist` Function

```
def create_whitelist(data, exclusion_criteria):
```

Parameters:

- **data**: A dictionary containing network activity data organized by URL hostname.
- **exclusion_criteria**: A list of words or phrases used to filter out URLs that should not be included in the whitelist. URLs are compared against this list of exclusion criteria, and if a URL contains any of the exclusion criteria, it is excluded from the whitelist. Consequently, only URLs that do not match any of the exclusion criteria are added to the whitelist. In subsequent steps, only URLs that have been added to the whitelist are considered for further analysis, while those that are excluded are ignored as they are deemed untrustworthy or irrelevant.

Function Definition

Listing 5.2: `create_whitelist` Function

```
def create_whitelist(data, exclusion_criteria):
    whitelist = {url: requests for url, requests in data.items()
                 if not any(exclusion in url for exclusion
                            in exclusion_criteria)}
    return whitelist
```

Functionality:

1. **Whitelist Initialization:** Initialize an empty whitelist dictionary to store URLs that meet the inclusion criteria.
2. **URL Filtering:**
   - Iterate through each URL hostname and associated activity data in the input dictionary.
   - Check if the URL hostname contains any of the specified exclusion criteria.
   - If the URL hostname does not match any exclusion criteria, add it to the whitelist dictionary along with its associated activity data.
3. **Return Whitelist:** Return the generated whitelist dictionary containing URLs that passed the exclusion criteria.

Performance Considerations

The `create_whitelist` function is efficient in terms of both time and space complexity:

- **Time Complexity:** The function iterates over each URL in the `data` dictionary once and performs a check against the `exclusion_criteria` list for each URL. If there are n URLs and m exclusion criteria, the time complexity is $O(n \times m)$.
- **Space Complexity:** The function creates a new dictionary to store the whitelisted URLs. In the worst case, where no URLs match the exclusion criteria, the space complexity is $O(n)$, where n is the number of URLs in the input `data` dictionary.

While the function is efficient for small to moderately sized datasets, its performance could be affected by a very large number of URLs or exclusion criteria. Optimizations, such as parallel processing or more efficient data structures, could be considered for handling larger datasets.

Role in Thesis:

The whitelist creation function plays a critical role in preprocessing network activity data by curating a subset of URLs for further analysis. By excluding URLs that are irrelevant or known to be unrelated to the research objectives, the function ensures that subsequent analyses focus on meaningful communication patterns within the network. Additionally, the whitelist enhances the interpretability of results by providing a refined dataset tailored to the specific research context, thereby facilitating more accurate insights and conclusions in the thesis.

Results and Output:

The filtered URLs, as per the established whitelist, are then presented in the output. This step serves to showcase the effectiveness of the whitelist mechanism in selectively retaining URLs that align with the predefined inclusion criteria, thereby contributing to the enhancement of the overall analytical precision in the context of network behavior analysis.

5.3. Average Power Calculation

The implementation of the power calculation step follows the figure's progression 5.2. After the whitelist filtering process, the determination of URL powers ensues, resulting in the creation of a power dictionary. Following the completion of power calculations for all URLs, the subsequent step involves computing the average power across the dataset. Following this computation, the process proceeds to normalize the URL powers by subtracting the average power. This normalization step may yield negative values for certain URLs. As such negative values are indicative of non-malicious behavior, URLs associated with negative power values are excluded from further analysis. Conversely, URLs with positive power values continue through subsequent steps of analysis.

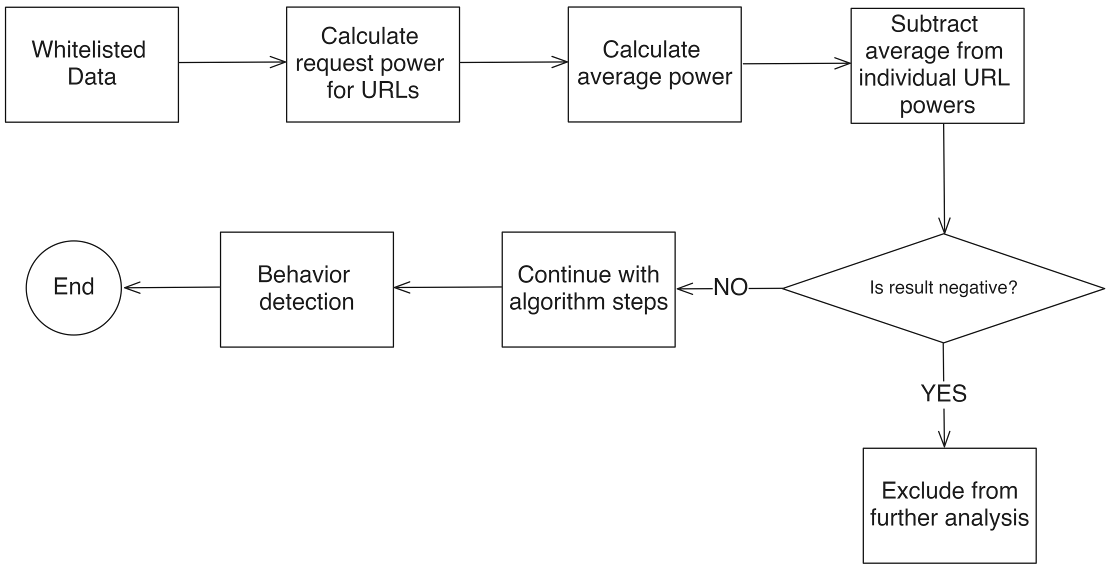

Whitelisted Data
Calculate request power for URLs
Calculate average power
Subtract average from individual URL powers
Is result negative?
NO
Continue with algorithm steps
Behavior detection
End
YES
Exclude from further analysis

Figure 5.2.: Average power calculation

The calculation of request power is a component aimed at quantitatively assessing the frequency of requests based on the temporal intervals between successive occurrences of URL hostnames within the dataset. This subsection delineates the functionality, parameters, and approach employed by the `calculate_request_power` function, providing an in-depth understanding of its mechanics and significance.

Key parameters such as the request list and the specific approach employed for power calculation are elaborated upon in detail. The request list comprises a comprehensive collection of individual requests, each containing pertinent information such as timestamps that denote the exact occurrence time of each request. By analyzing the temporal intervals between successive occurrences of URL hostnames within the dataset, the calculation of request power yields valuable insights into the frequency dynamics of network activity.

This analytical process involves determining the duration between each visit to the same URL hostname, effectively capturing the rhythm and regularity of access patterns.

The `calculate_request_power` function leverages these intervals to compute the power of requests, which is essentially a measure of how frequently specific URLs are accessed over a given time. This measure is key as it helps to identify normal versus abnormal patterns of network behavior, which can be indicative of potential security threats or unusual activities.

Furthermore, the calculated request power values facilitate a granular understanding of network interaction patterns, enabling a more precise and detailed analysis of user behavior and network usage. By identifying deviations from established norms, the methodology enhances the detection of anomalous behavior and potential security threats. The iterative nature of the calculation process ensures continuous refinement and optimization of the methodology's performance, thereby enhancing its efficacy in real-world cybersecurity scenarios. Each iteration involves reassessing and recalibrating the parameters to better fit the observed data, thus improving the accuracy and reliability of the power calculations over time. This continuous improvement cycle is essential for adapting to the evolving landscape of network security threats and ensuring that the methodology remains robust and effective in detecting and mitigating potential risks. Through this comprehensive approach, the `calculate_request_power` function plays a role in fortifying network security by providing deep, actionable insights into the frequency and patterns of network requests.

Function Signature

The function for calculating the average power is defined as follows:

Listing 5.3: `Calculate Power` Function

```
def calculate_request_power(request_list):
```

Parameters:

- **request_list:** A list of dictionaries representing individual requests, where each dictionary contains pertinent information such as timestamps ("`_time`") denoting the occurrence time of the request.

Function Definition

Listing 5.4: `calculate_request_power` Function

```
def calculate_request_power(request_list):
    power_dictionary = {}
    last_date_time = request_list[0]["_time"]

    for request_dict in request_list:
        current_date_time = request_dict["_time"]
        # Check if current_date_time is a string,
        # and convert it to a datetime object
        if isinstance(current_date_time, str):
            current_date_time = datetime.strptime(
                current_date_time, "%Y-%m-%dT%H:%M:%S.%fZ"
            )

        time_delta = int(
            (current_date_time - last_date_time).total_seconds()
        )

        # Add the power to the power dictionary
        power_dictionary[time_delta] = power_dictionary.get(
            time_delta, 0
        ) + 1
        last_date_time = current_date_time

    # Sort the dictionary for better visualization
    power_dictionary = dict(sorted(power_dictionary.items()))

    return power_dictionary
```

Functionality

1. **Initialization:** The function initializes an empty dictionary, `power_dictionary`, to store the computed power values.
2. **Time Interval Computation:** It iterates through the `request_list`, calculating the time delta (in seconds) between consecutive request occurrences.
3. **Power Assignment:** For each time delta, the function assigns a corresponding power value in the `power_dictionary`, representing the frequency of requests occurring within that interval.
4. **Output Generation:** Upon completion, the function returns `power_dictionary`, mapping time intervals to their respective power values.

Performance Considerations

The `calculate_request_power` function is efficient in terms of both time and space complexity:

- **Time Complexity:** The function iterates over each request in the `request_list` once and performs a timestamp comparison for each request. If there are n requests, the time complexity is $O(n)$.

- **Space Complexity:** The function creates a new dictionary to store the power values. In the worst case, where each request occurs at a unique time interval, the space complexity is $O(n)$, where n is the number of requests in the input `request_list`.

While the function is efficient for small to moderately-sized datasets, its performance could be affected by a very large number of requests. Optimizations, such as parallel processing or more efficient data structures, could be considered for handling larger datasets.

Role in Thesis

The `calculate_request_power` function plays a critical role in analyzing network activity data by computing the power of requests over time. By measuring the frequency of requests occurring within specific time intervals, the function helps identify patterns and trends in network behavior. This analysis is essential for understanding the dynamics of user interactions and system activities, which can inform various aspects of network security and performance monitoring. Additionally, the computed power values can serve as input for further analysis, such as detecting anomalies or assessing the impact of different network events.

Results and Outputs

The function yields `power_dictionary`, a structured representation of time intervals and their corresponding request power values, which is essential for analyzing and interpreting the frequency dynamics within the dataset. This `power_dictionary` serves as a foundational tool that encapsulates the quantitative assessment of how frequently specific URL hostnames are accessed over defined periods. Providing a detailed mapping of time intervals to request power values, allows for a nuanced examination of the temporal patterns of network activity. The structured nature of `power_dictionary` ensures that each data point is accurately cataloged and easily retrievable, facilitating efficient data manipulation and analysis. This computation of request power enables a granular understanding of network interaction patterns, offering deep insights into the behavior of users and systems over time. It helps to identify normal operational patterns, thereby making it easier to spot deviations that may indicate anomalous or malicious activity. This capability is particularly valuable for cybersecurity applications, where understanding the baseline frequency dynamics is critical for detecting threats. By contributing significantly to the overarching objective of characterizing the temporal dynamics inherent in the dataset, the `power_dictionary` plays a pivotal role in enhancing the overall robustness and effectiveness of the network security methodology. This structured representation not only aids in current analytical processes but also lays the groundwork for future research and development, allowing for continuous improvement and adaptation of the security framework in response to evolving threats.

5.4. Band-Pass Filtering

Bandpass filtering is a signal processing technique employed to dissect time-series data, allowing the extraction of specific frequency components nestled within a pre-defined range. This technique involves passing signals through a filter that permits only those components whose frequencies lie within a certain interval while attenuating or eliminating frequencies outside this range. The effectiveness of bandpass filtering lies in its ability to isolate and highlight the most pertinent aspects of the data, effectively reducing noise and enhancing the clarity of the signal. By focusing on a specific band of frequencies, bandpass filtering can reveal underlying patterns and trends that may not be immediately apparent in the raw data. This is particularly useful in various applications, including network traffic analysis, where it is essential to identify and analyze periodic behaviors or anomalies that occur within a certain frequency spectrum. The technique's precision and adaptability make it a powerful tool for extracting meaningful insights from complex datasets, enabling more accurate and reliable interpretations of time-series data.

This technique is used to remove noisy data by focusing on specific frequency ranges. This process effectively cleans up the data, making it clearer and more manageable for analysis.

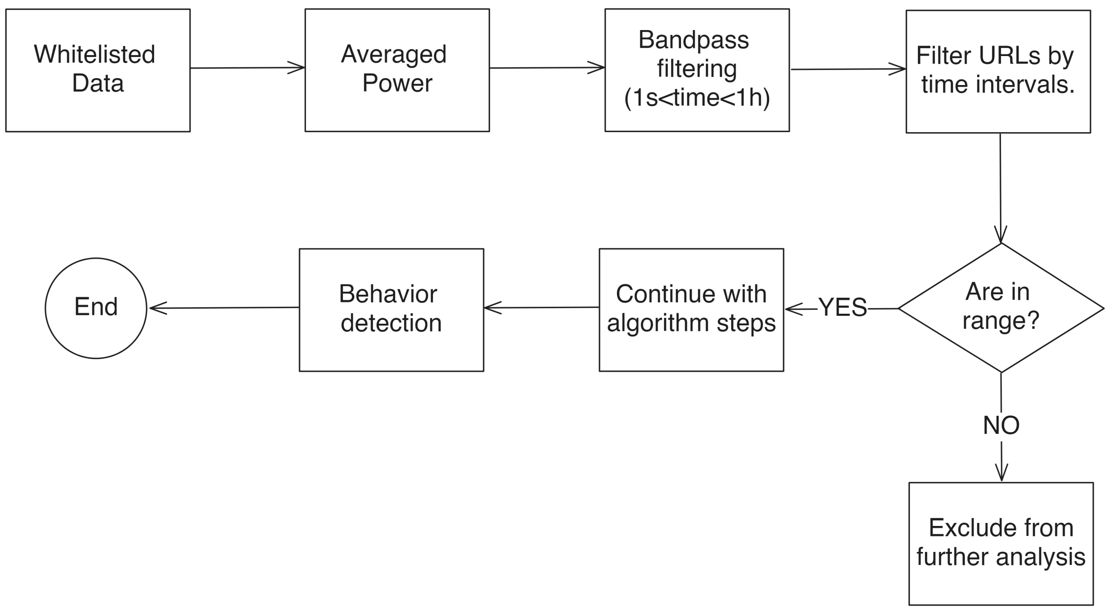

Whitelisted Data
Averaged Power
Bandpass filtering (1s<time<1h)
Filter URLs by time intervals.
Are in range?
YES
Continue with algorithm steps
Behavior detection
End
NO
Exclude from further analysis

Figure 5.3.: Average power calculation

Figure 5.3 illustrates the implementation of the bandpass filtering method, which is a sophisticated technique for refining the analysis of network traffic data. This method builds upon two critical steps from previous stages: the identification of whitelisted URLs and the calculation of averaged power values. The algorithm sets a lowcut frequency at 1 second and a highcut frequency at 1 hour, establishing a specific frequency range for the filtering process. By applying these criteria, the bandpass filter analyzes the time intervals associated with each URL. URLs whose time intervals fall within this predefined range are retained for further scrutiny, while those outside the range are systematically excluded. This selective filtering process is key as it hones in on URLs that demonstrate temporal patterns of interest, thereby aligning with the research objectives. By focusing the analysis on relevant frequency components, the bandpass filter enhances the precision and accuracy of the subsequent data examination. This targeted approach not only reduces the complexity and volume of data but also amplifies the significance of the retained information. The process ensures that only URLs exhibiting meaningful temporal dynamics are considered, facilitating a more nuanced understanding of network behaviors. This method is particularly valuable in cybersecurity research, where identifying and interpreting subtle variations in traffic patterns can reveal potential security threats or anomalies. The refined dataset resulting from this bandpass filtering method provides a robust foundation for further analysis, enabling researchers to derive more accurate and actionable insights from the data. By isolating the most relevant temporal patterns, the filtering process significantly improves the effectiveness of the overall analytical framework, making it a vital component in the comprehensive study of network traffic dynamics.

Function Signature

The function for applying bandpass filtering to the power values is defined as follows:

Listing 5.5: `Bandpass Filter` Function

```
def bandpass_filter(data, lowcut_time, highcut_time, sampling_rate, order=4):
```

Parameters:

- **data:** A list or array of power values representing the frequency of requests within specific time intervals.
- **lowcut_time:** An integer representing the lower boundary of the frequency range (in seconds). Components below this threshold are filtered out.
- **highcut_time:** An integer representing the upper boundary of the frequency range (in seconds). Components above this limit are filtered out.
- **sampling_rate:** A float representing the sampling rate of the data (in Hz). This is the rate at which data points are sampled per second.
- **order:** An integer specifying the order of the filter (default is 4). Higher-order filters have sharper cutoffs, providing better separation of frequencies, but may introduce more phase delay and computational complexity.

Function Definition

Listing 5.6: `bandpass_filter` Function

```
def bandpass_filter(data, lowcut_time, highcut_time, sampling_rate, order=4):
    nyquist = 0.5 * sampling_rate
    lowcut = lowcut_time / nyquist
    highcut = highcut_time / nyquist

    if lowcut >= 1 or highcut >= 1:
        raise ValueError("filter critical frequencies must be 0 < Wn < 1")
    b, a = butter(order, [lowcut, highcut], btype='band')
    return filtfilt(b, a, data)
```

Functionality

1. **Frequency Normalization:** Calculate the Nyquist frequency, which is half the sampling rate. Normalize the low and high cut-off frequencies by dividing them by the Nyquist frequency.

2. **Frequency Validation:** Ensure that the normalized frequencies are within the valid range $0 < Wn < 1$. Raise a `ValueError` if they are not.

3. **Filter Design:** Use the Butterworth filter design (`butter` function) to create a band-pass filter with the specified order.

4. **Data Filtering:** Apply the filter to the data using the `filtfilt` function, which performs forward and backward filtering to minimize phase distortion.

5. **Return Filtered Data:** Return the filtered data, which contains only the frequency components within the specified range.

Performance Considerations

The `bandpass_filter` function involves several steps, each with its own performance implications:

- **Time Complexity:** The filtering process involves creating the filter coefficients and applying the filter to the data. The `butter` function has a time complexity of $O(n)$, where n is the order of the filter. The `filtfilt` function has a time complexity of $O(m \cdot n)$, where m is the length of the data.

- **Space Complexity:** The function requires space to store the filter coefficients and the filtered data. The space complexity is $O(m)$, where m is the length of the data.

The function is efficient for moderate-sized datasets but may require optimization for very large datasets. Techniques such as parallel processing or more efficient filtering algorithms could be considered.

Role in Thesis

The `bandpass_filter` function is crucial for refining the power data by isolating significant frequency components within a specified range. This process enhances the accuracy of subsequent analyses by focusing on the most relevant data, reducing the impact of noise and irrelevant fluctuations. It provides a clearer understanding of the temporal patterns of network activity, which is essential for identifying meaningful trends and anomalies.

Results and Outputs

The function yields `filtered_data`, a list or array of power values that fall within the specified frequency range. This refined dataset is more suitable for detailed analysis and interpretation, allowing for better identification of significant patterns and behaviors. By isolating the critical frequency components, the bandpass filtering process contributes to a more robust understanding of the dynamics within the network.

5.5. Behavior Detection

In the last step, as shown in Figure 5.4, the algorithm reaches its final stage, which is behavior detection. This stage is pivotal in discerning the relevance and significance of the URLs retained after the filtering process. To accurately identify potentially malicious or anomalous URLs, the algorithm requires the establishment of a threshold value. This threshold is not arbitrary; it is determined through a combination of extensive experimentation and leveraging past experiences. The experimentation phase involves testing various threshold levels against historical data to observe their effectiveness in accurately flagging suspicious activities without generating excessive false positives. Past experiences, particularly insights gleaned from previous network security incidents and the operational context of the network, also play a role in fine-tuning this threshold. By integrating empirical data with historical knowledge, the threshold is calibrated to optimize the balance between sensitivity and specificity. Once set, this threshold becomes a benchmark against which the behavior of URLs is measured. URLs that exhibit characteristics surpassing this threshold are flagged for further investigation, as they may indicate potential security threats or deviations from normal network behavior. This final detection step is essential for transforming the filtered data into actionable intelligence, enabling network administrators and security professionals to focus their attention on the most critical and relevant threats. By effectively filtering out noise and highlighting significant anomalies, the behavior detection stage enhances the overall efficacy of the cybersecurity framework, ensuring that the network remains secure against evolving threats.

The output of the algorithm, as shown in Figure 5.5, delivers a targeted overview of URLs that require detailed examination. This list is produced by comparing each URL against a predefined threshold of 500, with those exceeding this threshold being flagged for potential concerns. URLs identified as surpassing this threshold are marked for closer investigation due to their deviation from normal behavior, which could indicate possible security threats or unusual activity. This alert system is crucial for prioritizing high-risk URLs, enabling security analysts to focus their efforts on the most significant issues. By efficiently filtering out less critical data, the

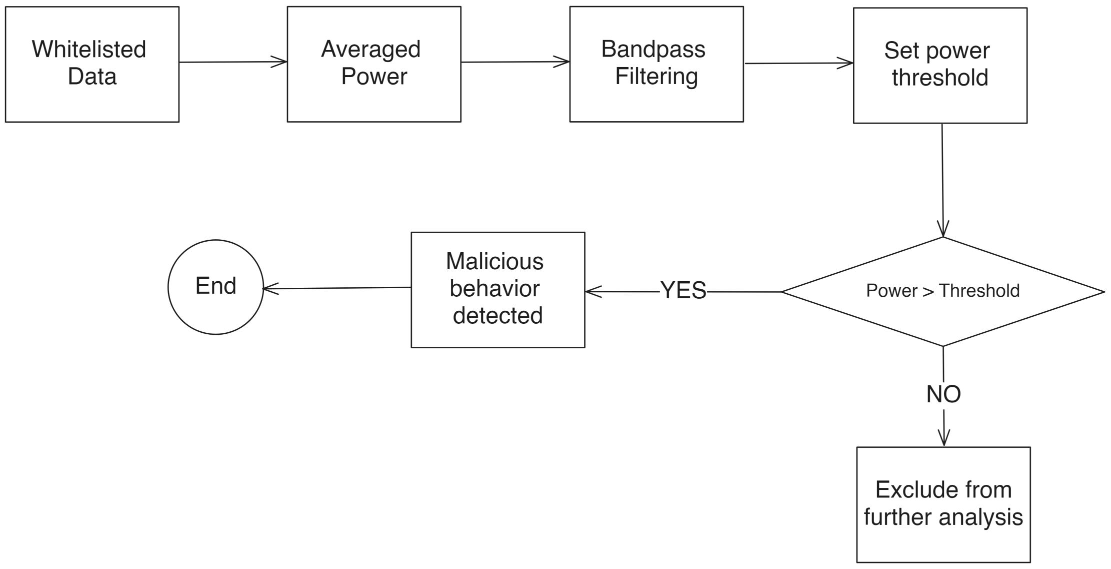


Figure 5.4.: Behavior Detection

system enhances threat detection accuracy and reduces false positives. Analyzing these flagged URLs can reveal hidden patterns or attack methods that may otherwise be missed. Thus, this proactive alert mechanism is vital for effective threat management, facilitating early detection and response to mitigate risks and safeguard the network against potential breaches.

The algorithm's output demonstrates its effectiveness in reinforcing network security by promptly identifying and addressing potential threats.

5.6. 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:**

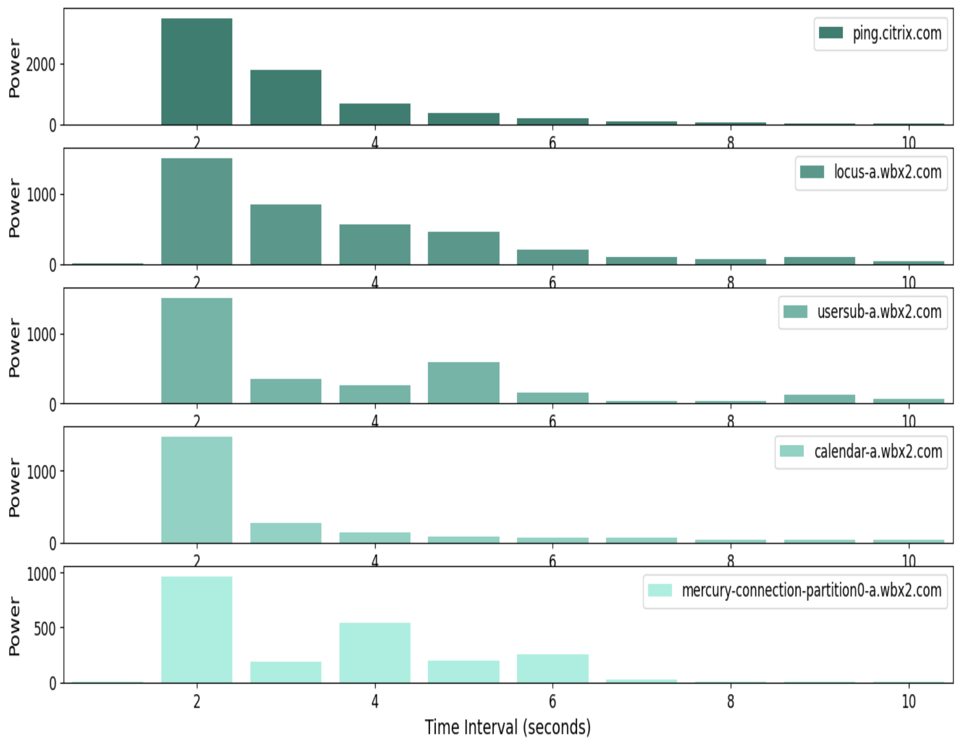


Figure 5.5.: Steps of the proposed method

- 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.

5.7. 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.

6. 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.

6.1. 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.

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.

6.1.1. In-Depth Analysis of Algorithm Output

To assess the accuracy of the algorithm's identification of malicious behavior, a verification process was executed, which involved systematically evaluating the algorithm's outputs against known benchmarks. The initial findings from this process unveiled patterns that hinted at possible malicious behavior, indicating that the algorithm was effectively identifying anomalies consistent with malicious activities. This verification process is critical for ensuring the reliability and robustness of the algorithm in real-world applications, as it confirms the algorithm's ability to detect subtle and complex patterns indicative of potential security threats.

To further refine the focus of the analysis, Figure 6.1 provides a detailed snapshot of the data from a particular day, highlighting specific areas where suspicious activity may be present. This figure showcases the URLs identified by the algorithm, each accompanied by a distinct time interval and power level. The next critical step involves detecting the behavior of these URLs to accurately report beaconing malicious behavior. This process requires checking the power levels of these URLs against a predefined threshold value. The threshold value was selected after numerous experiments and relies heavily on the analyst's expertise and experience to ensure its accuracy and reliability.

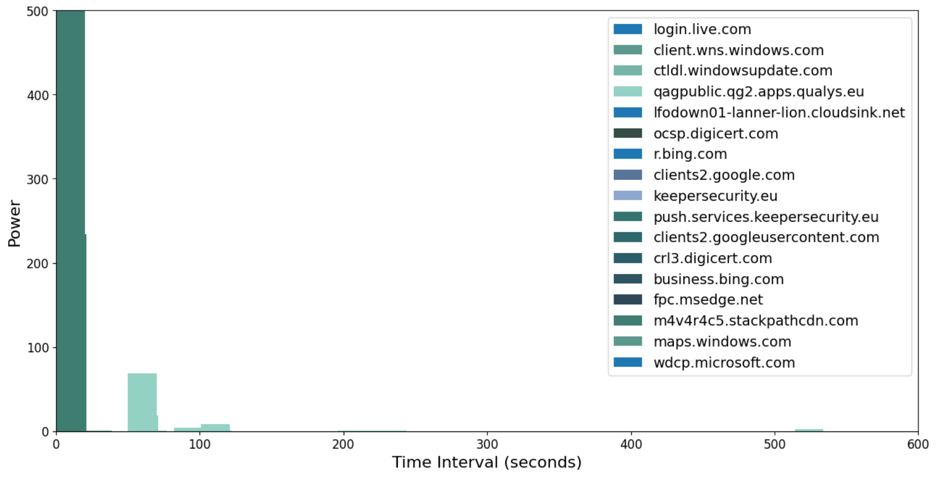


Figure 6.1.: Testing Data

To enhance clarity and isolate the most significant indicators of potential malicious activity, Figure 6.2 presents only those URLs that have exceeded the established threshold, which in this instance is set at 500. This strategic filtering immediately draws attention to the URL 'm4v4r4c5.stackpathcdn.com', which exhibits behavior that warrants closer scrutiny due to its elevated power levels. By isolating this particular domain from the broader dataset, the figure reveals a significant peak during a specific time interval, indicating a moment of heightened activity.

This peak is particularly noteworthy as it signifies that during the specified time interval, the power associated with 'm4v4r4c5.stackpathcdn.com' reached an unusually high value of 2877. This substantial power level is well above the predetermined threshold, clearly indicating an abnormal and potentially malicious pattern of activity. The suspicious behavior was observed within the time interval of 10-12, marking this period as a critical window for further investigation.

The isolation and examination of these peaks are important for understanding the nature of the potential threat. By focusing on the time intervals and power levels that exceed the threshold, analysts can more effectively identify and interpret patterns indicative of beaconing malicious behavior. This detailed approach underscores the importance of combining algorithmic outputs with the analyst's expertise to detect and respond to potential security threats accurately. It highlights how the rigorous verification process, alongside the use of targeted metrics and thresholds, enhances the ability to discern meaningful signals from noise, thereby improving the overall effectiveness of the beaconing detection methodology. This thorough analysis not only aids in identifying immediate threats but also contributes to the ongoing refinement of detection techniques, ensuring they remain robust and reliable in the face of evolving web traffic scenarios and potential malicious activities.

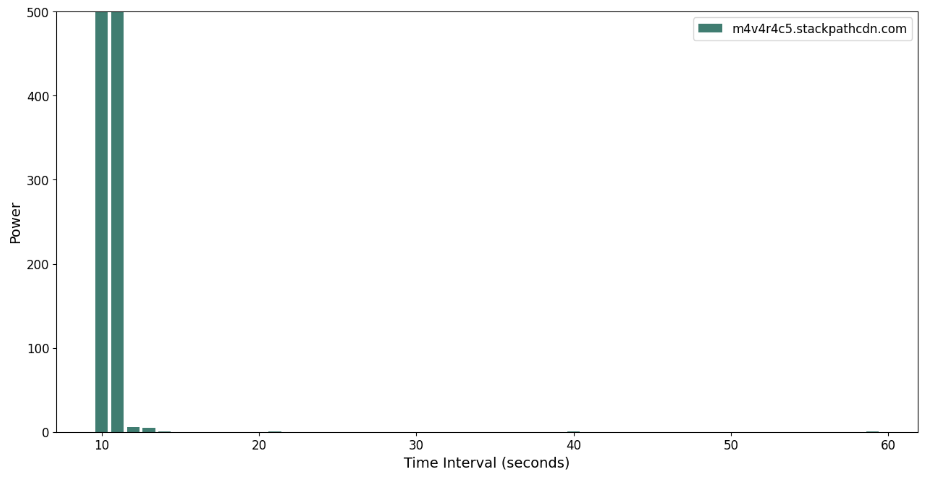


Figure 6.2.: Malicious Domain

Upon the identification of the URL, the subsequent analysis unveiled clear indicators suggesting malicious intent within the algorithm's output. To verify the authenticity of these suspicions, a comprehensive inquiry was launched into the nature of the detected behavior, aimed at determining whether it unequivocally originated from a malicious URL source. This inquiry involved a detailed examination of the URL's historical footprint, which uncovered a disconcerting pattern of activity. It became evident that a particular user had persistently engaged in phishing tactics, repeatedly attempting to access and manipulate the URL for nefarious purposes. Such deliberate and systematic actions underscored the malicious nature of the user's intentions. Recognizing the severity of the situation, immediate action was taken to alert cybersecurity experts, thereby initiating a thorough examination and swift resolution of the identified security breach.

The investigation revealed that the user behind the malicious activity employed sophisticated techniques to mask their actions, making detection more challenging. This necessitated a deeper dive into the user's digital footprint, examining IP addresses, timestamps, and the nature of the requests made to the URL. The collected evidence pointed to a concerted effort to exploit vulnerabilities within the system, highlighting the importance of the initial algorithmic detection and the subsequent manual analysis in identifying and mitigating the threat.

Furthermore, to thoroughly evaluate the algorithm's effectiveness, a comprehensive analysis was conducted on the data gathered across multiple days. This longitudinal study allowed for the identification of recurring patterns or evolving trends that may signify malicious intent. The subsequent sections present the results derived from the algorithm's examination of each day's data, providing insights into the consistency and reliability of the algorithm in detecting suspicious activities. By comparing daily outputs, analysts could identify not only persistent threats but also new and emerging ones, thereby enhancing the overall security posture.

The analysis also involved cross-referencing the detected malicious activities with known threat databases to determine if the identified URL and user behavior matched any previously recorded cyber threats. This step was key in understanding the broader context of the threat and in developing appropriate countermeasures. The collaboration with cybersecurity experts ensured that the findings were promptly addressed, and preventative measures were implemented to safeguard against future attacks.

In summary, the identification of the URL and the subsequent in-depth analysis highlighted the importance of a multi-layered approach to cybersecurity. The combination of advanced algorithms, historical data analysis, and expert intervention provided a robust framework for detecting and responding to malicious activities. This approach not only addressed the immediate threat but also contributed to the ongoing refinement of detection techniques, ensuring they remain effective against evolving cyber threats.

The figures below provide comprehensive snapshots of online activity on different days, highlighting key patterns and anomalies. In the figure 6.3, a detailed log documents the internet activity of a specific individual on a particular day. This log outlines the various websites visited, including both innocuous sites and one that raised suspicions. The figure shows the output of an algorithm designed to detect malicious behavior, marking the beginning of a detailed analysis.

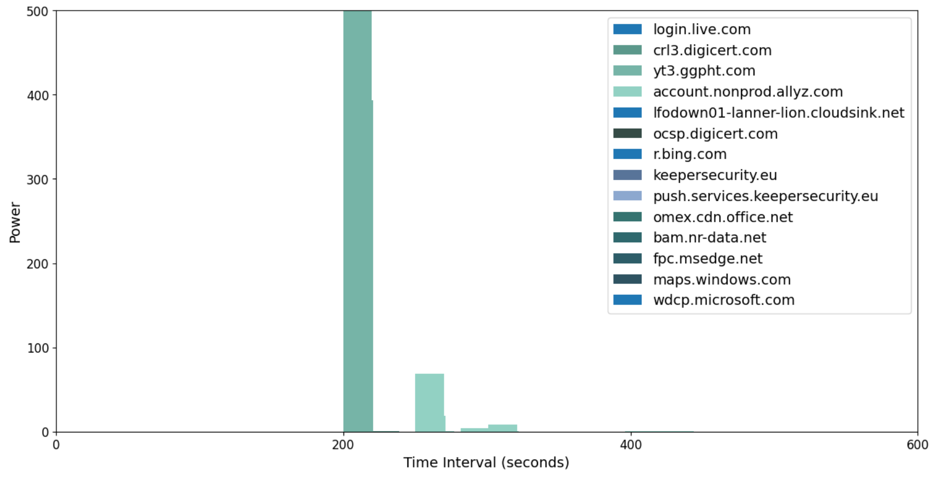


Figure 6.3.: Testing Data

The focus of this analysis is to look over all URLs that exhibit a specific power level. The algorithm identifies URLs with notable peaks in the output, particularly those with power levels exceeding a predetermined threshold value. This threshold was established through extensive experimentation and expert analysis, ensuring its effectiveness in filtering out benign activity while highlighting potential threats.

Shifting the focus to Figure 6.4, this figure zooms in on a specific timeframe, offering a closer examination of a particular website's behavior. During the period from 200 to 220, there was a notable surge in activity on ’yt3.ggpht.com’, which deviated significantly from its usual patterns. This irregularity prompted a deeper analysis, revealing indications of potentially malicious beaconing behavior.

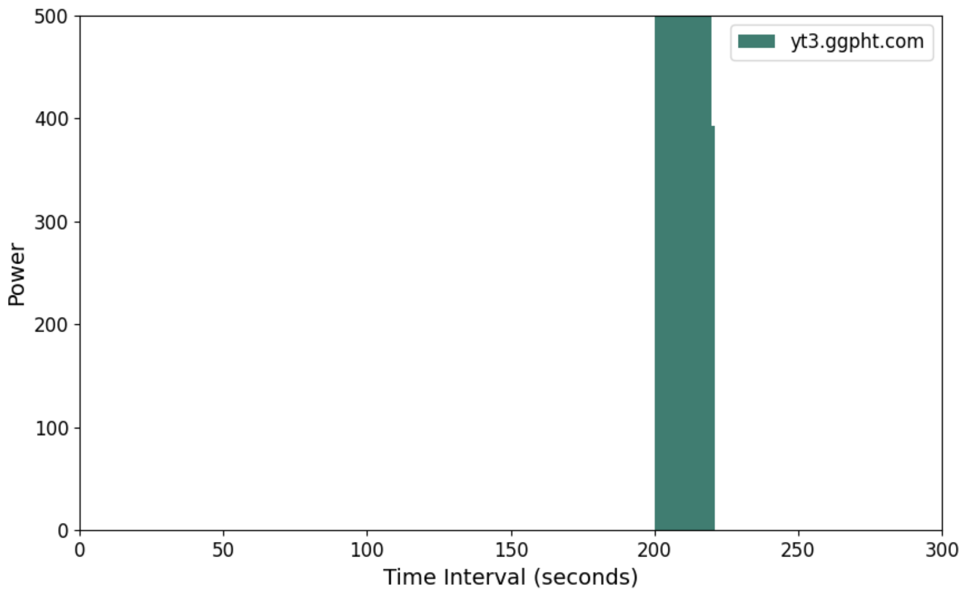


Figure 6.4.: Malicious Domain

The subsequent investigation into ’yt3.ggpht.com’ confirmed the presence of unauthorized activity on the website. This finding underscores the critical importance of such monitoring systems in identifying and addressing cybersecurity threats. By highlighting irregular patterns and behaviors, these systems act as vigilant guardians, ensuring the safety and integrity of digital spaces.

These visual representations play a vital role in maintaining online security. They not only facilitate the detection of malicious activity but also aid in the timely response to emerging threats. The detailed logs and focused analyses presented in Figures 6.3 and 6.4 exemplify the effectiveness of combining algorithmic detection with expert scrutiny experiences. This approach ensures that suspicious activities are not only identified but also thoroughly investigated and mitigated.

The integration of advanced algorithms and detailed visual representations provides a robust framework for monitoring and securing online environments. By continuously analyzing web traffic and identifying anomalies, these systems help protect against unauthorized activities and potential cyber threats. This multi-layered approach is essential for maintaining the integrity and security of digital spaces in an increasingly connected world.

7. 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.

7.1. 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:

7.1.1. 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.

7.1.2. 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.

7.1.3. 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.

7.2. 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:

7.2.1. 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.

7.2.2. 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.

7.2.3. 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.

7.2.4. 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.

8. Conclusion and Future Work

8.1. 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 essential 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.

8.2. 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

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

```python
from influxdb_client import InfluxDBClient
from datetime import datetime
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt

# Function to calculate request power
def calculate_request_power(request_list):
    power_dictionary = {}
    last_date_time = request_list[0]["_time"]

    for request_dict in request_list:
        current_date_time = request_dict["_time"]

        # Check if current_date_time is a string, and convert it to a datetime object
        if isinstance(current_date_time, str):
            current_date_time = datetime.strptime(current_date_time, "%Y-%m-%dT%H:%M:%S.%fZ")

        time_delta = int((current_date_time - last_date_time).total_seconds())

        # Add the power to the power dictionary
        power_dictionary[time_delta] = power_dictionary.get(time_delta, 0) + 1
        last_date_time = current_date_time

    # Sort the dictionary for better visualization
    power_dictionary = dict(sorted(power_dictionary.items()))

    return power_dictionary

# Function to apply bandpass filtering in terms of time
def bandpass_filter(data, lowcut_time, highcut_time, sampling_rate, order=4):
    nyquist = 0.5 * sampling_rate
    lowcut = lowcut_time / nyquist
    highcut = highcut_time / nyquist

    if lowcut >= 1 or highcut >= 1:
        raise ValueError("Digital filter critical frequencies must be 0 < Wn < 1")

    b, a = butter(order, [lowcut, highcut], btype='band')
    filtered_data = filtfilt(b, a, data)
    return filtered_data

# InfluxDB connection details
url = "http://localhost:8086"
token = "*****"
org = "Student"
bucket = "Net"
influx_username = '*****'
influx_password = '*****'

try:
    # Create an InfluxDB client
    client = InfluxDBClient(url=url, token=token, org=org)

    # Query data from InfluxDB
    query = f'from(bucket:"{bucket}") |> range(start: 2023-08-01T00:00:00Z,
        stop: 2023-08-02T00:00:00Z)'
    tables = client.query_api().query(query, org=org)

    # Extract points from the result
    points = [record.values for table in tables for record in table.records]

    # Process and organize the InfluxDB data
    print("Processing InfluxDB Data:")
    extracted_influx_objects = {}

    for point in points:
        url_hostname = point.get("url_hostname")

        # Check if url_hostname is already in the dictionary
        if url_hostname not in extracted_influx_objects:
            extracted_influx_objects[url_hostname] = []

        # Append under the corresponding url_hostname
        extracted_influx_objects[url_hostname].append(point)

        # Print the extracted InfluxDB data for debugging
        print(point)

    # Create a whitelist to filter out unwanted URLs
    def create_whitelist(data, exclusion_criteria):
        whitelist = {url: requests for url, requests in data.items() if not
            any(exclusion in url for exclusion in exclusion_criteria)}
        return whitelist

    # Define exclusion criteria for URLs
    exclusion_criteria = ['allianz', 'res']

    # Create a whitelist based on the exclusion criteria
    whitelist = create_whitelist(extracted_influx_objects, exclusion_criteria
        )

```

```python
    print(”Filtered URLs based on whitelist:”)
    for url_hostname in whitelist:
        print(url_hostname)

    # Create a table of power for each URL hostname with bandpass filtering in terms of time
    print(”“nPower Table with Bandpass Filtering in Terms of Time:”)
    for url_hostname, requests in whitelist.items():
        print(f”URL Hostname: {url_hostname}”)
        power_dictionary = calculate_request_power(requests)

        # Print the power dictionary for debugging
        print(”Power Dictionary:”, power_dictionary)

        # Extract keys and values from the power dictionary
        time_intervals = list(power_dictionary.keys())
        power_values = list(power_dictionary.values())

        # Apply bandpass filtering in terms of time
        lowcut_time = 5  # 5 seconds
        highcut_time = 1000  # 1000 seconds

        # Check if there are enough elements to calculate sampling rate
        if len(time_intervals) >= 2:
            sampling_rate = 1.0 / (time_intervals[1] - time_intervals[0])  # Sampling frequency

            try:
                filtered_power_values = bandpass_filter(power_values, lowcut_time, highcut_time, sampling_rate)
            except ValueError as e:
                print(f”Error: {e}”)
                filtered_power_values = power_values

            # Print the filtered power values for debugging
            print(”Filtered Power Values:”, filtered_power_values)

            # Calculate the average power
            average_power = sum(filtered_power_values) / len(filtered_power_values)

            # Print the average power for debugging
            print(”Average Power:”, average_power)

            # Subtract average power from all power values
            adjusted_power_values = [power - average_power for power in filtered_power_values]

            # Print the adjusted power values for debugging
            print(”Adjusted Power Values:”, adjusted_power_values)

            # Remove negative powers
            non_negative_power_values = [max(0, power) for power in adjusted_power_values]

            # Get indices for the time range of interest (5 to 1000 seconds)
            time_range_indices = [i for i, t in enumerate(time_intervals) if
                5 <= t <= 1000]

            # Print the time range indices for debugging
            print("Time Range Indices:", time_range_indices)

            # Plot the adjusted data within the specified time range
            plt.plot([time_intervals[i] for i in time_range_indices], [non_
                negative_power_values[i] for i in time_range_indices], label=
                url_hostname)

    # Check if there are multiple URLs in the whitelist before creating
        legend
    if len(whitelist) > 1:
        plt.legend()

    plt.xlabel("Time Interval (seconds)")  # Change x-axis label
    plt.ylabel("Adjusted Power")  # Change y-axis label
    plt.title("Adjusted Power over Time")  # Change the chart title
    plt.show()

except Exception as e:
    print(f"An error occurred: {e}")
```

A.2. Data Analysis Implementation

```python
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt

# Function to calculate request occurrence
def calculate_request_occurrence(request_list):
    occurrence_dictionary = {}
    last_date_time = None

    for _, request_row in request_list.iterrows():
        try:
            current_date_time = pd.to_datetime(request_row["_time"])

            if last_date_time is not None:
                time_delta = int((current_date_time - last_date_time).total_
                    seconds())

                # Add the occurrence to the occurrence dictionary
                occurrence_dictionary[time_delta] = occurrence_dictionary.get
                    (time_delta, 0) + 1

            last_date_time = current_date_time

        except (ValueError, TypeError) as e:
            print(f”Error in row: {_}, Timestamp value: {request_row['_time']}, Error message: {e}”)

    # Sort the dictionary for better visualization
    occurrence_dictionary = dict(sorted(occurrence_dictionary.items()))

    return occurrence_dictionary

# Function to apply bandpass filtering in terms of time
def bandpass_filter(data, lowcut_time, highcut_time, sampling_rate, order=4):
    nyquist = 0.5 * sampling_rate
    lowcut = lowcut_time / nyquist
    highcut = highcut_time / nyquist

    if lowcut >= 1 or highcut >= 1:
        raise ValueError(”Digital filter critical frequencies must be 0 < Wn < 1”)

    b, a = butter(order, [lowcut, highcut], btype='band')
    filtered_data = filtfilt(b, a, data)
    return filtered_data

# CSV file path for the cleaned and modified data
csv_file_path = r'C:“Allianz“4“1125“Modified_Beaconing.csv'

try:
    # Read data from CSV file and explicitly convert ”_time” to datetime
    df = pd.read_csv(csv_file_path)
    df[”_time”] = pd.to_datetime(df[”_time”], format='%H:%M:%S.%f', errors='coerce')

    # Process and organize the CSV data
    print(”Processing CSV Data:”)
    extracted_csv_objects = {}

    for _, row in df.iterrows():
        url_hostname = row.get(”url_hostname”)

        # Check if url_hostname is already in the dictionary
        if url_hostname not in extracted_csv_objects:
            extracted_csv_objects[url_hostname] = []

        # Append under the corresponding url_hostname
        extracted_csv_objects[url_hostname].append(row)

        # Print the extracted CSV data for debugging
        print(row)

    # Create a whitelist to filter out unwanted URLs
    def create_whitelist(data, exclusion_criteria):
        whitelist = {url: requests for url, requests in data.items() if not any(exclusion in url for exclusion in exclusion_criteria)}
        return whitelist

    # Define exclusion criteria for URLs
    exclusion_criteria = [’allianz’, ’res’]

    # Create a whitelist based on the exclusion criteria
    whitelist = create_whitelist(extracted_csv_objects, exclusion_criteria)

    print(”Filtered URLs based on whitelist:”)
    for url_hostname in whitelist:
        print(url_hostname)

    # Create a table of occurrence for each URL hostname with bandpass filtering in terms of time
    print(”“nOccurrence Table with Bandpass Filtering in Terms of Time:”)
    peak_url_hostname = None
    peak_occurrence_value = 0
    for url_hostname, requests in whitelist.items():
        print(f”URL Hostname: {url_hostname}”)
        occurrence_dictionary = calculate_request_occurrence(pd.DataFrame(requests))

        # Print the occurrence dictionary for debugging
        print(”Occurrence Dictionary:”, occurrence_dictionary)

        # Identify peak occurrence value and corresponding URL hostname
        if occurrence_dictionary:
            max_occurrence_value = max(occurrence_dictionary.values())
            if max_occurrence_value > peak_occurrence_value:
                peak_occurrence_value = max_occurrence_value
                peak_url_hostname = url_hostname

        # Extract keys and values from the occurrence dictionary
        time_intervals = list(occurrence_dictionary.keys())
        occurrence_values = list(occurrence_dictionary.values())

        # Apply bandpass filtering in terms of time
        lowcut_time = 5  # 5 seconds
        highcut_time = 1000  # 1000 seconds

        # Check if there are enough elements to calculate sampling rate
        if len(time_intervals) >= 2:
            sampling_rate = 1.0 / (time_intervals[1] - time_intervals[0])  # Sampling frequency

            try:
                filtered_occurrence_values = bandpass_filter(occurrence_values, lowcut_time, highcut_time, sampling_rate)
            except ValueError as e:
                print(f”Error: {e}”)
                filtered_occurrence_values = occurrence_values

            # Print the filtered occurrence values for debugging
            print(”Filtered Occurrence Values:”, filtered_occurrence_values)

            # Calculate the average occurrence
            average_occurrence = sum(filtered_occurrence_values) / len(filtered_occurrence_values)

            # Print the average occurrence for debugging
            print(”Average Occurrence:”, average_occurrence)

            # Subtract average occurrence from all occurrence values
            adjusted_occurrence_values = [occurrence - average_occurrence for occurrence in filtered_occurrence_values]

            # Print the adjusted occurrence values for debugging
            print(”Adjusted Occurrence Values:”, adjusted_occurrence_values)

            # Remove negative occurrences
            non_negative_occurrence_values = [max(0, occurrence) for occurrence in adjusted_occurrence_values]

            # Get indices for the time range of interest (5 to 1000 seconds)
            time_range_indices = [i for i, t in enumerate(time_intervals) if 5 <= t <= 1000]

            # Print the time range indices for debugging
            print(”Time Range Indices:”, time_range_indices)

            # Plot the adjusted data within the specified time range
            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)

    # Print the URL hostname with the peak occurrence
    if peak_url_hostname:
        print(f”“nURL Hostname with Peak Occurrence: {peak_url_hostname}”)
        print(f”Peak Occurrence Value: {peak_occurrence_value}”)

    # Check if there are multiple URLs in the whitelist before creating legend
    if len(whitelist) > 1:
        plt.legend()

    plt.xlabel(”Time Interval (seconds)”)  # Change x-axis label
    plt.ylabel(”Adjusted Occurrence”)  # Change y-axis label
    plt.title(”Adjusted Occurrence over Time”)  # Change the chart title
    plt.show()

except Exception as e:
    print(f”An error occurred: {e}”)
```

Bibliography

[1] Bitdefender, "Global cybersecurity threat map," accessed: 2024-08-13. [Online]. Available: https://threatmap.bitdefender.com/

[2] 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," *IntechOpen*, August 2021, reviewed: 07 July 2021, Published: 19 August 2021.

[3] InfluxData, "Influxdb 3.0 system architecture," accessed: 2024-08-13. [Online]. Available: https://www.influxdata.com/blog/influxdb-3-0-system-architecture/

[4] Y. Hu, Y. Li, X. Liu, X. Zhang, and L. Xu, "Baywatch: Robust beaconing detection to identify infected hosts in large-scale enterprise networks," *IEEE Access*, vol. 4, pp. 3689–3703, 2016.

[5] H. Zhu, W. Liu, K. Wang, Y. Hu, and D. Xue, "Global analysis with aggregation-based beaconing detection across large campus networks," *Computers, IEEE Transactions on*, vol. 67, no. 2, pp. 163–175, 2018.

[6] X. Yu, Z. Zhang, Z. Wu, and C. Liang, "Identifying malicious hosts involved in periodic communications," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 318–329, 2017.

[7] W. Cheng, Y. Zhang, J. Xiao, and Z. Li, "Abnormal behavior detection to identify infected systems using the apchain algorithm and behavioral profiling," *Journal of Computer Security*, vol. 26, no. 3, pp. 223–245, 2018.

[8] Y. Liu, H. Zhang, J. Sun, and S. Liu, "Periodic behavior in botnet command and control channels traffic," *International Journal of Network Security*, vol. 19, no. 2, pp. 268–279, 2017.

[9] H.-W. Chiu, S.-J. Chien, and H.-C. Liao, "Uncovering periodic network signals of cyber attacks," *Computer Networks*, vol. 101, pp. 108–120, 2016.

[10] Y. Li, W. Song, C. Wu, C. Liu, and X. Yang, "Detecting malicious beaconing communities using lockstep detection and co-occurrence graph," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1133–1146, 2018.

[11] Q. Xu, Y. Hu, and L. Li, "Apt beaconing detection: A systematic review," *IEEE Access*, vol. 7, pp. 174 013–174 027, 2019.

[12] M. Hagan, B. Kang, K. McLaughlin, and S. Sezer, "Peer-based tracking using multi-tuple indexing for network traffic analysis and malware detection," *IEEE Transactions on Information Forensics and Security*, 2020.

[13] A. Shalaginov, K. Franke, and X. Huang, "Malware beaconing detection by mining large-scale dns logs for targeted attack identification," *Computers & Security*, vol. 88, p. 101682, 2019.

[14] 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 *2015 10th Asia Joint Conference on Information Security (AsiaJCIS).* IEEE, 2015.

[15] Y. Borchani, "Advanced malicious beaconing detection through ai," *Computers & Security*, vol. 110, p. 102100, 2021.