Technische Hochschule Deggendorf
Fakultät Angewandte Informatik

Studiengang Master Angewandte Informatik

Leistungsbewertung des BAYWATCH-Frameworks: Eine Evaluierung mit realen und künstlichen Daten

BAYWATCH Framework Performance: An Evaluation with Real and Artificial Data

Masterarbeit zur Erlangung des akademischen Grades:

Master of Engineering (M.Eng.)

an der Technischen Hochschule Deggendorf

Vorgelegt von:
Aida Nikkhah Nasab
Matrikelnummer: 22208964

Am: March 2025

Prüfungsleitung:
Prof. Dr. Fischer

Ergänzende Prüfende:
Zineddine Bettouche

Erklärung

Name des Studierenden: Aida Nikkhah Nasab

Name des Betreuenden: Prof. Dr. Fischer

Thema der Abschlussarbeit:

Leistungsbewertung des BAYWATCH-Frameworks: Eine Evaluierung mit realen und künstlichen Daten . . . .

. . . .

. . . .

. . . .

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 interconnected digital landscape, Advanced Persistent Threats (APTs) exploit stealthy beaconing behavior to evade detection, posing significant risks to enterprise networks. This thesis investigates the performance of the BAYWATCH framework in identifying APTs by analyzing periodic communication patterns within extensive network log data.

The study employs a signal analysis pipeline that combines Fast Fourier Transform (FFT) for frequency-domain detection with autocorrelation function (ACF) for time-domain verification. This dual approach ensures robust identification of periodicities, even under noisy conditions. To systematically evaluate resilience, synthetic datasets with programmable jitter (2–150 seconds) and beacon intervals (10–300 seconds) are generated, alongside validation using real-world enterprise network traces. Key innovations include permutation-based FFT thresholding, bandpass filtering, and frequency-lag correlation, collectively improving detection accuracy while minimizing false positives.

Experiments demonstrate the framework's efficacy, achieving rapid processing times (less than 10 seconds post-filtering) and reliable detection across diverse scenarios.

This work contributes a scalable, efficient solution for early APT detection, validated in both controlled and operational environments. Future directions include real-time streaming analysis, machine learning integration for anomaly detection, and extension to IoT and cloud infrastructures. The thesis advances proactive cybersecurity strategies, offering a practical tool to safeguard large-scale networks against evolving threats.

Contents

Contents

1 Topical Overview

1.1 Problem Statement

In today's interconnected digital landscape, safeguarding information and securing network systems are critical priorities. Modern enterprises generate vast amounts of user log data daily, containing valuable insights yet presenting significant challenges in differentiating benign activities from potential security threats. As cyberattacks—particularly Advanced Persistent Threats (APTs)—become increasingly sophisticated, there is a pressing need for robust and proactive cybersecurity measures. A key challenge lies in efficiently analyzing massive volumes of log data to detect and mitigate malicious activities before they cause substantial harm to network infrastructures. One common tactic employed by APTs is beaconing behavior, where compromised systems communicate with command-and-control (C2) servers at regular intervals. Identifying such covert communication patterns is important for early threat detection. This thesis focuses on the detection of beaconing behavior to enhance cybersecurity in large-scale network environments.

1.2 Research Objectives

The primary objective of this research is to advance network security by developing improved methods for early detection and rapid response to potential cyber threats, with a particular emphasis on APTs. Specific research objectives include the development of advanced detection techniques that leverage distinctive behavioral patterns, the implementation of proactive security measures to enable swift responses, and the enhancement of beaconing behavior analysis through an innovative signal analysis pipeline. In addition, the research aims to continuously evaluate and improve security policies based on both real-world and synthetic data.

1.3 Research Questions

The research is guided by several key questions, including:

- How can beaconing behavior be effectively detected within large-scale network data to provide early warning of potential threats?

- What is the impact of periodicity in network communications on distinguishing between benign and malicious activities?

- Is the beaconing behavior detectable in generated synthetic data, and how does its detectability compare to that in real-world data?

1.4 Structure of the Thesis

The thesis is organized into a cohesive narrative that begins by establishing the foundational background and core concepts essential to understanding network security and periodicity detection. Following this, a review of related work contextualizes the current research within the broader field. The methodology chapter then details the advanced techniques introduced in the framework. Chapter 5, Data Analysis is an exploration of real-world network log data to uncover patterns and insights related to beaconing behavior, setting the stage for subsequent evaluations. Chapter 6 represents a detailed description of the procedures and techniques employed to generate synthetic beaconing data, which is used to validate the performance of the detection framework under controlled conditions. Chapter 7 represents Evaluation and Results. An investigation and comparison of the framework's performance on both real and synthetic data, summarizing key findings and contributions, and discussing potential improvements. Finally, Chapter 8 represents Conclusions and Future Work. It presents the overall conclusions of the research, outlines the contributions made, and proposes directions for future research in the field of network security.

The source code and implementation details of this research are available in the following repository: `https://mygit.th-deg.de/an28964/master-thesis`

2 Background

This chapter provides the foundational knowledge for understanding the context and significance of this research. It begins with an overview of the cybersecurity landscape, emphasizing the current state, emerging trends, and persistent challenges organizations face. 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, which is for detecting anomalies in cybersecurity contexts. On top of that, the chapter represents 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. Finally, the chapter introduces the BAYWATCH framework, which serves as the foundation for the research by providing a structured approach to detecting beaconing behavior in network traffic.

The field of cybersecurity is continually evolving, with new threats emerging as technology advances. Understanding these threats and the strategies to counter them is important for protecting sensitive information, ensuring operations continuity, and maintaining enterprise networks' integrity. 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 an ever-evolving and increasingly complex environment, where cyber threats continuously adapt to exploit emerging technologies and vulnerabilities. Organizations across industries face persistent challenges in securing their networks, data, and critical infrastructure against a wide array of threats, ranging from malware and ransomware to highly coordinated Advanced Persistent Threats (APTs) and nation-state-sponsored attacks. As adversaries refine their techniques, security strategies must evolve to stay ahead of these rapidly shifting threats.

Several key factors shape the modern cybersecurity landscape, making threat detection and mitigation increasingly challenging. The rapid digitization of industries, the widespread adoption of cloud computing, and the proliferation of Internet of Things (IoT) devices have significantly expanded the attack surface, providing adversaries with new entry points to exploit. While these technological advancements offer substantial benefits in terms of scalability and efficiency, they also introduce new vulnerabilities, often outpacing the ability of traditional security solutions to keep up.

Additionally, the emergence of cybercrime-as-a-service models, such as Ransomware-as-a-Service (RaaS) and Phishing-as-a-Service (PhaaS), has lowered the barrier to entry for cybercriminals, allowing even unskilled attackers to launch sophisticated campaigns with minimal effort. Threat actors leverage automation, AI-driven attack techniques, and deepfake technologies to enhance the effectiveness of their operations, making it more difficult for conventional security measures to detect and neutralize threats in real time.

Another critical challenge in cybersecurity is the shortage of skilled security professionals. As attacks grow in complexity, organizations struggle to recruit and retain experts capable of defending against sophisticated adversaries. This talent gap is exacerbated by the increasing demand for expertise in specialized areas such as threat hunting, digital forensics, AI-driven security analytics, and adversarial attack simulations. As a result, organizations are turning to automation, machine learning-driven anomaly detection, and advanced behavioral analytics to augment human capabilities and enhance their defensive strategies. Given these challenges, the need for detection frameworks is more urgent than ever.

This research focuses specifically on the detection of beaconing behavior, a common characteristic of command-and-control (C2) communications used in APTs, botnets, and malware infections. By analyzing periodic patterns in network traffic, this study aims to develop effective methodologies for identifying stealthy cyber threats and mitigating their impact. The next sections will further explore the theoretical foundations and practical applications of periodicity detection in cybersecurity, setting the stage for the proposed detection framework.
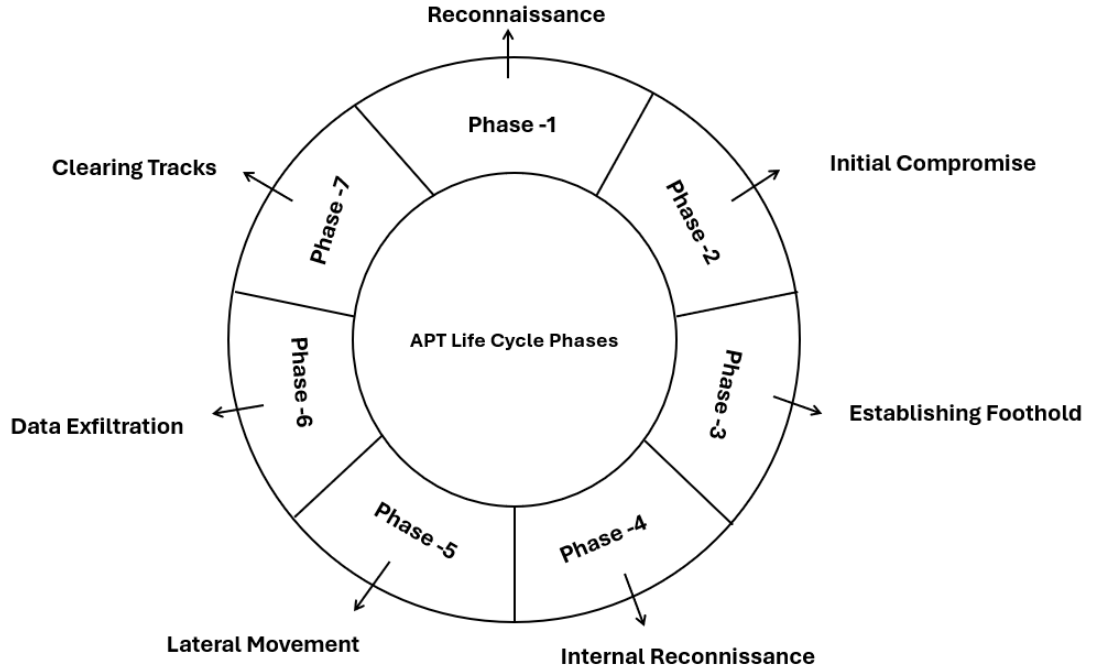
Figure 2.1: 7 phases of APT attack lifecycle [1]

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, posing significant risks to organizations, governments, and critical infrastructure. Unlike conventional cyber attacks that are often opportunistic and short-lived, APTs involve prolonged, highly targeted operations conducted by state-sponsored actors, cybercriminal syndicates, or advanced hacking groups.

What sets APTs apart is their stealth and persistence. Attackers typically employ multi-stage intrusion strategies, beginning with initial reconnaissance, followed by infiltration through zero-day exploits, spear-phishing campaigns, or supply-chain attacks. Once inside the target network, they establish command-and-control (C2) infrastructure, allowing them to maintain access over extended periods while evading detection. To achieve this, APT actors leverage sophisticated evasion techniques, including polymorphic malware, encrypted C2 channels, and lateral movement tactics to expand their foothold within the network.

A defining characteristic of APT campaigns is their low-and-slow approach, where attackers deliberately limit their activities to avoid triggering security alerts. Unlike other malware, which may cause immediate disruptions, APTs focus on long-term intelligence gathering, strategic espionage, and covert data exfiltration, often remaining undetected for months or even years. The consequences of a successful APT attack can be severe, leading to financial losses, reputational damage, intellectual property theft, and even national security threats.

Given their stealthy nature, detecting APTs requires advanced behavioral analysis and anomaly detection techniques rather than relying solely on signature-based security measures. One common indicator of APT activity is beaconing behavior, where compromised systems periodically communicate with remote C2 servers. Identifying these covert communication patterns within large-scale network traffic is a critical challenge in cybersecurity, making periodicity detection a valuable approach for uncovering hidden threats.

This research aims to enhance APT detection capabilities by focusing on beaconing behavior analysis in network traffic. By leveraging both real-world and synthetic data, the study explores effective methodologies for identifying periodic communication patterns, enabling organizations to detect and mitigate stealthy cyber threats before they cause significant harm.

Figure 2.1 illustrates the sequential stages of an APT attack, beginning with Phase 1 (Reconnaissance), where attackers gather target intelligence through open-source research or network scanning. Phase 2 (Initial Compromise) involves breaching defenses via methods like spear-phishing or exploiting vulnerabilities. Phase 3 (Establishing Foothold) ensures persistence through backdoors, credential theft, or malware installation. Phase 4 (Internal Reconnaissance) focuses on mapping the compromised network to identify high-value assets. Phase 5 (Lateral Movement) enables attackers to pivot across systems, escalating privileges to reach critical data. Phase 6 (Data Exfiltration) involves stealthily transferring stolen information via encrypted channels. Finally, Phase 7 (Clearing Tracks) erases forensic evidence (e.g., log deletion, tool removal) to evade detection. This progression underscores the need for frameworks like Baywatch to detect subtle, multi-stage behaviors—such as lateral movement, encrypted exfiltration, or log tampering—while balancing accuracy and scalability in real-world environments[1].

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 [2].

- **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 [3].

- **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 [4].

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 important for developing effective cybersecurity strategies.

Figure 2.2 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 [5].

- **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 [6].

- **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 [7].

- **Network Monitoring and Management:** Tools and practices for monitoring network traffic, identifying anomalies, and managing network resources to maintain performance and security.
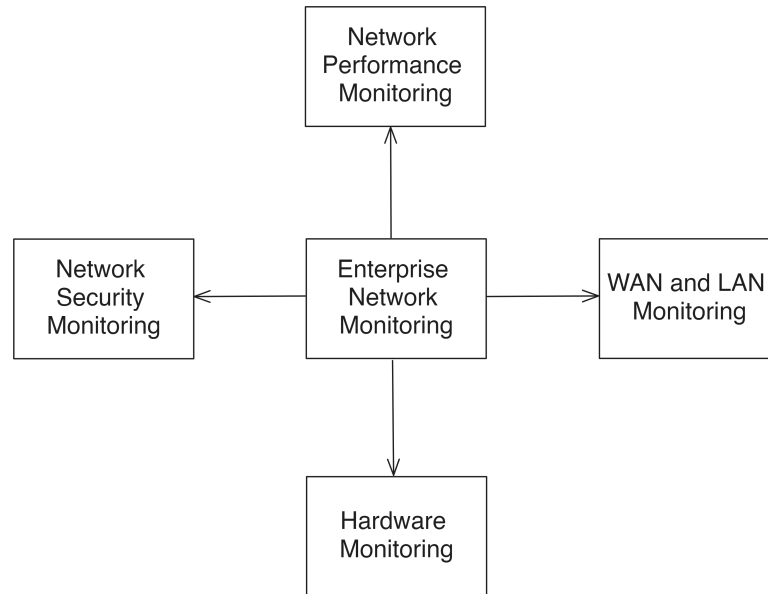
Figure 2.2: Enterprise network diagram

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 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 Time Series Databases and InfluxDB

Time-series databases (TSDBs) are optimized for storing and querying temporal data. In cybersecurity, they enable efficient analysis of network traffic patterns over time.

InfluxDB is a popular TSDB known for its high throughput and SQL-like query language (Flux) [8].

Key features include:

- Time-optimized storage for efficient data retrieval.

- Retention policies for automated data lifecycle management.

- Integration with visualization tools like Grafana.

InfluxDB supports real-time monitoring and historical analysis of network traffic, making it ideal for detecting anomalies like beaconing. For example, its ability to handle high-frequency timestamped data aligns with the BAYWATCH framework's requirements for processing large-scale network logs.
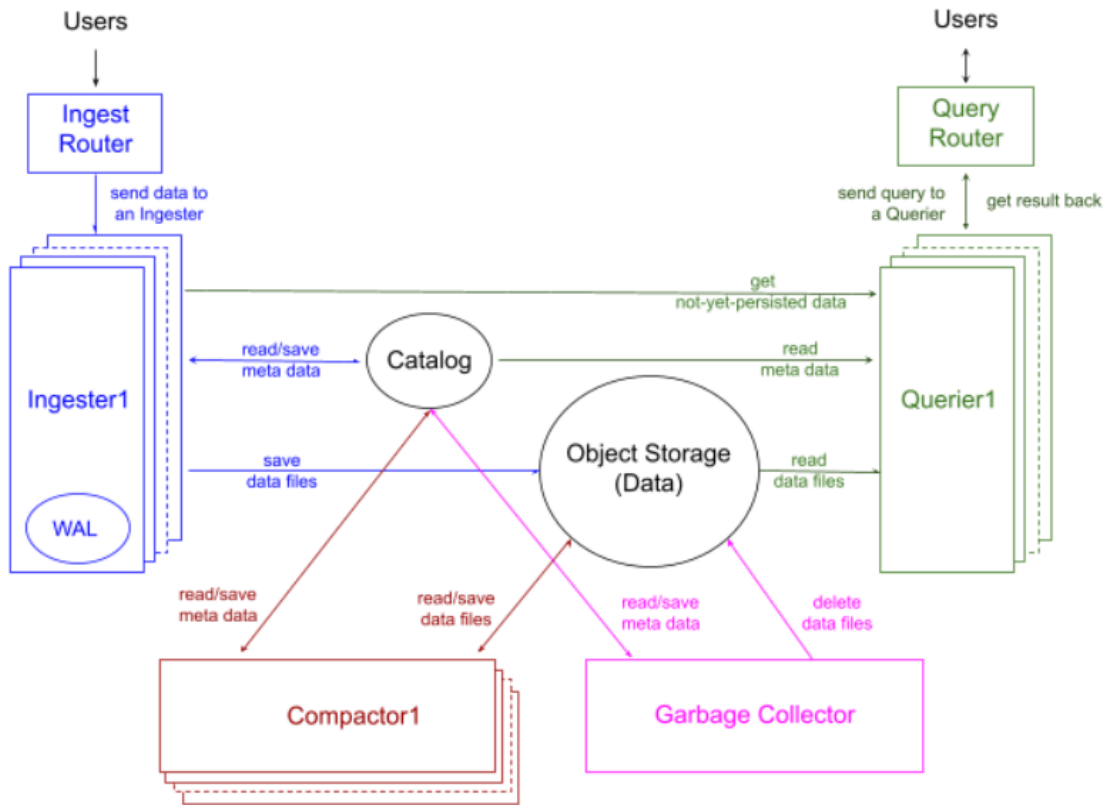
Figure 2.3: InfluxDB Architecture [8]

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 2.3 illustrates the workflow of a distributed system designed for high-throughput data ingestion and querying. Users initiate data ingestion via an Ingest Router, which routes incoming data to an Ingester node. The Ingester ensures durability by writing data to a Write-Ahead Log (WAL) and temporarily storing metadata and raw data files. A Compactor optimizes storage by merging and persisting data to Object Storage (e.g., cloud buckets) while updating the Catalog (metadata index) for efficient lookup. The Garbage Collector purges obsolete data files based on retention policies. For queries, Users submit requests through a Query Router, which directs them to a Querier node. The Querier retrieves results by combining real-time data from Ingesters (not yet persisted) and historical data from Object Storage, leveraging the Catalog for metadata navigation. This architecture emphasizes scalability (distributed Ingesters/Queriers), reliability (WAL), and cost-efficiency (Object Storage integration), while balancing low-latency access with persistent storage management.
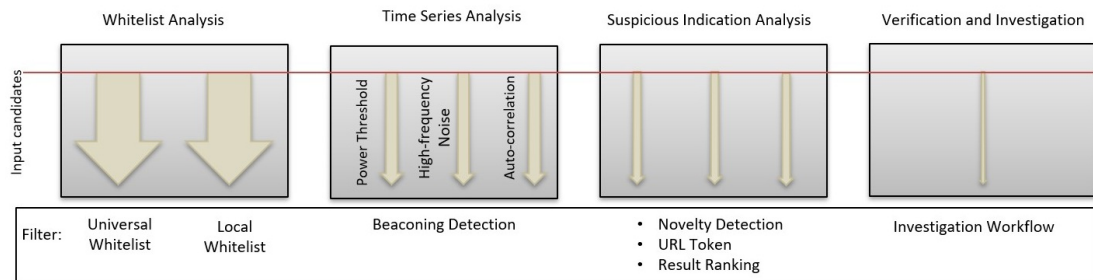
Figure 2.4: Algorithm steps

2.5 Overview of the BAYWATCH Framework

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

1. **Whitelist Analysis**: This phase eliminates known legitimate beaconing traffic using universal and local whitelists.

2. **Time Series Analysis**: This phase identifies periodic communication patterns in the remaining traffic using a robust periodicity detection algorithm.

3. **Suspicious Indicator Analysis**: This phase further filters out legitimate beaconing behavior by analyzing domain-specific indicators of malicious activity.

4. **Investigation and Verification**: The final phase involves manual investigation and verification of the remaining suspicious cases.

Figure 2.4 provides a detailed overview of the algorithm's processing steps, which occur in four distinct phases. In Phase 1, the input data undergoes whitelist analysis, where it is categorized into two separate whitelists. The Universal Whitelist contains common, globally trusted URLs such as major search engines (e.g., Google, Yahoo) and other widely recognized platforms. The Local Whitelist, on the other hand, includes URLs that are specifically trusted within the organization, such as internal Allianz resolution URLs. This step helps to filter out trusted sources, ensuring that only potentially suspicious or unknown URLs are subjected to further analysis in subsequent phases.

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

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

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

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

2.6 Whitelist Analysis

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

2.6.1 Universal Whitelisting

The universal whitelist contains globally trusted destinations, such as popular search engines, software update servers, and news feeds. These destinations are unlikely to be involved in malicious beaconing unless they are compromised. The whitelist is dynamic and can be updated with new trusted destinations. In the BAYWATCH framework, the universal whitelist is constructed using publicly available lists of popular domain names. These lists are curated based on the popularity and trustworthiness of the domains, ensuring that only well-known and widely used destinations are whitelisted.

2.6.2 Local Whitelisting

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

2.7 Time Series Analysis

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

2.7.1 Algorithm Overview

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

1. **Candidate Discovery**: This step identifies potential periodic components in the time series using the Fast Fourier Transform (FFT). The FFT converts the time series from the time domain to the frequency domain, allowing the algorithm to identify dominant frequencies that may correspond to periodic behavior.

2. **Pruning**: This step filters out high-frequency noise and less feasible candidates using statistical methods and bandpass filtering. The algorithm applies hypothesis testing to determine the statistical significance of the candidate periods and removes those that do not meet the significance threshold.

3. **Verification**: The final step verifies the validity of the remaining candidate periods using the circular autocorrelation function (ACF). The ACF measures the similarity between the time series and a shifted version of itself, providing a more fine-grained detection of periodic behavior.

2.7.2 Candidate Discovery Using FFT

The candidate discovery step begins by transforming the sequence of connection timestamps into a discrete time series. Let $T = \{t_1, t_2, \dots\}$ be the timestamps of connections between a communication pair. These timestamps are converted into an integer sequence $x(n) = \{x_0, x_1, \dots, x_{N-1}\}$, where $x_i > 0$ indicates that a connection occurred at time interval t_i, and $x_i = 0$ indicates no connection.

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

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

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

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

- The FFT can produce false positives for non-periodic signals, as it decomposes any signal into sinusoidal components.

- The FFT has limited resolution for low-frequency components, making it difficult to detect long-period beaconing.

- The FFT is sensitive to noise and interruptions in the time series, which can distort the periodogram.

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

Power Threshold for Candidate Frequencies

To address these limitations, the BAYWATCH framework uses a power threshold to distinguish between significant periodic signals and random noise. The power threshold is calculated using a permutation-based filtering approach, which estimates the maximum power that can be attributed to random noise in the time series.

The steps to calculate the power threshold are as follows:

1. **Permutation of the Time Series**: The original time series $x(n)$ is shuffled to create a permuted version $x'(n)$. This permutation destroys any periodic patterns in the time series while preserving its first-order statistics (e.g., amplitude).

2. **Periodogram of the Permuted Signal**: The FFT is applied to the permuted time series $x'(n)$, and the periodogram $P'(k)$ is computed. The maximum power $p_{max}(x')$ in the periodogram of the permuted signal is recorded.

3. **Repetition for Confidence**: To ensure a reasonable confidence level, the permutation process is repeated m times (e.g., $m = 20$). The $(C \times m)$-th highest power value (e.g., the 19th highest for $C = 95\%$ confidence) is selected as the power threshold p_T.

The power threshold p_T is then used to filter out insignificant frequencies in the original periodogram. Only frequencies with power $P(k) > p_T$ are considered as potential candidates for periodic behavior. This approach ensures that the algorithm focuses on frequencies that are unlikely to be caused by random noise.

2.7.3 Pruning Using Bandpass Filtering and Hypothesis Testing

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

Bandpass Filtering

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

Hypothesis Testing

The algorithm further prunes the candidate set by applying hypothesis testing to determine the statistical significance of the candidate periods. For each candidate period $\mathcal{P}$, the algorithm constructs a null hypothesis H_0: $\mathcal{P}$ is the true period of the time series. The observed intervals between connections are modeled as a normal distribution $N(\mathcal{P}, \sigma^2)$, where σ^2 represents the variance due to noise.

The algorithm performs a one-sample t-test on the observed intervals and calculates the p-value. If the p-value is below a significance threshold (e.g., $\alpha = 0.05$), the null hypothesis is rejected, and the candidate period is removed from the set. This step ensures that only statistically significant periods are retained for further analysis.

2.7.4 Verification Using Autocorrelation

The final step in the time series analysis is the verification of the remaining candidate periods using the circular autocorrelation function (ACF). The ACF measures the similarity between the time series and a shifted version of itself, providing a more accurate detection of periodic behavior.

The ACF is calculated as:

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

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

2.8 Suspicious Indicator Analysis

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

2.8.1 URL Path Token Filter

The URL path token filter analyzes the resource path within the URLs accessed by the hosts. The resource path often contains tokens that indicate the type of web resource being accessed. The filter measures the popularity of these tokens and filters out requests containing known and popular tokens, as they are unlikely to be involved in malicious beaconing.

2.8.2 Novelty Analysis

The novelty analysis step removes duplicate cases by filtering out source/destination pairs that have already been reported for beaconing behavior. This step ensures that only new and unique cases are passed on to the ranking algorithm for further analysis.

2.9 Investigation and Verification

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

2.9.1 Feature Set

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

2.9.2 Classifier

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

2.9.3 Bootstrapping Process

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

2.10 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 important for understanding the subsequent chapters, which will represent related work, methodology, implementation, experiments, and results. Additionally, this chapter has introduced the BAYWATCH framework, which serves as the core detection system for analyzing beaconing behavior. The BAYWATCH framework follows a structured multi-step process. In this work, using machine learning techniques like bootstrapping and classifiers is not implemented, and the focus is on time series analysis and periodicity detection in both real and synthetic data. The knowledge gained from this background will inform the development and evaluation of advanced techniques for detecting and mitigating cyber threats in enterprise networks.

3 Related Work

Hu et al. (2016) proposed BAYWATCH, a robust beaconing detection method designed to identify infected hosts in large-scale enterprise networks [9]. The method focuses on detecting beaconing behavior, which is commonly exhibited by compromised hosts communicating with external command and control servers. By analyzing network traffic patterns, BAYWATCH can efficiently detect infected devices while minimizing false positives. The system is specifically designed to scale in large enterprise environments, making it suitable for real-world deployment. The authors validate BAYWATCH through extensive evaluation using real-world network traffic, demonstrating its effectiveness in identifying infected hosts and improving network security.

Zhang et al. (2023) introduced a global analysis approach for aggregation-based beaconing detection across large campus networks [10]. Their method focuses on detecting beaconing behavior in network traffic by aggregating data from multiple sources within a campus network, which enhances detection accuracy. The approach is designed to scale across large networks and aims to minimize false positives by leveraging aggregation techniques. The authors validate their method through extensive experiments on real-world campus networks, demonstrating its effectiveness in identifying compromised hosts and improving network security in large-scale environments.

Apruzzese et al. (2017) proposed a method for identifying malicious hosts involved in periodic communications [11]. Their approach focuses on detecting abnormal periodic communication patterns in network traffic, which are often indicative of compromised hosts communicating with external servers. The authors introduce a novel technique for identifying such hosts by analyzing the timing and frequency of communication sessions. The proposed method is evaluated through experiments, demonstrating its effectiveness in identifying malicious hosts and enhancing network security by targeting irregular communication patterns.

Seo and Lee (2018) proposed an abnormal behavior detection method to identify infected systems using the APChain algorithm and behavioral profiling [12]. Their approach focuses on analyzing system behavior to detect deviations from normal activity, which may indicate the presence of malware or compromised systems. The APChain algorithm is used to model and track the behavior of systems, allowing for the identification of anomalous patterns associated with infected hosts. The authors validate their method through experiments, demonstrating its effectiveness in detecting abnormal behaviors and enhancing system security in real-world environments.

Huynh et al. (2016) focused on uncovering periodic network signals of cyber attacks [13]. The paper explores how periodic network traffic patterns can indicate cyber attacks, particularly in the context of detecting covert channels used by attackers for command and control. The authors propose a method to identify such periodic signals by analyzing network traffic over time. Their approach highlights the importance of periodicity in revealing malicious activity and introduces a visualization technique to facilitate the detection of these patterns. The study contributes to improving network security by enabling better detection of stealthy attack signals.

Jang et al. (2021) proposed a method for detecting malicious beaconing communities using lockstep detection and co-occurrence graphs [14]. The paper introduces an innovative approach to identifying groups of compromised hosts involved in coordinated beaconing behavior, a common indicator of malicious activity. By using lockstep detection and analyzing the co-occurrence of network events, the method can effectively pinpoint these malicious communities. The authors present this approach as part of a patent (US Patent 10,887,323), contributing to the detection of advanced persistent threats (APTs) and enhancing network security by identifying coordinated attacks.

Talib et al. (2022) conducted a systematic review on APT beaconing detection techniques [15]. The paper provides an extensive analysis of various methods used to detect Advanced Persistent Threats (APT) based on beaconing behavior, which is a common communication pattern in APT attacks. The authors review different detection techniques, including signature-based, anomaly-based, and machine learning methods, highlighting their strengths and limitations in identifying beaconing activities in network traffic. This review serves as a valuable resource for researchers and practitioners aiming to enhance APT detection and improve network security against sophisticated cyber threats.

Charan et al. (2021) explored the use of data mining and machine learning techniques for Advanced Persistent Threat (APT) attribution and detection in their study on DMAPT [1]. The paper focuses on the application of various data mining and machine learning methods to improve the identification and attribution of APTs, which

are often difficult to detect due to their stealthy nature. The authors discuss the effectiveness of different approaches in detecting APTs and their potential for enhancing threat detection capabilities in network security. The study provides valuable insights into the role of advanced analytics in tackling sophisticated cyber threats.

Hagan et al. (2018) proposed a peer-based tracking method using multi-tuple indexing for network traffic analysis and malware detection [16]. The approach aims to improve malware detection by analyzing network traffic patterns using multi-tuple indexing, which allows for more efficient tracking of peer interactions in the network. By examining the flow of traffic between different peers, the method identifies suspicious activities that may indicate the presence of malware. The authors validate their technique through experiments, demonstrating its effectiveness in detecting malicious traffic and enhancing network security by providing a more granular analysis of peer behavior.

Shalaginov et al. (2016) focused on malware beaconing detection by mining large-scale DNS logs for targeted attack identification [17]. The paper explores the use of DNS logs to detect beaconing behavior, which is commonly associated with malware communicating with external command and control servers. By analyzing large-scale DNS traffic data, the authors propose a method to identify targeted attacks based on the periodic patterns of beaconing. Their approach highlights the importance of leveraging DNS traffic for identifying malware infections, contributing to enhanced detection capabilities in large-scale network environments.

Yeh et al. (2018) investigated a malware beacon of botnet by analyzing local periodic communication behavior [18]. The paper focuses on identifying malware beaconing behavior in botnets by studying the periodic communication patterns between infected hosts and their command and control servers. The authors propose a method to detect these periodic behaviors, which are typically used by botnets to maintain control over compromised systems. Their approach highlights the importance of analyzing local traffic patterns for detecting botnet infections and contributes to improving malware detection techniques through the identification of communication anomalies.

Enright et al. (2022) introduced a learning-based zero-trust architecture for 6G and future networks [19]. The paper explores the integration of machine learning with zero-trust security models to address the evolving security challenges in next-generation networks, particularly 6G. The authors propose a framework that combines learning-based techniques with zero-trust principles to enhance the detection of malicious activity and improve overall network security. The study contributes to the development of more adaptive and robust security architectures for future networks, offering a promising solution to the emerging threats in 6G environments.

Van Ede et al. (2022) introduced Deepcase, a semi-supervised contextual analysis method for security events [20]. The paper presents a novel approach that combines semi-supervised learning techniques with contextual analysis to enhance the detection of security events. By leveraging contextual information, Deepcase can identify complex patterns and relationships in security data, improving the accuracy of event anomaly detection. The authors demonstrate the effectiveness of their approach in real-world security environments, showing its potential to enhance the detection and response capabilities of security systems in large-scale networks.

Ongun et al. (2021) introduced PORTFILER, a port-level network profiling approach for detecting self-propagating malware [21]. The paper presents a novel method that profiles network traffic at the port level to identify self-propagating malware, which often uses specific ports for communication and propagation. PORTFILER analyzes network behavior to detect irregularities and patterns associated with malware activity. By focusing on port-level communication, the approach improves malware detection, providing more accurate identification of self-propagating threats in real-time. The authors demonstrate the effectiveness of their method through experiments, showing its potential to enhance network security against rapidly spreading malware.

Niu et al. (2020) proposed a method for detecting malware on the Internet of Unmanned Aerial Vehicles (IoUAVs) by combining string matching and Fourier transformation techniques [22]. The paper addresses the growing concern of malware targeting UAV networks and introduces a hybrid approach that leverages string matching for identifying suspicious patterns in network traffic and Fourier transformation for analyzing periodic behaviors associated with malware. By combining these techniques, the authors enhance the detection accuracy of malicious activities, offering a more robust solution to securing UAV-based networks. The study contributes to the advancement of IoT security, particularly in the context of UAV systems, which are increasingly vulnerable to cyberattacks.

Duan et al. (2018) presented an approach for the automated generation and selection of interpretable features for enterprise security [23]. The paper focuses on improving enterprise security by developing methods for automatically generating and selecting meaningful, interpretable features from large datasets. These features can be used in security models to detect anomalies and potential threats more efficiently. The authors propose a framework that integrates automated feature engineering with machine learning techniques to enhance security monitoring systems. Their work contributes to the field by improving the interpretability and performance of security analytics,

making it easier for security teams to understand and respond to potential threats.

Haffey et al. (2018) focused on modeling, analyzing, and characterizing periodic traffic on a campus edge network [24]. The paper explores the behavior of periodic traffic patterns in campus networks, which are often indicative of scheduled communications, including those used by malware. The authors propose models to better understand and quantify these traffic patterns, helping to distinguish between legitimate and potentially malicious activities. Their work provides insights into how periodic traffic can be leveraged to enhance network security, particularly in the detection of botnets and other forms of malware that use regular communication intervals.

Recent research has focused on various aspects of enterprise security and malicious activity detection. Oprea et al. (2018) introduced MADE, a security analytics framework designed to enhance threat detection in enterprise environments [25] . The framework leverages advanced analytics to detect potential threats by analyzing large volumes of security data, enabling organizations to respond more effectively to cyber incidents. Ukrop et al. (2019) investigated the perception of IT professionals regarding the trustworthiness of TLS certificates, highlighting challenges in assessing certificate legitimacy and its implications for secure communications [26] . In a related study, Vissers et al. (2017) explored the ecosystem of malicious domain registrations within the .eu top-level domain (TLD), providing insights into the strategies used by attackers to exploit domain registration systems for malicious purposes [27] . Together, these works contribute to the broader understanding of security challenges in modern networks and propose solutions to improve detection and mitigation strategies.

4 Methodology

This chapter begins by presenting the data strategy rationale, outlining the significance of selecting appropriate data sources for analyzing periodic network behaviors. The discussion includes both real-world network logs and artificially generated datasets, emphasizing the importance of using a combination of these data sources to ensure comprehensive evaluation and validation of the detection framework. Following this, the chapter provides a detailed overview of the event log dataset, describing its structure, characteristics, and relevance to the study. Additionally, necessary preprocessing steps—such as data cleaning, normalization, and transformation—are discussed to ensure the dataset is well-prepared for analysis. Beyond real data, the chapter outlines the methodology used for generating synthetic datasets, which serve as a controlled testbed for validating the framework's effectiveness. The artificial data generation process is explained step by step, ensuring that it closely mimics real-world network traffic while allowing precise control over key parameters related to beaconing behavior.

4.1 Data Strategy Rationale

Real data represents actual network traffic, capturing the authentic, complex, and often noisy behavior of users in an enterprise environment. It reflects genuine usage patterns—including legitimate periodic traffic, gaps due to network delays or device outages, and inherent variability caused by diverse applications—that naturally emerge during normal operations. For example, noise can be introduced by network retransmissions and latency issues, while missing data often occurs when devices temporarily go offline or due to privacy-driven data filtering. These challenges arise organically and can obscure the detection of malicious beaconing behavior by blending with benign periodic activities.

In contrast, artificial data is generated under controlled conditions. Beaconing behavior is simulated with predetermined parameters (e.g., specific beacon frequencies and controlled jitter ranges), establishing a "ground truth" scenario where variations can be deliberately introduced. This controlled simulation replicates how challenges like noise and irregular intervals might manifest under different network conditions. By systematically varying these parameters, the sensitivity and robustness of the BAYWATCH framework can be precisely assessed, and the impact of specific challenges on detection accuracy can be thoroughly evaluated.

The combination of both data sources enables validation of the framework in authentic operational conditions (ensuring external validity) while also leveraging controlled simulations to isolate and address specific challenges. This dual approach ensures that the detection mechanism is both practical for real-world deployment and resilient against a broad spectrum of adversarial scenarios.

4.2 Real Data Source

The real-world data used in this study was collected from a large-scale enterprise network, specifically capturing user activities as they navigate through various URLs during their workday. This dataset offers an invaluable and detailed perspective on user interactions within the network, providing a comprehensive view of how users engage with web resources in a real-time operational environment. Each data entry is recorded, tracking specific user interactions, and is linked to precise timestamps, ensuring the ability to recreate a chronological sequence of activities. This feature is important for reconstructing browsing sessions and understanding the context in which different behaviors unfold. The dataset, stored in JSON format, is highly flexible and human-readable, which facilitates easier parsing, management, and analysis of large volumes of data. JSON's lightweight nature also makes it well-suited for representing nested information, ensuring that detailed records of user activity can be organized efficiently without sacrificing accessibility. Each entry in the dataset contains key information, such as the URL visited, the time of access, and metadata related to the user's session, allowing for fine-grained analysis of individual behaviors and network traffic patterns. This detailed level of logging is important for identifying trends, anomalies,

and specific patterns within user behavior, which could provide valuable insights into network usage over time. It enables the identification of peak usage periods, frequent transitions between particular URLs, and patterns of repeated activity that may indicate consistent or habitual behavior. Additionally, such data can be instrumental in recognizing deviations from typical usage patterns, which may be indicative of malicious activities like beaconing or other types of cyber threats. Through this dataset, it becomes possible to conduct both broad trend analysis and detailed individual case studies, contributing significantly to the understanding of user behavior in an enterprise network context.

4.2.1 Data Structure and Schema

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

- `IP_Address`: The IP_Address of the user's device is recorded to provide a unique identifier for each host within the network. This field is for tracing the source of network traffic and understanding the specific device interactions with the network. By capturing the IP address, it becomes possible to track user activity across different sessions, correlate network events, and identify patterns of behavior. It also allows for detecting unusual or unauthorized access attempts.

- `logdate`: The date and time of the user interaction is recorded in a standardized date-time format, such as "2023-08-01T06:06:37.000Z." This format follows the ISO 8601 standard, which includes the year, month, and day (YYYY-MM-DD) followed by a "T" separator, the time in hours, minutes, and seconds (HH:mm:ss), and an optional milliseconds part (.000). The "Z" at the end indicates that the time is represented in Coordinated Universal Time (UTC).

- `url_hostname`: The hostname of the URL visited by the user is recorded in the dataset to track the specific websites or servers accessed during user interactions. This field provides valuable insights into the types of resources being requested, which can be used to identify patterns in browsing behavior, such as frequently visited websites or network traffic trends. By capturing the hostname, the dataset enables analysis of user activities at a higher level, helping to identify potential security concerns, such as access to malicious or unauthorized sites. This information can be used to detect anomalies in network traffic and enhance the accuracy of network monitoring and beaconing detection efforts.

- `user`: An optional field denoting the user identifier is included in the dataset to track individual user interactions. However, for privacy and security reasons, usernames are deliberately omitted during the import process to prevent the exposure of personally identifiable information (PII). Instead of using direct identifiers, anonymized or pseudonymous identifiers may be used to preserve user privacy while still enabling the analysis of user behaviors. This approach ensures that sensitive data is protected in compliance with data protection regulations, such as GDPR, while allowing for meaningful analysis of network activity and interactions.

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

```
1  { "$schema": "http://json-schema.org/draft-07/schema#",
2    "type": "object",
3    "properties": {
4      "logdate": { "type": "string", "format": "date-time"},
5      "urlhostname": { "type": "string"},
6      "user": { "type": "string"}},
7    "required": ["logdate", "urlhostname"] }
```

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

4.2.2 Data Collection and Scale

The dataset was collected over a single day, specifically on a typical workday, August 1, 2023 (Tuesday), generating nearly 73 gigabytes of information. This large-scale data collection captures a comprehensive range of details, providing an in-depth look at user interactions across various online platforms and services. The dataset encompasses the following key components:

- **Host Information**: The dataset records the IP addresses of user devices, offering a unique identifier for each host. This enables the tracking of individual devices, their browsing activities, and the overall flow of data across the network.

- **Timestamps**: Each user interaction is timestamped with high precision, capturing the exact date and time of the activity. This temporal data is for performing time-based analysis of browsing patterns, such as identifying peak usage periods, periods of inactivity, and frequent transitions between different URLs.

- **URL Hostnames**: The dataset logs the hostnames of the URLs visited by users, providing valuable insights into their browsing destinations. This information helps to identify the types of websites visited, potential external threats, or any anomalous traffic patterns indicating malicious activity.

- **User Interactions**: A chronological record of each user activity is maintained, allowing for the detailed reconstruction of browsing sessions. This data enables the identification of typical user behavior, which can then be compared against unusual patterns that might indicate beaconing or other cybersecurity threats.

The dataset's scale and granularity make it an ideal resource for analyzing user behavior across different time intervals. The detailed information captured, such as user activity, URL destinations, and precise timestamps, plays a pivotal role in identifying significant patterns, detecting anomalies, and forming the basis for robust beaconing detection strategies. The rich data offers the necessary insights to evaluate the effectiveness of different detection techniques and improve the cybersecurity framework for large-scale networks.

4.2.3 Data Management and Preprocessing

To effectively manage and analyze the collected dataset, a sophisticated data management system was implemented. This system leverages **InfluxDB**, a time-series database specifically optimized to handle high volumes of temporal data efficiently. Given the size and complexity of the dataset, careful planning was necessary to ensure seamless data processing, storage, and retrieval for subsequent analysis. The data management process was executed in several systematic steps to ensure that all data was appropriately handled, organized, and analyzed. These steps are outlined below:

1. **Data Import**: The first step in the data management process involves importing the dataset into InfluxDB. Custom Python scripts were developed to automate the entire import process, eliminating the need for manual intervention. These scripts are responsible for creating a dedicated "bucket" within InfluxDB, a specific storage container designed to organize and manage the incoming data. This automated approach not only streamlines the process but also minimizes the possibility of human error, ensuring that data is consistently organized in a structured and efficient manner.

2. **Schema Implementation**: To maintain the integrity and consistency of the data, a predefined schema was applied during the import process. The schema serves as a blueprint that defines the structure and format of the data, ensuring that all entries conform to the same standards. This step is in maintaining uniformity across the dataset and preventing errors during analysis. If any entry fails to meet the required format, it is automatically rejected by the system, and a validation message is generated. This message serves as a notification, allowing the system administrator to identify any issues and take corrective actions before proceeding with further data analysis.

3. **Initial Data Analysis**: After successfully importing the dataset, an initial analysis was conducted to gain a better understanding of its overall behavior and characteristics. This preliminary analysis focused on several key aspects of the data:

   - **Observing Data Trends**: A comprehensive examination of the dataset was conducted to identify overarching trends throughout the day. This step involves analyzing the time distribution of user interactions and identifying periods of peak activity as well as potential idle times.

- **Identifying Frequently Accessed URLs**: The most frequently accessed URLs were identified, and their corresponding access frequencies were calculated. This analysis provided valuable insights into user browsing behavior, helping to pinpoint commonly visited sites or services.
- **Analyzing Time Intervals**: The intervals between user requests were analyzed to uncover patterns such as frequent browsing sessions or longer periods of inactivity. This aspect of the analysis is key to identifying unusual activity that might indicate an anomalous pattern, such as beaconing.
- **Host Activity Distribution**: The distribution of hosts and their associated activity patterns were examined. This step enables the identification of trends specific to individual devices or users, offering further context for understanding network traffic behavior.

The process of inserting data into InfluxDB proved to be challenging due to the sheer volume of data being processed. It took nearly three days to complete the insertion on the hardware system, which was equipped with an Intel Core i5 processor, 16GB of RAM, and a 500GB SSD running Windows 11. The significant duration was primarily attributed to the high amount of data being imported, as well as the need for efficient data validation and schema enforcement. Despite the hardware's relatively moderate specifications, the system managed to handle the import process, though performance was impacted by the scale of the dataset and the complexity of the preprocessing steps.

4.2.4 Challenges with Real-World Data

Analyzing real-world network traffic presents several challenges that must be addressed to ensure accurate detection and analysis of beaconing behavior. Unlike synthetic datasets, real-world data is influenced by numerous external factors, such as network congestion, varying user behaviors, and environmental inconsistencies, making it necessary to apply robust preprocessing and analytical techniques.

Noise and Variability

Real-world network traffic is inherently noisy, with fluctuations in connection timing caused by factors such as network delays, packet loss, retransmissions, and load balancing mechanisms. These unpredictable variations can obscure meaningful periodic patterns, making it difficult to identify beaconing behavior accurately. To address this, preprocessing techniques such as smoothing filters, moving averages, and statistical normalization are applied to reduce the effects of noise while preserving periodic trends.

Missing Data

Incomplete data is a common challenge when dealing with real-world traffic. Devices may disconnect, move out of the observation range, or experience network failures, leading to gaps in the dataset. These missing entries can distort pattern recognition and reduce the reliability of beaconing detection. Additionally, robust analytical models that can tolerate incomplete datasets, such as probabilistic approaches, help ensure that missing data does not significantly impact detection accuracy.

Legitimate Periodic Traffic

One of the most significant challenges in beaconing detection is distinguishing between malicious activity and legitimate periodic communication. Many benign applications, including email polling, cloud synchronization, system health checks, and software update mechanisms, exhibit regular intervals of network activity that may resemble beaconing behavior. To prevent false positives, a multi-layered approach is used, incorporating behavioral profiling, frequency analysis, and contextual evaluation. This involves analyzing supplementary metadata, such as destination IP addresses, communication frequency, protocol usage, and packet payload characteristics, to identify deviations from expected benign behavior.

By addressing these challenges through rigorous preprocessing, statistical modeling, and contextual analysis, the impact of real-world data inconsistencies is minimized, leading to more accurate and reliable beaconing detection in large-scale enterprise networks.

4.3 Artificial Data Source

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

4.3.1 Design of Artificial Data

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

- **Host Information**: The IP addresses of the user devices, enabling the tracking of individual hosts and their activities. Since this is generated data, the IP address is set to 127.0.0.1 for all synthetic beacons.

- **Timestamps**: The precise date and time of each user interaction, enabling temporal analysis of browsing patterns. The timestamps of the generated data are aligned with the real dataset, specifically on August 1, 2023 (Tuesday), to maintain consistency in temporal analysis.

- **URL Hostnames**: The hostnames of the visited URLs, providing insights into the destinations of user traffic. The format of these generated URLs follows the pattern "beacon{i}.example.com", where "i" represents a unique identifier for each synthetic beacon.

- **User Interactions**: A chronological record of user activities, facilitating the identification of trends and anomalies. The sequence of interactions is designed to resemble real-world browsing behavior to ensure realism in pattern analysis.

- **Is Artificial**: A tag (labeled as "yes") was added to distinguish the artificial data from real-world data. This tag ensures that the synthetic data can be easily identified and separated during analysis.

To ensure consistency and reliability across all generated entries, the structure of the JSON files for artificial data is governed by a Document Type Definition (DTD). Below is an example of the JSON schema used for defining the dataset:

```
1   { "$schema": "http://json-schema.org/draft-07/schema#",
2     "type": "object",
3     "properties": {
4       "logdate": { "type": "string", "format": "date-time"},
5       "urlhostname": { "type": "string"},
6       "user": { "type": "string"}},
7       "IsA": { "type": "string" },
8     "required": ["logdate", "urlhostname", "IsA"] }
```

4.3.2 Jitter and Beacon Frequency Variations

In simulating real-world beaconing behavior, two critical parameters are varied: beacon frequency and jitter. Beacon frequency refers to the regular interval at which a beacon signal is transmitted, while jitter introduces randomness into these intervals to mimic natural network variations or deliberate obfuscation tactics.

Beacon Frequency Intervals:
The following beacon frequency intervals were utilized in the simulations:

$$\text{Intervals: } [10, 20, 30, 40, 50, 60, 120, 300] \text{ seconds}$$

Each interval represents a different rate of beacon transmission, ranging from high-frequency (every 10 seconds) to low-frequency (every 5 minutes). This variation allows for the assessment of the detection framework's sensitivity across a spectrum of beaconing behaviors.

Jitter Ranges:
To introduce variability into the beaconing intervals, the following jitter ranges were applied:

Jitter ranges: $[2, 5, 10, 120, 150]$ seconds

These jitter values add a random variation to each beacon interval, simulating conditions from minimal timing fluctuations to substantial deviations. For instance, a jitter of 2 seconds introduces slight randomness, while a jitter of 150 seconds represents significant variability, which may cause the beaconing behavior to be undetectable. By systematically varying both beacon frequency and jitter, the robustness of the BAYWATCH framework is evaluated under diverse conditions, ensuring its effectiveness in detecting beaconing behaviors across a range of real-world scenarios.

Combined Impact of Frequency and Jitter:
The interplay between beacon frequency and jitter is important. By systematically varying both beacon frequency and jitter, the robustness of the BAYWATCH framework is evaluated under diverse conditions, ensuring its effectiveness in detecting beaconing behaviors across a range of real-world scenarios. It will be presented in the implementation chapter.

4.3.3 Scenarios Tested

To evaluate the robustness and effectiveness of the BAYWATCH framework, a comprehensive set of scenarios was designed by varying both beacon frequency intervals and jitter ranges. This dual-parameter variation ensures that the framework is tested under conditions that closely mimic real-world network behaviors. By systematically combining the beacon frequency intervals with the jitter ranges, multiple scenarios were created to test the BAYWATCH framework. For instance:

High-Frequency, Low-Jitter Scenario: Beacon interval of 10 seconds with a jitter of 2 seconds.

Moderate-Frequency, Moderate-Jitter Scenario: Beacon interval of 60 seconds with a jitter of 10 seconds.

Low-Frequency, High-Jitter Scenario: Beacon interval of 300 seconds with a jitter of 150 seconds(half of the interval).

Each combination presents unique challenges, allowing for a thorough assessment of the framework's capability to detect beaconing behavior under varying conditions.

Different malware or legitimate applications may exhibit varying beacon frequencies. Testing across a spectrum of intervals ensures that the framework can accurately detect both rapid and infrequent beaconing behaviors. Real-world network conditions introduce randomness in communication timings. By incorporating different jitter levels, the framework's resilience to timing variations and its ability to distinguish between regular and irregular patterns are assessed. Through this multifaceted testing approach, the BAYWATCH framework's robustness and adaptability to diverse network behaviors are thoroughly evaluated.

4.3.4 Integration with Real-World Data

To evaluate the effectiveness of the BAYWATCH framework, artificial data was integrated alongside real-world network traffic. This hybrid approach allows for a comprehensive assessment by leveraging the strengths of both data types. The real-world dataset captures the complexity and unpredictability of live network environments, offering insights into how the framework performs under realistic conditions. In contrast, the artificial dataset provides controlled scenarios, enabling rigorous testing of specific beaconing patterns, edge cases, and known attack behaviors. A critical component of this integration is the "is Artificial" tag, which explicitly marks synthetic data entries. This tag ensures that artificial records can be easily distinguished from genuine network traffic during preprocessing and analysis. It prevents unintentional contamination of results while allowing direct comparisons between the detection rates of known synthetic beacons and real-world network anomalies. By combining real and artificial data, the evaluation process benefits from both the unpredictability of real-world conditions and the precision of controlled simulations. This dual approach ensures that the BAYWATCH framework remains robust to noise, variations, and legitimate periodic traffic while maintaining high accuracy in detecting malicious beaconing activity. Additionally, the artificial data serves as a validation benchmark, allowing the refinement of detection thresholds and feature extraction techniques before deployment in production environments.

4.4 Summary

This chapter outlines the methodology for evaluating the BAYWATCH framework, combining real-world enterprise network logs and synthetic datasets to ensure comprehensive validation under both authentic and controlled conditions. Real data, collected from a large-scale enterprise network, captures the inherent complexity of user interactions—including noise, missing entries, and legitimate periodic traffic—while artificial data simulates beaconing behavior with precise parameters (e.g., intervals of 10–300 seconds, jitter ranges of 2–150 seconds) to establish ground truth scenarios. The real dataset, structured in JSON format with fields for IP addresses, timestamps, and URL hostnames, underwent rigorous preprocessing (e.g., schema validation, noise reduction) and was managed via InfluxDB for efficient time-series analysis, despite hardware limitations during its three-day ingestion. Challenges such as distinguishing malicious beaconing from benign periodic activities were addressed through multi-layered filtering and contextual metadata analysis. Artificial data, tagged for easy identification, tested the framework's robustness against diverse evasion tactics, including high-jitter and low-frequency beaconing. By integrating synthetic data into real traffic, the methodology enabled a dual evaluation: real-world data assessed external validity, while controlled simulations isolated performance metrics (e.g., detection sensitivity, false-positive rates). This hybrid approach ensured BAYWATCH's adaptability to both unpredictable network environments and sophisticated adversarial strategies, setting a foundation for empirical validation in subsequent chapters.

5 Data Analysis

This chapter represents the detailed analysis of the dataset, focusing on understanding user behavior, temporal patterns, and network interactions. By employing advanced visualization techniques and statistical methods, this chapter aims to uncover meaningful insights into the dataset's structure, identify patterns, and detect potential anomalies. The analysis is divided into four main sections: **Visualization of URL Request Counts**, **24-Hour URL Visit Analysis**, **Time Interval Analysis of URL Requests**, and **Distribution of Hosts Based on Unique URLs Contacted**. Each section provides a comprehensive exploration of the data, supported by visualizations and detailed interpretations.

5.1 Visualization of URL Request Counts

Understanding the distribution and frequency of URL requests is for identifying patterns, detecting anomalies, and distinguishing between normal and potentially suspicious browsing behavior. Analyzing these request patterns provides insights into user interactions, network load, and possible beaconing activity. This section presents visualizations of URL request counts using both logarithmic and linear scales to facilitate a comprehensive comparison of visit frequencies across different URLs. The logarithmic scale is particularly useful for handling large variations in request counts, allowing infrequent and high-volume URLs to be analyzed side by side without extreme disparities overshadowing meaningful trends.

Figure 5.1 illustrates the distribution of request counts across different URL hostnames, providing insights into traffic patterns within the dataset. The x-axis represents the index of URLs sorted in descending order of request count, ensuring that the most frequently accessed URLs appear on the left. The y-axis, displayed on a logarithmic scale, captures the wide range of request frequencies, making it easier to observe both high-traffic and low-traffic URLs within the same visualization. The figure highlights a steep drop-off in request counts, where a small subset of URLs receives significantly higher traffic while the majority of URLs are accessed infrequently. This pattern suggests a power-law distribution, a well-documented phenomenon in network and web traffic analysis, where a few highly popular destinations dominate overall traffic while long-tail URLs contribute to a smaller proportion of requests. By visualizing this distribution, the analysis enables a clearer understanding of user browsing behavior, network load distribution, and potential anomalies. Identifying the most visited URLs and their relative popularity supports further investigation into traffic patterns, content access trends, and potential indicators of automated or malicious activity.

5.2 24-Hour URL Visit Analysis

Understanding the temporal patterns of URL visits is for analyzing user behavior, identifying peak usage periods, detecting anomalies, and optimizing network resource allocation. By examining how URL requests fluctuate over 24 hours, valuable insights can be gained into traffic trends, workload distribution, and deviations from expected activity. This section presents an analysis of the temporal distribution of URL visits, highlighting variations in user activity throughout the day. Identifying peak access times helps in understanding common usage patterns, such as increased traffic during working hours or spikes related to scheduled automated processes. Additionally, periods of unusually high or low activity may indicate network congestion, scheduled maintenance, or potential security threats such as automated scanning or beaconing behavior. By visualizing these trends, the study enables a more comprehensive assessment of network dynamics, facilitating proactive resource management and anomaly detection. This analysis provides a foundation for optimizing infrastructure performance and enhancing network security by distinguishing between normal fluctuations and suspicious activity.

Figure 5.2 illustrates the number of visits to different URLs over 24 hours, providing insights into temporal browsing patterns. The X-axis represents the hours of the day (ranging from 00:00 to 24:00), while the Y-axis
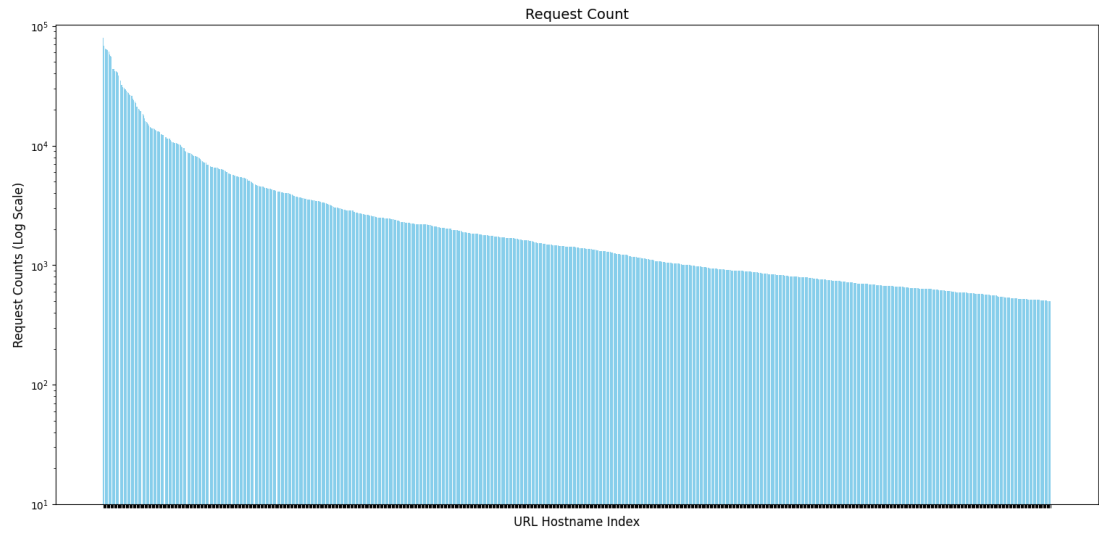
Figure 5.1: Request counts of URLs (log scale). The X-axis represents URL hostnames index that is sorted in descending order, and the Y-axis shows the number of visits on a logarithmic scale.
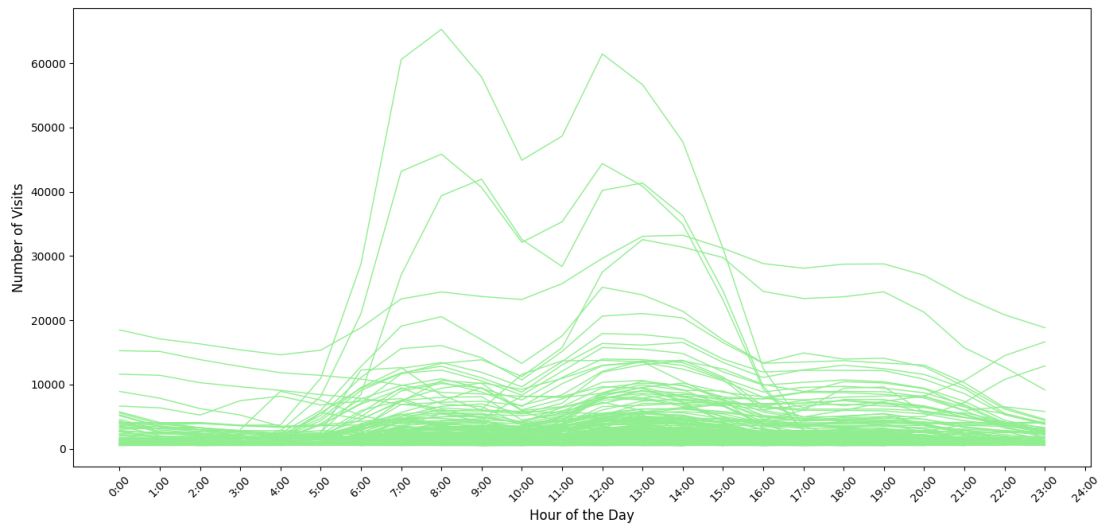

Figure 5.2: Number of visits by hour (24 hours). The X-axis represents the hours of the day, and the Y-axis shows the number of visits.

indicates the total number of URL visits recorded within each hour. The visualization reveals distinct trends in user activity, with a significant drop in visit counts around 04:00, followed by a steady increase as the day progresses. This pattern suggests that network activity is heavily concentrated during working hours, with minimal traffic during the early morning period. The sharp decline at 04:00 is particularly noteworthy, as it marks a transition between two distinct phases of network usage: "day activity" (00:00–04:00) and "night activity" (04:00–24:00). This segmentation allows for a more detailed examination of user behavior across different time intervals. For instance, the low-activity period may correspond to automated background processes, while the surge in traffic later in the day reflects active user interactions. Identifying these temporal trends is important for optimizing resource allocation, detecting anomalies, and distinguishing between normal fluctuations and potential security threats such as scheduled beaconing or automated network scans.

A comparison is conducted between the average number of visits during two distinct periods of the 24-hour cycle: the "day" period (00:00–04:00) and the "night" period (04:00–24:00). The "day" period exhibits an average of approximately 2,000 visits, while the "night" period shows a higher average of around 2,500 visits. This indicates that user activity is more concentrated during the night period, which could be influenced by factors such as the working hours of the organization or the time zones of users within the network. Such a pattern suggests that employees or users from different geographical locations may have varying peak usage times, with a noticeable increase in activity during the night hours. This insight into temporal trends in user behavior can be analytical for organizations aiming to optimize network resource allocation and improve the efficiency of security monitoring. For instance, understanding these peak periods allows network administrators to better allocate bandwidth or enhance security measures during times of higher activity. Additionally, this analysis can help in identifying potential anomalies, such as spikes in traffic, which may warrant further investigation to ensure the integrity and security of the network.

5.3 Time Interval Analysis of URL Requests

Analyzing the time intervals between URL requests is important for identifying patterns in user interactions and detecting potential beaconing behavior. By studying the time gaps between consecutive requests, it is possible to uncover patterns of periodicity that may indicate either normal user activity or suspicious behavior, such as beaconing attempts. This section examines the distribution of time intervals between requests, using both seconds and minutes as the units of analysis. In particular, the analysis focuses on uncovering regular patterns that are often associated with beaconing, where requests are sent at fixed intervals. By comparing the distribution of time intervals across different time frames, such as shorter intervals (seconds) and longer intervals (minutes), we can identify potential correlations and trends in user activity. For example, frequent and predictable intervals may suggest automated systems or malicious actors attempting to maintain communication over a network. Understanding these time distributions is a key part of detecting and mitigating beaconing behavior, while also distinguishing it from normal user browsing activity.

Figure 5.3 illustrates the distribution of time intervals between URL requests, with the Y-axis displayed on a logarithmic scale. The X-axis represents time intervals in seconds, divided into 65 bins, where each bin corresponds to a one-second interval ranging from 0 to 65 seconds. The use of a logarithmic scale on the Y-axis is particularly useful for visualizing the wide range of request counts. By compressing the scale for higher values and expanding it for lower values, the logarithmic scale enables a clearer and more detailed comparison of the frequency of requests across different time intervals. The visualization reveals a consistent pattern where the number of requests decreases as the time interval between them increases. However, there is a noticeable spike in the number of requests at every 10-second interval, suggesting periodicity in user behavior. This periodicity could be indicative of regular user activities, such as polling mechanisms, automated updates, or recurring checks for new information. These behaviors are common in legitimate network traffic and can help establish a baseline for normal activity. The identification of such periodic patterns is important in network traffic analysis, as it helps differentiate between regular activity and potential malicious behavior. For instance, if a URL exhibits similar periodic patterns but with irregular or unexpected intervals, it could be a sign of beaconing—a technique often used by malware to maintain communication with a command-and-control (C&C) server. In this case, the analysis could reveal anomalies in the intervals that deviate from expected patterns, potentially indicating a botnet or other malicious activity. By comparing these patterns against known baselines of legitimate traffic, it becomes easier to identify and flag suspicious requests for further investigation.
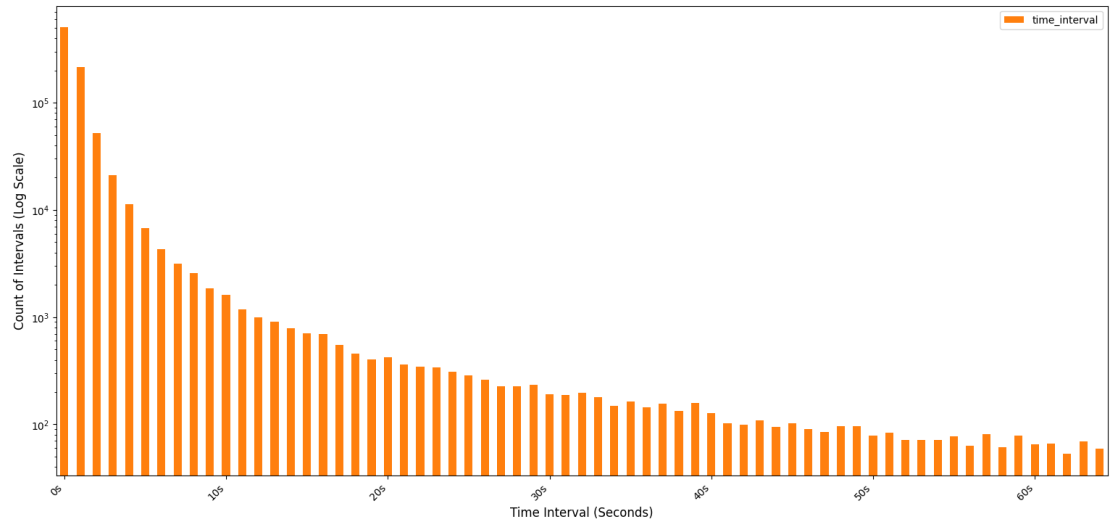
Figure 5.3: Distribution of time intervals between URL requests (0–65 seconds). The X-axis represents time intervals (bins) in seconds, and the Y-axis shows the number of requests on a logarithmic scale.
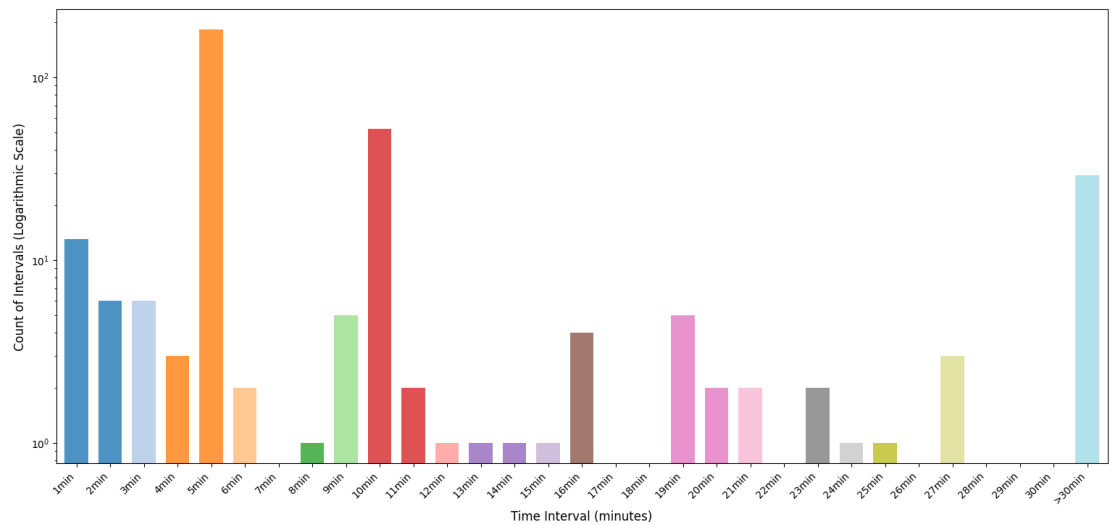

Figure 5.4: Distribution of time intervals between URL requests (in minutes). The X-axis represents time intervals (bins) in minutes, and the Y-axis shows the number of requests on a logarithmic scale.

Figure 5.4 extends the analysis of time intervals between URL requests to a larger time scale, with the X-axis each representing a one-minute interval, except for the last bin, which aggregates data from intervals longer than 31 minutes. To avoid losing beaconing data at the edges, each bin spans ±30 seconds; for example, the 1-minute bin represents data from 30 to 90 seconds. The Y-axis remains on a logarithmic scale, ensuring that both high-frequency and low-frequency intervals are visible and can be compared effectively. This use of a logarithmic scale enables the identification of trends across various time scales, making it a powerful tool for understanding patterns in network traffic. Similar to the analysis presented in Figure 5.3, the visualization reveals a decreasing trend in the number of requests as the time interval between them increases. This suggests that user interactions are typically clustered within shorter time intervals, with longer gaps between requests. However, a notable spike in request frequency appears every 5 minutes, indicating a periodic pattern at a larger time scale. This periodicity is consistent across all URLs in the dataset, suggesting that it represents a common behavior such as scheduled tasks, automated updates, or regular user interactions. These spikes could correspond to routine activities in many systems or applications that are configured to perform tasks at fixed intervals—such as background data synchronization, refresh cycles, or regular system health checks. The observed periodic behavior is particularly significant in the context of detecting malicious beaconing activity. Malicious software, including botnets and malware, often utilizes similar periodic behavior to maintain communication with command-and-control (C&C) servers, operating at regular intervals. By identifying these regular spikes in request frequency, organizations can establish a baseline for normal network behavior and detect any deviations that might indicate unauthorized or suspicious activities. The consistent periodicity observed across the dataset could thus serve as a key indicator for detecting potential threats and taking proactive security measures. The logarithmic scale is crucial for effectively visualizing the wide range of time intervals and request counts. The logarithmic scale compresses the scale for higher values and expands it for lower values, allowing for a more balanced view of both common and rare events. This enhanced visualization capability enables a clearer understanding of the temporal dynamics of user interactions and supports the identification of periodic patterns, which are important for detecting stealthy beaconing behavior in network traffic. Ultimately, this approach aids in distinguishing between normal and abnormal patterns, enhancing the framework's ability to identify potential security threats.

5.4 Distribution of Hosts Based on Unique URLs Contacted

Understanding the interaction patterns of hosts within the network is essential for identifying key services, detecting anomalies, and optimizing network performance. By analyzing the distribution of hosts based on the number of unique URLs they contacted, insights can be gained into the concentration of network activity and the diversity of services being accessed. This analysis helps highlight the most active hosts and their browsing behaviors, providing valuable information for pinpointing critical network resources, determining high-traffic users, and identifying potential security concerns. For example, an unusually high number of unique URL requests from a single host may indicate an abnormal pattern, which could suggest automated processes or even malicious behavior. By focusing on the number of unique URLs accessed by each host, this section offers a clear understanding of how traffic is distributed across the network and how hosts interact with various services. Additionally, this analysis aids in understanding the level of engagement with different network segments, assisting network administrators in optimizing resource allocation and managing network load during peak times."

Figure 5.5 illustrates the distribution of hosts (IP addresses) based on the number of unique URLs they contacted. The X-axis represents the number of unique URLs, ranging from 1 to 15, while the Y-axis shows the count of hosts within each category. The visualization highlights that the majority of hosts interact with only a small number of unique URLs. Specifically, approximately 17,500 hosts contacted exactly two unique URLs, while around 15,000 hosts interacted with only one unique URL. As the number of unique URLs increases, the number of hosts decreases significantly, although there are still many hosts contacting more than a few URLs. This pattern suggests that network activity is highly concentrated around a small set of destinations, with most hosts accessing only a limited range of resources. For example, hosts that contact only one or two unique URLs are likely interacting with essential services such as internal tools, authentication servers, or frequently accessed websites. In contrast, hosts contacting a larger number of unique URLs may represent more diverse or specialized activities, such as administrators, developers, or automated systems performing a variety of tasks across the network. This distribution of host behavior emphasizes the importance of leveraging whitelists to filter out known legitimate traffic, ensuring that analysis can focus on detecting potentially suspicious activities. The concentration of network traffic on a
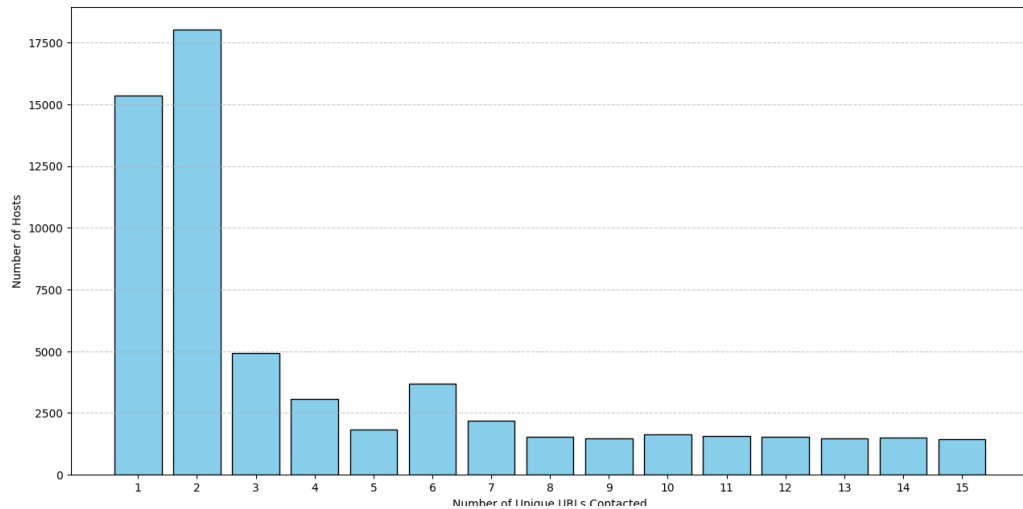
Figure 5.5: Distribution of hosts based on unique URLs contacted. The X-axis represents the number of unique URLs contacted by each host, and the Y-axis shows the count of hosts in each category.

limited set of URLs also carries significant implications for network monitoring and security. By identifying the most frequently accessed URLs, organizations can prioritize security measures for resources that are most likely to be targeted by malicious actors. URLs that experience high traffic are often the focal points of cyberattacks, such as phishing schemes, malware distribution, or command-and-control (C&C) communication. By directing attention to these critical resources, organizations can enhance their ability to detect and mitigate emerging threats. Additionally, monitoring the distribution of hosts based on the number of unique URLs they access can help identify anomalous behavior. For instance, a host that unexpectedly begins contacting a large number of unique URLs could indicate suspicious activity, such as a compromised device engaged in reconnaissance or data exfiltration. Establishing a baseline for normal host behavior allows organizations to more effectively identify deviations that may require further investigation, enhancing overall network security.

Analysis of URL Connections

After checking the URLs that were reached by these hosts, several conclusions can be drawn regarding the nature and behavior of the hosts:

- **Certificate and Security Validations:**
  Some URLs, such as `ocsp.digicert.com`, `ocsp.globalsign.com`, `crl.globalsign.com`, and `ctldl.windowsupdate.com`, are associated with certificate status checking and other security validations. These connections highlight that the hosts are actively performing routine checks to ensure the validity of digital certificates. This activity is indicative of a continuous effort to maintain secure communication channels, verify certificate integrity, and prevent man-in-the-middle (MITM) attacks. The inclusion of URLs related to certificate revocation and status checking suggests a heightened emphasis on maintaining secure connections in the network environment.

- **Operating System and Application Updates:**
  Several URLs, including `update.googleapis.com`, `www.msftconnecttest.com`, and `tldl.windowsupdate.com`, are indicative of hosts checking for operating system or application updates. These domains are typically associated with automated update mechanisms, where endpoints periodically reach out to ensure that their software and security patches are up to date. This also includes connectivity tests to verify network accessibility and ensure systems are functioning properly. These connections are crucial for maintaining the integrity and functionality of the hosts, keeping them secure and performing optimally through regular updates.

- **Enterprise and Cloud Services:**
  Domains such as `saml.allianz.com`, `www.allianz.de`, `autodiscover.allianz.de`, `service-now.com`, and `workspace.citrix` point to hosts interacting with enterprise-level services commonly found in corporate environments. These include services for Single Sign-On (SSO), IT service management, and remote workspace access. The connection to platforms like Citrix suggests that users are accessing virtual desktop environments or cloud-based services, enabling flexible work arrangements. Additionally, integration with platforms like ServiceNow highlights that these hosts may be involved in internal IT service management and troubleshooting, which is a critical component of organizational operations, particularly in large enterprises with complex infrastructures.

- **Monitoring, Telemetry, and Feature Management:**
  Connections to services like `infrastructure-command-api.eu.newrelic.com`, `infra-api.eu.newrelic.com`, and `launchdarkly.com` indicate that the hosts may be transmitting telemetry data for system monitoring, performance analytics, and feature management. These domains are linked to widely used platforms for infrastructure monitoring and feature flag management. Such activities suggest that hosts are contributing to the organization's overall monitoring strategy, providing valuable data on the performance, uptime, and behavior of applications and services. Regular telemetry collection is a key aspect of proactive system management, allowing IT teams to detect potential issues early and optimize performance.

- **Security and Threat Management Vendors:**
  Some URLs associated with security vendors, such as `updates.checkpoint.com`, `services.checkpoint.com`, `diag-services.checkpoint.com` and `support-connector-service.manage.trendmicro.com`, reflect the presence of security systems designed for threat management. These connections highlight that the hosts are integrated with advanced security solutions, which are important for identifying and mitigating cyber threats. Whether it's through regular updates, diagnostics, or real-time threat intelligence feeds, these systems play a role in defending the network against evolving security risks. The integration of such services ensures that the network is constantly protected from new and emerging threats, with automated systems managing much of the workload to keep human intervention to a minimum.

- **Diversity in Connection Sets:**
  Hosts that interact with a higher number of unique URLs are typically engaging with a more diverse set of services, which may reflect a broader range of operations. These activities include access to enhanced security protocols, system updates, and various enterprise services. The diversity in connections may also point to more specialized use cases, where certain hosts are tasked with overseeing a larger portion of the network's activities, such as system administrators or network engineers. The variety in services accessed by these hosts allows for better resource allocation and helps the organization ensure that different types of services receive appropriate attention. These hosts are likely performing more complex tasks, such as monitoring, data analysis, or security audits, and their broad range of connections is an important feature of their role in the network.

All the hosts in one day are 208,516; however, until now, only 61,207 hosts have been analyzed. So, 147,309 other hosts are waiting to be analyzed. Here are some bullet points about the hosts that have been analyzed:

- **10.201.24.129** is the host that connected to 29,858 unique URLs. This host is the most active host in the network.

- **10.16.17.11**, **10.73.17.16**, **10.17.105.167**, **172.31.161.1** are hosts that connected to 15,159, 14,567, 13,004, and 10,151 unique URLs, respectively. These hosts are the second, third, fourth, and fifth most active hosts in the network.

- 997 hosts have connected to unique URLs between 1,000 and 7,000.

- 7,965 hosts have connected to unique URLs between 500 and 999.

- 70,519 hosts have connected to unique URLs between 100 and 499.

- 67,823 hosts have connected to unique URLs between 16 and 99.

5.5 Summary

The data analysis presented in this chapter offers a detailed and comprehensive examination of the dataset's structure, user behavior, and network interactions. By utilizing a variety of visualization tools and statistical methods, the chapter identifies and uncovers key patterns that not only contribute to a better understanding of the data but also provide actionable insights for optimizing network performance and enhancing security measures. The analysis begins with a focus on URL request counts, offering a clear view of the frequency and distribution of web traffic. This helps highlight which URLs are most frequently accessed by hosts within the network, shedding light on the overall popularity of various resources. Understanding the distribution of these request counts is for determining which URLs should be prioritized in network monitoring and security management. The high-traffic URLs, in particular, are often more susceptible to attacks, such as phishing, malware distribution, or even DDoS attacks. By recognizing these hotspots, network administrators can more effectively allocate resources to ensure that these critical URLs are properly secured and monitored. Further investigation into the 24-hour visit patterns of hosts reveals how user activity is distributed across time. By analyzing these temporal patterns, the chapter sheds light on peak usage times, user behavior trends, and possible anomalies. A close examination of these patterns provides a deeper understanding of when the network is most active and helps detect deviations that might indicate unusual or malicious behavior. For instance, atypical spikes in activity at specific hours of the day could signal security incidents such as bot traffic or unauthorized access attempts. This aspect of the analysis is for optimizing network resources and managing traffic loads during high-usage periods, ensuring the network's stability and performance. Another aspect of the analysis involves the time intervals between requests. This segment of the study reveals how hosts interact with the network, providing insights into the frequency of user requests and the temporal gaps between them. This can help identify periodic or repetitive behavior, which may indicate underlying issues such as inefficient resource usage or even intentional attempts at evading detection. The analysis of time intervals is for identifying malicious activities, such as beaconing—a pattern in which an infected device sends regular, seemingly benign requests to a specific URL to maintain communication with a command-and-control server. Detecting such behaviors can play an important role in early-stage threat detection, as it allows for the identification of compromised devices or ongoing cyberattacks before they escalate. The distribution of hosts based on the number of unique URLs they contact provides a further layer of insight into user and network behavior. This analysis highlights the concentration of network activity and reveals how different hosts interact with various resources. For example, some hosts may only contact a limited number of URLs, often related to essential services, while others might interact with a broader set of resources. The latter group may represent specialized functions or more complex network activities. By understanding the distribution of hosts across different sets of URLs, organizations can better prioritize their security efforts and ensure that high-risk activities are closely monitored. This distribution can also help distinguish between normal and anomalous behaviors, offering clues about potential security threats or misconfigurations within the network. Collectively, these findings emphasize the importance of focusing on high-traffic URLs and understanding the temporal patterns in user activity. By identifying periodic behaviors or unusual request intervals, it becomes possible to detect anomalies that could indicate malicious intent or system vulnerabilities. The insights provided by this analysis are important for creating more effective detection mechanisms within the BAYWATCH framework, laying a strong foundation for the development of robust network security tools and strategies. The use of advanced visualization techniques and statistical analysis in this chapter is instrumental in uncovering these patterns. These tools provide a clear and intuitive way to visualize complex data sets, helping to identify trends and outliers that may otherwise go unnoticed. This approach not only contributes to a deeper understanding of the dataset but also facilitates the identification of areas that require further investigation or intervention. By offering a comprehensive view of the network's structure and behavior, this chapter provides a solid foundation for enhancing network security, improving performance, and developing more effective detection and mitigation mechanisms for potential threats. In conclusion, the data analysis conducted in this chapter offers a thorough understanding of network dynamics, highlighting key areas for improvement in both security and performance optimization. By examining the dataset's structure, user behavior, and network interactions through various lenses, this chapter delivers valuable insights that can guide future research and the implementation of more sophisticated network management strategies. These findings are for building a proactive security posture, ensuring the network remains resilient against evolving threats while maintaining optimal performance.

6 Implementation

Although the original BAYWATCH framework was reimplemented in Python for enhanced integration and maintainability, the focus here is strictly on the additional contributions that extend beyond the original design. The BAYWATCH framework incorporates an advanced signal analysis pipeline and a comprehensive evaluation methodology using both real network traces and synthetically generated beaconing data. These enhancements are aimed at improving temporal pattern detection under varied jitter, interval, and frequency conditions.

6.1 System Enhancements

The framework extends the original beaconing detection methodology with several keys:

6.1.1 Advanced Signal Analysis Pipeline

A multi-stage processing pipeline has been developed to isolate genuine periodic signals from noise. The pipeline consists of the following stages:

1. **Bandpass Filtering:**
   In the opening act of the analysis, the raw time-series data is prepared by removing unwanted noise while preserving its hidden rhythmic patterns. The process begins with a calculation: determining the Nyquist frequency, which is defined as half the sampling rate. This frequency sets the ultimate limit for the representation of signal components.

   Next, the desired temporal constraints—specified in seconds—are transformed into the language of frequencies. This is achieved by taking the reciprocal of the low and high periods, thus converting them into lower and upper frequency bounds. These bounds pinpoint the interval in which the periodic signals are expected to emerge.

   With these frequency limits in hand, they are normalized against the Nyquist frequency, establishing a precise passband. A Butterworth filter is then crafted to operate within this band, celebrated for its smooth, flat response that faithfully preserves the integrity of the true periodic components.

   Finally, the filtering is applied in a zero-phase manner, meaning that the data is processed in both forward and reverse directions. This dual-pass approach ensures that no phase distortion creeps into the signal, thereby maintaining the original timing of the periodic events.

   Thus, through this thoughtful orchestration of calculations and filtering, the bandpass stage effectively isolates the frequency region where the periodic signals reside, setting the stage for subsequent, more refined analyses.

2. **Permutation-Based FFT Thresholding:**
   Following filtering, the signal is transformed into the frequency domain using a Fast Fourier Transform (FFT). In order to distinguish significant periodic components from random noise, a dynamic threshold is computed. This threshold is derived by repeatedly randomizing the filtered data and analyzing the resulting spectral amplitudes. The underlying idea is that random permutations will destroy any inherent periodicity; therefore, frequency components in the original signal that exceed the threshold—determined based on a high confidence level—are likely to represent true periodic behavior.

3. **Autocorrelation Peak Detection:**
   In parallel with the FFT analysis, the pipeline examines the time-domain characteristics of the filtered signal by computing its autocorrelation. This process highlights repeating patterns over time, with prominent peaks indicating potential periodicities. By identifying these peaks, the method is able to capture the candidate time lags at which the signal exhibits a high degree of self-similarity, reinforcing the evidence of an underlying periodic structure.

4. **Frequency-Lag Correlation:**
The final stage of the pipeline cross-validates the findings from the frequency and time domains. Here, the candidate frequency components obtained from the FFT are correlated with the candidate lags identified through the autocorrelation analysis. A tolerance is applied to account for minor discrepancies. Only those frequency components that consistently align with the time-domain peaks are retained as genuine periodic signals. This correlation ensures that the periodicity detected in the frequency spectrum is supported by corresponding temporal patterns, thereby enhancing the overall reliability of the detection process.

This comprehensive signal analysis pipeline, combining bandpass filtering, permutation-based FFT thresholding, autocorrelation peak detection, and frequency-lag correlation, constitutes a novel contribution of the BAYWATCH framework. It is designed to be robust in the presence of noise and adaptable to variations in beaconing patterns, thereby significantly enhancing the detection of periodic signals in complex network traffic data.

6.1.2 Evaluation with Beaconing Data

A thorough evaluation was conducted to assess the effectiveness of the proposed approach in detecting beaconing behavior within network traffic. This evaluation was carried out using two distinct data sources, each serving a specific purpose in testing and validating the methodology:

- **Real Network Traces:** Real-world data was collected from an operational enterprise environment to ensure that the evaluation reflected authentic network activity. These traces contained legitimate user interactions, background network traffic, and various enterprise-related communications. By utilizing real network data, the study ensured that the proposed detection mechanisms could operate effectively under practical conditions, accounting for the complexities of actual user behavior, business applications, and security controls. The dataset included a diverse range of network activities, such as web browsing, automated system updates, authentication requests, and enterprise cloud service interactions, providing a realistic testbed for identifying potential beaconing patterns.

- **Synthetic Beaconing Data:** In addition to real-world data, synthetic beaconing traffic was generated to emulate a broad spectrum of beaconing behaviors under controlled conditions. This dataset was specifically designed to vary key parameters such as beacon interval and jitter, ensuring the evaluation covered a wide range of realistic and adversarial scenarios. The generated beaconing traffic included different types of periodic communications. This synthetic data allowed for systematic testing of detection thresholds, sensitivity to timing variations, and the robustness of anomaly detection mechanisms.

By combining real-world network traces with controlled synthetic beaconing data, the evaluation process ensured a comprehensive and rigorous assessment of the detection approach. The real network traces provided insight into how the method performed in a complex, dynamic enterprise environment, while the synthetic data allowed for precise benchmarking of detection capabilities across different beaconing strategies. This dual approach helped validate the robustness of the system against both common and advanced beaconing techniques, ensuring its applicability in practical network security scenarios.

6.2 Experimental Design of Synthetic Beaconing Data

The synthetic experiments simulated various beacon configurations to study the impact of temporal noise on detection accuracy. Each beacon candidate is characterized by:

- **Base Interval (s):** The expected or nominal time gap between consecutive beacon transmissions, assuming no variations. For example, if a beacon transmits every 10 seconds, its base interval is 10 seconds.

- **Jitter (s):** The amount of variation in the beacon's transmission times. Instead of transmitting at fixed intervals, a beacon might send signals slightly earlier or later than expected. If the jitter is 5 seconds, the actual transmission times may vary within ± 5 seconds of the base interval, meaning a beacon scheduled every 10 seconds might actually transmit between 5 and 15 seconds.

- **Jitter/Interval Ratio (%):** This ratio quantifies the extent of timing variability relative to the base interval and is computed as:

$$\frac{\text{Jitter}}{\text{Interval}} \times 100\%. \tag{6.1}$$

  A lower ratio (e.g., 1–5%) indicates minimal variation, making detection easier, whereas a higher ratio (e.g., 40–50%) implies significant irregularity, complicating detection.

- **Frequency (Hz):** The primary frequency of the beacon signal, representing how often a beacon transmits per second. It is calculated as the inverse of the average interval:

$$\text{Frequency} = \frac{1}{\text{Interval}}. \tag{6.2}$$

For instance:

- A beacon with a 10-second interval and 2-second jitter has effective intervals between 8 and 12 seconds.

- Another beacon with a 60-second interval and 10-second jitter produces intervals between approximately 50 and 70 seconds.

- More extreme cases include a beacon with a 300-second interval and 150-second jitter (half of the interval), resulting in intervals from 150 to 450 seconds. At the end, it is concluded that the jitter amount that is half of the interval is not an option for the beaconing detection.

6.3 Results and Analysis

Table 6.1 presents an analysis of various beacon candidates. The table provides metrics that describe the periodicity of beacon transmissions, including interval duration, jitter, and jitter-to-interval ratio. These metrics are used to understand the complexity of beacon detection and the impact of temporal noise on signal identification. The table is sorted by interval in increasing.

This table provides an overview of beacon transmission characteristics, highlighting differences in periodicity and randomness. Beacons with high jitter introduce more unpredictability, making them harder to detect, while those with low jitter and frequent transmissions are easier to identify. The results underscore the importance of understanding temporal patterns in beacon signals and the impact of noise on detection accuracy.

Figure 6.1 illustrates the synthetic beacon candidates with varying jitter levels. The x-axis represents the time intervals frequencies between beacon transmissions, while the y-axis shows the amplitude of the signals. The graph demonstrates how different jitter levels affect the periodicity and amplitude of beacon signals. Beacons with low jitter exhibit clear periodic patterns, making them easier to detect, while those with high jitter show more irregularity, complicating identification. The figure presents the results obtained after applying all stages of the detection algorithm to the selected beacon URLs. Each URL exhibits distinct candidate points where periodic behavior has been detected. The analysis reveals a significant variation in the number of candidate points across different beacon URLs. Specifically, some beacons, such as "beacon1.example.com" through "beacon4.example.com", doesn't have any candid in output, indicating that their periodic signals are either weak or occur over long intervals (here every 300 seconds), making them undetectable. This suggests that beaconing behaviors with longer intervals are inherently more difficult to detect, as their signals appear less frequently in the analyzed data. On the other hand, the algorithm performs more effectively in detecting beacons with shorter intervals. Other beacons, such as "beacon5.example.com" and "beacon6.example.com", show a substantially stronger periodicity.

A clear example of this can be seen in "beacon7.example.com", where a detected frequency of 0.05 Hz corresponds to a periodic beaconing behavior every 20 seconds. The detection of this short-interval beacon illustrates the algorithm's strength in identifying high-frequency periodic transmissions, as their repetition leads to more pronounced spectral features in the output. Overall, the figure demonstrates that while long-interval beacons pose detection challenges, the algorithm excels in identifying shorter-interval beacons with strong periodicity.

6.4 Discussion and Conclusion

The BAYWATCH extensions refine beacon detection by evaluating the impact of temporal noise and jitter on detection accuracy. The evaluation highlights that a low jitter/interval ratio (ideally below 10%) can aid detection,

Table 6.1: Beacon Candidates Ranked by Interval

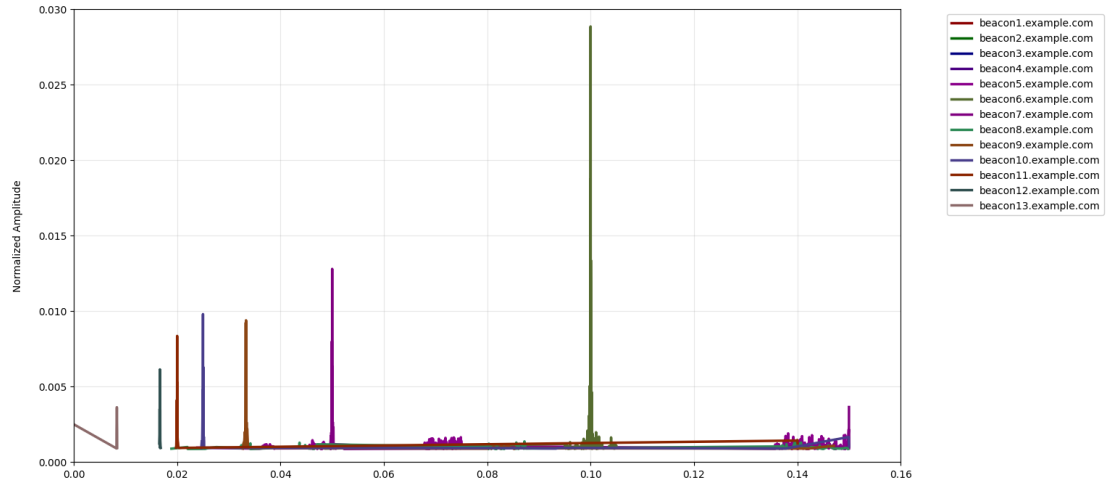| Beacon URL	Interval (s)	Jitter (s)	Jitter/Interval Ratio (%)
beacon5.example.com	10	5	50.00
beacon6.example.com	10	2	20.00
beacon7.example.com	20	5	25.00
beacon8.example.com	20	10	50.00
beacon9.example.com	30	5	16.67
beacon10.example.com	40	5	12.50
beacon11.example.com	50	5	10.00
beacon12.example.com	60	10	16.70
beacon13.example.com	120	10	8.30
beacon1.example.com	300	2	0.67
beacon2.example.com	300	5	1.67
beacon3.example.com	300	120	40.00
beacon4.example.com	300	150	50.00

Figure 6.1: Synthetic Beacon Candidates with Varying Jitter Levels, the x-axis represents the time intervals frequencies between beacon transmissions, while the y-axis shows the amplitude of the signals.

but attackers are unlikely to operate under such ideal conditions. Instead, the framework demonstrates resilience under moderate noise through adaptive thresholding and correlation techniques. The synthetic experiments provide insights into the challenges posed by irregular transmissions, emphasizing the importance of understanding periodic patterns in adversarial settings. By combining real-world network traces with synthetic beaconing data, the framework ensures a comprehensive evaluation, assessing its capability to detect malicious beaconing behavior under varied conditions. These results highlight the framework's role in enhancing network security by improving anomaly detection against stealthy threats.

The enhancements in the BAYWATCH framework, as detailed in this chapter, primarily focus on improving its evaluation and applicability rather than incorporating a fundamentally new signal analysis pipeline. By reimplementing the base framework in Python and systematically analyzing the effects of jitter, BAYWATCH achieves improved accuracy and scalability in beacon detection. The comprehensive evaluation with both real and synthetic data underscores the critical impact of jitter on detection performance and provides clear guidelines for optimal parameter settings in practical network security applications.

7 Experiments and Discussions

This chapter presents a comprehensive evaluation of the framework to validate its efficacy in detecting malicious beaconing behavior in large-scale networks. The experiments are designed to address two objectives: first assessing the framework's robustness and accuracy under controlled noise conditions using synthetic datasets, and second evaluating its practical performance in real-world enterprise network environments. Synthetic data, generated with programmable noise levels and periodic patterns, enables systematic testing of framework's core algorithms, such as the Fast Fourier Transform (FFT) and autocorrelation-based verification. Subsequently, the framework is deployed on a real-world dataset. This dual approach not only validates the theoretical soundness of the methodology but also demonstrates its scalability and operational feasibility. By synthesizing findings from both artificial and real-world scenarios, this chapter provides insights into framework's strengths, limitations, and applicability in modern cybersecurity defense systems.

7.1 Validation Steps

The validation process in the framework consists of three steps designed to identify malicious beaconing behavior. These steps ensure that only truly periodic and suspicious communication patterns are flagged, while minimizing false positives caused by noise or legitimate periodic traffic. The validation steps are as follows:

1. **FFT Candidate Detection with Power Threshold**: The first step involves applying the Fast Fourier Transform (FFT) to the time series of connection timestamps. The FFT converts the time-domain data into the frequency domain, revealing potential periodic patterns. A power threshold is then applied to filter out insignificant frequencies caused by noise. Only frequencies with amplitudes exceeding this threshold are retained as FFT candidates.

2. **ACF Verification**: The second step verifies the FFT candidates using the autocorrelation function (ACF). The ACF measures the similarity between the time series and a shifted version of itself, ensuring that the detected periodicity is consistent over time. By applying the ACF to different sequences of the data, the framework identifies ACF candidates that exhibit strong temporal consistency.

3. **Combination of FFT and ACF Results**: In the final step, the FFT and ACF results are combined to confirm beaconing behavior. The ACF results are transformed into the frequency domain, allowing direct comparison with the FFT candidates. A URL is flagged as malicious beaconing only if it is identified as a candidate by both the FFT and ACF steps. This cross-validation ensures high confidence in the detected beaconing behavior.

This multi-step validation process is for distinguishing malicious beaconing from legitimate periodic traffic and noise. By combining the strengths of FFT (frequency-domain analysis) and ACF (time-domain consistency), the framework achieves high accuracy and robustness in detecting advanced threats like botnets and APTs.

7.1.1 FFT Candidate Detection with Power Threshold

The first step is to identify candidate frequencies using the Fast Fourier Transform (FFT). The FFT converts the time-domain signal (i.e., the sequence of connection timestamps) into the frequency domain, revealing periodic patterns that may indicate beaconing behavior. The function is used to compute the FFT and filter out insignificant frequencies based on a global threshold.

The global threshold τ is a parameter in the FFT candidate detection process. It is determined using a permutation-based approach, where the original time series is shuffled to destroy any periodic patterns. The maximum amplitude in the shuffled spectrum is used as the threshold, ensuring that only frequencies with power exceeding random noise are considered candidates. This approach minimizes false positives caused by spurious signals.
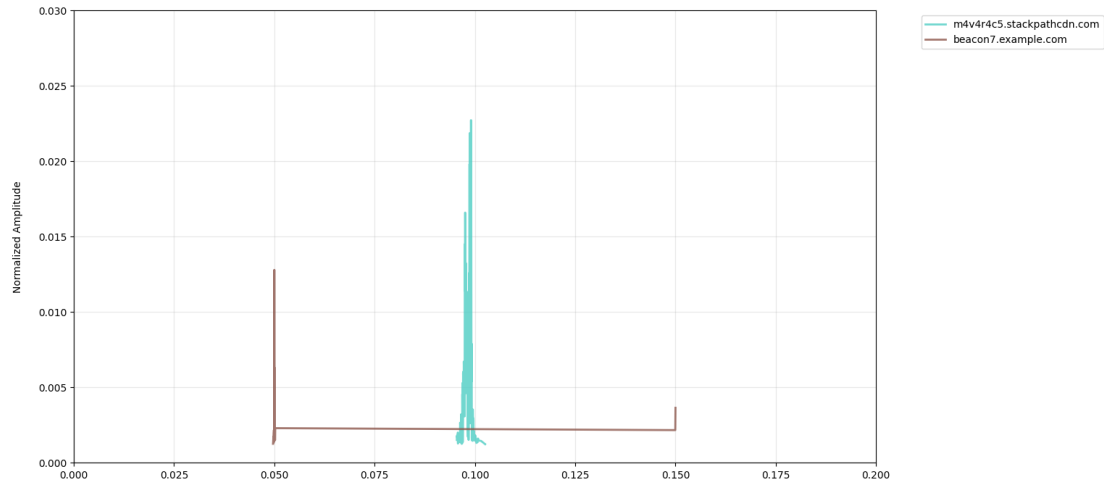
Figure 7.1: Frequency Spectrum with FFT & ACF Candidates. The x-axis represents frequency (Hz), and the y-axis represents amplitude. The figure shows candidates for the domains "m4v4r4c5.stackpathcdn.com", and "beacon7.example.com"

7.1.2 ACF Verification

After identifying candidate frequencies using the FFT, the BAYWATCH verifies their validity using the autocorrelation function (ACF). The ACF measures the similarity between the time series and a shifted version of itself, providing a more robust detection of periodic behavior. This step is for eliminating false positives caused by noise or transient patterns.

7.1.3 Combination of FFT and ACF Results

The final step combines the results from the FFT and ACF steps to confirm malicious beaconing behavior. A URL is flagged as a beaconing candidate only if it is identified by both the FFT and ACF analyses. This cross-validation ensures that only high-confidence signals are detected, minimizing false positives and enhancing the framework's accuracy.

Figure 7.1 presents the analysis of two selected URLs "m4v4r4c5.stackpathcdn.com", and "beacon7.example.com", derived from both real and synthetic data. The first URL, extracted from real data, exhibits a clear beaconing behavior. The second URL corresponds to a synthetic beacon, artificially generated to simulate a periodic transmission pattern.

The x-axis represents the frequency range, corresponding to different time intervals, while the y-axis indicates the amplitude of the detected signals. The beacon, derived from real data, exhibits periodic behavior with a frequency of approximately 0.1 Hz and a transmission interval of 10 seconds. Similarly, the synthetic beacon shows candidate frequencies at 0.05 Hz and 0.15 Hz. However, the peak at 0.05 Hz indicates that the URL exhibits beaconing behavior with a periodicity of 20 seconds.Conversely, the non-beaconing URL "fpc.mesedge.net" was also selected, and the algorithm was applied to it, but it did not yield any results. This analysis demonstrates the framework's ability to accurately detect beaconing behavior in both real and synthetic datasets, providing a reliable method for identifying malicious activities in network traffic.

Table 7.1 presents candidate data obtained from both real network traces and beaconing analysis. The table is organized into four columns: the first lists the measured attributes, including host IP addresses, URLs, observed frequencies (in Hertz), amplitude, and Is Beacon values of the periodic signals. The subsequent columns represent three distinct candidates. Candidate 1 is characterized by the host IP address "127.0.0.1", a URL "beacon7.example.com",

Table 7.1: Data Candidates from Real and Artificial Data

Attribute	Candidate 1	Candidate 2	Candidate 3
Host IP addresses	127.0.0.1	10.16.102.224	10.100.59.132
URLs	beacon7.example.com	m4v4r4c5.stackpathcdn.com	fpc.mesedge.net
Frequencies ($\sim$Hz)	0.05 & 0.15	0.1	-
Amplitude	0.014	0.024	-
Is Beacon	Yes	Yes	No

frequencies "0.05 & 0.15" Hz (indicating multiple frequency components), and a maximum amplitude of 0.014. Candidate 2 features the host IP address "10.16.102.224", the URL "m4v4r4c5.stackpathcdn.com", a single frequency component at 0.1 Hz, and a maximum amplitude of 0.024. Candidate 3, on the other hand, is associated with the host IP address "10.100.59.132", the URL "fpc.mesedge.net", and does not exhibit significant periodic frequencies or amplitudes, as indicated by the "-" symbols. last row presents that which URL is detected as a beaconing behavior. By applying the detection algorithm to this dataset and analyzing the output, it becomes evident that the algorithm effectively identifies periodic signals in both real and synthetic beaconing behaviors. The results highlight the robustness of the method, demonstrating its ability to distinguish between beaconing and non-beaconing activity while accurately capturing different periodic transmission intervals.

7.2 Discussion

The framework's combination of Fast Fourier Transform (FFT) and autocorrelation function (ACF) proved highly effective in detecting malicious beaconing behavior. By leveraging the complementary strengths of FFT (frequency-domain analysis) and ACF (time-domain consistency), the framework achieved a high detection accuracy while minimizing false positives. The FFT efficiently identified periodic signals within network traffic, allowing for the detection of recurring patterns that might indicate beaconing behavior. Meanwhile, the ACF ensured that these periodic signals were consistent over time, reinforcing the detection confidence. The cross-validation step, where only frequencies confirmed by both FFT and ACF are flagged as beaconing candidates, played an important role in enhancing detection robustness. This approach significantly reduced false positives by filtering out transient, non-malicious periodic signals, ensuring that the framework remained effective even in noisy environments with fluctuating network activity.

Despite its strong performance, the framework has certain limitations that must be addressed for broader applicability. First, its reliance on historical data means that it cannot detect zero-day beaconing behavior. Since the detection process requires a sufficient time window to analyze periodicity, newly emerging threats that do not yet exhibit clear patterns may evade detection until enough data has been accumulated. This limitation underscores the need for complementary real-time anomaly detection techniques that can provide additional layers of defense against previously unseen threats.

Second, while the framework effectively filters out most noise, it occasionally flags legitimate periodic traffic as suspicious. Some automated processes, such as scheduled software updates, real-time financial market feeds, and news syndication services, naturally generate periodic network traffic that may be mistakenly classified as beaconing behavior. This issue could be mitigated by integrating adaptive whitelisting mechanisms that dynamically update based on observed traffic patterns and external threat intelligence feeds. By continuously refining its whitelist based on empirical observations, the framework could reduce false positives without compromising its ability to detect actual threats.

A factor in evaluating the effectiveness of the proposed algorithm is its execution time. After applying the necessary preprocessing and filtering steps on real-world network data, the implemented algorithm—designed to cross-check FFT and ACF results—completed execution in under 10 seconds. This runtime was measured on a system equipped with an Intel Core i5 processor, 16GB RAM, and a 500GB SSD running Windows 11. The rapid execution time highlights the efficiency of the implemented detection pipeline, demonstrating that the approach is computationally feasible for deployment in real-world enterprise environments.

The fast execution is largely attributed to the effectiveness of the preprocessing steps. By applying a combination of noise reduction techniques, frequency filtering, and data normalization, the dataset was refined before undergoing FFT and ACF analysis. This preprocessing stage reduced computational complexity, allowing the core detection algorithm to focus on meaningful periodic signals rather than processing extraneous noise. As a result, the framework efficiently extracted periodic patterns from network traffic without significant delays, making it suitable for near analysis in security operations centers (SOCs).

Furthermore, the framework's modular design allows it to be adapted and extended based on specific security needs. Different configurations can be applied depending on the threat landscape, network architecture, and available computational resources. For instance, organizations with high network traffic volumes can implement parallelized versions of the FFT and ACF computations to further accelerate processing times. Additionally, integrating machine learning models trained on historical network traffic could enhance the framework's ability to distinguish between benign and malicious periodic behavior.

Another potential enhancement involves incorporating contextual analysis of detected beaconing traffic. While the current approach focuses primarily on periodicity, future iterations could integrate additional metadata, such as domain reputation scores, geolocation data, and endpoint behavior analysis. By correlating detected beaconing patterns with known threat intelligence sources, the framework could improve its ability to distinguish between benign automated services and actual command-and-control (C2) communications used in cyberattacks.

In conclusion, the proposed framework represents a significant advancement in the detection of malicious beaconing behavior. Its modular architecture, scalability, and high accuracy make it a practical tool for enterprise threat detection and network security monitoring. By addressing its current limitations and exploring potential enhancements, such as adaptive whitelisting and machine learning-based classification, the framework could evolve into an even more powerful component of modern cybersecurity defense systems. Its ability to efficiently analyze network traffic and detect covert communication channels underscores its value as a proactive defense mechanism against advanced persistent threats (APTs) and other sophisticated cyber threats.

8 Conclusion and Future Work

8.1 Conclusion

The BAYWATCH framework represents a robust and scalable solution for detecting malicious beaconing behavior in large-scale enterprise networks. By combining the strengths of Fast Fourier Transform (FFT) for frequency-domain analysis and autocorrelation function (ACF) for time-domain consistency, the framework achieves high accuracy in identifying periodic communication patterns indicative of malware infections. The multi-step filtering approach, which includes whitelisting, time series analysis, and suspicious indicator analysis, ensures that only high-confidence beaconing cases are flagged for further investigation. This significantly reduces false positives while maintaining a low false negative rate.

The framework was evaluated using both synthetic and real-world datasets. In synthetic datasets, BAYWATCH demonstrated strong noise tolerance, successfully detecting beaconing behavior. These results highlight the practical applicability of BAYWATCH in real-world enterprise environments, where it can serve as a tool for detecting advanced threats like Advanced Persistent Threats (APTs).

Key contributions of this work include:

- An implementation of the BAYWATCH framework's 8-step filtering methodology, designed to effectively distinguish between legitimate and malicious beaconing behavior.
- An implementation optimized for scalability, leveraging efficient data structures and parallel processing techniques to analyze large-scale network traffic over extended time windows.

The research is guided by several key questions, which are addressed below:

1. **How can beaconing behavior be effectively detected within large-scale network data to provide early warning of potential threats?**

   Beaconing behavior can be effectively detected in large-scale network data by analyzing communication patterns for periodicity and regularity. By applying advanced signal processing techniques, such as FFT and ACF, the framework can identify subtle beaconing signals amidst noisy traffic. The multi-step validation process ensures high accuracy in distinguishing between benign and malicious periodic behavior, providing early warning of potential threats.

2. **What is the impact of periodicity in network communications on distinguishing between benign and malicious activities?**

   Periodicity in network communications plays an important role in distinguishing benign from malicious activities. While legitimate applications may exhibit periodic traffic patterns, malicious activities often involve irregular or concealed periodicity to evade detection. Analyzing the consistency and regularity of communication intervals aids in differentiating between benign and malicious behaviors.

3. **Is the beaconing behavior detectable in generated synthetic data, and how does its detectability compare to that in real-world data?**

   Beaconing behavior is detectable in generated synthetic data; however, the detectability may differ from that in real-world data. Synthetic datasets often lack the complexity and variability found in real-world traffic, which can affect the performance of detection models trained exclusively on synthetic data. By evaluating the framework on both synthetic and real-world datasets, researchers can assess its robustness and generalizability across different environments.

8.2 Future Work

While the BAYWATCH framework has shown promising results, several avenues for future research could further enhance its capabilities and applicability:

8.2.1 Real-Time Analysis

The current implementation of BAYWATCH relies on historical data for beaconing detection. Future work could explore real-time streaming analysis to enable the detection of beaconing behavior as it occurs. This would require the development of efficient algorithms for processing high-velocity network traffic in real time, as well as integration with real-time threat intelligence feeds.

8.2.2 Adaptive Whitelisting

To reduce false positives caused by legitimate periodic traffic (e.g., news feeds), adaptive whitelisting mechanisms could be implemented. These mechanisms would dynamically update the whitelist based on observed traffic patterns and external threat intelligence, ensuring that only truly suspicious behavior is flagged.

8.2.3 Machine Learning Integration

Machine learning techniques could be integrated into the framework to improve the classification of legitimate and malicious periodic traffic. For example, supervised learning models could be trained on labeled datasets to identify subtle patterns indicative of malicious behavior. Unsupervised learning techniques could also be used to detect anomalies in network traffic that may not exhibit strict periodicity.

8.2.4 Extension to Other Data Sources

While the current implementation focuses on web proxy logs, the framework could be extended to analyze other types of network traffic, such as DNS queries, NetFlow data, and firewall logs. This would provide a more comprehensive approach to detecting advanced threats and improve the framework's ability to identify covert communication channels.

8.2.5 Handling Aperiodic Beaconing

Although rare, some advanced threats may employ aperiodic beaconing strategies to evade detection. Future work could explore techniques for detecting such behavior, such as analyzing the entropy of communication intervals.

8.2.6 Deployment in Diverse Environments

Finally, future work could focus on deploying and evaluating the framework in diverse network environments, such as cloud infrastructures, IoT networks, and industrial control systems. This would help identify environment-specific challenges and adapt the framework to different use cases.

8.3 Final Remarks

The BAYWATCH framework advances the field of cybersecurity by offering a robust, scalable solution for detecting malicious beaconing—a critical indicator of advanced threats like APTs and botnets. Its success lies in the hybrid methodology that combines spectral analysis (FFT) for frequency detection and autocorrelation (ACF) for temporal validation, enabling precise identification of periodic patterns even amidst noisy, real-world network traffic. By integrating multi-layered filtering—whitelisting, time series scrutiny, and behavioral heuristics—the framework minimizes false positives while maintaining sensitivity to stealthy threats, addressing a longstanding challenge in anomaly detection.

The framework's practical efficacy is underscored by its validation across both synthetic and real-world datasets, demonstrating resilience to jitter, network noise, and evasion tactics. This dual-data approach not only confirms its adaptability to controlled scenarios but also its readiness for enterprise deployment, where complexity and scale demand solutions that balance accuracy with computational efficiency. Moreover, BAYWATCH's modular architecture allows seamless integration with existing security infrastructures, such as threat intelligence platforms, enhancing organizational capabilities to preemptively identify and mitigate covert attacks.

Looking ahead, the framework's evolution could redefine proactive cybersecurity strategies. Future enhancements like real-time analysis and adaptive whitelisting would enable dynamic threat response, while machine learning integration could uncover subtler, non-periodic attack patterns through unsupervised anomaly detection. Extending BAYWATCH to diverse data sources—such as DNS logs, IoT telemetry, or cloud-native traffic—would broaden its applicability across modern IT ecosystems, from edge computing to industrial control systems.

Ultimately, this work bridges a vital gap in cyber defense, providing enterprises with a tool to combat increasingly sophisticated threats. By continuing to refine BAYWATCH's capabilities and deploying it across sectors like healthcare, finance, and critical infrastructure, the cybersecurity community can foster a more resilient digital landscape, safeguarding sensitive data and operational continuity in an era of relentless cyber escalation.

Bibliography

[1] P. V. S. Charan, P. M. Anand, S. K. Shukla, P. V. 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," in *Data Mining - Concepts and Applications*.   IntechOpen, Aug. 2021. [Online]. Available: https://www.intechopen.com/chapters/77974

[2] D. D. Caputo, S. L. Pfleeger, J. D. Freeman, and M. E. Johnson, "Going Spear Phishing: Exploring Embedded Training and Awareness," *IEEE Security & Privacy*, vol. 12, no. 1, pp. 28–38, Jan. 2014, conference Name: IEEE Security & Privacy. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6585241

[3] Y. Guo, "A review of Machine Learning-based zero-day attack detection: Challenges and future directions," *Computer Communications*, vol. 198, pp. 175–185, Jan. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422004248

[4] D. A. Eisenberg, D. L. Alderson, M. Kitsak, A. Ganin, and I. Linkov, "Network Foundation for Command and Control (C2) Systems: Literature Review," *IEEE Access*, vol. 6, pp. 68 782–68 794, 2018, conference Name: IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8535018

[5] S. W. O'Malley and L. L. Peterson, "A dynamic network architecture," *ACM Trans. Comput. Syst.*, vol. 10, no. 2, pp. 110–143, May 1992. [Online]. Available: https://dl.acm.org/doi/10.1145/128899.128901

[6] H. Krawczyk, K. G. Paterson, and H. Wee, "On the Security of the TLS Protocol: A Systematic Analysis," in *Advances in Cryptology – CRYPTO 2013*, R. Canetti and J. A. Garay, Eds.   Berlin, Heidelberg: Springer, 2013, pp. 429–448.

[7] R. K. Thomas, "Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments," in *Proceedings of the second ACM workshop on Role-based access control - RBAC '97*.   Fairfax, Virginia, United States: ACM Press, 1997, pp. 13–19. [Online]. Available: http://portal.acm.org/citation.cfm?doid=266741.266748

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

[9] X. Hu, J. Jang, M. P. Stoecklin, T. Wang, D. L. Schales, D. Kirat, and J. R. Rao, "BAYWATCH: Robust Beaconing Detection to Identify Infected Hosts in Large-Scale Enterprise Networks," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2016, pp. 479–490, iSSN: 2158-3927. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7579765

[10] Y. Zhang, H. Dong, A. Nottingham, M. Buchanan, D. E. Brown, and Y. Sun, "Global Analysis with Aggregation-based Beaconing Detection across Large Campus Networks," in *Proceedings of the 39th Annual Computer Security Applications Conference*, ser. ACSAC '23.   New York, NY, USA: Association for Computing Machinery, Dec. 2023, pp. 565–579. [Online]. Available: https://dl.acm.org/doi/10.1145/3627106.3627126

[11] G. Apruzzese, M. Marchetti, M. Colajanni, G. G. Zoccoli, and A. Guido, "Identifying malicious hosts involved in periodic communications," in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, Oct. 2017, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8171326

[12] J. Seo and S. Lee, "Abnormal Behavior Detection to Identify Infected Systems Using the APChain Algorithm and Behavioral Profiling," *Security and Communication Networks*, vol. 2018, no. 1, p. 9706706, Jan. 2018, publisher: John Wiley & Sons, Ltd. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1155/2018/9706706

[13] N. Anh Huynh, W. Keong Ng, A. Ulmer, and J. Kohlhammer, "Uncovering periodic network signals of cyber attacks," in *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, Oct. 2016, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7739581

Bibliography

[14] J. Jang, D. H. Kirat, B. J. Kwon, D. L. Schales, and M. P. Stoecklin, "Detecting malicious beaconing communities using lockstep detection and co-occurrence graph," Jan. 2021. [Online]. Available: https://patents.google.com/patent/US10887323/en

[15] M. Abu Talib, Q. Nasir, A. Bou Nassif, T. Mokhamed, N. Ahmed, and B. Mahfood, "APT beaconing detection: A systematic review," *Computers & Security*, vol. 122, p. 102875, Nov. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404822002693

[16] M. Hagan, B. Kang, K. McLaughlin, and S. Sezer, "Peer Based Tracking using Multi-Tuple Indexing for Network Traffic Analysis and Malware Detection," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug. 2018, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8514165

[17] A. Shalaginov, K. Franke, and X. Huang, "Malware Beaconing Detection by Mining Large-scale DNS Logs for Targeted Attack Identification," Apr. 2016.

[18] Y.-R. Yeh, T. C. Tu, M.-K. Sun, S. M. Pi, and C.-Y. Huang, "A Malware Beacon of Botnet by Local Periodic Communication Behavior," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 02, Jul. 2018, pp. 653–657, iSSN: 0730-3157. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8377941

[19] M. A. Enright, E. Hammad, and A. Dutta, "A Learning-Based Zero-Trust Architecture for 6G and Future Networks," in *2022 IEEE Future Networks World Forum (FNWF)*, Oct. 2022, pp. 64–71, iSSN: 2770-7679. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10056611

[20] T. v. Ede, H. Aghakhani, N. Spahn, R. Bortolameotti, M. Cova, A. Continella, M. v. Steen, A. Peter, C. Kruegel, and G. Vigna, "DEEPCASE: Semi-Supervised Contextual Analysis of Security Events," in *2022 IEEE Symposium on Security and Privacy (SP)*, May 2022, pp. 522–539, iSSN: 2375-1207. [Online]. Available: https://ieeexplore.ieee.org/document/9833671/?arnumber=9833671

[21] T. Ongun, O. Spohngellert, B. Miller, S. Boboila, A. Oprea, T. Eliassi-Rad, J. Hiser, A. Nottingham, J. Davidson, and M. Veeraraghavan, "PORTFILER: Port-Level Network Profiling for Self-Propagating Malware Detection," in *2021 IEEE Conference on Communications and Network Security (CNS)*, Oct. 2021, pp. 182–190. [Online]. Available: https://ieeexplore.ieee.org/document/9705045/?arnumber=9705045

[22] W. Niu, J. Xiao, X. Zhang, X. Zhang, X. Du, X. Huang, and M. Guizani, "Malware on Internet of UAVs Detection Combining String Matching and Fourier Transformation," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9905–9919, Jun. 2021, conference Name: IEEE Internet of Things Journal. [Online]. Available: https://ieeexplore.ieee.org/document/9220767/?arnumber=9220767

[23] J. Duan, Z. Zeng, A. Oprea, and S. Vasudevan, "Automated Generation and Selection of Interpretable Features for Enterprise Security," in *2018 IEEE International Conference on Big Data (Big Data)*, Dec. 2018, pp. 1258–1265. [Online]. Available: https://ieeexplore.ieee.org/document/8621986/?arnumber=8621986

[24] M. Haffey, M. Arlitt, and C. Williamson, "Modeling, Analysis, and Characterization of Periodic Traffic on a Campus Edge Network," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Sep. 2018, pp. 170–182, iSSN: 2375-0227. [Online]. Available: https://ieeexplore.ieee.org/document/8526883/?arnumber=8526883

[25] A. Oprea, Z. Li, R. Norris, and K. Bowers, "MADE: Security Analytics for Enterprise Threat Detection," in *Proceedings of the 34th Annual Computer Security Applications Conference.* San Juan PR USA: ACM, Dec. 2018, pp. 124–136. [Online]. Available: https://dl.acm.org/doi/10.1145/3274694.3274710

[26] M. Ukrop, L. Kraus, V. Matyas, and H. A. M. Wahsheh, "Will you trust this TLS certificate?: perceptions of people working in IT," in *Proceedings of the 35th Annual Computer Security Applications Conference.* San Juan Puerto Rico USA: ACM, Dec. 2019, pp. 718–731. [Online]. Available: https://dl.acm.org/doi/10.1145/3359789.3359800

[27] T. Vissers, J. Spooren, P. Agten, D. Jumpertz, P. Janssen, M. Van Wesemael, F. Piessens, W. Joosen, and L. Desmet, "Exploring the Ecosystem of Malicious Domain Registrations in the .eu TLD," in *Research in Attacks, Intrusions, and Defenses*, M. Dacier, M. Bailey, M. Polychronakis, and M. Antonakakis, Eds. Cham: Springer International Publishing, 2017, pp. 472–493.