SAARLAND UNIVERSITY

Faculty of Mathematics and Computer Science
Department of Computer Science
Dissertation

Getting to the root of SSH Compromises: A Multi-Dimensional Characterization of the SSH Threat Landscape

Dissertation zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)

der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

vorgelegt von
Cristian Munteanu

Saarbrücken, 2025

Notes on style:
As most of the work presented in this dissertation was done in collaboration with other researchers, the scientific plural "we" is used.

Saarland University
Faculty MI – Mathematics and Computer Science
Department of Computer Science
Campus - Building E1.1
66123 Saarbrücken
Germany

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In-noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Declaration of original authorship

I hereby declare that this dissertation is my own original work except where otherwise indicated. All data or concepts drawn directly or indirectly from other sources have been correctly acknowledged. This dissertation has not been submitted in its present or similar form to any other academic institution either in Germany or abroad for the award of any other degree.

Place, Date (Unterschrift / Signature)

Acknowledgements

I would like to express my sincere gratitude to everyone who supported me throughout this journey.

My supervisors–people who guided me, taught me, and showed remarkable patience throughout the process. Anja–for your invaluable guidance and wisdom. George–for your unwavering support and your positive attitude, which kept me optimistic and moving forward. Tobias–for teaching me the importance of slowing down, reflecting, and thinking critically before jumping to conclusions.

I would also like to thank the INET team for making the office a comfortable and welcoming place where I truly enjoyed working, learning, and collaborating. Thank you, Iris, for helping me navigate the paperwork. Thank you, Joerg, for putting up with me and my crazy IT requests. Thank you, Pascal, for making me laugh, and Tiago, for your daily morning greetings and advice. Thank you, Taha, Sebastian, Florian, Fariba, Danesh, Sina, Johannes, Simran, Vaishnavi, and Emilia–for the brainstorming sessions, great conversations, and shared laughs that made every day better.

Above all, I would like to express my deepest gratitude to my wife, Valeria. Her patience, strength, and the many sacrifices she made throughout this journey were fundamental to my reaching this milestone. Her unwavering support has profoundly shaped both who I am and what I have achieved. To my children, Sofia and Alex, my parents, Dinu and Aurelia, my brother Adrian, and my dear friends, Vadim and Cristina–thank you for your constant encouragement and for always being there for me. You've made this journey one of the most meaningful and rewarding experiences of my life.

Abstract

The Internet has become a critical infrastructure, attracting a vast amount of activity, including malicious threats. The Secure Shell Protocol (SSH), the successor of Telnet–designed for secure machine-to-machine communication–is one of the most widely used protocols on the Internet. Due to its ubiquity, SSH has become a prime target for attackers. Over the years, SSH attacks have evolved, and their frequency has only increased.

In this thesis, we investigate the nature of these attacks, including their origins, methods, and targets. We conduct a retrospective study and a longitudinal analysis using a large honeyfarm, as well as an active analysis to identify compromised SSH servers.

Through a global network of honeypots, we analyze approximately 750 million SSH sessions over a three-year period. The dataset, collected from 221 honeypots across 55 countries, reveals stark variations in activity–some honeypots observe millions of connections, while others record only a few thousand. We also analyze attacker behavior, uncovering a shift toward more exploratory attacks and increased reconnaissance efforts. Additionally, attackers increasingly leverage recently registered Autonomous Systems (ASes) to store and distribute malicious files. Our findings suggest that attackers are becoming more aware of honeypot presence, with some actively seeking to evade detection.

To extend our analysis, we propose a method to identify compromised SSH servers at scale. We exploit SSH's authentication behavior, where a challenge is only issued if a public key is installed. This approach *neither* grants access to compromised systems (unlike testing known attacker passwords), *nor* requires privileged access for auditing.

Applying this methodology to a comprehensive Internet scan, we identify over 21,700 compromised systems across 1,649 ASes in 144 countries. These include critical infrastructure where attackers have installed at least one of 52 verified malicious SSH keys provided by a threat intelligence company. Our investigation also uncovers insights into malicious campaigns such as the 'fritzfrog' IoT botnet and threat actors like 'teamtnt'. Moreover, we collaborate with a national CSIRT and the Shadowserver Foundation to notify affected entities and facilitate remediation efforts. We run our measurements continuously and automatically share notifications.

Zusammenfassung

Das Internet hat sich zu einer kritischen Infrastruktur entwickelt und zieht eine Vielzahl von Aktivitäten an–darunter auch bösartige Bedrohungen. Das Secure Shell-Protokoll (SSH), der Nachfolger von Telnet, wurde für sichere Maschine-zu-Maschine-Kommunikation entwickelt und ist eines der am weitesten verbreiteten Protokolle im Internet. Aufgrund seiner Allgegenwärtigkeit ist SSH zu einem bevorzugten Ziel für Angreifer geworden. Im Laufe der Jahre haben sich SSH-Angriffe weiterentwickelt, und ihre Häufigkeit hat stetig zugenommen.

In dieser Arbeit untersuchen wir die Eigenschaften dieser Angriffe, einschließlich ihrer Ursprünge, Methoden und Zielsysteme. Wir führen sowohl eine retrospektive Untersuchung als auch eine longitudinale Analyse unter Verwendung einer umfangreichen Honeyfarm durch, ergänzt durch eine aktive Analyse zur Identifikation kompromittierter SSH-Server.

Über ein globales Netzwerk von Honeypots analysieren wir etwa 750 Millionen SSH-Sitzungen über einen Zeitraum von drei Jahren. Der Datensatz, gesammelt von 221 Honeypots in 55 Ländern, zeigt deutliche Unterschiede in Aktivitäten–einige Honeypots registrieren Millionen von Verbindungen, während andere nur wenige Tausend beobachten. Wir analysieren auch das Verhalten der Angreifer und stellen dabei eine Verschiebung hin zu explorativeren Angriffen sowie zunehmende Aufklärungsaktivitäten fest. Zudem nutzen Angreifer vermehrt kürzlich registrierte Autonome Systeme (ASes), um bösartige Dateien zu speichern und zu verbreiten. Unsere Ergebnisse deuten darauf hin, dass Angreifer sich zunehmend der Präsenz von Honeypots bewusst sind–einige versuchen sogar aktiv, einer Erkennung zu entgehen.

Zur Erweiterung unserer Analyse schlagen wir eine Methode zur Identifikation kompromittierter SSH-Server im großen Maßstab vor. Dabei machen wir uns das Authentifizierungsverhalten von SSH zunutze: Eine Herausforderung für den Schlüsselaustausch wird nur dann aufgegeben, wenn ein öffentlicher Schlüssel installiert ist. Dieser Ansatz *gewährt* weder Zugang zu kompromittierten Systemen (anders als das Testen bekannter Angreifer-Passwörter), *noch* erfordert er privilegierten Zugriff zur Überprüfung.

Durch Anwendung dieser Methode auf einen umfangreichen Internetscan identifizieren wir über 21.700 kompromittierte Systeme in 1.649 ASes in 144 Ländern. Darunter befinden sich auch kritische Infrastrukturen, auf denen Angreifer mindestens einen von 52 verifizierten, von einem Threat-Intelligence-Unternehmen bereitgestellten, bösartigen SSH-Schlüsseln installiert haben. Unsere Untersuchung liefert darüber hinaus Einblicke in bösartige Kampagnen wie das IoT-Botnetz `fritzfrog` und Akteure wie `teamtnt`. Zudem arbeiten wir mit einem nationalen CSIRT und der Shadowserver Foundation zusammen, um betroffene Einrichtungen zu benachrichtigen und Maßnahmen zur Schadensbegrenzung zu ermöglichen. Unsere Messungen laufen kontinuierlich, und Benachrichtigungen werden automatisch geteilt.

List of Publications

Parts of this dissertation are based on pre-published work. These works are co-authored with other researchers as listed below.

International Conference Publications

Cristian Munteanu, Oliver Gasser, Ingmar Poese, Georgios Smaragdakis, and Anja Feldmann. 2023. “Enabling Multi-hop ISP-Hypergiant Collaboration”. In Proceedings of the 2023 Applied Networking Research Workshop (ANRW ’23). Association for Computing Machinery, New York, NY, USA, 54–59. doi.org/10.1145/3606464.3606487

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 (IMC ’23). Association for Computing Machinery, New York, NY, USA, 282–296. doi.org/10.1145/3618257.3624826
Results are presented in Chapter 3.

Cristian Munteanu, Yogesh Bhargav Suriyanarayanan, Georgios Smaragdakis, Anja Feldmann and Tobias Fiebig. “Attacks Come to Those Who Wait: Long-Term Observations in an SSH Honeynet”. In Proceedings of the 2025 ACM on Internet Measurement Conference (IMC ’25).
Association for Computing Machinery, New York, NY, USA, 628–644. doi.org/10.1145/3730567.3764475
Results are presented in Chapter 4.

Cristian Munteanu, Georgios Smaragdakis, and Anja Feldmann. 2024. “Poster: The State of Malware Loaders”. In Proceedings of the 2024 ACM on Internet Measurement Conference (IMC ’24). Association for Computing Machinery, New York, NY, USA, 745–746. doi.org/10.1145/3646547.3689659
Results are presented in Chapter 4.

Cristian Munteanu, Georgios Smaragdakis, Anja Feldmann and Tobias Fiebig (in press). “Catch-22: Uncovering Compromised Hosts using SSH Public Keys”. In Proceedings of the 34th USENIX Security Symposium. USENIX. hdl.handle.net/21.11116/0000-0010-89BB-1
Results are presented in Chapters 5 and 6.

Contents

List of Publications **viii**

1 Introduction **1**

1.1 Contributions . . . 4

1.2 Publications and Collaborations . . . 5

1.3 Overview and Structure . . . 5

2 Background **8**

2.1 Secure SHell Protocol . . . 8

2.2 SSH Attacks . . . 9

2.2.1 Brute-Force Attacks . . . 9

2.2.2 Protocol Weakness Attacks . . . 9

2.2.3 Countermeasures . . . 10

2.3 Attackers Behavior . . . 11

2.3.1 Threat-Actor Characterization . . . 12

2.4 SSH Honeypots . . . 12

3 Unwanted SSH Traffic **16**

3.1 Honeyfarm Dataset . . . 17

3.2 Honeypot Session Categories . . . 20

3.3 Honeypot Clients . . . 25

3.3.1 Client IPs per country . . . 25

3.3.2 Client IPs Over Time . . . 25

3.3.3 Client IPs across countries by category . . . 28

3.3.4 SSH vs. Telnet Activity . . . 29

3.3.5 Client IPs vs. Honeypots . . . 29

3.3.6 Regional Diversity . . . 31

3.4 Commands . . . 33

3.4.1 Common Commands . . . 33

3.4.2 Popular Hashes and Attack Campaigns . . . 34

3.4.3 Campaign Timeline and Freshness . . . 36

3.4.4 Hashes vs. Honeypots . . . 37

3.4.5 Hashes vs. Clients 39
3.4.6 Activity of Attack Campaigns 39
3.5 Discussion 39
3.6 Summary 40

4 SSH Attacker Characterization **42**
4.1 Datasets 43
4.1.1 Honeyfarm Profile 43
4.1.2 Abuse Datasets 43
4.1.3 Autonomous System Data 44
4.2 Commands 45
4.3 Malware Files 52
4.4 Malware storage location 54
4.5 Attacker login attempts 58
4.6 *"mdrfckr"*—case study 61
4.7 Discussion 64
4.8 Summary 65

5 SSH Active Scans **66**
5.1 Background 67
5.2 Methodology 69
5.2.1 Public-Key Presence Identification 69
5.2.2 Lab Experiment 69
5.2.3 Controlled Environment Tests 70
5.2.4 False-Positive Detection 72
5.2.5 Malicious Keys 72
5.2.6 Tested Usernames 73
5.2.7 Ethical Considerations 73
5.3 Experiment 75
5.3.1 Measurement Infrastructure 75
5.3.2 Scan Execution Methodology 75
5.3.3 Scanning Schedule 76
5.3.4 Descriptives and Comparison to Censys 76
5.4 Summary 77

6 Compromised SSH Hosts **78**
6.1 Introduction 78
6.2 Compromised Host Overview Port 22 79

6.3 Geographic Distribution 82
6.4 Malicious Keys & Actors 82
6.5 Port 2222 Findings 88
6.6 Notable Events 89
6.7 Discussion 92
6.7.1 Internet Measurement Opt-Out and Ethics 92
6.7.2 Legal Challenges 92
6.7.3 Feedback on IPv6 Measurements 93
6.7.4 Evasion Tactics and Scalability Challenges 93
6.8 Summary 94

7 Conclusion **95**
7.1 Summary 95
7.2 Future Directions 97

A Commands classification Table **100**

List of Abbreviations **101**

List of Figures **102**

List of Tables **105**

Bibliography **107**

Chapter 1
Introduction

The Internet has become an indispensable part of modern life, serving as the backbone for essential services, industries, and everyday interactions. Its role continues to expand as digital infrastructure evolves, supporting an ever-growing volume of online activities and global connectivity. The foundation of this vast digital ecosystem lies in a diverse set of protocols, each serving a distinct purpose to enable seamless connectivity and communication across networks.

Since its introduction in 1995, the Secure Shell (SSH) Protocol has become the de facto standard for secure remote access to shell-enabled systems [1]. Originally designed for Unix and Linux Operating Systems, SSH has since spread across the digital landscape, reaching various operating systems. With its growing adoption, SSH has been deployed on million devices, often enabled by default. This widespread use is evident in the up to 40 million publicly accessible SSH servers on the Internet [2].

Designed to create a secure communication channel, SSH supports multiple authentication mechanisms, including public/private key authentication and, traditionally, password authentication. Due to the continued prevalence of password authentication, attackers frequently conduct brute-force campaigns to exploit weak or default passwords [3, 4]. As a widely used protocol for secure communication and remote system management, SSH has become a prime target for malicious actors [5, 6].

Attacks on SSH are a significant concern, driving extensive research on the subject. Numerous studies have examined SSH security, beginning with brute-force and dictionary attacks [4, 7, 8], followed by more advanced techniques such as Plaintext Recovery Attacks [9], Keystroke Timing Attacks [10], and the Terrapin attack [11].

While prior research has extensively examined password-based attacks and cryptographic vulnerabilities, fewer studies have explored how attackers continue to adapt their strategies in response to security enhancements. Despite SSH remaining a fundamental component of secure network infrastructure, there has been a noticeable gap in longitudinal studies examining attacker activity, behavioral changes over time, and the analysis of compromised SSH servers. This lack of recent empirical studies create a blind spot in understanding modern SSH threats and the extent of real-world compromises. As such, **the main topics of this thesis** is to analyze how has SSH attack activity changed over time and how compromised hosts can be effectively detected through non-intrusive large-scale Internet measurements. To systematically explore these topics, we have structured our investigation around four well-defined research questions.

The first research question that we address in this thesis is: **Can we understand unwanted SSH traffic on a global scale?** To investigate, we leverage honeypots—network-connected systems designed to mimic a vulnerable system or network to attract, detect, and analyze threats by deceiving attackers into interacting with it [12]. They can be deployed alongside production systems to alert network administrators about specific attack strategies or independently in the wild to capture broader attack campaigns [13]. For decades, the security research community has relied on honeypots both as a security mechanism [14, 15], and as a tool for studying network security, discovering emerging threats, and analyzing attack strategies in the wild [16, 17, 18, 19]. The type of honeypot determines the data collected and the specific types of unwanted traffic that can be analyzed. SSH honeypots, in particular, record both metadata (e.g., client IP addresses, port numbers) and behavioral data (e.g., executed commands, interactions with other devices, and created files), and have been instrumental in studying malware, such as Mirai [5, 20].

A wide range of honeypot software is freely available [21, 22, 23, 24], enabling researchers to easily deploy systems and collect data. However, most deployments consist of only a handful of honeypots, often poorly distributed and maintained for limited periods. In contrast, commercial companies such as GreyNoise[25] and NetScout [26] operate large-scale well-distributed honeypot infrastructures, known as 'honeyfarms'. The data collected by these honeyfarms allows the development of security products, detection of new threats, and the enhancement of consumer security, but is rarely available for researchers or the public.

Through our collaboration with Global Cyber Alliance (GCA) [27], we analyze data collected from a honeyfarm comprising 221 honeypots deployed in 55 countries across 64 networks. The honeypots are operational from November 2021 to August 2024 and use IPv4 addresses that had not previously been associated with honeypots or darknets. By examining three years of data, we assess both the broad trends captured by the collective honeyfarm, as well as the distinct insights offered by individual honeypots. In this period, the GCA honeypots distinguished between scanning, reconnaissance, and intrusion behaviors, offering insights into unsolicited Internet activity.

It is crucial to consider a temporal perspective when analyzing SSH attacks (or any other type of attack) since malicious actors continuously develop and deploy new methods and techniques over time. This evolution is a natural consequence of the ongoing tug-of-war between those who devise attack strategies and those who develop defensive mechanisms. As such, we formulate our second research question: **How does attacker behavior change over extended periods?** Attackers employ a variety of techniques, ranging from simple brute-force attempts to sophisticated botnets. While previous research has provided valuable insights into attacker behavior, studies often focus on specific botnets [28] or rely on short-term honeypot observations [29], offering important but time-limited or attack-specific perspectives. Building on these foundations, our work extends prior research by analyzing a large honeyfarm data collection over a long period, shedding light on the evolving nature of malicious activities at a broader scale.

This is critical because defending against attacks is an ongoing struggle between those seeking to exploit systems and those working to protect them. Leveraging our collaboration with GCA we observe trends over a long-term period. This allows us to detect recurring attack patterns, correlate trends with external events, and refine defensive strategies accordingly.

Monitoring attacker behavior over an extended period allows us to identify command execution patterns, detect persistent threats, and analyze ongoing attack campaigns. Gaining such insights is crucial for developing effective and adaptive defense mechanisms. We conduct a longitudinal analysis of SSH-based attacks, evaluating the importance of continuous observation in Internet security research. Our findings reveal that a significant portion of attackers not only deploy malware but also attempt to establish persistent access to compromised machines.

While honeypots serve as a valuable source of information, they remain a passive approach to understanding the spread of an attack. Deploying a large number of honeypots may capture millions of attack attempts, but it does not provide insight into the actual impact–specifically, how invasive the attack was or how many real hosts were compromised.

Therefore, we pose our third research question: **Can we design a non-intrusive methodology to identify compromised hosts at scale?** Once an attacker compromises a system, changing its password may not be a viable option to establish persistence, as a sudden inability to log in could alert the system's legitimate owner [30]. Instead, attackers often install their own public SSH keys into compromised accounts, as observed in several documented incidents [11] and honeypot studies [31]. Many compromised systems run SSH servers, providing attackers with a convenient method of persistent access [32].

SSH implementations behave differently depending on whether a public key is actually installed when a user attempts authentication via a key fingerprint. Typically, a challenge is only issued if the provided key is registered for that user [33]. This behavior is an intentional design choice in the SSH protocol [34] and is not considered a security issue by OpenSSH developers [35]. However, this mechanic provides a unique opportunity: if an attacker's public SSH key is obtained, it can be used to remotely identify compromised systems. Using this methodology, we pioneer a large-scale measurement technique to identify compromised systems on the Internet. The design of the measurement tool follows current standard ethical considerations [36] and has obtained approval from the local (Ethical Review Board) ERB.

As such, our final research question is: **Can we successfully identify and characterize compromised SSH hosts in the wild?** Through a collaboration with Bitdefender [37], a threat intelligence company specializing in Malware and Internet Security, we obtained a dataset of 52 malicious SSH public keys. These keys were discovered in honeypots, malware analyses, and incident investigations. We developed a measurement methodology that identified over 21 thousand compromised systems worldwide, including critical infrastructure, government agencies, research institutions, and civil infrastructure. Our analysis allows us to characterize attacker behavior in unprecedented detail, mapping who compromises which systems and where. We integrated our findings into a notification campaign via the Shadowserver Foundation [38] and later through Germany's national Computer Security Incident Response Team (CSIRT) at the Federal Office for Information Security (BSI). We continue to perform automated scans and distribute notifications through our partners.

1.1 Contributions

This section gives an overview of the contributions of this thesis.

With regard to the first research question, i.e., can we understand unwanted SSH traffic on a global scale, we make the following contributions:

- We analyze more than 721 million sessions collected at 221 honeypots deployed in 55 countries over 3 years. Based on the recorded data, we characterize the sessions as no credential, failed login, no command execution, command execution, and command execution with file download.
- We analyze the geographical distribution of intrusive IPs and identify a distinction between general scanning IPs and those exhibiting more aggressive, intrusive behavior. Additionally, we observe that while some honeypots are more heavily targeted—seeing up to 30× more unique files—they still account for only a small fraction (10%) of the total data collected by the honeyfarm.

Regarding the second research question, i.e., how does attacker behavior change over extended periods, our contribution can be summarized as follows:

- Over three years, we analyze 163 million intrusive sessions, revealing a significant shift in attacker behavior starting in 2023. Attackers' methods for executing files on compromised systems have evolved, and by clustering over 16,000 malicious files, we classify various botnets, including Mirai, Gafgyt, and XXorDDoS, providing insights into their long-term activity.
- More than 70% of malware storage locations are hosted in newly registered Autonomous Systems (ASes), often less than five years old. Additionally, we find that attackers actively scan for honeypots, sometimes abusing them as proxies for further attacks. Our analysis of a persistent attack campaign throughout the observation period highlights its aggressive nature and operational characteristics.

With regards to our third research question, i.e., can we design a non-intrusive methodology to identify compromised hosts on large-scale, our contributions are:

- We design and implement a methodology that accurately identifies compromised SSH servers by leveraging unique characteristics of the SSH protocol. The tool is publicly available as an artifact of our research [39].
- Through extensive in-lab testing, we refine our tool to enhance performance while minimizing potential risks to networks during scanning operations.

With regards to our final research question, i.e., can we successfully identify and characterize compromised SSH hosts in the wild, our contributions are:

- We detect over 21,700 compromised SSH hosts across 1,649 ASes using 52 malicious public keys on both IPv4 and IPv6. Our analysis reveals detailed attacker behavior, including persistent IoT botnets, potential nation-state actors, and unique attack-specific keys.
- By cross-referencing our active measurements with data from the GCA honeyfarm, we verify that 24 of the compromised hosts are actively launching attacks, while two others serve as malware storage locations, demonstrating ongoing attacker operations.

1.2 Publications and Collaborations

Parts of this thesis are based on previously published work conducted in collaboration with other researchers. Below, we outline the main contributions of the thesis author to the research incorporated in this document:

Chapter 3: Unwanted SSH Traffic: This chapter analyzes unwanted SSH traffic using data from a large-scale honeypot deployment. The work was published at IMC 2023 in collaboration with the authors listed in [31]. The thesis author's main contributions include: (i) Formulation of the research questions, (ii) Joint background research, (iii) Complete data analysis, including interpretation and visualization.

Chapter 4: Attacker Behavior: This chapter investigates how attacker behavior evolves over time. The work was published at IMC 2025 in collaboration with the authors listed in [40]. The thesis author's main contributions include: (i) Formulation of the research questions, (ii) Background research, (iii) Full analysis of the data, including discussion and visualizations.

Chapters 5 and 6: Compromised SSH Hosts: These chapters present a methodology for non-intrusively identifying compromised SSH hosts at scale and analyze the data collected from an Internet-wide experiment using this methodology. This work was published at USENIX Security 2025 in collaboration with the authors listed in [41]. The thesis author's main contributions include: (i) Formulation of the research questions, (ii) Background research, (iii) Design and testing of the experimental setup, (iv) Joint data analysis, including discussion and visualizations.

No AI tools were used for the analysis or writing of this thesis or publications. All research, data processing, and text were carried out manually by the author and collaborators.

1.3 Overview and Structure

The aforementioned topics are presented in this thesis according to the following structure:

Chapter 2 provides background information relevant to this thesis, covering the design of the SSH protocol, various SSH attacks, attacker behavior, and the role of SSH honeypots.

To investigate unwanted SSH traffic on the Internet, Chapter 3 presents an in-depth analysis of data collected by a well-distributed honeyfarm. The results of this study have been published in [42].

Chapter 4 offers a longitudinal analysis of SSH intruder behavior, examining shifts in their tactics over time. The results of this analysis are currently under submission.

Shifting focus from passive analyses of SSH attacks to an active measurement approach, Chapter 5 details the methodology and preparation required to identify compromised SSH hosts in the wild through Internet-wide scanning. The tool is published as an artifact [39].

Chapter 6 presents the results obtained from our measurement tool following an Internet-wide scan. These findings are published in [41].

Finally, Chapter 7 summarizes the key findings of this thesis, addresses the research questions, and situates them within a broader context. The chapter concludes with reflections on potential future research directions.

Chapter 2
Background

This chapter provides background information for this thesis. We start off with the Secure SHell Protocol in Section 2.1. After we discuss various attacks on SSH in Section 2.2 and the attacker's behavior in Section 2.3. Finally, we then turn our attention towards the Honeypots described in Section 2.4.

2.1 Secure SHell Protocol

SSH has first been published in 1995 by Tatu Ylönen to replace by then prevalent plain-text commands, e.g., `rsh` (remote shell), `rcp` (remote copy), and `telnet`, for remotely interacting with systems. The standards track RFCs, RFC4251-RFC4254 [43, 34, 44, 45], were only published in 2006, and later updated by several documents [46, 47, 48, 49, 50, 51, 52], mostly to update algorithms used for encryption or authentication.

SSH provides multiple authentication methods to securely log into a remote system, ensuring both security and flexibility. The most common method is **password-based authentication**, where users provide a username and a password to gain access. While straightforward, this method is vulnerable to brute-force attacks and credential theft, making it less secure if not combined with additional security measures, such as fail2ban [53]. A more secure alternative is **public key authentication**, where a cryptographic key pair (public and private key) is used. The user stores the private key on their local machine and adds the public key to the `authorized_keys` file on the remote server. This method eliminates the need for passwords, providing stronger security while enabling automated access for scripts and applications.

Beyond these traditional methods, SSH also supports more advanced authentication mechanisms. **Multi-factor authentication (MFA)** enhances security by requiring an additional verification step, such as a one-time password (OTP) generated by an authenticator app. Another option is **Kerberos authentication**, which enables single sign-on (SSO) by using tickets issued by a trusted authentication server, avoiding password reuse across multiple logins. Organizations managing large-scale infrastructures often utilize **SSH certificate-based authentication**, where a central certificate authority (CA) signs user keys, simplifying key management and access control. Lastly, **keyboard-interactive authentication** provides a flexible approach where the server dynamically requests additional credentials, such as security questions or temporary codes, further strengthening SSH security beyond traditional login methods.

SSH is a layered protocol, with an underlying transport layer (RFC4253 [44]) over which sessions for individual purposes, e.g., sFTP, X11 forwarding, port forwarding, or–commonly–an interactive terminal session can be started (RFC4254 [45]). For each session, different authentication and authorization requirements may be set, with authentication being defined in RFC4252 [34].

One of the most prevalent SSH implementation is OpenSSH [54] by the OpenBSD project. In addition, especially on embedded devices, Dropbear–a more resource conscious implementation–is widely used [55]. On a descriptive level, a study by Gasser et al. [56] presents a comprehensive analysis of SSH deployments in the wild, examining the prevalence, configuration, and security practices of SSH servers across the Internet.

2.2 SSH Attacks

SSH is a popular and extensively deployed protocol, making it a frequent target for attacks. Despite its improvements over the years, brute-force and dictionary attacks remain some of the most common threats against SSH.

2.2.1 Brute-Force Attacks

Enumerating passwords, possibly restricted to a word-list of known or default passwords, to bypass authentication for remote systems is most likely the oldest 'attack' on the Internet, and was already part of the Morris Worm [57]. Naturally, in the context of SSH, brute force attacks as part of Internet background noise have been a constant since the protocol has been wildly deployed [4, 8, 7, 30, 58].

Especially in recent years, the emergence of the 'Internet-of-Things' (IoT) led to a new upswing of SSH related brute-forcing, due to many of these devices having been deployed with weak and/or known default passwords [59, 60]. Accordingly, several botnets of IoT devices used for, e.g., (d)DoS attacks, including Mirai, have been created using mostly brute-force attacks [5, 6, 61, 62, 63]. While industry regulations and policy measures, particularly in the context of IoT [64, 65], are increasingly enforced, the fundamental issue persists.

Similarly, newer versions of SSH, by default, prohibit password authentication for the `root` user or, as in recent Ubuntu releases, disable remote `root` login entirely. From a technical standpoint, the most effective way to mitigate brute-force attacks on SSH is to disable password authentication in favor of more secure alternatives, such as token-backed public key authentication. However, passwords remain prevalent, especially on embedded devices and systems managed by less experienced users.

2.2.2 Protocol Weakness Attacks

Another type of SSH attack targets weaknesses within the SSH protocol itself. In 2001 Song et al. [10] examines how SSH, despite its encryption mechanisms, leaks information through inter-keystroke timing and packet size patterns. The authors demonstrate that an attacker can infer typed content by analyzing the timing of keystrokes sent in interactive SSH sessions. By applying statistical techniques such as Hidden Markov Models, they show that password guessing attacks can be significantly accelerated.

Later, in 2009, Albrecht et al. [9] introduces several plaintext-recovery attacks targeting the SSH protocol. The authors implemented a proof-of-concept attack against OpenSSH, demonstrating the ability to recover 14 bits of plaintext from any given ciphertext block with a probability of 2^{-14}, and 32 bits with a probability of 2^{-18}. These attacks exploit design flaws in the SSH Binary Packet Protocol (BPP) [44], particularly when a 128-bit block cipher is used in Cipher Block Chaining (CBC) [66] mode under default configurations. The vulnerabilities arise due to SSH's packet processing mechanisms, which were not adequately accounted for in prior security analyses.

In a more recent work, in 2024, Bäumer et al. [11] introduces a novel man-in-the-middle (MitM) exploit targeting the SSH protocol. By manipulating sequence numbers during the SSH handshake, an attacker can perform a prefix truncation, effectively deleting initial packets in the communication stream without detection. This vulnerability compromises SSH channel integrity, particularly when using encryption modes like ChaCha20-Poly1305 [67] or CBC with Encrypt-then-MAC [68]. The authors demonstrate practical implications, such as downgrading public key algorithms during user authentication and disabling keystroke timing defences. An Internet-wide scan revealed that approximately 71.6% of SSH servers support vulnerable encryption modes, underscoring the widespread impact of this flaw.

2.2.3 Countermeasures

A significant body of research on SSH attacks explores various countermeasures, often utilizing techniques such as network flow analysis. Drašar et al. [69] addresses the challenges of detecting dictionary attacks in high-speed networks, where traditional deep packet inspection methods are often too resource-intensive. The authors propose a model of malicious traffic inspired by industry practices, focusing on simplicity and effectiveness, which allow countermeasures against dictionary-attacks.

In 2012, Hellemons et al. [70] introduces SSHCure, an intrusion detection system (IDS) designed to detect SSH-based attacks in high-speed networks. Traditional IDSs often rely on packet inspection, which can be resource-intensive and less effective in encrypted or high-throughput environments. In contrast, SSHCure utilizes flow-based monitoring, analyzing aggregated data flows rather than individual packets, making it more scalable and efficient.

Similar to SSHCure, but covering multiple network protocols (including SSH), Hofstede et al. [71, 72] propose a network-based monitoring approach that leverages flow data exported using the IP Flow Information Export (IPFIX) protocol [73]. This method focuses on analyzing per-connection histograms of packet payload sizes, enabling the detection of brute-force attacks and subsequent compromises, even when the traffic is encrypted. By examining the distribution of packet sizes within network flows, the system can identify irregular patterns indicative of malicious activity.

Jonker et al. [74] further investigate the phenomenon of 'flat' traffic patterns in SSH brute-force attacks. 'Flat' traffic refers to network connections that exhibit similar characteristics in terms of packet count, byte count, and duration, often resulting from repetitive application-layer actions like login attempts during brute-force attacks. The researchers observed that, contrary to expectations, not all SSH dictionary attacks produce flat traffic patterns. Factors such as TCP retransmissions and control packets can introduce variability, making detection based solely on flow-level data challenging.

Recent research has focused on designing and analyzing countermeasure techniques that leverage machine learning [75, 76, 77, 78, 79, 80]. These studies employ various machine learning algorithms, including decision trees, random forests, Naïve Bayes, k-Nearest Neighbors (kNN), and deep learning models such as Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Stacked Auto-Encoders (SAE), to detect malicious login attempts with high accuracy while maintaining computational efficiency. By analyzing flow-based network features, these approaches minimize reliance on deep packet inspection, enhancing scalability for big data environments. The findings demonstrate that different models can achieve high detection rates, providing an effective solution for real-time security monitoring and intrusion prevention.

However, especially machine-learning based approaches often show a high number of false positives. In 2024, Sachin et al. [58] introduces a defense mechanism that successfully blocks 99.5% of SSH brute force attacks, significantly outperforming state-of-the-art rate-based blocking methods while reducing false positives by 83%.

A different approach to countermeasures against botnets is presented in GQ [81], a framework designed to safely and effectively measure and analyze modern malware by containing its execution within a controlled environment. This approach enables researchers to study malware behavior in detail while preventing any potential harm to the broader network.

In 2019, Cao et al. [82] introduced CAUDIT, a framework for continuously monitoring and auditing SSH servers to detect and prevent brute-force attacks. The paper discusses various brute-force methods and the use of public key authentication by malicious actors, providing a comprehensive view of the challenges in securing SSH servers against such attacks.

2.3 Attackers Behavior

A different approach to investigating SSH attacks focuses on analyzing attacker behavior. SSH honeypots, for instance, can capture valuable information such as executed commands and downloaded files, providing deeper insights into adversaries' tactics [3, 83]. This data helps improve the understanding of intruder strategies and enhances the characterization of their attack methods, ultimately contributing to more effective defensive measures.

Recent works apply different clustering [84] and natural language processing (NLP) techniques [85] to group attackers' IPs. In work by Shamsi et al. [84], the authors set up honeypots for 5 protocols (including SSH) in 3 regions of the AWS cloud for a duration of 20 months. They applied different clustering algorithms and identified features to cluster the attacker IPs. The work by Barron et al. [29] deployed 102 honeypots for 4 months and varied their characteristics, e.g., location and difficulty to break, to study the response by attackers.

Another body of work focuses on the initial phase of SSH connections to gain insights into the attacker's IPs. Ghiëtte et al. [86] deployed 4,500 honeypots for a month, focusing on fingerprinting the software stacks and tools used by adversaries to establish the SSH connection. They identified 49 tools used for SSH compromising attempts.

Wu et al. [87] deployed a honeypot on a /16 IPv4 prefix and reported their analysis of data spanning 1,000 days. However, due to the massive number of IP addresses within that prefix, i.e., 65k, their deployment is a lightweight low-interaction honeypot that does not record the executed commands [82]. Hence, their analysis includes only those attributes of the attacker that are observable upon connection, for example, SSH clients and IP addresses.

Once attackers compromise a system, usually the first order of action is ensuring persistence [32]. From honeypot studies [31], it is known that attackers regularly first 'scout' for vulnerable systems, and later–en mass–log in to systems for establish persistence and provisioning subsequent use, e.g., use as a loader, dropper, additional brute-forcer etc. In honeypot studies, several attack groups established persistence by installing their own public keys into the `authorized_keys` file, with some groups overwriting all existing keys [31].

2.3.1 Threat-Actor Characterization

As noted by Barron et al. [29], attribution and attacker profiling are among the most difficult aspects of defending against organized or state-sponsored threats. Studies also suggest that attackers' strategies, persistence, and the likelihood of specific attack types are influenced by target configurations, vulnerabilities, and perceived value[88, 89]. These studies emphasize the importance of differentiating between human-driven attacks and automated bot activity.

Honeypots play a crucial role in this characterization, as illustrated by Sadique et al. [90], who analyzed botnet behavior to understand attackers' actions post-compromise. Stone-Gross et al. [28] conduct an extensive and in-depth analysis of the Torpig botnet, providing valuable insights into its behavior and attack characteristics. A recent paper by Izhikevich et al. [91] showed that attackers are increasingly sophisticated when targeting honeypots.

In addition, other studies have pursued predictive methods, such as statistical models to forecast attacks [92] or machine learning applied to honeypot data to detect severe SSH attacks [93]. Research has also shown that malware developers increasingly use sophisticated techniques like polymorphism to evade detection [94, 95, 96].

2.4 SSH Honeypots

This section provides background information on SSH honeypots, different types of honeypots, and honeyfarms. In general, a *honeypot* mimics a real system to attract, log, and inspect potentially malicious activities [19, 97]. Honeypots can be deployed either as virtual machines or on physical hardware [98, 99, 100] and exist in various specialized forms, such as web honeypots [101, 102], mobile honeypots [103], or those designed specifically for worm detection [104]. Regarding SSH honeypots, Kippo [105] and Cowrie [22] are among the most widely used today [19], and commonly form the foundation of studies of malicious activity [106, 107].

To facilitate access to attackers, SSH honeypots enable logins without any or with easily guessable credentials. After a successful login, a honeypot tries to imitate a real system, e.g., by providing shell access, allowing to execute commands, or enabling downloads from remote servers. At the same time, the honeypot system logs all activity to be then able to analyze the behavior of potential malware.

Honeypots can be classified into different types using a variety of metrics. One commonly used metric to distinguish honeypots is the possible interactions that they provide. This allows us to classify them into low-interaction, medium-interaction, and high-interaction honeypots [108]. Low-interaction honeypots such as Honeyd [21] or Nepenthes [23] simulate only a very restrictive set of system behaviors. High-interaction honeypots such as The Honeynet Project [109], on the other hand, try to imitate a much broader set of services and interactions, providing also access to real-world operating systems without simulation. Medium-interaction honeypots such as Honeytrap [24] or Cowrie [22] can be seen as somewhere in between these two extremes, i.e., they are more sophisticated than low-interaction honeypots but less sophisticated than high-interaction honeypots.

Honeypots can also be run as a fleet to detect potentially malicious behavior more easily. When running more than one honeypot within the same network, this is referred to as a *honeynet*. Similarly, when multiple honeypots are deployed in different networks, this is referred to as a *honeyfarm*.

In this thesis, we make use of a data collected by GCA honeyfarm. The honeyfarm consists of 221 medium-interaction *Cowrie* [22] honeypot instances distributed over 64 networks in 55 countries worldwide.

Researchers have deployed honeypots to gain insights into attacks on various vulnerabilities in applications, devices, or protocols. These tools are deployed on multiple scales and network types, such as in clouds [110, 29], on campuses [82, 29, 111], or in eyeball networks [20]. Studies have used *vertical honeypots* exposing particular services and protocols, or *horizontal honeypots* offering multiple services and applications [111]. Another body of work have also used logs from real-world operational systems to study unsolicited or malicious requests [112, 113].

Honeypot Detection: If a honeypot is easily detected as such, this can negatively affect its effectiveness since adversaries may avoid honeypots to hide their malicious activities. Multiple studies have investigated these honeypot detection techniques [114, 115, 116], and presented techniques to measure the deployment of popular honeypot systems. In addition, there are also online services [117] that provide estimates on the probability of an IP being a honeypot.

Amplification Attacks: Amplification DDoS attacks are one of the popular areas where researchers used honeypots for their studies [17]. Since amplification attacks mostly make use of IP address spoofing, they are limited to UDP-based protocols such as NTP, CharGen, and DNS [118, 119, 120, 121, 18].

IoT Honeypots: In response to the increasing number of IoT devices, researchers have also deployed honeypots that simulate vulnerable IoT devices to study attacks on the IoT ecosystem. IoT honeypots have proven to be invaluable in identifying various malware families targeting IoT devices [59, 60], contributing information for botnet counteraction [122, 123, 124]. For example, Griffoen et al. [20] deployed 7,500 specialized Telnet-based IoT honeypots across eight networks for three weeks, primarily focusing on Mirai botnets [5] and their underlying infrastructure. Another study by Dang et al. [110] deployed 108 honeypots across multiple cloud providers and two ISPs to study fileless attacks (attacks that do not require downloading malware) on Linux-based IoT devices, categorizing the attacks based on the executed commands.

Companies Operating Honeypots: Maintaining and operating a large global fleet of honeypots across different types of networks for extended periods of time can become prohibitively expensive for individual research groups. Therefore, we observe security-focused companies [125], non-profit organizations [126], or their alliances operate such deployments and occasionally share their insights via threat intelligence tools [127], dashboards [128, 127] or white papers. In this thesis we take an in-depth look into unsolicited SSH (and Telnet) connection attempts by leveraging a globally distributed honeyfarm across 3 years that is available to researchers.

Network Telescopes: Similar to honeypots–designed to capture unwanted traffic–the network telescopes monitor traffic reaching Internet address space that is not assigned to hosts but is advertised to the global routing system [129]. Although network telescopes can provide insights about scanners [130], attacks [131, 132], and other Internet phenomena and characteristics [133, 134, 135], they are typically passive. Thus, no connections are established or monitored. In recent years, reactive network telescopes have been proposed to address this issue. In the work of Hiesgen et al. [136], a reactive network telescope "Spoki" responds to asynchronous TCP SYN packets and engages in TCP handshakes initiated in the second phase of two-phase scans. With the collected data, they investigated Mirai attacks based on the SYN packet and malicious behavior based on the payload of the initial TCP packet. However, honeypots can report the intruder's full activity and thus are more powerful in detecting malware, unauthorized access, and command execution. Large networks or content providers also operate network telescopes. Richter et al. [113, 137] have a deep look into the scanning activity on the Internet. Using the logs of a large content delivery network (CDN) they show that about 87% of the incoming connection are scanners. Table 2.1 presents a concise comparison of related work with respect to interaction types, scope of analysis, and study duration.

Honeypot Limitations: Nawrocki et al. [17] offer a different perspective on honeypots, also conducting an in-depth analysis of their limitations. Their findings highlight that one of the primary constraints lies in the placement and vantage points of honeypots. In this thesis we extend this discussion by identifying additional limitations, such as the inability of a honeypot to capture files downloaded by attackers due to the methods used to transfer them. This gap in monitoring suggests that modern attacks have become sophisticated enough to evade certain honeypot configurations, highlighting the need for ongoing advancements in honeypot technology to keep up with evolving threat tactics.

Table 2.1: Comparison of Interaction Type, Scope, and Time Analysis Across Related Studies

Year	2004	2010	2012	2016	2017		2019		2020	2022			2023	2025
Studies	Moore et al. [129]	Kotenko et al. [123]	Dainotti et al. [130]	Pa et al. [59]	Antonakakis et al. [5]	Luo et al. [60]	Dang et al. [110]	Prokofiev et al. [122]	Griffoen et al. [20]	Kepner et al. [124]	Hiesgen et al. [136]	Richter et al. [137]	Nawrocki et al. [17]	This Thesis
Interaction														
Active interaction	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓
Passive interaction	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
Reactive measurements	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
Scope of Analysis														
Specific botnets	✗	✓	✗	✓	✓	✓	✗	✓	✓	✗	✓	✗	✗	✓
Broad attacks	✓	~	✓	~	~	~	✓	~	~	✓	✓	✓	✓	✓
Time														
Long time study	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓
Short time study	✗	✓	✗	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓

✓ = Yes ✗ = No ~ = Partial

Chapter 3
Unwanted SSH Traffic

Honeypots have long served as an essential tool in the Internet security landscape, offering a controlled environment to detect and analyze malicious activities on the Internet. By simulating vulnerable systems, honeypots attract attackers and allow researchers to observe their behavior—from scanning and reconnaissance to exploitation and malware deployment. Despite their widespread use in both academic research and commercial applications, the data collected by honeypots is often fragmented or limited in scope, making it difficult to gain a comprehensive understanding of global threat activity. Furthermore, while large-scale commercial honeyfarms exist, the data they gather is typically proprietary, leaving researchers with limited insight into the full breadth and diversity of observed threats.

In this chapter, we leverage a rare opportunity to collaborate with the Global Cyber Alliance (GCA), a non-profit operator of a large-scale honeyfarm. We analyze data collected over a 32-month period from 221 medium-interaction honeypots deployed across 55 countries and 64 networks. This dataset allows us to study unsolicited and unwanted activity on the Internet at an unprecedented scale. By examining over 750 million sessions, we provide a detailed characterization of the types of threats visible to honeypots, both individually and collectively. Our findings reveal significant differences in visibility across identical honeypots, uncover long-lived attack patterns, and highlight the strategic behavior of adversaries. These insights not only inform the design of future honeypot deployments but also offer actionable knowledge for improving threat detection and mitigation efforts globally.

The contributions of this chapter can be summarized as follows:

- We analyze more than 750 million sessions from December 2021 to August 2024, collected at 221 honeypots in the studied honeyfarm. We characterize the sessions as no credential, failed login, no command execution, command execution, and command execution with file download.
- We study the characteristics of around 4.8 million honeypot client IPs from more than 24.4 thousand networks worldwide. We notice that more than 40% of these IPs participate in more than one type of activity (e.g., port scans vs. executing commands), and 20% of all activity in our dataset is observed for more than a week.

- We report that around half of the client IPs are located in a different geographical region than the targeted honeypot. However, 25% of client IPs that connect and execute commands or download malicious code are in close proximity to the honeypots.
- We analyze more than 70 thousand unique hashes of files generated by clients on the honeypots. We find striking differences in the number of IPs associated with each hash: from tens of thousands to only a handful of IPs. We conclude that some high-profile attacks can be mitigated by blocking a handful of IPs, while others are more difficult to mitigate.
- Some of the popular hashes are visible for a year or more. However, the dominance of hashes varies over time. On average more than 500 unique hashes are visible per day, with the number of first-seen hashes varying between 2% and 60% of all daily hashes.
- Our study shows prominent differences between honeypots regarding the number of sessions, IPs, hashes of files generated by commands, and first-seen hashes in the honeypot. Around 20% of the honeypots observe $5\times$ to $30\times$ more unique file hashes than the rest of the honeypots. Yet, even the honeypots with the highest number of observed hashes contribute less than 10% of the overall hashes in the honeyfarm.
- We also report that honeypots that observe the highest number of sessions or client IPs are not the ones that collect the highest number of files hashes generated by intruders generating or modifying files. However, the honeypots that collect the highest number of file hashes are typically the ones that observe the hashes earlier than the rest in our honeyfarm.

3.1 Honeyfarm Dataset

Our work relies on data from an operational honeyfarm that we obtain by establishing a collaboration with Global Cyber Alliance (GCA) [27], a large honeyfarm operator. The data is processed and analyzed in situ, using the provided interface. We anonymize sensitive data to comply with the requirements set forth by the collaboration. One of their projects involves building and operating an SSH/Telnet honeyfarm. We leverage almost 3 years of data from their honeyfarm, which has been operational since December 2021 to August 2024.

The goal of deploying honeypots in this honeyfarm is to have them geographically distributed across different countries as well as in different networks, with a focus on residential networks. The honeyfarm consists of 221 identically configured honeypots in 55 countries and 64 Autonomous Systems (ASes). Each honeypot is realized using a customized version of the Cowrie Honeypot suite [22], a medium interaction SSH and Telnet honeypot that is designed to log possible brute force attacks as well as shell interactions that a possible intruder executes. The selection of the Cowrie honeypot software is driven by its ease of use and because it covers a relevant attack vector for IoT devices: SSH and Telnet. In Figure 3.1, we show a visualization of the 55 countries where the 221 honeypots are deployed. While most countries host a single honeypot, some, e.g., the US and Singapore, host multiple ones.

During our observation period from December 1, 2021, until August 24, 2024 all 221 honeypots are active. Within this time frame, each honeypot reports summaries for each SSH or Telnet session. Each successful TCP connection handshake by a client on either the SSH port 22 or the Telnet port 23 creates a new session in the honeyfarm database. A

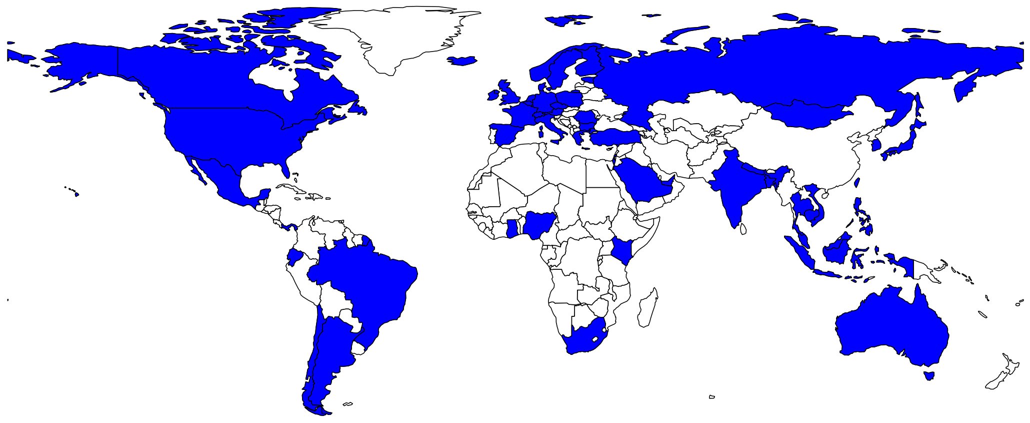


Figure 3.1: Presence of 221 honeypots in 55 countries.

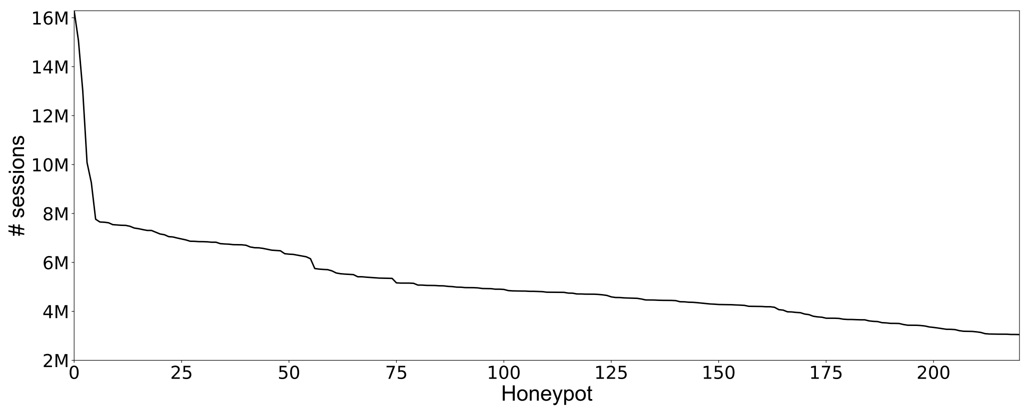


Figure 3.2: Honeypot activity: Number of sessions per honeypot (sorted by # of sessions).

session is ended either by a TCP connection tear down from the client or a timeout by the honeypot, which is configured to be three minutes. For each session, the honeypot records basic session information, which includes the start time, the end time (including timeout), the IP and port of the honeypot as well as the client. In addition, if SSH is used, it records the client SSH version from the SSH handshake if available.

Moreover, the honeypot records the interactions of the client with the honeypot, namely, used credentials for login and executed commands. For each login attempt, it records the credentials used as strings and whether the used credentials are accepted The honeypots are configured to allow password-based SSH authentication using the username root and by supplying any password except “root”. Public-key-based SSH authentication is not supported. For Telnet, the same authentication rules as for SSH are in place. After a successful login, the client has access to a Unix-like shell that emulates common Unix commands. As such, the honeypot records each command executed by the client in a list of “known” or “unknown” commands. Known commands are emulated by the honeypot; unknown ones are simply recorded. If a command includes a URI (this includes anything retrieved from a remote target, including retrievals via FTP, HTTP, SCP, etc.), the URI is recorded as well. If a command results in a creation or modification of a file, a hash of the file content is recorded.

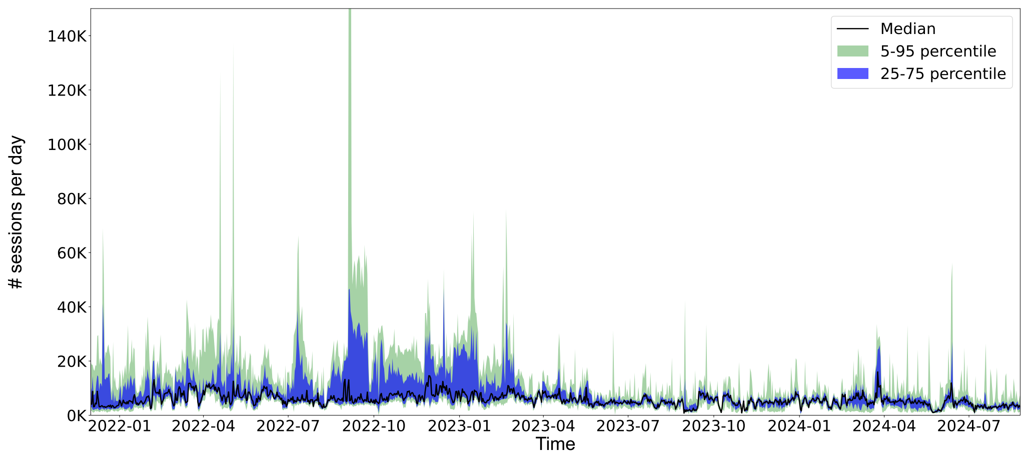


Figure 3.3: Honeyfarm activity across time for top 5% of honeypots with the most sessions: median, IQR, and 5th/95th percentile ranges.

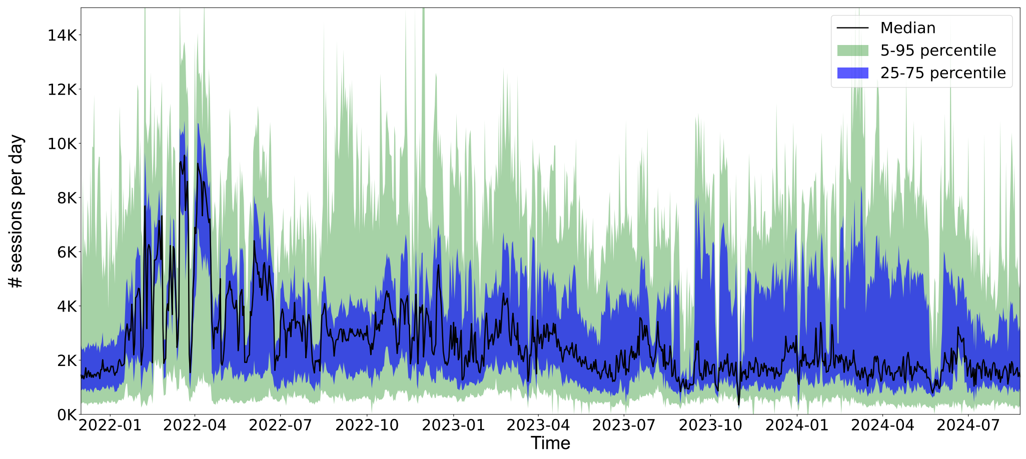


Figure 3.4: Honeyfarm activity across time for honeypots, showing median, IQR, and 5th/95th percentile ranges for the number of sessions per honeypot.

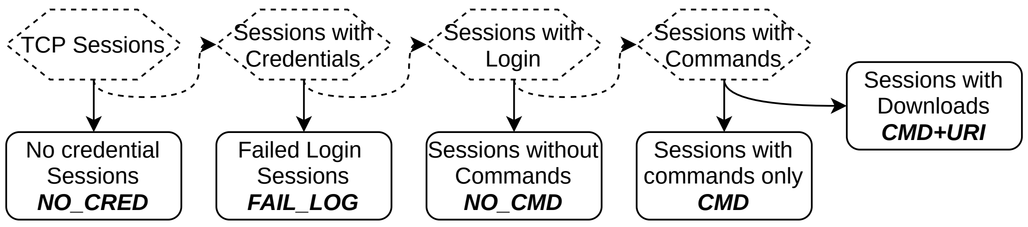

TCP Sessions
Sessions with Credentials
Sessions with Login
Sessions with Commands
Sessions with Downloads CMD+URI
No credential Sessions NO_CRED
Failed Login Sessions FAIL_LOG
Sessions without Commands NO_CMD
Sessions with commands only CMD

Figure 3.5: Flow diagram: Session classification.

Overall, a total number of more than 750 million sessions are recorded throughout the measurement period. A majority of 72.41% of these sessions use SSH, while 27.59% use Telnet to connect. The median daily number of sessions across all honeypots is roughly 1.6 million. We note that the daily activity differs greatly across honeypots, see Figure 3.2, which shows the number of sessions per honeypot sorted by activity. We find that the top 10 honeypots see 14% of all sessions. Moreover, we observe a knee in the distribution around 11, after which the difference in activity for the remaining honeypots is drastically smaller. Indeed, even the least targeted honeypot still sees more than 360,000 sessions. The 10 most popular honeypots are located in 10 different countries in 7 different ASes. As such, their popularity does not appear to be linked to a specific network or geographical location. Furthermore, it is also not an artifact of a specific time interval, as we see next.

Indeed, the minimum/maximum daily activity of a single honeypot involves 94/1,634,886 sessions. Note that the observed activities change across time according to the activity of scanners and possible attackers. Given the large differences in popularity for the top 5% of honeypots and the remaining ones, we show the median number of daily sessions over time in Figure 3.3 (black line) as well as the interquartile range (IQR, 25% – 75%, blue line) together with the 5th and 95th percentile range (5% – 95%, light green line) for the top 5% of honeypots only. Figure 3.4 shows the same analysis for all the honeypots.

We clearly see activity spikes, e.g., on September 5, 2022, when some honeypots see a substantial increase in activity, see Figure 3.3. Other spikes are visible in May 2022. Still, we see that the median typically follows the 75% and 95% line, as shown in Figure 3.4. The 25% and the 5% lines are often smoother and do not show as many spikes, with the exception of the 25% line for spring 2022, where it follows the spikes.

To summarize: First, not all honeypots get the same attention from scanners and attackers in terms of the number of sessions. We see that the most targeted honeypot sees more than 30$\times$ the number of sessions than the least targeted one. Moreover, the top 10 honeypots account for 14% of all sessions in the dataset. Finally, the daily activity can go above 1.5M sessions per honeypot.

3.2 Honeypot Session Categories

From our data, we can identify different classes of client honeypot interactions. Namely, we use the following subcategories for our sessions, see Figure 3.5:

NO_CRED: This category includes all sessions where the client never attempts to log in. Accordingly, the honeypot never sees any credentials. Such sessions can, e.g., be the result of "scans" for open ports without a "login attempt".

Protocol	NO_CRED	FAIL_LOG	NO_CMD	CMD	CMD+URI	Total
SSH and Telnet	26.3%	38.7%	11.6%	23%	0.4%	100%
SSH	24.74%	98.92%	96.97%	92.45%	67.12%	72.41%
Telnet	75.26%	1.08%	3.03%	7.55%	32.88%	27.59%

Table 3.1: Percentage of total sessions per category (top row), and per protocol (SSH/Telnet only) percentage of sessions in each category (second and third column).

FAIL_LOG: Sessions in this category contain login attempts which do *not* succeed. The honeypot only allows login via username and password and not via keys (see Section 3.1). Hereby, the username must be "root", and the password must be any string except "root". The motivation for this password policy is to check for "root login" attempts rather than "regular user login" attempts.

NO_CMD: This category includes sessions with a successful login that never executed any follow-up commands. Note, that there might have been unsuccessful login attempts prior to the successful one within the same session.

CMD: Sessions in this category have a successful login and execute commands (known or unknown ones). However, they do not access any external resources via a URI.

CMD+URI: These sessions have a successful login and executed commands by the client. In addition, the client tries to explicitly access an external resource via a URI.

Based on these categories we define the following different client behaviors:

Scanning: The sessions in the NO_CRED category can best be described as scanning behavior, as no credentials are being sent.

Scouting: The sessions in the FAIL_LOG category can be characterized as scouting behavior, since the clients tried to log in using some credentials.

Intrusion: The sessions in the NO_CMD, CMD, and CMD+URI categories can be described as intrusion behavior, since the clients were able to get access to a shell, with some even executing commands.

Table 3.1 summarizes the results of the session classification. The top row contains the results for all sessions. The bottom ones are the fractions for SSH and Telnet, respectively. Note that SSH accounts for three-quarters of all sessions (see the last column). Overall, more than a quarter of all sessions only check if the ports are open without attempting to log in. This category is dominated by Telnet accounting for more than three fourths. Failed login attempts account for 38% of all sessions. This is fully dominated by SSH with more than 98%. In 11.6% of all sessions, the clients successfully log in, but do not execute any commands—again dominated by SSH. In 23% of all sessions, the client executes commands, and in more than 0.4%, the client accesses external resources. Here, SSH accounts for more than 60% of all these sessions.

For those sessions involving commands, we note that about one third create or modify files, and 0.5% create or modify multiple files (at least two). The remaining ones do not involve file system access. Table 3.2 lists the ten most popular successful passwords. Some of them are predictable and are to be expected (e.g., "admin", "1234", or

Password	
3245gs5662d34	admin
1234	dreambox
vertex25ektks123	12345
password	broadguam1
passw0rd	alpine

Table 3.2: Top 10 most used successful passwords (row-major order).

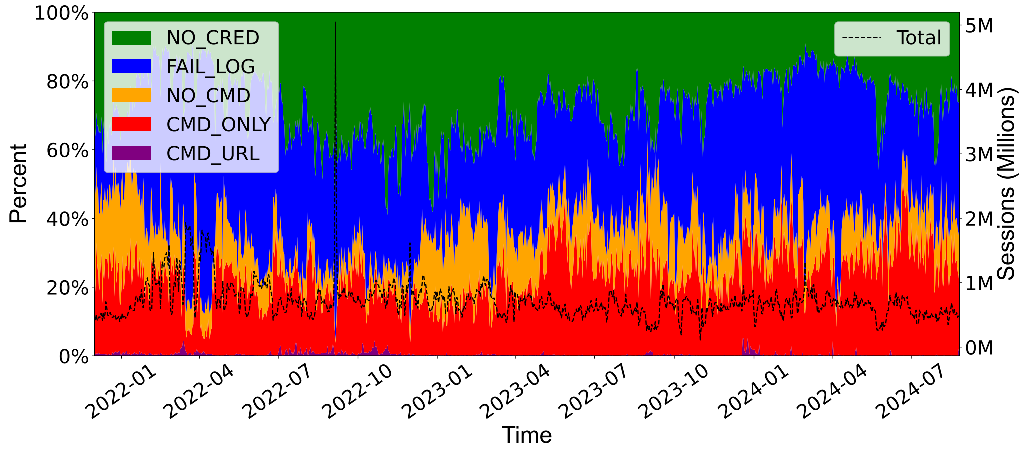


Figure 3.6: Honeyfarm activity: Across time (black line, right axis) and by category (in percentages, left axis).

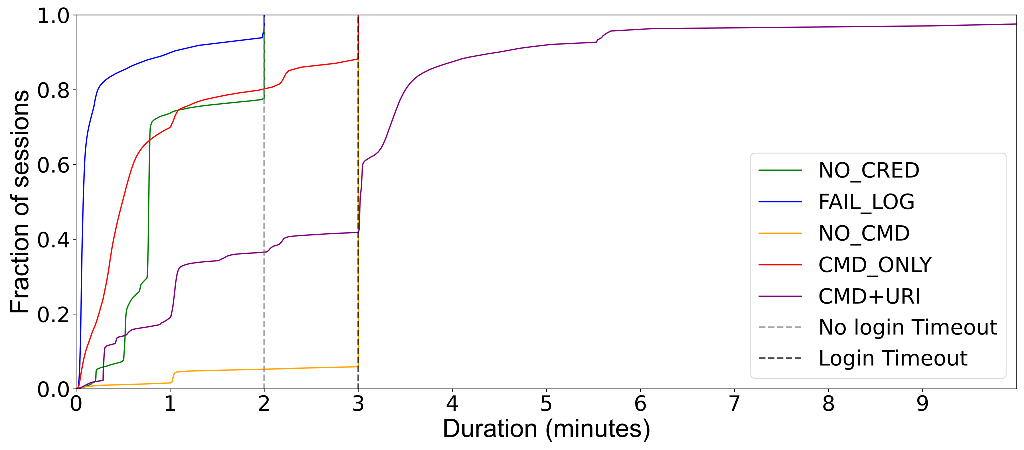
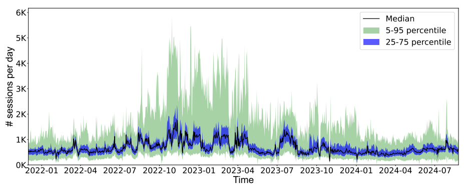


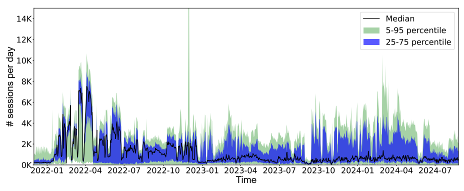
Figure 3.7: ECDF of honeyfarm session duration by category.

“passw0rd”), while others are very specific. This may indicate that these specific passwords might be part of a larger attack campaign, or have access to a leaked password database. Among the most often used usernames we see the strings “nproc”, “admin”, and “user”. However, as they are not root, the login is unsuccessful in these cases.

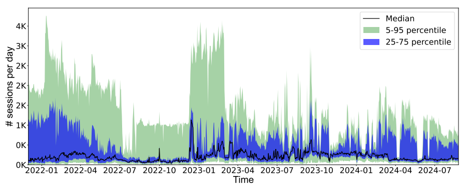
Next, in Figure 3.6, we show how the fraction of the sessions in each category changes over time using a stacked area plot (left y-axis). In addition, the figure shows the total



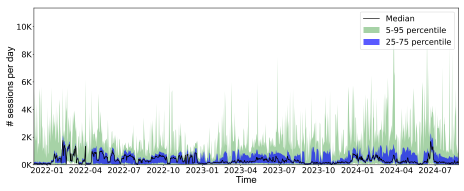
(a) NO_CRED



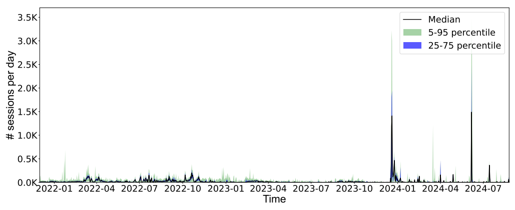
(b) FAIL_LOG



(c) NO_CMD



(d) CMD

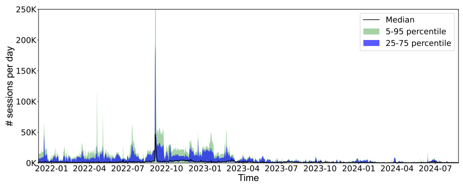


(e) CMD+URI

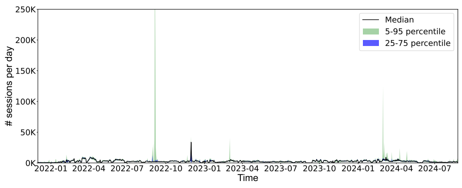
Figure 3.8: Honeypots activity by category across time: Showing median, IQR, and 5th/95th percentile ranges for the number of sessions per honeypot.

activity in the number of sessions as a black line (right y-axis). We note that in the period of Spring 2022 and Spring 2023 the fraction of sessions in the NO_CRED category increases, indicating that the fraction of scanning activity increases. Additionally, at the beginning and end of our observation period, the fraction of sessions with successful logins that never execute any commands (NO_CMD) makes up more than 20%. Upon closer inspection, we find that it is a single prefix that originates most of these sessions, which is mainly active during these time periods. According to RIPEstat [138], this prefix is originated by a Russian datacenter. The fraction of sessions with commands is relatively constant throughout our observation period. It only decreases during the spikes in Spring 2022, December 2023, as well as the large spike on September 5, 2022, when the overall activity is dominated by sessions with failed logins. We also see that the sessions with URIs occur in bursts, most likely related to specific types of attacks.

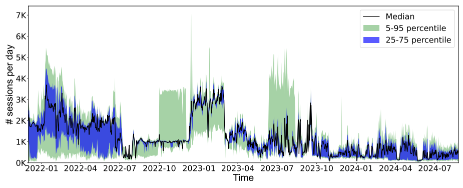
In Figure 3.7 we show the empirical cumulative distribution function (ECDF) of the session durations by category. In addition, the plot highlights two timeout values: the one for no login (dashed gray line) and the successful login (dashed black line). Overall, we see that the session durations increase with increasing interactions between the client and the honeypot. We confirm that the honeypots terminate many sessions due to timeouts. This is, in particular, the case for more than 90% of all sessions that log in successfully but never execute any command (NO_CMD). Interestingly, most sessions which never attempted to log in (NO_CRED) or with failed login (FAIL_LOG)



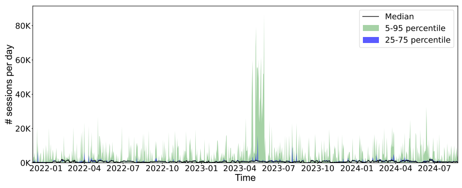
(a) NO_CRED



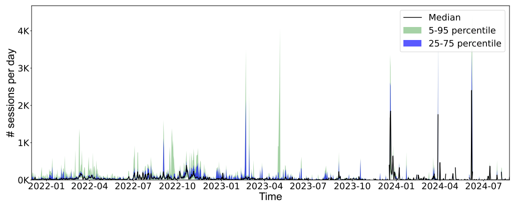
(b) FAIL_LOG



(c) NO_CMD



(d) CMD



(e) CMD+URI

Figure 3.9: Honeypots activity by category across time for the top 5% honeypots: Showing median, IQR, and 5th/95th percentile ranges for the number of sessions per honeypot.

are terminated after 3 unsuccessful tries (0.3% of all SSH session by the honeypot) or by the client. In addition, we note that a substantial number of sessions with commands (CMD) trigger the login timeout and that some sessions with URIs (CMD+URI) cross the timeout—they last longer than three minutes. This is due to the reset of the timeout period while waiting for the external resource.

So far, we looked at the overall activity of the honeyfarm. Next, we show how the number of sessions in each category varies by honeypot across time in Figure 3.8. Therefore, we reuse our earlier visualization for all sessions, recall Figure 3.3 and 3.4, but this time we have separate plots for each of the categories.

We show the activity for the top 5% honeypots (sorted by session activity) in Figure 3.9. We notice that for both NO_CRED and FAIL_LOG, three of the honeypots observed a significant amount of sessions on September 9, 2022. Further investigation did not show any difference between the sessions received by these honeypots and the rest. In the case of CMD and CMD+URI, we see that only a single or two honeypots are reporting spikes during the entire period of our study. In Figure 3.9(c) we notice that the top 5% of the honeypots observe a similar activity. An intense activity from the start of our study period (December 2021) until July 2022, followed by a drop and another rise in the first three months of 2023.

Overall, we see that the activity in each category differs; this is underlined also by the fact that the y-axis scale differs. The most obvious observations are (a) that NO_CRED has a substantial baseline activity for all honeypots at all times, indicating that scanning does not stop; (b) that FAIL_LOG has a similar shape to the overall one, which relates to the point that it contributes a substantial share of sessions; (c) spikes are often due to activity seen by only a small subset of the honeypots, e.g., the spike on November 5, 2022, in FAIL_LOG; for CMD there is a large variation with little baseline activities across all honeypots except during the spring of 2022; (d) there is a large variation in terms of targeted honeypots for NO_CMD sessions, and there are almost no spikes visible in the other categories; and (e) both sessions in the CMD as well as the CMD+URI category are quite spiky, indicating that there are time periods with substantial activity. We also note that towards the end of 2022, a subset of the honeypots are scanned more often, i.e., the variance increases for NO_CRED sessions.

To summarize: By splitting the honeypot sessions into different categories, we notice the first characteristic behaviors. We observe that the NO_CMD sessions tend to be terminated by timeouts while most of the FAIL_LOG and NO_CRED sessions are closed before the one minute mark. Another observation is that the NO_CRED category is more stable over time, while the rest of the categories tend to have multiple spikes across our observation period. The first hints at honeypots keep being scanned at a constant rate, while the latter suggests that the scouting and more specifically, the intrusion categories are more volatile over time. These bursts of activities might be linked to different attack campaigns.

3.3 Honeypot Clients

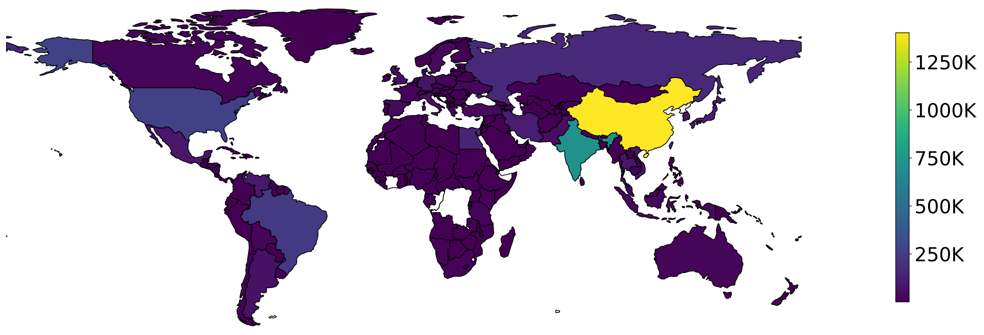
Next, we turn our attention to the characteristics of the clients that establish connections to the honeyfarm.

3.3.1 Client IPs per country

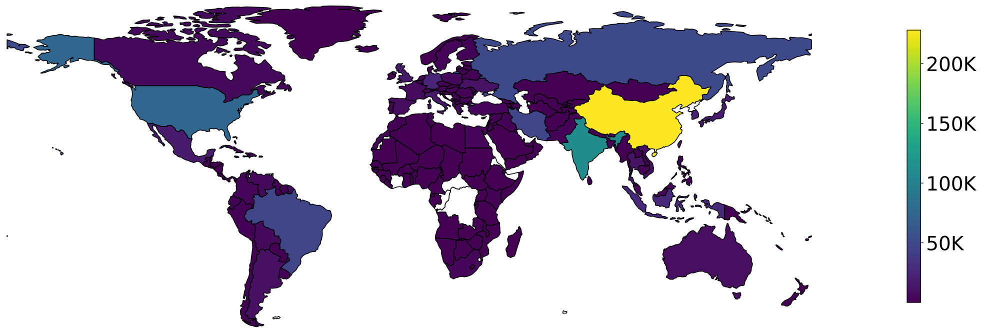
Overall, the monitored honeypot sessions involve more than 4.8 million unique IPv4 addresses from more than 24.4k ASes during the 32 months observation period. Note, these IPs are unlikely to be spoofed as all sessions involve a successful TCP handshake. To determine the physical location of the clients, we use the commercial API of Maxmind [139] to map the IPs to countries. Figure 3.10(a) shows the distribution of the clients on the world map using a logarithmic scale. Overall a large fraction of the honeyfarm client IPs originate from China (31%), India (9%), the US (8%), Russia (5%), Brazil (5%), Taiwan (5%), Mexico (3%), and Iran (3%).

3.3.2 Client IPs Over Time

Then, we study how the daily population of client IPs changes over time. Figure 3.11 shows the number of unique IPv4 addresses observed per day for each category for the duration of our study. The first surprise is the substantial increase in the scanning activity (NO_CRED) after about two months. It seems that it takes scanners some time to discover the honeypots and include them in "regular" scans. Indeed, at the same time, the command execution is also much more prominent (see CMD line). Overall, we notice that the number of client IPs varies substantially per category. The number of client IPs with NO_CRED sessions is higher than those with FAIL_LOG and CMD sessions. The number of client IPs involved in NO_CMD is small, except during the last five months



(a) All sessions.



(b) Sessions with commands (CMD + CMD+URI).

Figure 3.10: Global distribution of client IPs connecting to the honeypots.

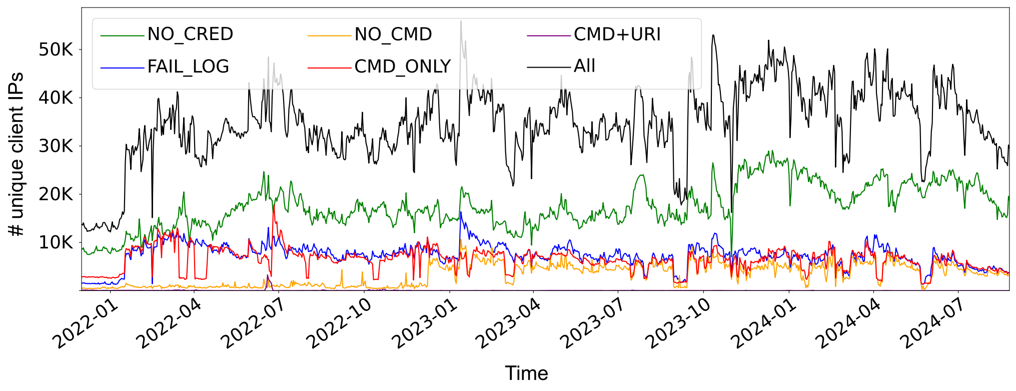
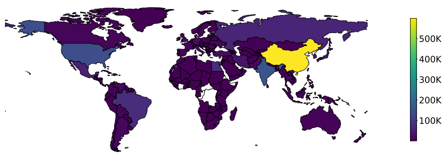
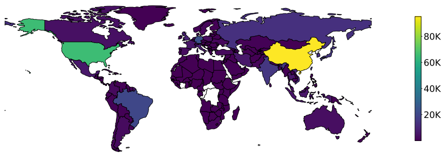


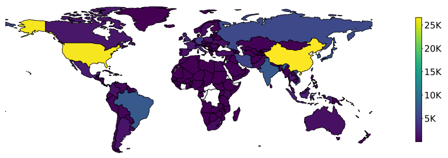
Figure 3.11: Number of unique IPs per session type.



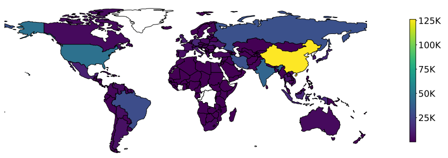
(a) NO_CRED.



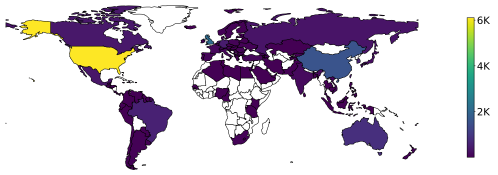
(b) FAIL_LOG.



(c) NO_CMD.



(d) CMD.



(e) CMD+URI.

Figure 3.12: Honeypot client IPs that establish sessions per country split by category.

of our study. The number of client IPs with CMD+URI is very low for most of our study period.

There is also a substantial variation in the number of client IPs per category. For example, the number of unique IPs for NO_CRED varies between 10k and 50k during the 32 months of our study. We also notice that the number of unique IPs for FAIL_LOG and CMD are of the same order of magnitude, around 10k IPs between the beginning of 2021 and the beginning of 2023. However, there is a noticeable change at the end of 2021 from around 3k to 10k and a substantial increase for FAIL_LOG in 2023 from 10k to more than 12k. The daily number of client addresses that are categorized as NO_CMD is relatively low, around 1k IPs, for the first part of our study period. However, there is a noticeable increase after December 2022 reaching the levels of CMD at the level of 10k IPs. The number of client IPs that are categorized as CMD+URI sessions is overall small, with a noticeable spike in June 2022 with more than 2,500 IPs.

We note that the number of daily unique IPs is small compared to the number of IPs observed during the full period of our study. This implies that, typically, the same set IPs do not contact the honeyfarm continuously. It also indicates that a large number of client IPs can be utilized for scouting as well as other more suspicious interactions, and with low frequency. This makes these IPs more difficult to be detected as potential malicious ones when the window of observation is small.

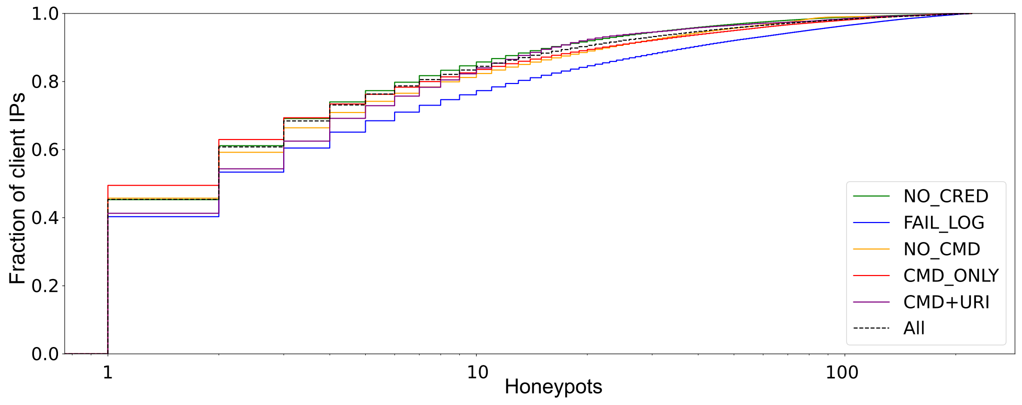


Figure 3.13: ECDF of number of honeypots contacted per client by category.

3.3.3 Client IPs across countries by category

Next, we revisit the geographic distribution of the clients but this time per category. Overall, we observe similar distributions but with some variations. Figure 3.10(b) shows the distribution of IPs on the world map for CMD + CMD+URI. For comparison, we show the maps for all categories in Figure 3.12. NO_CRED sessions involve 4.1 million unique IPv4 addresses in 21 thousand ASes during our study period. Again, the largest fraction of them originates from the US, China, Taiwan, Russia, and Iran. A large fraction of client IPs are attributed to FAIL_LOG sessions, see Figure 3.12(b), are hosted in Asia. Although the US is still the most popular origin, China, Japan, Vietnam, Singapore, and India are at the top of the list. Moreover, the set of these IPs is relatively smaller compared to NO_CRED. We observe 723 thousand unique IPv4 addresses in 16.7 thousand ASes. The number of client IPs that are classified as CMD are higher to those classified as FAIL_LOG. Indeed, we observe 940 thousand unique IPv4 addresses in 16.1 thousand ASes, see Figure 3.10(b).

Around 384 thousand of these IPs also establish FAIL_LOG sessions with honeypots in the honeyfarm. This is to be expected as compromised hosts are used to establish connections with different credentials. At the top of the list of countries that host IPs that establish CMD sessions are the US, China, Japan, India, and Brazil, see Figure 3.10(b). The number of IPs involved in the NO_CMD session is smaller compared to CMD. Indeed, we observe only 160 thousand IPs in 8.5 thousand ASes. This can be attributed to the fact that the majority of client IPs that successfully login to any of the honeypots in the honeyfarm execute commands. At the top of the list are Russia, Germany, the US, Vietnam, and Sweden.

Finally, only a small number of client IPs are involved in CMD+URI sessions, as such sessions are only a very small fraction of all sessions. The clients in this category are 20 thousand IPv4 addresses in 1.6 thousand ASes. Among the top countries that host these client IPs are the US, Netherlands, France, Bulgaria, and Romania, see Figure 3.12(e). The main take-away is that as the interactions become more involved, the top countries hosting the clients changes.

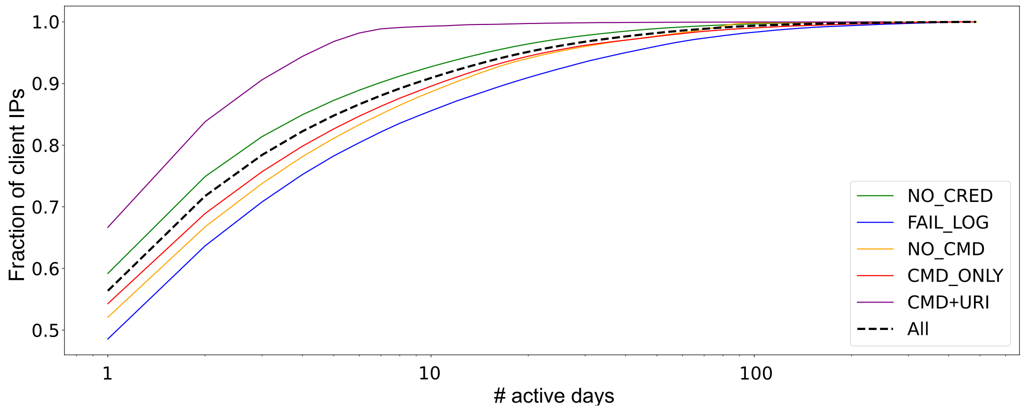


Figure 3.14: ECDF of number of days that a client IP is observed by category.

3.3.4 SSH vs. Telnet Activity

Next, we look at differences per protocol. Overall, recall that SSH is much more common than Telnet. However, for NO_CRED we observe 75% Telnet and only 25% SSH sessions. This is also true when looking at the country granularity. However, some countries differ. For example, for Vietnam, Brazil, and Lithuania, SSH dominates. For other countries, e.g., Italy and Canada, SSH contributes less than 10%. NO_CMD is clearly dominated by SSH except for some special cases, e.g., Switzerland and Bulgaria. Manual cross-checks show that these are due to one spike in activity.

3.3.5 Client IPs vs. Honeypots

Given our earlier observation regarding the reuse of client IPs, we now investigate the relationship between client IPs and honeypots more closely. In Figure 3.13 we show an ECDF of the number of honeypots within our honeyfarm that are contacted by each client IP. We acknowledge that dynamic IP address assignment may skew the distribution towards lower numbers.

Overall, we see that more than 40% of all client IPs only contact a single honeypot. Yet, 18% contact more than 10, and 2% contact more than half of the honeypots (> 110). Surprisingly, clients that interact more with the honeypots are likely to contact more of them. The largest exception, though, is the FAIL_LOG category. One possible explanation is that FAIL_LOG sessions can indicate a reconnaissance behavior of an intruder in order to detect more potentially vulnerable hosts.

Next, we ask how long client IPs are interacting with our honeyfarm. Figure 3.14 shows an ECDF of the number of days that a client IP contacts any of the honeypots in our honeyfarm. Note, that most client IPs are only visible for a single day. Yet, more than 100 client IPs are active almost every day ($> 90\%$) during our study period. We again see that FAIL_LOG sessions are most prominent. On the other hand, IPs that involve CMD+URI are seen for the lowest number of days. Indeed, these are mainly consecutive days. This hints at the possibility that potential attackers are trying out different strategies.

So far, we have taken the viewpoint of a client IP. Now we are switching to the the viewpoint of honeypots and ask how many different clients are contacting each one of the honeypots within our honeyfarm. Figure 3.15 shows the number of unique client

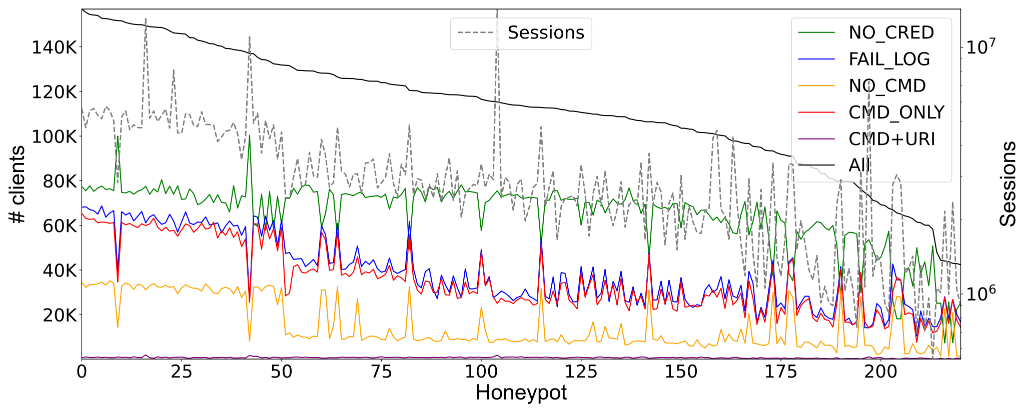


Figure 3.15: Number of clients per honeypot per category.

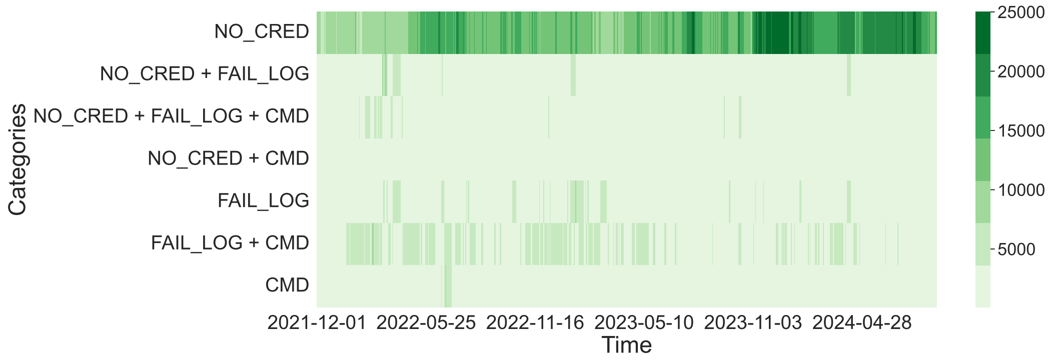


Figure 3.16: Number of clients per combinations of category across time (per day).

IPs per honeypot sorted by the number of clients. We see that a few honeypots (the top 10 ones) are contacted by many more clients than others. Surprisingly, these are not the same as the ones that see the largest number of sessions, recall Figure 3.2. For comparison, we add the number of sessions to Figure 3.15 using the right y-axis using log scale.

We also include the number of client IPs per category within the figure. Note, that these do not add up, as some clients have sessions in multiple categories. Overall, the number of clients involved in scanning is more than twice as many as those involved in more advanced honeypot interactions. The curves for clients with sessions in FAIL_LOG and CMD sessions follow each other closely, whereby the one for FAIL_LOG is just slightly larger. Sessions with NO_CMD and CMD+URI are initiated from just a small fraction of the clients.

To underline that some IPs are indeed active in multiple categories, we check how many client IPs are involved in sessions from multiple categories on the same day in Figure 3.16. We pick three of our categories. We investigate NO_CRED as this indicates scanning, FAIL_LOG as this may be the first step of interaction with the honeypot, and CMD as this involves substantial interactions with the honeypot. We find that more than 2 million IPs are only involved in scanning (NO_CRED). We notice that scanning is often combined with login attempts as well as command execution, e.g., in Spring 2022.

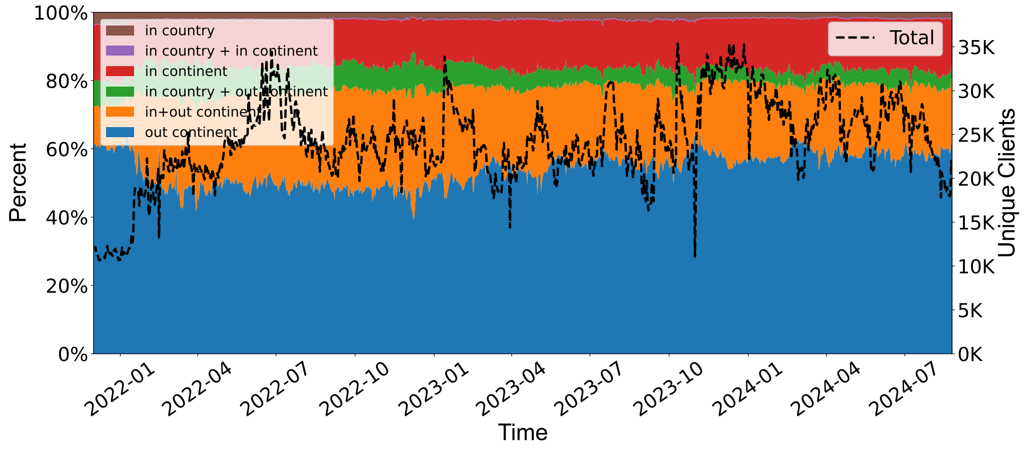


Figure 3.17: Regional diversity: client vs. honeypot across time (left y-axis) and number of clients (right y-axis). All Sessions.

Scanning combined with commands rarely occurs on the same day. However, failed logins together with commands are very common but hardly occur on the same day as failed login only. Commands only complement those that also do failed login, e.g., in Summer 2022.

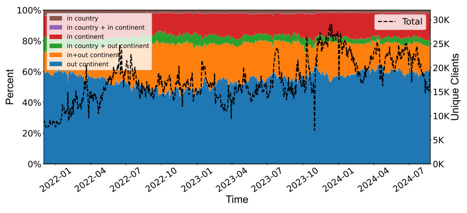
3.3.6 Regional Diversity

Next, we take advantage of the geographic diversity of the honeyfarm. This allows us to check if the client and the contacted honeypot are in the same country, on the same continent, or in completely different geographic regions. In Figure 3.17, we show the regional diversity of client/honeypot interactions across time as a fraction of clients. In addition, the plot also shows the number of clients observed per day. Overall, we see that most clients contact honeypots that are not on the same continent. This makes up more than 50% of all interactions every single day. The next most common class involves clients that contact a honeypot on the same continent and on another continent. This indicates that locality is not the primary motivation when clients choose address space for scanning or testing for vulnerabilities. The fraction of clients whose interactions stay within a single country is tiny. Even if we include clients that interact within the country as well as outside, the fraction still remains small. One caveat is that the honeyfarm has no deployment in China, yet a substantial share of clients (31%) is geolocated in China.

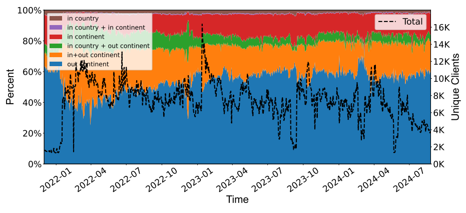
When further digging into the data by looking at the different categories, we note similar behavior for NO_CRED, FAIL_LOG, NO_CMD, as well as CMD. Figure 3.18 shows the regional diversity of clients vs. honeypots by category. In almost all categories, except CMD+URI, we notice that a significant amount of sessions are established between a honeypot and a client located on a different continent.

However, those sessions that involve URIs show more geographic proximity, see Figure 3.18(e). Here, the fraction of sessions out of the continent is substantially smaller, while the fraction in and out of the continent is substantially larger. This indicates that geographic locality may matter more when clients start picking targets for specific tasks.

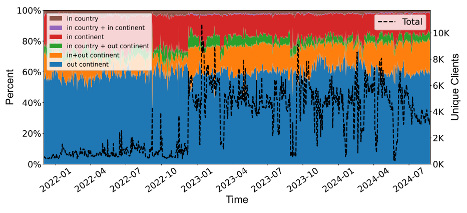
To summarize: First, the number of unique client IPs increases over time. This can be explained by the fact that it takes a while until the honeypot IPs become well-known.



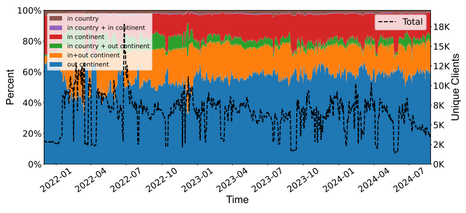
(a) NO_CRED.



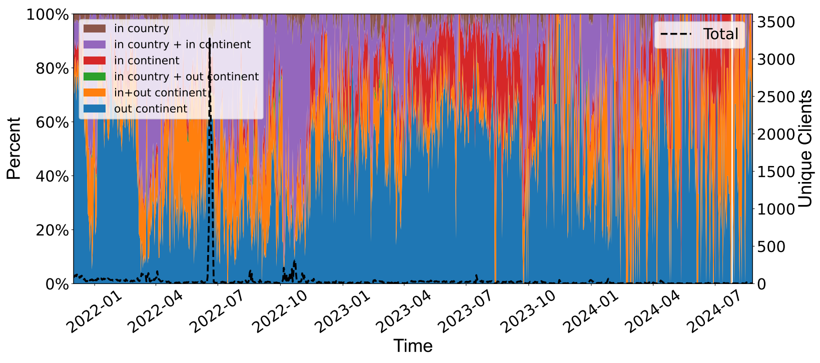
(b) FAIL_LOG.



(c) NO_CMD.



(d) CMD.



(e) CMD+URI.

Figure 3.18: Regional diversity: client vs. honeypot by category.

Commands		
grep name	awk	cat /proc/cpuinfo
free -m	crontab -l	w (whois)
uname -a	bash	chpasswd
mkdir .ssh	rm -rf .ssh	chmod -R go= /.ssh
top	echo "ssh-rsa AAA..."».ssh/authorized_keys	
shell	cat /proc/mounts	wget
tftp	history -c	chmod 777 bins.sh

Table 3.3: Top 20 commands (row-major order).

Second, more than half of client IPs contact multiple honeypots. Also, we notice a substantial number of IPs involved in more than one type of session. This sort of behavior makes the differentiation between a "malicious" and a "harmless" actor much more difficult. Third, more than 50% of client IPs are seen active only for one day. In CMD+URI sessions, this value increases to almost 70%. This result shows that IP blocking might not be a good option, yet we will later show that this is not always the case. Finally, intrusion sessions, specifically the CMD+URI sessions, tend to originate closer to the honeypot. This could be linked to the "malicious actor" choosing a closer location to the target from where to initiate the attack.

3.4 Commands

In this section, we look closely at the commands and associated file hashes resulting from a client's interaction with any of our honeypots. This involves roughly 19% of all sessions.

3.4.1 Common Commands

To understand common commands executed by honeypot clients, we take the recorded command strings, split them at command separators (";" and "|"), and look at the most popular ones, see Table 3.3. Among them, we find many typical Unix commands that give information about the system, such as `free`, `uname`, `w` for whois, etc., or access to files with such information, e.g., `cat /proc/cpuinfo`. Another class of popular commands relates to script execution such as `awk` or `shell`, or remote file access such as `wget` or `tftp`. Other commands relate to SSH and its handling of public keys, e.g., `mkdir .ssh`. Indeed, we also find the entry of a trojan horse SSH public key via `echo` into the SSH `authorized_keys` file among the most popular commands. Changing rights, e.g., `chmod 777`, can also be seen among popular commands. Another set of commands relates to changes of credentials, e.g., `chpasswd`. These observations are similar to those seen, e.g., by Koniaris et al. [107] and Dang et al. [110].

Overall, there is a lot of diversity in these commands with extra spaces, quotes, etc., which makes an overall classification challenging. Still, it is clear that typical Unix commands related to information gathering, script execution, file transfer, and credentials are among the most popular ones. In addition, about one-third of the commands involve creating or modifying a file for which the honeypot records a hash of the file content. These hashes can then be used to identify files with the same content across honeypot sessions. In effect, the hashes are signatures of the client honeypot interactions and, as such, can be

Hash	# Sessions ↓	# Unique Client IPs	# Days	Tag	# Honeypots
H1	46,736,536	213,985	970	trojan	221
H2	172,505	7	557	malicious	204
H3	105,102	1,288	20	mirai	203
H4	87,610	3	33	mirai	74
H5	87,514	89	452	mirai	96
H6	73,046	357	125	malicious	228
H7	55,107	1	3	unknown	152
H8	54,884	3	6	mirai	82
H9	54,464	488	6	trojan	209
H10	54,331	18	33	mirai	202
H11	52,312	129	6	trojan	214
H12	50,895	4	17	malicious	179
H13	47,679	14	56	malicious	207
H14	47,240	1	31	miner	211
H15	44,173	66	29	mirai	151
H16	42,633	3	8	unknown	163
H17	40,826	2	4	trojan	148
H18	40,031	4	18	unknown	184
H19	39,688	200	12	miner	207
H20	38,866	1,492	587	malicious	213

Table 3.4: Top 20 hashes sorted by the number of sessions.

used to identify specific attack campaigns. Almost all sessions only involve a single file. Only 0.5% of all the sessions involve two, e.g., when they create one and then, later on, change it. A few sessions, namely 282, involve more than 10 file operations, generating more than 10 hashes.

3.4.2 Popular Hashes and Attack Campaigns

During the 32 months of our study, we observe 70,317 unique file hashes created by honeypot clients. To understand which files may correspond to known malware, we cross-check all hashes in the VirusTotal malware database [140]. Of the more than 70k hashes, we find information for only less than one thousand hashes. Of these, most are labeled as malicious or are associated with specific attack threats, e.g., Mirai, or attack families, e.g., Trojan.

Given this low coverage, we use additional databases for manual cross-checks for the most popular hashes, namely ClamAV [141], FileScan.IO [142], InQuest [143], CERT.PL MWDB [144], and YOROI YOMI [145]. Table 3.4 summarizes the results. Among the 20 most popular hashes—by number of sessions—we find 6 hashes related to Mirai, 5 malicious ones, 4 trojan ones, 3 unknown ones, and 2 miners (one for Bitcoin and one for Ethereum). We find that the number of sessions differs substantially. The first trojan is the one that is also among the top commands, recall Table 3.3. In terms of sessions, it dominates all other commands as it is more than 200 times as popular as the next one. It also involves many unique client IPs that contact all possible honeypots in our honeyfarm. Moreover, it is active on all days throughout our observation period.

The next popular hash is a malicious one that has rather different characteristics. It only

Hash	# Sessions	# Unique Client IPs ↓	# Days	Tag	# Honeypots
H1	46,736,536	213,985	970	trojan	221
H21	10,741	7,285	569	mirai	173
H22	16,676	5,897	9	suspicious	205
H23	4,684	2,213	16	unknown	205
H20	38,866	1,492	587	unknown	213
H24	1,803	1,310	63	mirai	126
H3	105,102	1,288	20	mirai	203
H25	1,429	1,081	71	mirai	171
H26	1,211	1,067	30	mirai	113
H27	1,191	704	3	malicious	185
H28	626	523	32	mirai	153
H9	54,464	488	6	trojan	209
H29	761	448	301	mirai	116
H30	2,809	416	8	mirai	193
H31	21,754	387	7	suspicious	197
H32	417	365	26	mirai	145
H33	5,475	365	501	mirai	238
H6	73,046	357	125	malicious	228
H34	13,439	316	11	trojan	45
H35	244	235	65	unknown	87

Table 3.5: Top 20 hashes sorted by number of client IPs.

Hash	# Sessions	# Unique Client IPs	# Days ↓	Tag	# Honeypots
H1	46,736,536	213,985	970	trojan	261
H20	38,866	1,492	587	malicious	213
H21	10,741	7,285	569	mirai	173
H2	172,505	7	557	malicious	204
H33	5,475	365	501	mirai	238
H5	87,514	89	452	mirai	96
H29	761	448	301	mirai	116
H36	3,552	50	226	mirai	218
H37	10,834	4	172	mirai	197
H38	7,532	5	151	mirai	4
H39	8,309	4	145	trojan	193
H40	1,089	96	133	mirai	144
H6	73,046	357	125	malicious	228
H41	8,100	4	121	unknown	174
H42	138	5	113	trojan	72
H43	11,703	13	104	trojan	105
H44	125	96	103	mirai	61
H45	90	6	75	unknown	60
H46	82	6	75	trojan	56
H47	465	146	71	trojan	133

Table 3.6: Top 20 hashes sorted by the number of active days.

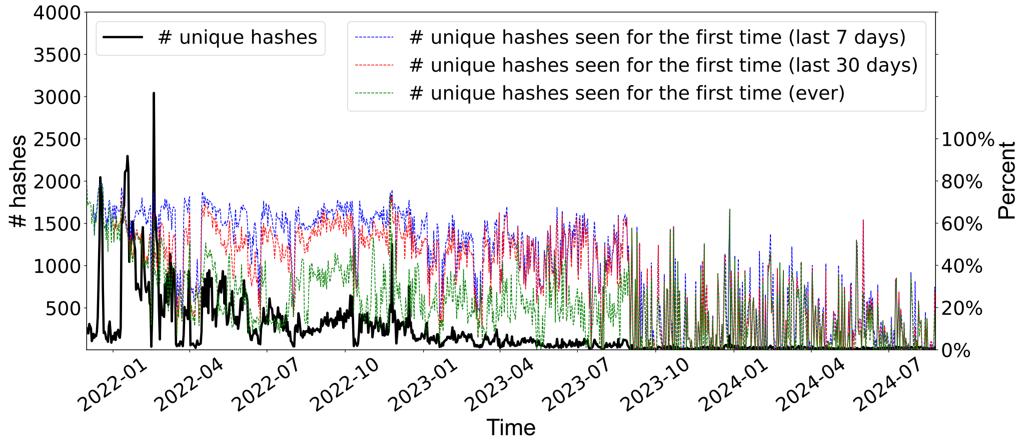


Figure 3.19: Unique Hash activity: Across time (black line, left axis) and fraction of fresh hashes (right axis). We distinguish between overall fresh hashes—not seen before, and hashes that are fresh within the last 30 or 7 days.

involves 7 client IPs, is active for roughly half of the time period (with some breaks in between), but still contacts almost all honeypots. Others, e.g., H4 and H7, also involve fewer than 5 clients but last much shorter or target only a subset of the honeypots.

Interestingly, there are multiple hashes associated with the Mirai attack, but they have all different characteristics, e.g., H4 and H10 lasts for 33 days, while H5 lasts for more than a year. H3 involve more than 1000 client IPs, yet H4 and H8 only 3 client IPs. One of the miners is only observed from a single client but with a large number of sessions (for that one client). It is active for a month and contacts almost all honeypots. The second one is active for only 12 days but involves 200 clients. Overall, 8 of the top 20 hash involve less than 5 client IPs as seen by our honeyfarm, while 11 of them are seen by more than 200 (90%) of the honeypots.

To highlight the diversity, we show the top 20 hashes sorted by the number of unique client IPs in Table 3.5 and sorted by the number of days in Table 3.6. Interestingly, Mirai-related hashes are dominant among the long-lasting attacks, with 9 different hashes labeled as Mirai.

Among the takeaways of this analysis are that attack vectors vary largely by the number of sessions, number of unique client IPs, active days, and number of contacted honeypots. As such, a honeyfarm needs to have diversity in terms of geographic coverage, network coverage, etc. This diversity implies that some of the attacks may be easy to block, e.g., if they only involve a small number of client IPs. Others are more difficult to stop, e.g., if they involve a large botnet of clients.

3.4.3 Campaign Timeline and Freshness

Figure 3.19 shows the number of unique hashes that are collected per day at the 221 honeypots of the honeyfarm. The number of hashes varies substantially from a few tens up to three thousand per day. Note, the spikes that can occur at any point and are not restricted to the first few months of the honeyfarm operation.

Given that attack campaigns have different durations, we also analyze if the hashes are

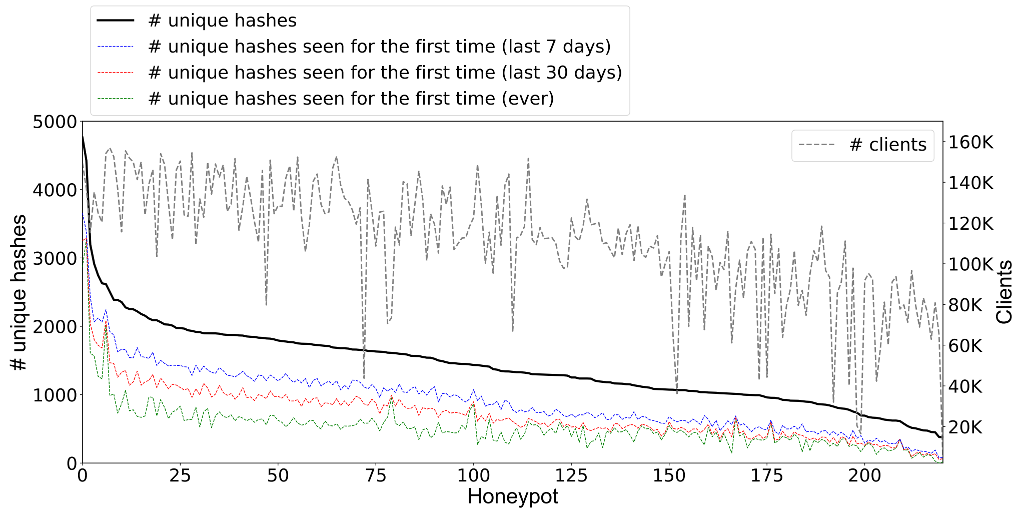


Figure 3.20: Unique (fresh) hashes: Per honeypot (left axis) and # of client IPs (right axis).

fresh, i.e., that they were not observed before. To not bias the data by our observation period, we define two additional freshness metrics using a sliding window approach, namely not observed within the last 7 days and the last 30 days.

We find that a substantial fraction of unique hashes observed in a day are fresh. In Figure 3.19, we show the fresh hashes as a percentage of the unique hashes observed per day with a green dashed line. This percentage varies substantially from 2% up to 60%. There is no apparent correlation between the number of unique hashes and the number of fresh hashes.

When moving to the two additional freshness metrics, we see that the fraction of fresh hashes is increasing up to 60%. With less memory, i.e., as we move from all via 30 days to 7 days, the percentage of fresh hashes increases. Overall, this plot highlights that "new" attacks or variants of attacks that generate fresh hashes are hitting the honeyfarm every day. We also see that some attacks are active for some time, then pause and restart.

Overall, we observe a downward trend in the number of unique hashes captured by the honeyfarm over time. The most pronounced decline occurs toward the end of Summer 2023, after which the system consistently detects an average of approximately 10 unique hashes per day. This trend may suggest one of two possibilities: (1) the honeypots are gradually being identified by malicious actors, leading to their avoidance, or (2) adversaries are adapting their strategies and deploying files in ways that evade detection by the honeypot infrastructure. These observations highlight the critical importance of both the anonymity and the detection capabilities of honeypots in monitoring malicious activity effectively.

3.4.4 Hashes vs. Honeypots

Figure 3.20 shows the number of unique hashes recorded by each honeypot sorted by the number of unique hashes. We again observe a large variability, i.e., the top 10 honeypots observe many more hashes than the rest. More concretely, they see a factor of 20 more hashes than the ones in the tail. However, the top honeypot does not see more than 5%

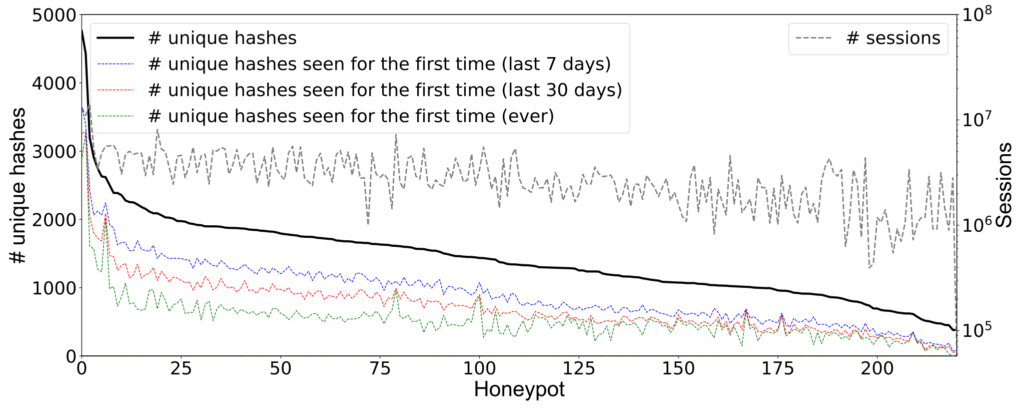
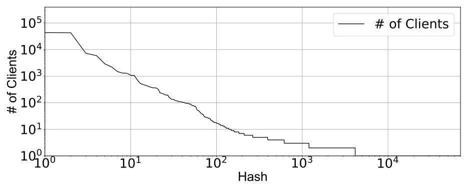
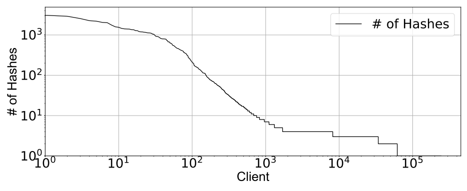


Figure 3.21: Unique (fresh) hashes: Per honeypot (left axis) and # of sessions (right axis).



(a) #clients IPs per hash (log log scale).



(b) #Hashes per client IPs (log log scale).

Figure 3.22: Distribution of unique Hashes over client IPs and unique client IPs over Hashes.

of all the hashes, and the top 10 see less than 15% of all hashes. The grey dotted line shows the number of clients for each honeypot. Figure 3.21 is a similar figure where the grey dotted line shows the number of sessions for each honeypot. Interestingly, however, the honeypots that see the highest number of unique hashes are not necessarily the honeypots that are contacted by the largest number of clients (gray line in Figure 3.20), nor are the ones with the most sessions (gray line in Figure 3.21).

Still, when we look at the freshness of the hashes, the top 10 honeypots remain the same, even though the individual ranks change by a few spots. Thus, they still contribute most of the fresh hashes. The same holds for the time-limited freshness metrics with a memory of 7 resp. 30 days. Indeed, the ranking remains almost stable throughout the whole set of honeypots.

When looking at how many hashes are seen by how many honeypots, we find that more than 60% of all the hashes are observed by a single honeypot only. However, more than 6.8% are seen at more than 10 honeypots, and more than 200 are seen by more than half of the honeypots; among them are many of those presented in Table 3.4. We also check the distribution of the hashes among countries and find a similar long-tail distribution. These numbers indicate that besides those hashes that are observed everywhere, there is a very long tail. Given that security incidents often try to hide in the long tail, it is important to have a honeyfarm infrastructure that has sufficient vantage points and is geographically and topologically diverse.

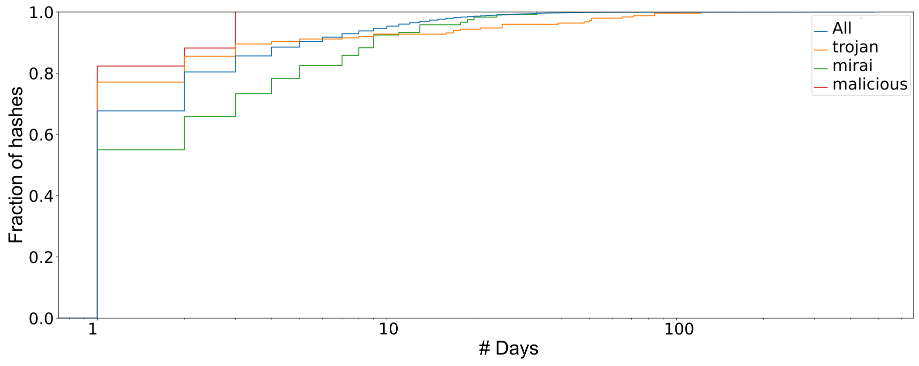


Figure 3.23: ECDF: length of attack campaigns by attack type.

3.4.5 Hashes vs. Clients

On the one hand, we observe hashes that involve many client IPs. On the other hand, some involve only a single client IP. To better understand this relationship, in Figure 3.22(a) we plot for each hash (log x-axis) the number of unique client IPs that involve this hash (log y-axis) sorted by the number of client IPs. We find the typical effects of a long-tailed distribution. Some hashes involve many clients, while others only have a few. The ones with many clients are likely easier to observe but may be harder to block. While the ones in the tail may be harder to observe but easier to block.

We also plot, in Figure 3.22(b), for each client IP (log x-axis) the number of unique hashes that involve this client (log y-axis) sorted by the number of hashes. Again we see the effects of the long-tail distribution. Some hashes/campaigns are driven by a small number of IPs others involve large sets of IPs. Such large sets of IPs may point to a sizable botnet.

3.4.6 Activity of Attack Campaigns

To better understand the activity of each attack campaign, we next investigate how many days we see a specific hash in any of our honeypots. Figure 3.23 shows the corresponding empirical cumulative distribution function (ECDF). Hereby, we look at all hashes and the subclasses we derived using VirusTotal. The latter distinguishes hashes associated with the Mirai attack, hashes tagged as trojan, and hashes tagged as malicious. The main takeaway is that (a) most hashes are only active for a single day, (b) hashes tagged as trojan tend to be active on more days compared to others, and (c) hashes tagged as Mirai are typically active for less than 30 days.

3.5 Discussion

On Multi-role Client IPs. Our analysis shows that 40% of all client IPs are involved in more than one type of connections. Thus, these devices, either compromised or purposefully provisioned as attack systems, serve different roles, ranging from scanning (checking for open ports) to scouting (trying login credentials), as well as intrusion and interaction (logging in and executing commands) with potentially vulnerable devices on the Internet. This shows that client IPs that are performing scanning, over time may perform security-threatening tasks. We were able to characterize such behavior as we utilize a highly distributed honeyfarm and collect connection data over a long period (32 months). Such observations highlight the value of a distributed honeyfarm and the need to collect data at honeypots over a long duration.

The Birth of a Honeyfarm. The analysis of our newly launched honeyfarm deployed on 221 hosts (whose addresses have never been used as honeypots before) in 55 countries and 64 networks shows, that from day one to the end of our study (32 months later) the level of activity remains overall similar. Interestingly, the levels of scanning and scouting are almost similar to the levels of intrusion and interaction with the honeypots at the beginning of our study. It takes more than a month until the level of scouting increases and more than 6 months for scanning. After a couple of months in the honeyfarm's lifetime, scanning and scouting activity combined surpasses the intrusion category. At the end of the 32 months of operation of the studied honeyfarm, we did not observe a noticeable drop in scouting or interaction activity. Nevertheless, we notice a drop in the number of collected hashes. It remains unclear, however, whether this reduction is due to attackers successfully identifying the honeypots or adopting alternative tactics that evade their detection.

Federated Honeyfarms. Our study clearly shows the benefits of operating a honeyfarm over individual honeypots. However, even the top honeypots only observe a small fraction of the unique set of hashes observed in the complete honeyfarm. We believe that the research and security community should collaborate to share the data and intelligence collected by honeyfarms operated and deployed by independent organizations. We expect this will substantially improve the visibility of activities such as scouting, intrusion, and interaction but also has the potential to identify such activity earlier than independent honeyfarms can achieve today.

Honeyfarms and Security Reality. The analysis of the data collected in the honeyfarm also shows that the majority of the honeypots observe high-profile attack campaigns. Many of these campaigns last for weeks, and many of the top 20 campaigns in terms of activity in our honeyfarms even last for months, with some being visible throughout the entire duration of our study, i.e., almost three years. Often, these campaigns are well orchestrated, and we see the same attack coming from numerous IPs.

However, this is not always the case. We also observe large campaigns which are active for more than a year, that originate from only a handful of IPs. It is frustrating to see that no action is taken to block IPs participating in such campaigns. Unfortunately, our analysis shows strong indications that network and cloud providers are not well informed or do not have the appropriate filters in place to block client IPs that participate in scouting, intrusion, or interaction with potentially vulnerable hosts. It is even more concerning that long-lasting campaigns observed by a substantial fraction of the honeypot population in our honeyfarm utilize only a handful of IPv4 addresses for their campaign, and still no blocking or take-down of these IPs takes place for months. Thus, although honeyfarms are proven to be very effective in detecting attack campaigns, this is only a part of the chain of network defenses that alone can not be a panacea.

3.6 Summary

In this Chapter, we analyze data from a newly deployed and operational honeyfarm consisting of 221 honeypots deployed in 55 countries and 64 networks for 32 months and shed light on the unwanted and unsolicited activity in the Internet. We performed our analysis per honeypot, i.e., *individually*, as well as per honeyfarm, i.e., *collectively*. Our analysis shows striking differences across identical honeypots operated worldwide.

Hereby, we taxonomized the different connections established with honeypots to illuminate different scanning, scouting, and intrusion behaviors. Depending on the metric, the top honeypots may receive more than 30× more sessions or sessions from 20× more client IPs than honeypots in the tail. Note, which honeypots are the top ones differs substantially by metric. As such, contrary to our intuition, those with the largest number of hashes do not have the most sessions or client IPs. However, we reported that the set of honeypots that observed the highest number of hashes is likely to observe new hashes earlier. These insights can be used to inform new installations of honeypots within honeyfarms depending on the objectives of the honeyfarm, e.g., early detection of new hashes vs. high visibility of scanning activity. We have also observed that the intruders generate many different files. However, none of the honeypots observed more than 5% of all seen unique files. To capture the tail, which are likely the more interesting intrusions, one has to have scale and diversity in the honeyfarm deployment.

Our analysis shows that many of the attacks, based on the hashes involved, are visible for an extended period so that they could in principle be easily detectable. Nevertheless attacks varied and new ones appeared on a daily basis. Moreover, some attacks always targeted the same subset of honeypots with just a few client IPs, while others used many client IPs likely from major botnets and contacted almost all honeypots. The former are in principle easy to take down, the latter are more difficult but they may be useful to track down botnets.

Chapter 4
SSH Attacker Characterization

In this chapter, we continue our investigation into unwanted SSH traffic by building on the insights introduced previously. The prior chapter provided an overview of unsolicited connection attempts observed through the lens of SSH honeypots, offering a descriptive snapshot of the attack landscape. However, descriptive analysis alone is insufficient to fully understand or characterize the behavior and strategies of malicious actors [5, 6]. The trends identified through data collected by honeypots not only reveal a steady growth in the volume of attacks but also an evolution in their methods. This is particularly evident in the context of global events, such as the Russia-Ukraine war, which has triggered a marked increase in hostile activity across the Internet [146].

Attackers employ a range of techniques, from worms and viruses to sophisticated botnets. While prior research has provided valuable insights into attacker behavior, these studies often focus on specific botnets [28] or rely on data collected during short timeframes from honeypots [29]. Such approaches, though informative, are limited in their ability to capture the dynamic and evolving nature of malicious activity.

The goal of this study is to move beyond static snapshots of attacker behavior and investigate how it changes over extended periods. This is critical because defending against attacks is an ongoing struggle between those seeking to exploit systems and those working to protect them. As attackers refine their tactics and strategies, defenders must stay informed and adapt accordingly to mitigate evolving threats.

Monitoring attacker behavior over longer periods enables us to uncover broader patterns, detect recurring threats, identify waves of activity, and correlate these trends with external events. This knowledge is essential for designing effective and adaptive defense mechanisms. In this chapter, we continue our analysis of the data collected through a collaboration with a large-scale honeyfarm operator and conduct a longitudinal analysis of SSH-based attacks. This approach provides valuable insights into how attacker behavior evolves over time, highlighting the necessity of continuous observation and adaptation in the development of defensive strategies.

Our contributions in this chapter can be summarized as follows:

- We investigate 163 million intrusive sessions that execute commands over 3 years and classify these commands based on their purpose. We identify a clear shift in attacker behavior starting in 2023.

- We analyze commands attempting to execute a file and observe that attackers' preferences for loading files onto compromised targets have evolved over time.
- We examine more than 16 thousand malicious files (hashes) and apply a clustering algorithm based on executed commands to classify various bots, including *Mirai*, *Gafgyt*, *XXorDDoS*, and others, to shed more light on their activity over time.
- We analyze the IPs used as malware storage locations and find that more than 70% are hosted in recently registered Autonomous Systems (ASes) no older than five years. Additionally, we observe that some attackers tend to reuse the same storage IPs over extended periods.
- We discover that attackers are actively scanning for honeypots and, in some cases, abuse them to use as proxies for attack execution.
- Finally, we investigate an aggressive attack campaign visible throughout the entire observation period and provide insights into its behavior and characteristics.

4.1 Datasets

In this section, we present the profile of the honeyfarm we collaborate with as well as other data sources we use.

4.1.1 Honeyfarm Profile

As we already presented in Chapter 3, we collaborate with a large non-profit honeyfarm operator that provides access to researchers and operates 221 identically configured Cowrie honeypots—medium-interaction SSH and Telnet traps—in 55 countries and 64 ASes, focusing on residential networks. The honeypots log session metadata (timestamps, IPs, ports), SSH client versions, login credentials and outcomes, and any shell commands executed. Only password-based SSH login is allowed, using the 'root' username with any password except "root"; public key authentication is unsupported. Over 32 months (December 2021–August 2024), the honeyfarm recorded over 750 million sessions; 546 million of those were SSH-based. We categorize sessions into scanning (no login attempt, 45M), scouting (unsuccessful login, 258M), intrusion (successful login without command execution, 80M), and command execution (successful login with commands, 163M).

4.1.2 Abuse Datasets

The honeypots gather detailed information about attacks, including malicious files. However, only the hashes of these files are collected, and the files themselves are not stored.

To better understand the intent behind the attacks, it is necessary to correlate the file hashes with known attack types. A valuable source of information for such correlations are abuse databases, which compile and categorize malicious file hashes.

In our analysis, we utilize publicly available abuse databases to cross-reference and validate our findings. Among these, we rely on several prominent services that aggregate and maintain extensive records of malicious activities in the wild:

abuse.ch [147] is an open access platform that tracks and detects threats with a strong emphasis on malware software and botnets. Internet Service Providers (ISPs), network and cloud operators, security and hardware vendors, government agencies, and law enforcement agencies rely and integrate Abuse's feed in commercial or open source products to detect and fight attacks in their organizations. The platform also enables the sharing of threat intelligence data with the research and operational security community. We utilize the threat intelligence feed related to IP reputation and malware software samples.

Team Cymru [148] is a threat-intelligence company that provides security solutions, threat analytics, and risk assessment for enterprises and hosts on the Internet. By collecting and analyzing various network data it constructs block lists and reputation scores.

VirusTotal [140] is a threat-intelligence company that specializes on the collection and reverse engineering of malware binaries. It is one of the most popular references for malware detection and characterization. It utilizes reports by many other security companies and volunteers to tag suspicious files (searchable via their `sha256` hash or binary). We got access to VirusTotal for all the hashes in our study.

ArmstrongTechs Project [149] ArmstrongTechs is a threat-intelligence firm specializing in various services to enhance organizational security. Additionally, ArmstrongTechs maintains a GitHub repository titled "Indicators-of-compromise-IOCs", which includes information on various threats and indicators of compromise.

In the following, we refer to all of these services together as abuse datasets.

4.1.3 Autonomous System Data

To better understand which infrastructure the attackers are using we categorize the investigated IPs according to the AS that announces it. For this purpose we use a service for looking up historic announcement information for IPv4 [150]. This service returns for each IP and timestamp a historic perspective which include the announcement time period, AS number, as well as AS organization details. To further categorize the returned ASes we use `bgp.tools` [151] as well as PeeringDB [152].

Both bgp.tools and PeeringDB employ a variety of tags and labels to describe and classify ASes comprehensively. Collectively, these platforms categorize ASes into 19 distinct tags. For our analysis, we focus specifically on differentiating between the following types:

CDNs - Content Delivery Networks.

Hosting - Hosting providers (including web-hosting, VPN).

ISPs/NSPs - Internet Service providers.

Others - Other types of networks (including governmental, academic, corporation, personal networks, or unlabeled).

We didn't consider labels such as "IPv6-only" or "Tranco 10k", as these are infrequent in our dataset and do not significantly contribute to understanding overarching patterns in attacker behavior.

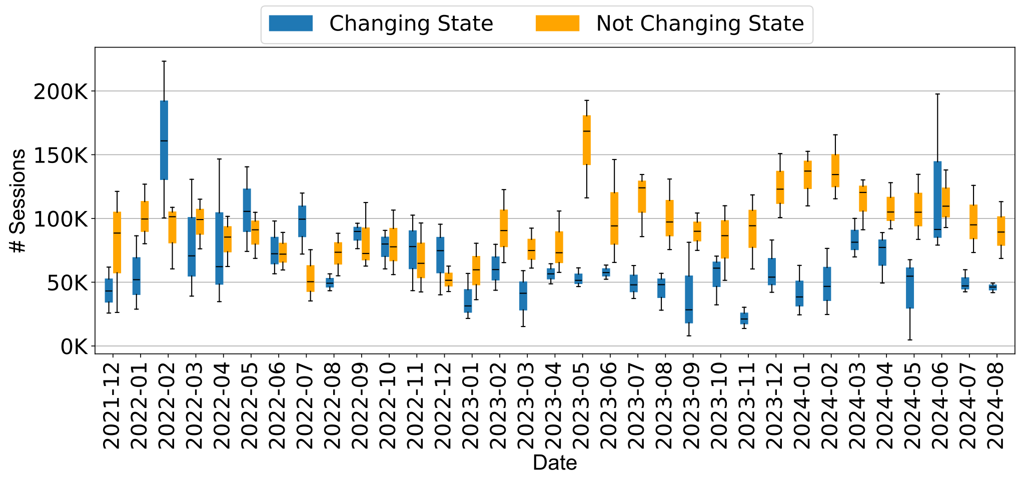


Figure 4.1: Sessions that contain commands split in two categories. In blue, we have the sessions that change the state of the honeypot and in orange otherwise.

4.2 Commands

To initiate our analysis, we examine the types of commands observed in the sessions collected by the honeypots. Our focus is on sessions in which attackers successfully logged in and executed at least one command, totaling approximately 163 million sessions. We categorize these sessions into two distinct types based on the nature of the commands executed. The first category includes sessions where commands do not alter the state of the honeypot. These commands are generally non-intrusive, primarily aimed at gathering information about the system, such as identifying existing files, inspecting running processes, or assessing the machine's capabilities. We observe that nearly 94 million sessions fall within this type. The second category consists of sessions in which at least one command modifies the honeypot's state. Such commands may edit or delete files or execute actions that actively alter the system's original state. The remaining of 69 million sessions belong to this category.

Figure 4.1 presents the distribution of both types of sessions over time. Each boxplot in the figure represents the daily distribution of sessions of each type for the corresponding month. During the initial phase of our data collection, spanning from 2021 to early 2023, both session types occur at similar rates, with a notable spike observed in early 2022. This spike correlates with well-documented attacks linked to the onset of the Russia-Ukraine conflict [153] and – as we show in Figure 4.4(a) – is dominated by one botnet.

In early 2023, however, a shift becomes apparent, marked by an increase in sessions involving commands that do not alter the honeypot's state. This trend suggests a growing preference among attackers to explore the system rather than immediately execute malicious commands after gaining access. One possible explanation for this shift could be an evolution in attack strategies: attackers might prioritize reconnaissance to better understand a target system before executing commands, potentially as a tactic to evade detection by antivirus software, honeypots, or other security mechanisms.

Classification: To better understand the commands, we conducted an in-depth analysis and manually classified them into 59 distinct categories using a series of regular expression (regex) rules [154]. These categories represent generic intrusive bot behaviors that

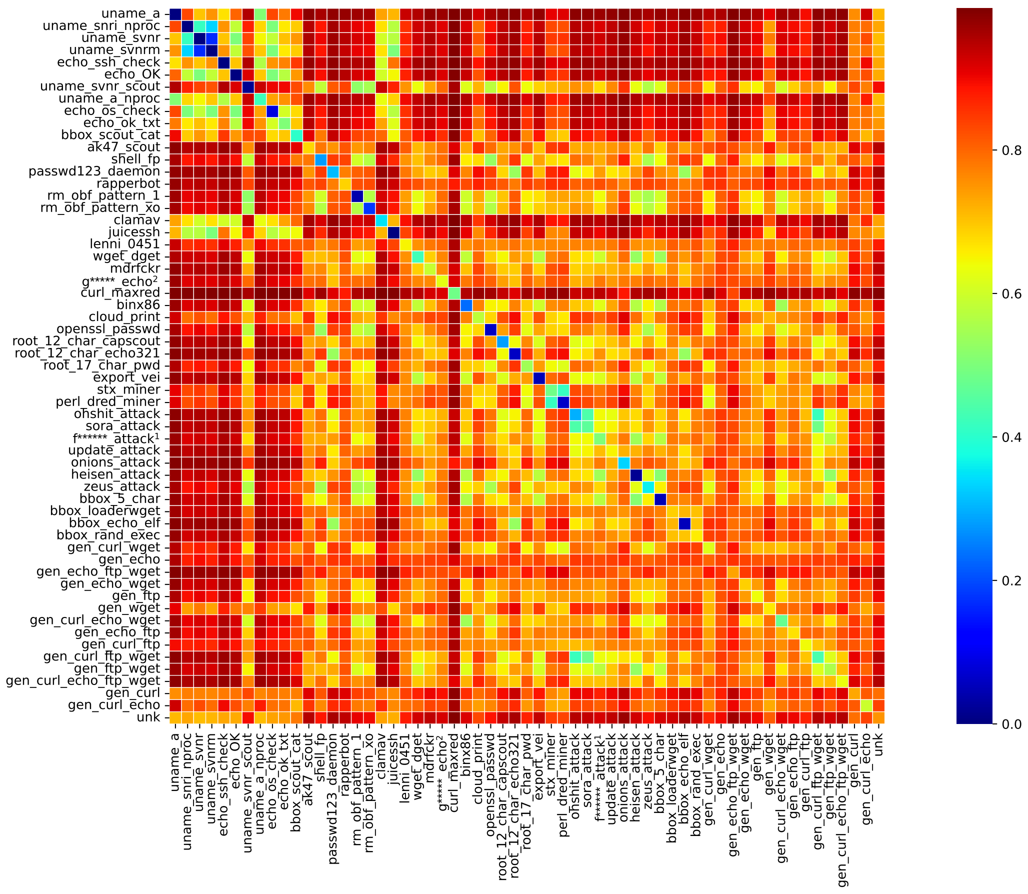


Figure 4.2: Inter-bot category normalized DLD.

exhibit similar characteristics. Specifically, 58 categories were generated based on regex matching, while the 59th category includes all commands that could not be classified. Additionally, 44 categories define in a bot or a specific attack wave, while 14 describe generic intrusive behavior of the bot–specifically the methods of introducing malicious files (*wget*, *ftp*, *curl* or simply *echo*).

Out of the 162 million sessions analyzed, over 161 million were successfully categorized into the 58 regex-based groups. The remaining 1 million sessions fall under the `unknown` category.

In Figure 4.2, we present the average DLD between different command categories. Additionally, Table A.1 (see Appendix A) shows all the category labels and the regex commands used to select the commands. Note the clear separation between the ten command categories located in the top-left corner and the rest. These commands do not alter the honeypots' state (downloading or writing files), but collect system information. Generally, *clamav* and *juicessh*, could also be considered part of this cluster.

State modification: First, we take a closer look at the sessions that do not alter the state of the honeypot. Figure 4.3 presents a detailed breakdown of the bot categories observed in these sessions. We observe that the majority of the sessions (over 95%) can be attributed to the top three bots, with the leading one, `echo_OK`, alone accounting for more than

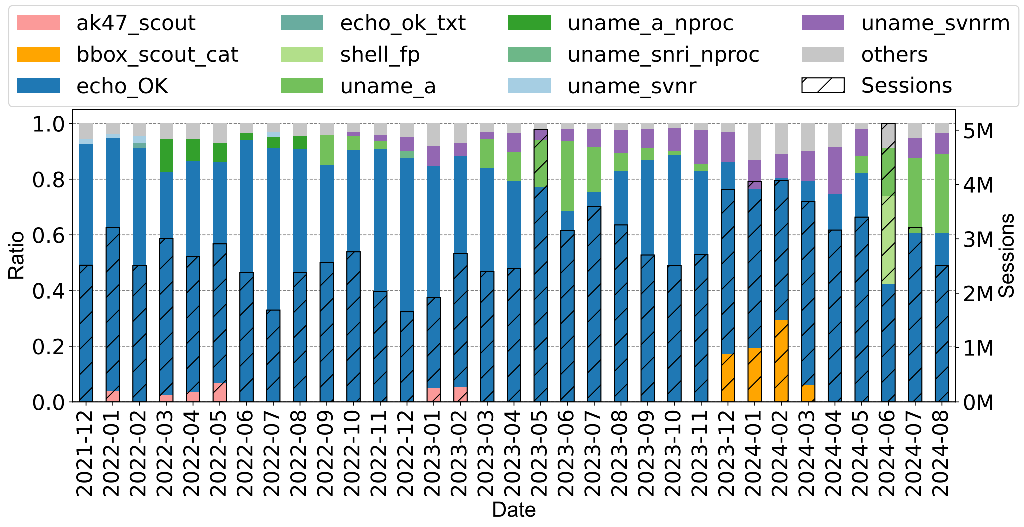


Figure 4.3: Sessions that do not change the state of the honeypot showing the Top 3 bots/month.

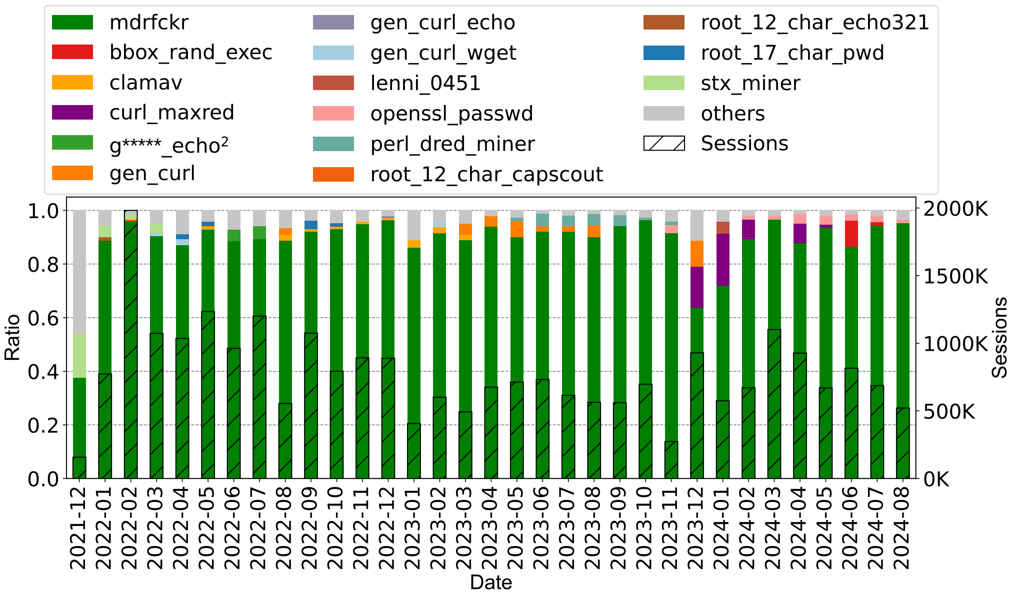
80%. Another notable observation is the presence of both continuous/consistent activity patterns (e.g., `echo_OK`, `uname_svnrm`) and wave-like or campaign-based behavior (e.g., `bb_scout_cat`, `uname_a`).

We next examine sessions that modify the honeypot's state, focusing on whether they involve specific file executions or downloads. Our analysis reveals that over 54 million sessions involve commands that add, modify, or delete files without executing them. The remaining 15 million sessions contain commands attempting to execute a file on the system.

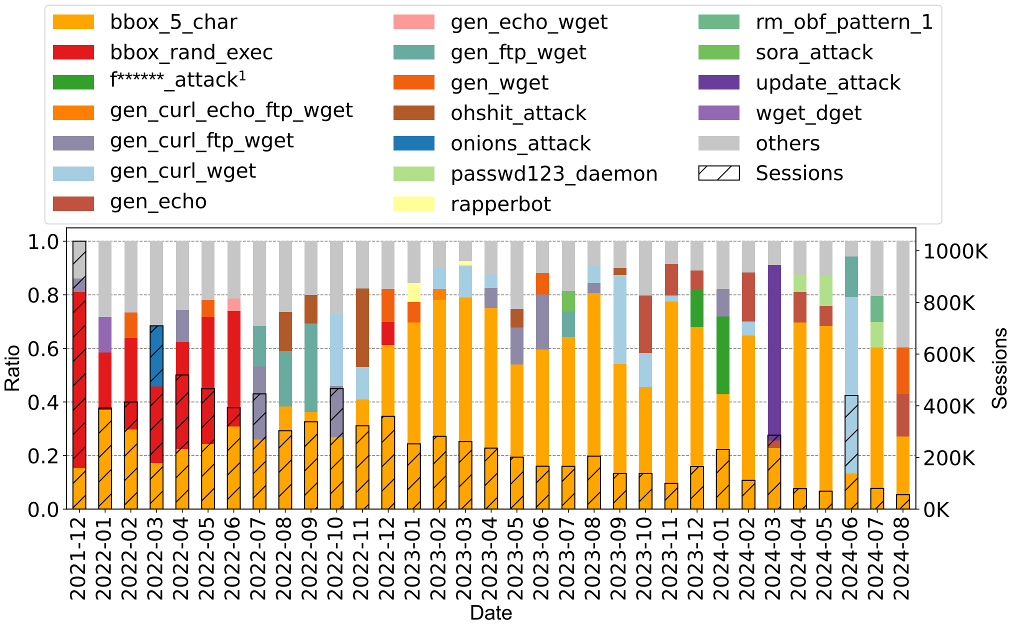
Figure 4.4 presents a temporal overview of the sessions that modify the state of the honeypot. On the left, Figure 4.4(a) provides a detailed analysis of the bot categories that add, modify, or delete files without executing them over time. As with the sessions that do not alter the honeypot's state, the majority of this activity (over 90%) is dominated by a single bot–`mdrfckr`–which will be discussed in detail in Section 4.6.

We observe a high number of sessions each month (>500k) in which attackers attempt to add, write, or delete files without executing them. This tactic may indicate that intruders are more focused on the following: (1) establish persistent access, e.g. by adding a public key, and as such "storing" the compromised device for later use; (2) placing a malicious script on the device for execution at a later time (e.g., coordinating with other devices for a DDoS attack); (3) inserting a script intended for silent execution upon a specific trigger, such as modifying a `.init` file or the crontab; (4) generating a random file and verifying its presence in a subsequent session to test system consistency, as inconsistency may signal to attackers that the device could be a honeypot. To date, no research has been published analyzing SSH attacks using stateful honeypots. However, there are existing proposals for the design of stateful honeypots, as discussed in [155].

Web attacks: At the start of 2024 we observe a wave of a different bot (`curl_maxred`). Closer analysis of the bot's sessions reveals that the honeypots repeatedly execute `curl` commands aimed at specific domains or IP addresses. This activity originates from four client IPs associated with a Russian hosting provider. These clients connect via SSH to 180 of the 221 honeypots in our honeyfarm, generating nearly 200k sessions in total. In each session, the attacker executes approximately 100 `curl` commands to various



(a) Sessions that add, modify, or delete files without executing them.



(b) Sessions that attempt to execute files.

Figure 4.4: Sessions that change the initial state of the honeypot.

```
curl https://<X.X.X.X>/ -s -X GET --max-redirs 5 --compressed
--cookie '<hidden-cookie>' --raw --referer '<hidden-URL>'
curl https://<X.X.X.X>/ -s -X POST --max-redirs 5 --compressed
--cookie '<hidden-cookie>' --raw --referer '<hidden-URL>'
...
```

Figure 4.5: Snippet of a curl with "cookie attack". Referrer, cookies, domain and IP redacted.

destinations. As a result, this group of four IP addresses is responsible for generating 20 million `curl` requests between January and April 2024.

A sample of the commands used in this campaign is shown in Figure 4.5. The `curl` commands are harmless to the honeypot, as they do not introduce malicious software. These commands attempt all HTTP methods with varying cookies and target over 100 domains or IP addresses, with some accessed more than 300k times throughout the campaign.

The targeted domains reveal shared characteristics: most are Russian or Ukrainian sites associated with sectors such as economy, trade, cryptocurrency, e-commerce, Telegram bots, drugs, and gaming. Essentially, the attacker appears to be using the honeypot as a "proxy" with the intent to launch attacks against these sites. From our perspective, the nature of these attacks likely involves either DDoS attempts or credential exploitation through the use of stolen cookies.

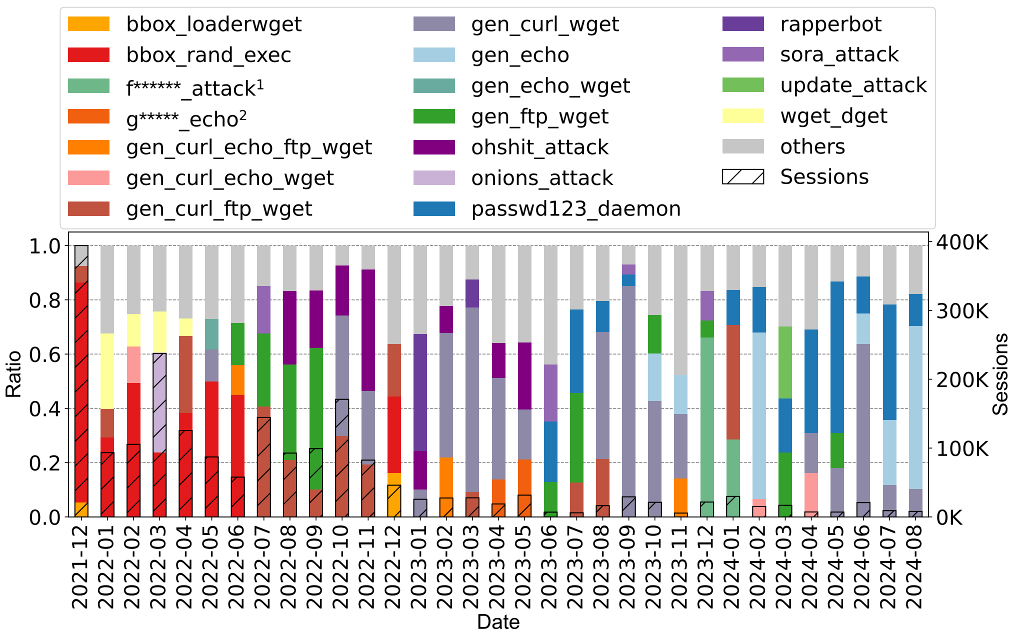
Both medium- and high-interaction SSH honeypots are intentionally designed to properly execute commands like *wget* or *curl* in order to retrieve and store malicious files for later analysis. Although honeypots are inherently non-malicious and serve solely to collect threat intelligence, here, their ability to execute such commands enable attackers to misuse them as intermediaries for malicious activity.

File exec: We continue our analysis by examining the sessions that attempt to execute files (see Figure 4.4(b)). First, we observe a higher diversity of bots, with the top three accounting for only about 50% of all sessions. Notably, the leading bots—`bb_5_diff_char_v2` and `bbox_unlabelled`— leverage the `/bin/busybox` [156] tool to execute potentially malicious scripts. This is not surprising; due to its lightweight implementation, *busybox* has been widely reported by multiple IoT honeypot studies as a preferred method for deploying and executing malware on IoT devices [157, 158, 159]. We also observe that the `bbox_unlabelled` campaign abruptly ends in mid-2022, which may indicate either a takedown or an intentional cessation of activity—especially since no other bot category appears to take its place thereafter.

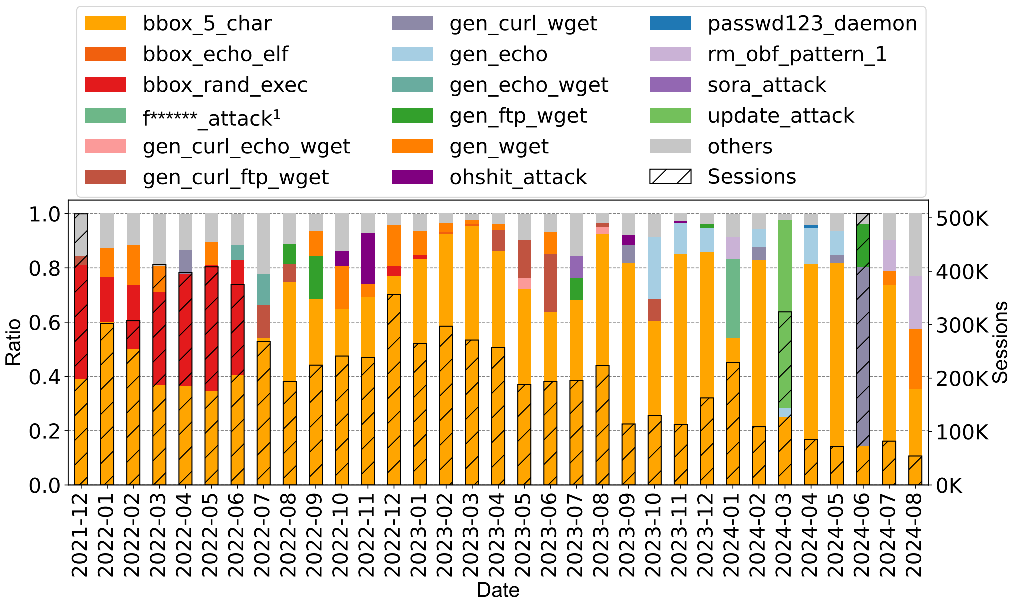
Another important trend is visible in the number of sessions over time (secondary Y-axis), which shows a clear decline. Starting from late 2022, we observe a marked downward trend, with no new bots executing files. Approximately 60% of activity during this period can be attributed to the `bb_5_diff_char_v2` bot. This decline suggests that attackers may be shifting toward more stealthy attacks.

Honeypot files: We continue our investigation by delving into the sessions we consider most intrusive—those containing commands that attempt to execute a file. Figure 4.6 displays the distribution of sessions that run a command which tries to execute files over time. Sessions where we successfully identify the file's hash, are labeled as "file exists",

[1] Slur against Jewish people redacted.
[2] Slur against homosexual men redacted.



(a) Sessions with exec commands, files exist



(b) Sessions with exec commands, files missing

Figure 4.6: Sessions that attempt to execute files.

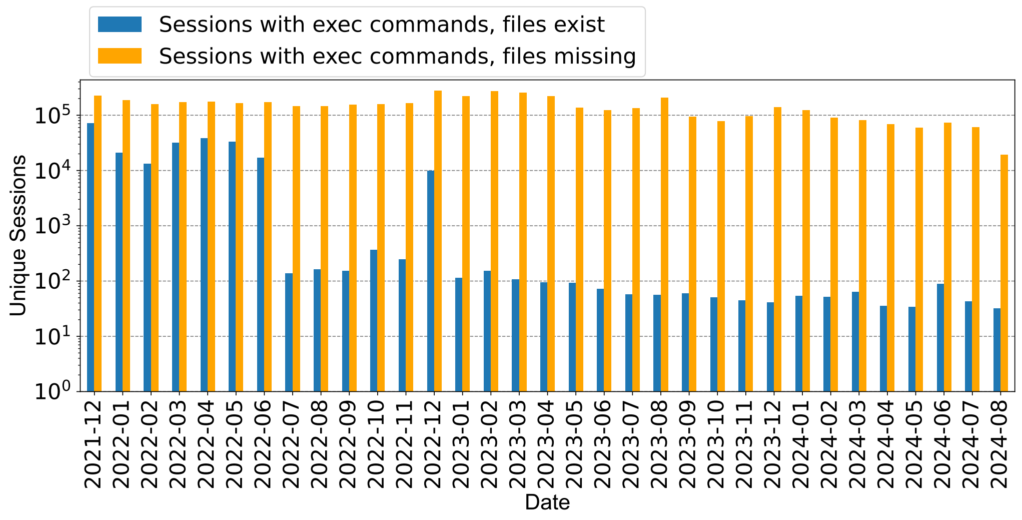


Figure 4.7: Sessions that execute files. Blue color represents sessions where the executed file is found (and a hash of the file is recorded), while orange color shows sessions where the file was missing (and no hash is recorded). Unique number of sessions (based on commands).

while sessions labeled "file missing" indicate no associated hashes. In the later case, the filename being executed was not found in any previous honeypot sessions, nor was it downloaded through commands (on the shell offered by the honeypot) in the current or prior sessions. This implies that attackers may try to copy the file via `scp`, `ftp` or `rsync` on the honeypot from a remote location. As these methods are not emulated by the deployed Cowrie implementation, the honeypots cannot catch the file.

We find more than 3 million sessions where the "file exists" and around 12 million sessions with "file missing". Figure 4.6(a) presents the temporal distribution of the session where the executed file exists, while Figure 4.6(b) represents the temporal distribution of sessions where the executed file is missing.

While there is a noticeable downward trend in sessions attempting to execute files, we observe an even more significant drop in sessions where the file actually exists. In 2022, the honeypots recorded over 100k "file exists" sessions per month. However, starting in 2023, this number dropped sharply to approximately 5k sessions per month.

This shift could potentially be explained by two factors: first, bots may be misconfigured, resulting in incomplete or improperly executed attacks; second, malicious actors may have adapted their tactics. Specifically, the attackers either (a) recognized that honeypots are unable to capture files transmitted directly via protocols such as `scp`, `ftp`, or `rsync`, or (b) opted to simplify their operations by bypassing the need for a malware loader entirely.

Another interesting observation concerns the bot category `bbox_unlabelled` (shown in red). This bot appears to have had multiple variants–some using protocols like *wget* and *tftp*, which allowed the honeypots to capture the dropped files, and others using protocols that prevented the honeypots from retrieving the files. It is plausible that this bot reached its end-of-life phase due to its extensive use of diverse protocols, which may have facilitated a deeper understanding by defenders and, ultimately, the development of effective countermeasures–especially in contrast to a similar bot, `bb_5_diff_char_v2`, which remains active.

In analyzing command uniqueness, see Figure 4.7, we find that sessions labeled as "file missing" exhibit a higher number of unique commands, indicating greater variability compared to sessions where the file is present. This suggests an increased use of obfuscation techniques in these sessions. The spikes in command uniqueness for sessions with files correspond to two distinct bot waves: a prolonged campaign at the start of 2022 and a shorter, intense burst of attacks in December 2022. A thorough analysis showed that both of the waves are most likely related to Mirai, as we are able to find the hashes recorded in those waves labeled as "Mirai" in the abuse datasets.

This longitudinal analysis of command types over time reveals a clear change in the modus operandi (MO) of SSH brute-force attackers. We observe a clear preference for bots exhibiting scouting behavior. By categorizing the executed commands, we can effectively visualize that the number of bots displaying aggressive behavior decreases over time, in contrast to those that refrain from executing files. Additionally, we identify a scenario in which the honeypots are vulnerable to misuse, serving as potential proxies for launching further attacks. These trends underscore the critical need for continual adaptation in anti-malware defenses, especially in honeypot technologies.

4.3 Malware Files

To gain more insight into the attacker behavior, in this section, we focus specifically on sessions in which files are loaded onto the honeypot. In more than 3 million sessions, the intruder explicitly executes commands on the honeypot to download a file. The honeypots do not retain the original files after download; instead, they generate a SHA-256 hash based on each file's content. Over the duration of our study, we identify a total of 16,257 unique hashes where the file is executed. By cross-referencing these file hashes with abuse databases, we are able to classify them as follows:

Malicious: An exec file or a script that reads/writes protected OS files. Typically, a Virus or a Trojan.

Mirai: One of the first significant botnets targeting exposed networking devices running Linux. Nowadays, it targets a wide range of networked embedded devices such as IP cameras, home routers (many vendors involved), and other IoT devices. Since the source code was made public many variants of the Mirai family appeared, infecting many networked devices around the world.

Dofloo: (Aka AESDDoS) is a popular malware used to create large scale botnets that often launch DDoS attacks and run cryptocurrency miners on the infected machines.

Gafgyt: A malware family which infects Linux systems to launch distributed denial-of-service attacks (DDoS).

CoinMiner: An unwanted malicious software which uses the victim's computational power (CPU and RAM mostly) to mine coins (for example Monero or Zcash).

XorDDos: A Linux Trojan malware with rootkit capabilities that is used to launch large-scale DDoS attacks.

Unfortunately, with the data from the abuse databases we are able to identify 681 hashes (≈5%), leaving more than 95% of all collected hashes unlabeled. The distribution of the identified hashes is as follows: Mirai – 85, Dofloo – 1, Gafgyt – 3, CoinMiner – 6, XorDDos – 3, and Malicious – 583. This indicates that we can accurately attribute only around 100 hashes to specific malware families. One reason for the low identification rate is that

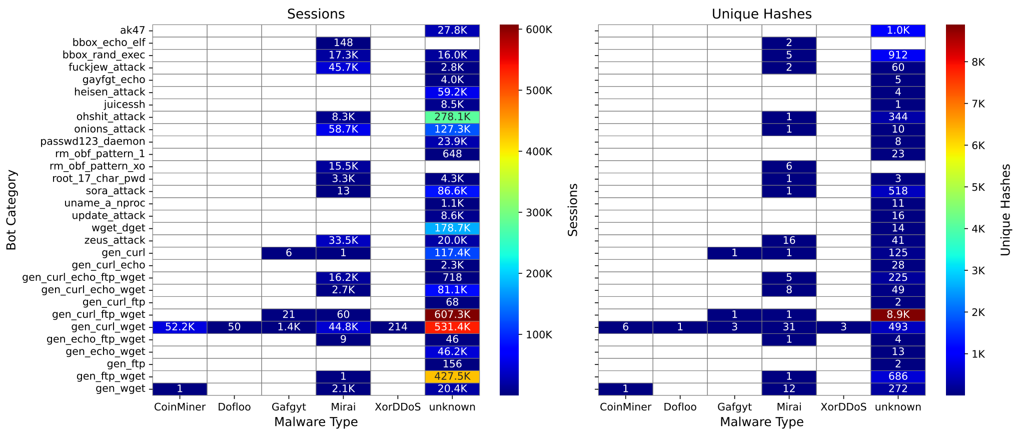


Figure 4.8: Heatmap of the bot categories and file types

abuse databases are unlikely to include all variants of the same malware, particularly since even a single bit flip can alter the hash and render the variant unrecognizable. Another reason is the lack of reports, since not every user reports malicious activity.

Categorization. Our next goal is to overcome the limitation posed by the small number of hashes identified in abuse databases. To achieve this, we apply bot categorization based on executed commands, as described in Section 4.2. The idea is to group together all hashes that belong to the same bot, inferred from their command behavior.

For this analysis, we used only the labels corresponding to specific malware families (e.g., Mirai, Dofloo), while merging the *Malicious* label with *unknown*, as it does not provide enough information to characterize a bot. The resulting heatmap showing the relationship between bot categories and known malware types is presented in Figure 4.8. On the left side, the heatmap represents the number of sessions, while on the right side it shows the number of unique hashes. As expected, the heatmap is heavily skewed toward the *unknown* label, with most sessions (and unique hashes) falling under that category.

We also observe that out of 16,257 unique hashes, 269 appear in multiple bot categories. Among them, 8 belong to Mirai, 2 to Gafgyt, and 1 to CoinMiner. These overlapping hashes are, as expected, found in more generic bot categories such as `gen_curl_wget`, `gen_curl_ftp_wget`, `gen_wget`, and `gen_curl`. For these generic categories, we also observe a mixture of multiple malware types. This is likely due to the fact that sessions in these categories contain only generic download commands, which can be reused by various actors to distribute different types of malware.

Interestingly, we also find that nearly 250K sessions are labeled as *Mirai*, based on the 85 hashes identified in the abuse databases. Here, we distinguish two cases: those where all hashes have been identified (e.g., `bbox_echo_elf` or `rm_obf_pattern_xo`), and those where a mix of *unknown* and *Mirai* hashes is present. In the former case, we can assert with higher confidence that these bots are indeed designed to spread only variants of Mirai. In the latter, however, the classification is less straightforward. For example, the `sora_attack` bot contains only one Mirai hash out of 519, while `bbox_rand_exec` includes 5 Mirai hashes and 912 unknown ones.

To better understand the temporal behavior of these bots, we analyzed their activity over time (Figure 4.9). Some bots, such as `bbox_rand_exec` or `ohshit_attack`, exhibit long-term persistence, whereas others, such as `onions_attack` or `zeus_attack`, appear to represent short-lived campaigns.

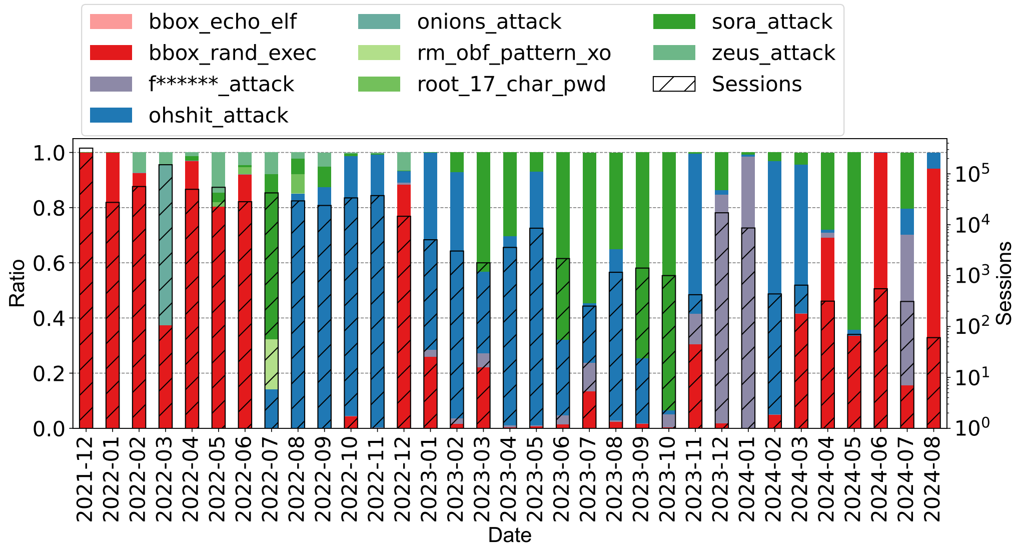


Figure 4.9: Distribution of the Mirai related bots over time

To determine whether a bot consistently distributes a single malware type (e.g., Mirai), we examined the presence of known Mirai hashes throughout its entire activity period (Figure 4.10). Our analysis shows that for bots such as `bbox_rand_exec`, `bbox_echo_elf`, `onions_attack`, and `zeus_attack`, Mirai hashes appear together with unknown ones throughout almost the entire observation period. In contrast, this is not the case for `ohshit_attack` and `sora_attack`. This suggests that the former bots are most likely involved in spreading Mirai malware, while the latter either contain a mix of malware families or successfully evade detection mechanisms that would have recorded their hashes.

Overall, this approach demonstrates strong potential for identifying and characterizing bots, as well as for understanding their evolution over time—especially in scenarios where the actual malware sample is unavailable. In an optimistic perspective, some of the identified bot categories could even serve as effective bot signatures, allowing security systems to block attacks before the malware is downloaded onto the target machine.

4.4 Malware storage location

In this section, we examine the IP addresses involved in download commands that serve as malware storage locations. We start by investigating whether the honeypot client IP—the IP address responsible for brute-forcing and gaining access—is the same as the malware storage location IP. Surprisingly, we find that in 80% of sessions involving downloads, the malware storage location IP differs from the connected client IP. Furthermore, although we see the download commands originating from over 32k unique IPs connecting to the honeypot during our study, only around 3k IPs served as malware storage locations. Given this one-order-of-magnitude difference, we examined whether these malware storage IPs had been previously reported. Abuse databases (recall Section 4.1.2) reveal that 56% of the malware loader IPs have been reported.

Enriching the dataset with additional attributes, such as the origin AS and AS type associated with each IP (refer to Section 4.1.3), allows us to leverage this supplementary

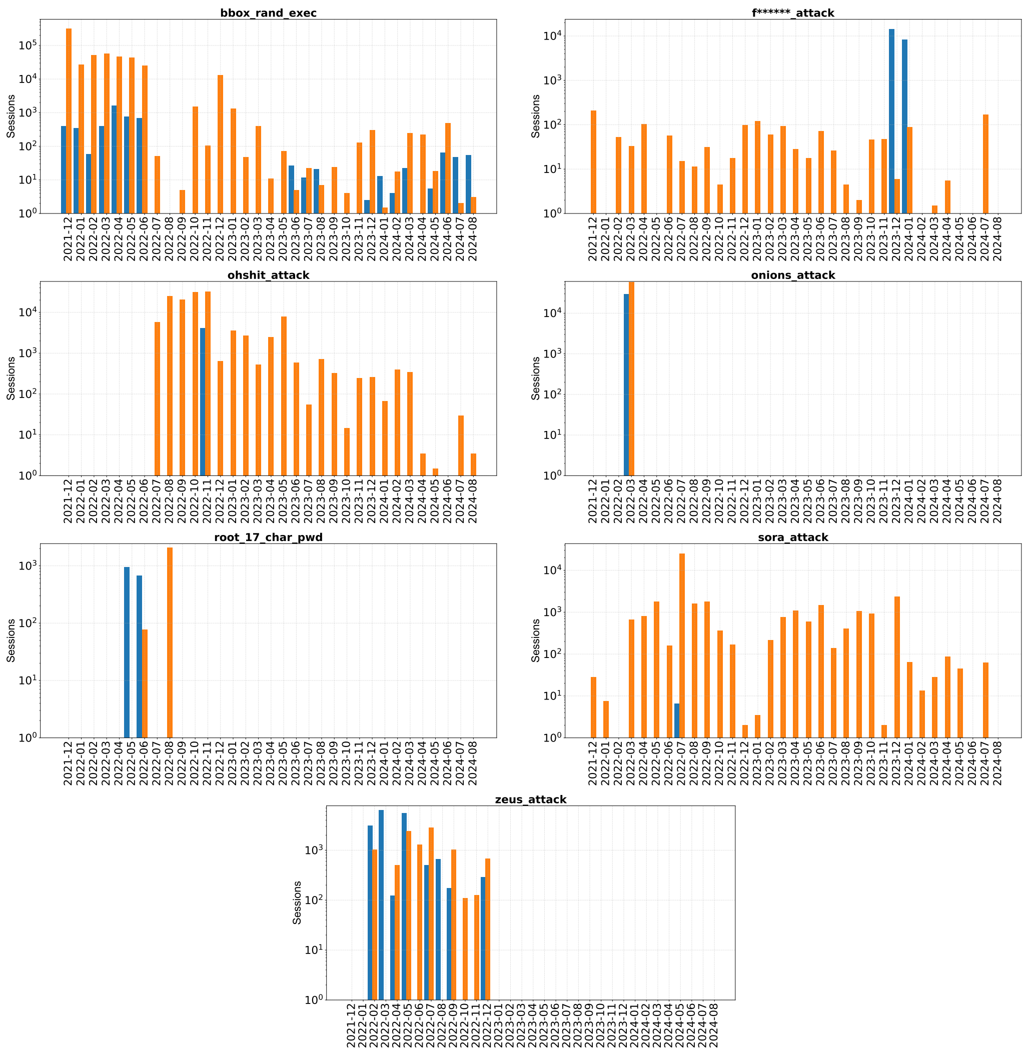


Figure 4.10: Mirai related bot categories over time

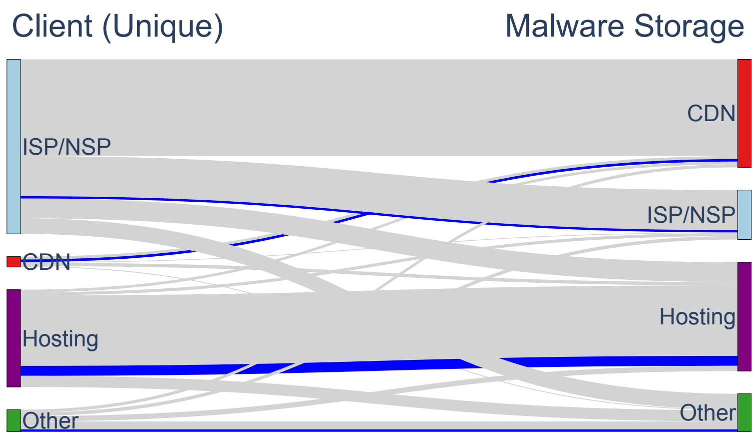


Figure 4.11: Sankey plot: # of client IPs by AS type vs. # malware storage IP by AS type. Flows correspond to client IP, malware storage IP pairs.

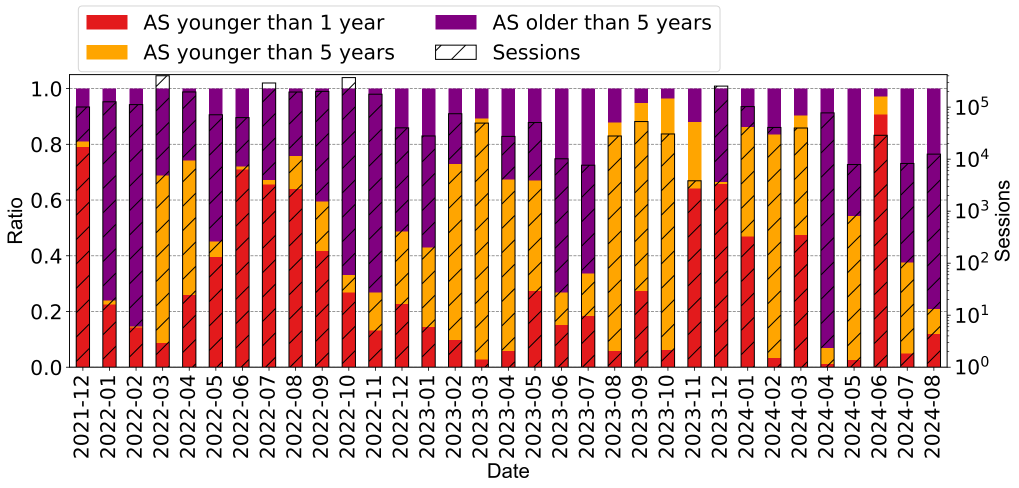
information effectively. We group the client IPs and the malware location IPs by AS type and generated a Sankey diagram of IP pairs, as shown in Figure 4.11. On the left side, we display the client IPs of the sessions, and on the right, the malware location IPs. Flows where the client IP matches the loader IP are shown in blue, while flows where they differ are shown in gray. Most client IPs are in ISP/NSP AS types, which aligns with expectations, as attackers often use end-hosts for attacks. Conversely, the majority of malware storage IPs are hosted in cloud environments, particularly in CDNs or Hosting ASes, with only a small fraction within ISP/NSP networks.

Figure 4.13 presents the distribution of AS types for these malware storage locations over our investigation period. As shown, the majority of malware downloads originate from Hosting ASes, with sporadic appearances of IPS/NSP and CDN ASes, which aligns with expectations. A somewhat surprising finding is the number of "Other" ASes, particularly the significant spike at the end of 2023. We categorized as "Other" any AS that was governmental, academic, personal networks, corporation or unlabeled. Upon manual verification of these "Other" ASes, we found that all provide some form of hosting service, whether web-hosting or generic VM-hosting. This reinforces the observation that attackers predominantly rely on Hosting ASes for malware storage.

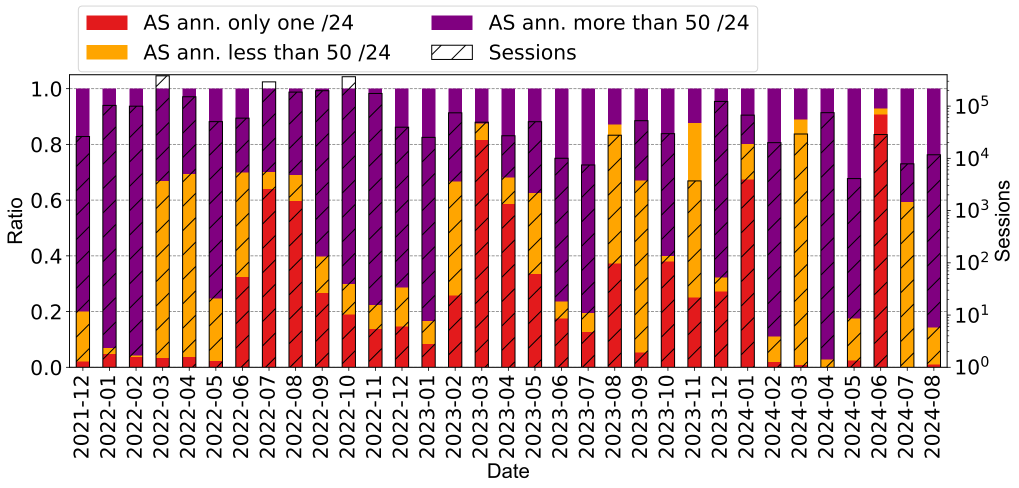
Storage ASes. We conducted a deeper analysis of the ASes used to host malicious files and identified a total of 388 ASes. Among these, 358 are either labeled as hosting ASes or provide hosting services (e.g., renting VMs or website hosting), while 30 are ISPs. Additionally, 36 of these ASes are currently "down", meaning they do not announce any prefixes.

Next, we examined the age of the ASes at the time of the malware download on the honeypot (see Figure 4.12(a)). We found that in more than 35% of cases, the AS was registered in the last year, and in more than 70% of cases, the AS was registered in the last 5 years. This led us to hypothesize that attackers are more successful in abusing recently registered, or smaller ASes or may even deploy their own AS.

We also analyzed the size of the ASes based on the number of /24 prefixes they announce, focusing only on IPv4 since our investigation centers on attacks targeting IPv4 (see Figure 4.12(b)). We found that approximately 20% of ASes announce only a single /24 prefix, and around 50% of ASes announce fewer than 50 /24 prefixes. For this analysis, we deaggregated the announced prefixes to enable a fair comparison of AS



(a) AS age of malware storage locations.



(b) AS size of malware storage locations.

Figure 4.12: AS analysis of malware storage locations.

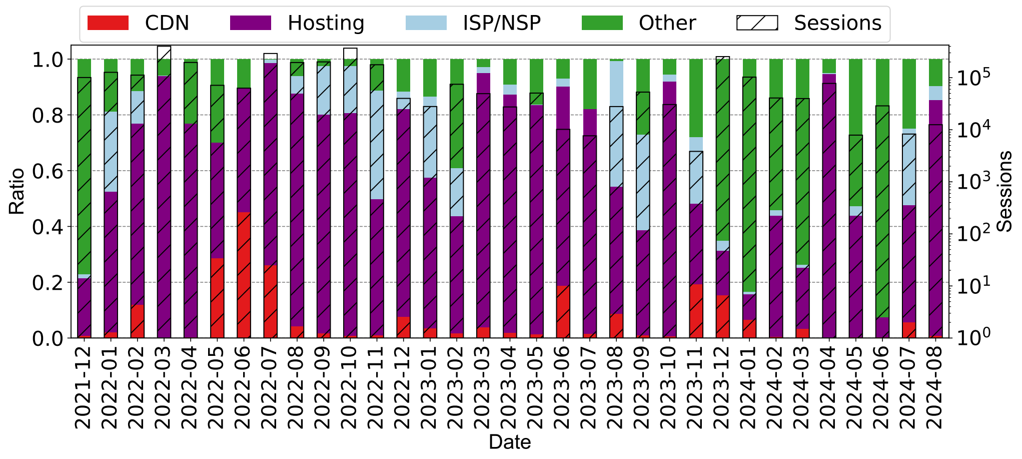


Figure 4.13: AS Types of Malware storage location seen over time.

sizes. These findings further support our hypothesis that attackers may prefer using recently registered and smaller ASes for malicious storage purposes.

To gain a deeper understanding of the malware storage ecosystem, we analyzed the duration for which an IP remains active in the honeyfarm. Specifically, we examined the reappearance of IPs across different time intervals: one week, four weeks, one year, and the entire dataset collection time window (see Figure 4.14).

For the one-week recall interval, we observe that in 50% of cases, the IP storing the malicious file is active for only one day. Another 20% remain visible for up to four days, while approximately 30% are active for the entire week. When analyzing longer recall intervals, we find a significant number of IPs being reused over time. On average, 25% of IPs reappear after at least six months. However, this number can reach nearly 50%, as observed during the spring of 2023.

The repeated use of the same IP for hosting malicious files over extended periods suggests that the machines storing these files were not taken down. Additionally, the reappearance of these IPs after long intervals (exceeding six months) suggests that the attacker may employ a pool of machines, rotating their usage to evade temporary blocklists. Another important takeaway from this figure is that the long duration of our investigation enables us to better observe the extent of IP reusability over time.

4.5 Attacker login attempts

In this section we focus on specific login attempts as well as top login credentials used by malicious actors.

We start our investigation with the most commonly used passwords for accessing the honeypots. Recall that the honeypots are configured to allow password-based SSH authentication using the username “root” and by supplying any password except “root”. Figure 4.15 reveals the most frequent passwords used over the entire observation period. While passwords *“admin”* or *“1234”* are not at all surprising (most like the combination “root:admin” is now on the top of all brute-force dictionary lists), the prominence of other passwords was unexpected.

The plot also highlights an interesting correlation between the passwords “dreambox”

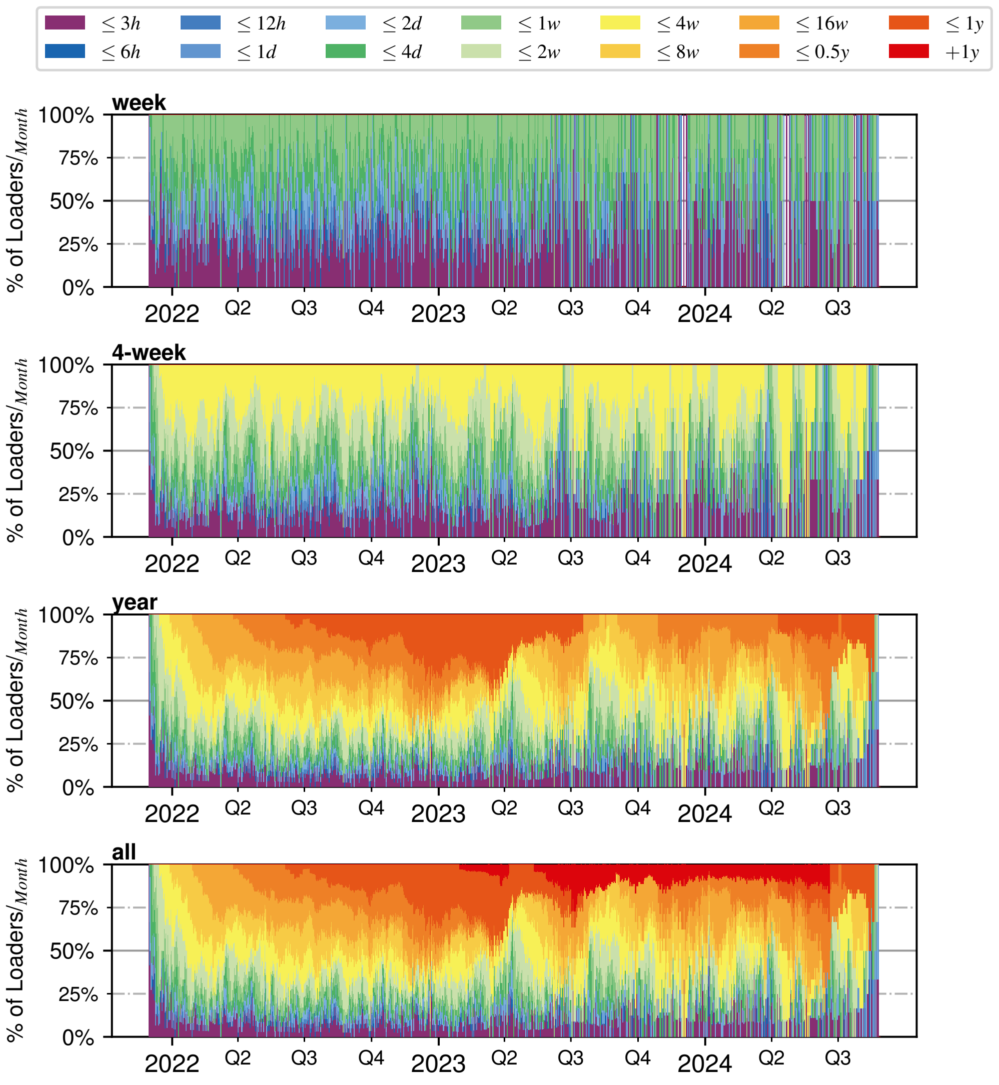


Figure 4.14: Malware storage activity days recorded over time.

and "vertex25ektks123", which appear to be synchronized in usage—a pattern that is not coincidental. Both passwords are known default credentials for popular TV boxes. The password "dreambox" is commonly set as the default for all `Dreambox Enigma(1)` models—specifically models 500, 500+, 5600, 5620, 600, 7000, and 7020 [160, 161]. Similarly, "vertex25ektks123" is the default password for the Dasan H660DW TV Box [162].

Sessions using these passwords could potentially be part of the same botnet structure, as they (1) appear to target similar device types (e.g., TV boxes) and (2) exhibit a consistent modus operandi: the bot first attempts to download a file from a remote server using the `wget` command and then executes it. Abuse database checks reveal a small number of hashes associated with these sessions, all labeled as "Mirai". This finding indicates that this botnet leverages default credentials to exploit specific IoT devices, likely incorporating them into the broader Mirai botnet.

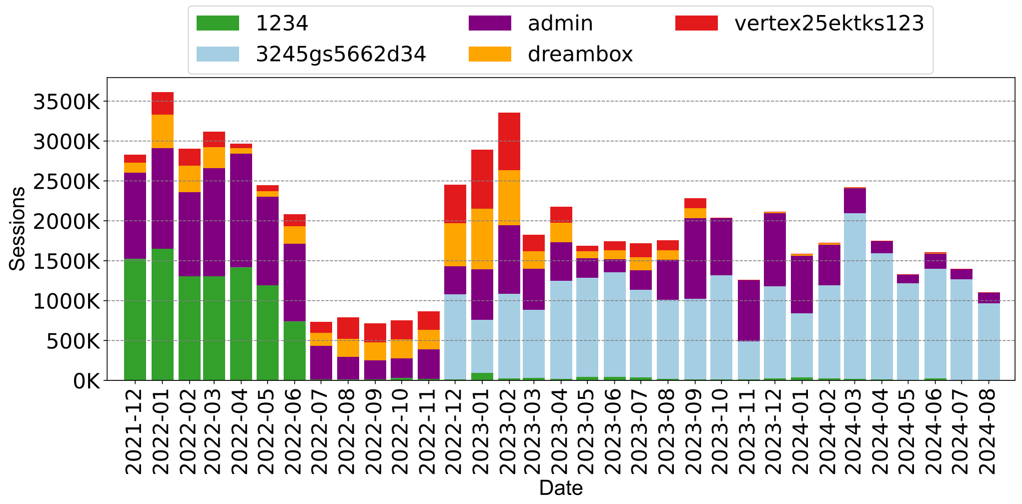


Figure 4.15: Attack characterization: Sessions that record intrusions for username 'root' using one of the Top 5 passwords recorded by the honeyfarm.

The most intriguing case is the top password in our dataset, "3245gs5662d34". We observe no commands executed after login with this password, suggesting that some attackers may focus on gaining access without further interaction—potentially marking targets for future attacks or testing for vulnerabilities.

Our data shows a substantial number of sessions using this password—over 24 million—starting from December 8, 2022, at 18:00 (UTC). Despite the scale of these login attempts, no commands are executed post-login, as the bot simply logs in and performs no additional actions. Analyzing the client IPs reveals that this attack originates from over 125k unique IPs. This volume is notable and is only comparable in scale to one other significant attack in our dataset, which we refer to as "*mdrfckr*" and discuss in Section 4.6.

This password appears in multiple blog posts and forums, yet specific information about the actor or the intended purpose remains unclear. One study [163] suggests that it might be a default credential for the Polycom CX600 IP telephone.

The sudden appearance of this password might indicate a more targeted attack against specific devices or infrastructure. If the Polycom CX600 IP telephone is indeed the target, the attacker might exploit this default credential to gain unauthorized access and potentially use the device to spy or gather information. Moreover, if this is the case, the timing of the attack could be linked to the acquisition of Poly (formerly Polycom) by HP [164]. The Polycom CX600 IP telephone is considered a legacy device, originating from the era when Polycom used Microsoft Lync and has not received updates since 2017 [165].

Cowrie Default Credentials. While investigating intrusion logins, we identified that the Cowrie honeypots allowed logins for a user other than "root", specifically the username "phil". This is not accidental, as the usernames "richard" and "phil" are well-known default usernames in the Cowrie honeypot [166]. In 2020, a Cowrie update changed the default username from "richard" to "phil". Analysis of these usernames revealed consistent login attempts throughout the observation period. Since our honeyfarm runs a later version of Cowrie, we observed successful connections using the "phil" username. Figure 4.16 shows the number of sessions connecting with these default credentials—"phil"—as well as attempts to connect with "richard" over time.

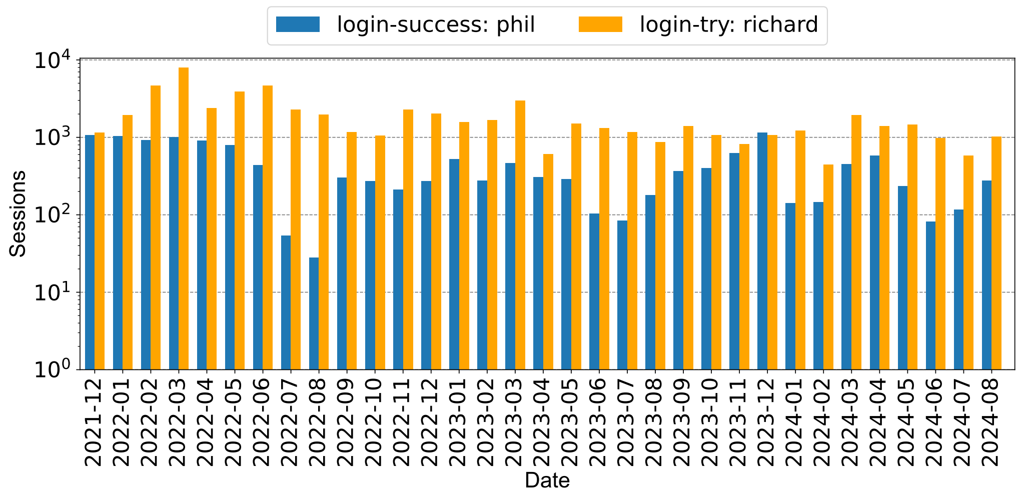


Figure 4.16: Attack characterization: Login attempts using default Cowrie usernames, "richard" and "phil", observed over time.

Although the sessions connecting with the "phil" credential are relatively low in volume (≈30k), they originate from over 10k unique IPs and span more than 1k ASes, suggesting a broader scouting activity. Our analysis indicates that most of these IPs are based in ISP/NSP or Hosting ASes.

Most interestingly, in the majority of cases (over 90% of all sessions), the attackers do not execute any commands after successfully logging in. Moreover, in these cases, the client IP does not reconnect to the honeypot afterwards. This behavior likely indicates that these intrusions are primarily scouting activities, where attackers aim to identify honeypots.

This example indicates that attackers in the field are knowledgeable and adaptive. It is well-documented [94, 95, 96] that attackers employ strategies to evade anti-malware defenses, such as obfuscation, hiding file extensions, encoding in base64, or promptly removing files. As previously discussed in Chapter 2.4, honeypots play a crucial role in enhancing Internet security. We see that attackers are increasingly aware of the deployments of honeypots and the risks associated with their methods being recorded. Malicious actors appear to perform reconnaissance activities, like scanning for honeypot-specific usernames, rather than indiscriminately launching attacks, signaling an evolution in attacker tactics toward more cautious approaches.

4.6 "*mdrfckr*"—case study

The largest attack in our dataset, both by session count and unique client IPs, is from a bot we refer to as "*mdrfckr*". The name is derived from the SSH public key label that the attacker installs to maintain persistence. Over the observation period, we recorded more than 46 million sessions from over 270k unique client IPs associated with this attack. The "*mdrfckr*" bot is believed to be linked to the Outlaw Hacking Group [167, 168], which has been active since 2018.

Analysis of commands collected by the honeypots shows that this attack installs a public key, ensures the victim cannot easily remove it (by locking out the victim through "root" password changes), and conducts reconnaissance on the machine. External sources, such as reports from actual victims [169], indicate that after establishing persistence, the attacker uses `rsync` to load malicious files and modifies the crontab file to execute them. This method allows the attacker to evade honeypot logs by minimizing detectable activity.

This attack, often associated with cryptocurrency mining, has been analyzed multiple times. Researchers have examined both the executed commands [170, 171] and the specific malicious files downloaded by the attacker [172, 168]. Abuse databases label the public key hash used in this attack as either "CoinMiner" or "Malicious", consistent with its association with cryptocurrency mining activities.

Cross-checking the IPs associated with the credential "3245gs5662d34" (see Section 4.5) attack against those used by the "*mdrfckr*" bot during the same period reveals a 99.4% overlap. Additionally, the number of sessions recorded monthly for both attacks is nearly identical, as shown in Figure 4.17. This finding suggests two possible scenarios. First, it could imply that the "3245gs5662d34" credential attack and the "*mdrfckr*" bot are managed by the same actor, indicating that the "*mdrfckr*" actor has expanded its operations in a new direction. Alternatively, both attacks may be run by separate actors who share the same infrastructure, pointing toward a shared "malicious infrastructure as a service" model.

Another notable observation is a behavior change in part of the "*mdrfckr*" sessions, coinciding precisely with the onset of the "3245gs5662d34" attack. This new variant, which we refer to as "mdrfckr-variant", displays several key changes from the original "*mdrfckr*" attack: (1) it no longer changes the root password, (2) removes the files `/tmp/auth.sh` and `/tmp/secure.sh`, (3) kills any processes associated with `auth.sh` and `secure.sh`, and (4) clears the `/etc/hosts.deny` file. As shown in Figure 4.17, this variant—orange colored squares—is at least an order of magnitude smaller in scale than the original ("mdrfckr-initial", blue colored circles).

Further investigation into these modifications reveals that the files `/tmp/auth.sh` and `/tmp/secure.sh` are associated with another "CoinMiner" botnet known as *WorkMiner* [173], which first appeared in 2021 as a successor to the *Mozi* botnet [174]. These scripts are designed to defend infected machines against other brute-force attacks by adding offending IPs to the `/etc/hosts.deny` file. This behavior suggests that the "*mdrfckr*" actor seeks to disable *WorkMiner's* blocking functionality, likely to ensure uninterrupted access to the machine. However, only these two specific scripts are removed, leaving *WorkMiner's* mining operations unaffected.

To further understand the "*mdrfckr*" attacker, we cross-referenced their IPs with labeled malicious IP lists, including the C2-Daily Feed [175] and the Killnet Proxy IP list [176]. When correlating the "*mdrfckr*" IP list with the Killnet list, we discovered 988 overlapping IPs. Killnet [177] is a known pro-Russian group, and IPs in their list are often associated with DDoS attacks, indicating that the "*mdrfckr*" actor may be involved in more than just cryptocurrency mining.

Examining the typical behavior of the "*mdrfckr*" bot over the entire observation period (see Figure 4.18), we notice that it generally generates around 100k sessions per day from approximately 7k unique IPs. However, there are brief periods where this activity drops to around 100 sessions per day from only $\approx$10 unique IPs. The low activity at the end of 2021 likely resulted from the recent deployment of the honeyfarm, which needed time to become a known target.

We confirm that the honeypots were fully operational during these periods. Furthermore,

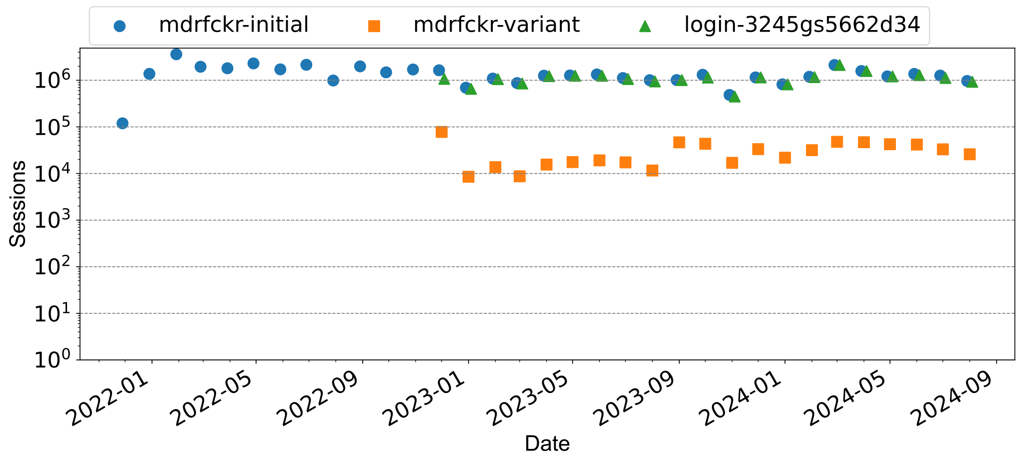


Figure 4.17: Attack case study: Behavior change in the "*mdrfckr*" bots and correlation to "3245gs5662d34" credential attack.

the rapid return to "normal" activity—recovering within hours—suggests a deliberate reduction rather than a takedown, which would likely exhibit a more gradual recovery in activity levels. Our hypothesis is that these drops could correlate with other coordinated attack activities or strategic pauses, suggesting that the "*mdrfckr*" actor may intermittently reallocate resources, possibly toward other attacks. This hypothesis is supported by following findings. Only during these low-activity periods do we observe sessions where the "*mdrfckr*" bot executes base64-encoded scripts after installing the SSH key. We decoded and analyzed these scripts, finding that they are variations of three distinct functionalities:

Cryptominer setup: Similar to the cryptominer found in the repository of a previously compromised victim.

Shellbot installation: This backdoor uses IRC to provide botnet control and is often associated with DDoS attacks.

Cleanup script: A script that thoroughly terminates a set of specific processes, likely to hinder detection.

Note, that since the honeypot cannot capture files transferred via `rsync`—a technique known to be used by this actor—it is likely that we are missing some malicious files that could further clarify this out-of-the-ordinary behavior.

To analyze this further, we examined the IPs responsible for uploading base64-encoded scripts and identified 1,624 unique IPs. These IPs primarily belong to Hosting AS and ISP/NSP providers. Most of these IPs appeared only once, and there was no overlap among IPs across different periods of reduced activity, indicating a dispersed and dynamic infrastructure, even during out-of-the-ordinary attacks.

Focusing on base64 scripts, the "cleaning" script appears to specifically target and remove processes and scripts related to a set of 8 IPs. These IPs exhibit a variety of open ports:

- Four IPs have port 22 open (SSH).
- One IP has ports 1337 and 9999 open and is running ZNC [178], an IRC bouncer often used in botnet command-and-control (C&C) infrastructure.
- One IP has ports 80 and 3306 (MariaDB) open.

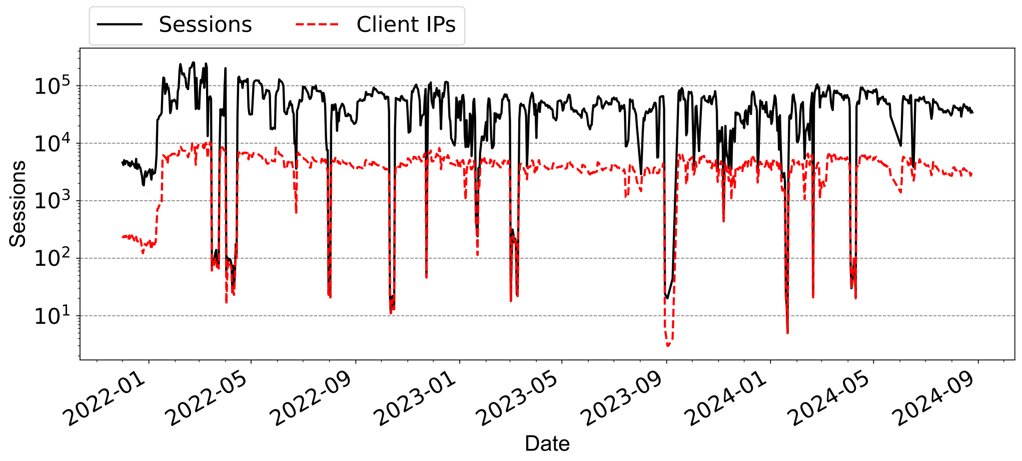


Figure 4.18: Attack case study: A temporal view of all sessions involving "*mdrfckr*" actor recorded by the honeyfarm.

- One IP has port 8080 open.
- One IP has port 43, 80 and 443 open.

Given the consistent inclusion of these IPs in the script, their strategic network positioning, and the nature of their open ports, it is plausible that these IPs function as part of the botnet's command-and-control (C&C) infrastructure. The longstanding presence of these IPs in the script, unchanged over time, highlights their critical role in maintaining control and coordination across the botnet.

Using the SSH scanning technique described in Chapter 5, our analysis (see also Chapter 6) revealed that the "*mdrfckr*" key has compromised over 13k servers globally, making it the most prevalent malicious SSH key identified in their dataset.

4.7 Discussion

Call for Better Honeypots. The current state of the art in honeypot technology appears to be insufficient to address evolving attacker strategies. For example, Cowrie honeypots are increasingly being targeted by attackers who exploit default credentials to identify them. Moreover, certain attackers bypass the honeypot's defenses entirely by abusing it as a proxy for their attacks. Others employ techniques such as `rsync` to evade detection, transferring files without being captured during the attack process. These trends highlight the urgent need for updated and more resilient honeypots if we are to keep pace with the constantly changing tactics of malicious actors.

Understanding malicious actors. Our analysis of the "mdrfckr" botnet exemplifies how attackers obfuscate their true objectives by engaging in multiple malicious activities simultaneously. This adaptability and evolution in tactics, likely in response to changing goals or environmental conditions, underscores the complexity of profiling malicious actors. By diversifying their actions, attackers can obscure their overall intentions, making it difficult to fully understand the scope and motivations of their operations through a single dataset.

Longitudinal analysis. This study underscores the growing importance of longitudinal analysis in understanding SSH attacks. While short-term snapshots can highlight immediate trends or specific incidents, they often fail to capture the gradual evolution of attacker strategies and behaviors. Long-term analysis reveals patterns such as the reuse of infrastructure, shifts in tactics, and the emergence of more exploratory methods, which are critical for comprehensively understanding the threat landscape. Without extended observation, key insights—such as the reappearance of malicious resources after dormancy or the adaptation of bots to evade detection—can be easily overlooked. As attackers become increasingly adaptive, longitudinal analysis is essential for developing effective, forward-looking defense mechanisms that address these evolving threats.

Events correlation. During our investigation of the *mdrfckr*, we sought to identify any significant global events that coincided with the observed drops in bot activity. Our analysis reveals a strong correlation between these periods of reduced activity and several documented attacks during the same timeframes.

2022.03.16—2022.03.24 and 2022.04.02—2022.04.12 A series of (pro-Russian) attacks targeting Ukrainian infrastructure. An actor called "IRIDIUM" suspected in attacks, performed a massive DDoS attack against targets [153].

2022.08.01—2022.08.02: Series of attacks on Infrastructure of a European country supporting Ukraine [179, 180].

2022.10.10—2022.10.16: "Sandworm" (a pro-Russian hacker group) attack against power grid of Ukraine [181, 182, 183], as well as Killnet DDoS attacks against US Airports [184].

2023.03.02—2023.03.10: Attack against "KyivStar"—largest Ukraine mobile operator [185].

2023.09.01—2023.09.08: DDoS attacks against Ukraine public administration and media [186].

2024.01.19—2024.01.21: APT29 (aka Midnight Blizzard)—data theft attack [187].

2024.04.04—2024.04.10: Another "Sandworm" attack against Ukraine's infrastructure [188, 189].

While the correlation between these events and bot activity does not imply causation, we find the pattern both interesting and concerning. In any case, these findings suggest that continuous monitoring of the bot's activity is essential.

4.8 Summary

In this chapter, we conduct a comprehensive longitudinal analysis of SSH-based attacks over a three-year period, shedding light on the evolving strategies and behaviors of attackers. By examining intrusive sessions, executed commands, malicious files, and the infrastructure supporting these attacks, we identified significant shifts in attacker tactics, including increased exploratory behavior and changes in how files are loaded onto compromised targets. Our findings also revealed a trend toward the use of recently registered ASes as malicious storage IPs, as well as evidence of attackers actively scanning for honeypots and, in some cases, exploiting them for their own purposes.

These insights highlight the importance of adapting defense mechanisms to address the evolving tactics of attackers. The study emphasizes the need for continuous observation and deeper analyses of malicious activity to uncover trends and changes in behavior. While our findings demonstrate the potential of integrating data from diverse sources to gain broader insights, further collaboration and research are necessary to enhance our understanding of attacker strategies and improve the effectiveness of defensive measures in an ever-changing threat landscape.

Chapter 5
SSH Active Scans

In this chapter, we shift from a passive observation of SSH attacks to an active and exploratory stance. Rather than relying solely on honeypots and waiting for attackers to engage with our systems, we take the initiative to seek them out in the wild. This proactive approach is inherently more complex and demands a well-thought-out methodology. The main goal is to identify hosts that are somehow connected to a specific attack – particularly those that possess unique information which may indicate compromise or even suggest they are the source of the malicious activity.

During our analysis of various attack vectors and the techniques used by bots to maintain persistence on compromised machines (see Chapter 3 and 4), we identified a recurring pattern: attackers often install their own public SSH keys on the systems they infiltrate. After an initial compromise, attackers typically avoid changing the system's password, as doing so could raise suspicions among legitimate users [30]. This practice has been documented both in large-scale incidents [11] and through honeypot-based studies [31]. These keys serve as a stealthy mechanism for re-entry, bypassing traditional password-based access and reducing the likelihood of detection.

Interestingly, a single public key may appear on multiple hosts, potentially linking victims and attackers. While we cannot determine the exact role of a host – whether it was the origin or target of the attack – without direct access, the presence of an attacker's public key provides a valuable lead for tracing compromise. The key insight here is that the SSH protocol behaves differently depending on whether a public key is actually authorized. Specifically, when a user presents a public key during authentication, a challenge is sent *only if* the key is already installed for the corresponding account [33]. This behavior is not considered a vulnerability but rather a conscious design decision, as noted in the protocol specification [34] and by the OpenSSH developers [35].

This design choice opens the door to a powerful technique: remote identification of compromised systems. If one manages to obtain an attacker's public SSH key – for example, from a honeypot or malware sample – it becomes possible to remotely check other systems for the presence of that key. This allows for mapping out a broader attack footprint without needing privileged access to the scanned machines. By turning this subtle protocol behavior to our advantage, we demonstrate a method for tracing the distribution of attacker-controlled keys across the Internet, potentially revealing hidden links between victims and their attackers.

The contributions of this chapter are summarized as follows:

- The design of a large-scale Internet measurement methodology that enables identifying whether a specific SSH public key is installed on an SSH server, without requiring authentication or privileged access.
- An in-lab feasibility study evaluating the methodology's effectiveness by testing it against the most widely deployed SSH server implementations.
- A controlled environment evaluation of the tool, including the development and validation of a false-positive detection mechanism.
- A discussion of ethical considerations, followed by the deployment of the methodology in a real-world, full-scale Internet measurement experiment.

5.1 Background

In addition to, e.g., password based authentication, SSH supports a public key based authentication method, see RFC4252, Section 7 [34]. For public key based authentication, a public key is installed for a user account. The server can then use this public key to encrypt a challenge which only a party with access to the private key can decrypt. That party can then proof possession of the private key by returning the plain-text challenge to the server.

We depict the technical process of a full SSHv2 authentication process step-by-step in Figure 5.1. After the TCP handshake the SSH Client and SSH Server exchange version information. Subsequently, the client initiates a Key Exchange.

Once all of these steps are completed, the client starts user authentication. First a `SSH2_MSG_SERVICE_REQ` is sent, and if a `SSH2_MSG_SERVICE_ACCEPT` is received the authentication continues. If the server accepts user authentication, the clients starts by sending a `SSH2_MSG_USERAUTH_REQ` message containing the 'username' and fingerprint of the 'pubkey' it wants to use for authentication.

The server will verify if the 'pubkey' is present in the 'authorized_keys' file (or related resource for public keys) of the user. Only if the key is present, the server returns a 'challenge'—a message encrypted with the 'pubkey'. This allows the server to skip the computationally costly public-private key authentication if the public key is not present. The user authenticate by decrypting the 'challenge' using the correct 'privatekey'. Authentication concludes when the plain-text challenge is returned to and verified by the server.

Naturally, this setup enables identifying whether a specific public key is installed for a user, without requiring knowledge of the private key. The general mechanic was first described by Siebenmann in 2016 [190]. In 2019, Golubin noted that this mechanic can be utilized to identify infrastructure used by specific GitHub users, given that GitHub makes users' SSH keys publicly available [33]. In 2021, Kaiser submitted a pull-request to the OpenSSH project [35] to change this behavior, referencing CVE-2016-20012, which in the meantime had been assigned to this behavior. Within the discussion around the pull request, however, OpenSSH developers clarified that this is, indeed, intended behavior [35].

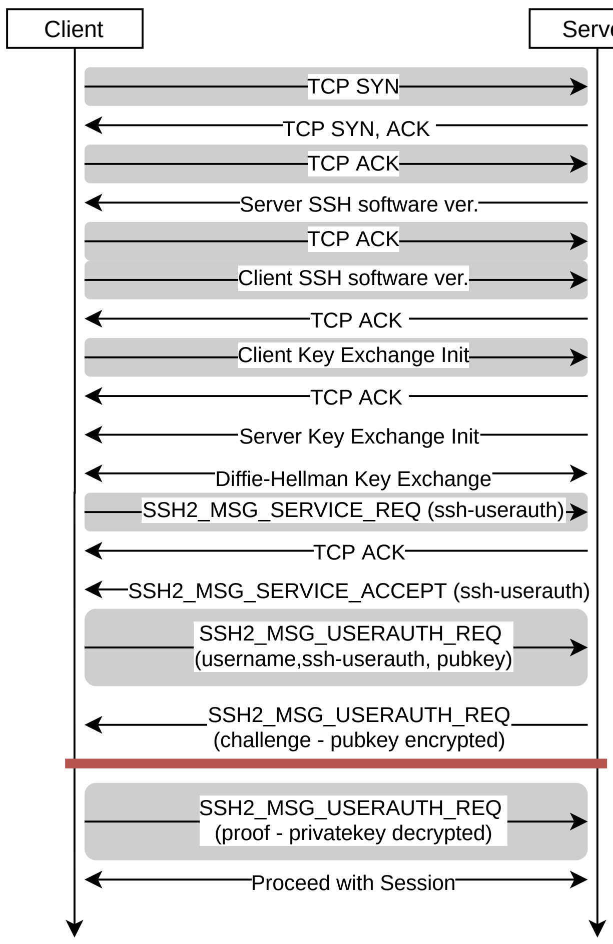

Client
Server
TCP SYN
TCP SYN, ACK
TCP ACK
Server SSH software ver.
TCP ACK
Client SSH software ver.
TCP ACK
Client Key Exchange Init
TCP ACK
Server Key Exchange Init
Diffie-Hellman Key Exchange
SSH2_MSG_SERVICE_REQ (ssh-userauth)
TCP ACK
SSH2_MSG_SERVICE_ACCEPT (ssh-userauth)
SSH2_MSG_USERAUTH_REQ (username,ssh-userauth, pubkey)
SSH2_MSG_USERAUTH_REQ (challenge - pubkey encrypted)
SSH2_MSG_USERAUTH_REQ (proof - privatekey decrypted)
Proceed with Session

Figure 5.1: Step-by-Step SSH authentication. The red line indicates when a client knows if a public key is present or not.

5.2 Methodology

In this section, we discuss the general methodology we use, including the ethical considerations. For details on our dataset, i.e., the scanning schedule etc., please see Section 5.3.

5.2.1 Public-Key Presence Identification

The core of our methodology is the behavior of the SSH protocol during public key authentication, see Section 5.1. Using the method described there, see also Figure 5.1, we can identify if a specific public key was installed for a specific user account by sending the fingerprint of a public key, and checking whether we did receive challenge.

Hence, in principle, for a given set of public keys and possible usernames on a given host, we can:

1. Start an authentication process
2. Send the username and pubkey fingerprint to the server
3. Either
   (a) Receive a challenge, and hence know that the specific *public key* for the user *user*, i.e., anyone in possession of the corresponding *private key* (but *not* us) could log into the server as that *user*, OR,
   (b) Do *not* receive a challenge, and hence know that the corresponding key is not installed for that user.

Please note that there is a limited chance for false negatives. Even though *if* the server would not send a challenge for a key even though it *was* installed, normal authentication with that key would also not be functional. In turn, however, a sever might send challenges despite a key not being present, which we will address in Section 5.2.4.

Furthermore, as the only two parameters required to execute this method are (i) a public key, and, (ii) a username, there is *no possibility* to log in/complete the authentication attempt for us, as we *only* are in possession of *public* keys. As such, the thick red line in Figure 5.1 also marks the point where our probes for a user/key combination end. We implemented this method as a patch for Zgrab2, which we make publicly available at `https://edmond.mpg.de/dataset.xhtml?persistentId=doi%3A10.17617%2F3.LVPCS6`.

Such probes do show up in authentication logs as failed authentication attempts, being indistinguishable from 'normal' brute-force attacks. We hence apply dedicated consideration to this issue in our ethics discussion, see Section 5.2.7.

5.2.2 Lab Experiment

Prior to implementing our methodology on Internet-scale, we performed a set of lab experiments to ensure its efficacy and safety given a broad set of SSH implementations. For that, we first tested our implementation on several versions of OpenSSH[54], Dropbear[55], BitviseSSH[191], and WolfSSH[192]. We selected these implementations based on the observed distribution of implementations in scan-data from censys.io[2], see Table 5.1, and evaluate each version between the newest and oldest listed in the table.

Our in-lab validation consisted of four distinct tests:

Deployment: The first step is to ensure that each SSH server can be successfully built and deployed. This process presents challenges, particularly with older versions; for example, Dropbear SSH versions 0.44 to 0.46 cannot be compiled due to the unavailability of required custom libraries.

Zmap Test: Upon successful deployment, we conduct a Zmap scan to confirm that the SSH server's port can be accurately identified as open. This step is essential for validating the initial phase of our scanning protocol.

Zgrab2 Test: For this test, we install a set of public keys on the SSH server and used Zgrab2 to issue a 'preauth' request. We assess the server's response to determine if the requested public key is recognized.

Public Key Login: Finally, we perform a public key login to verify that the installed keys are functional and that the SSH server correctly performs public key authentication.

The results of our in-lab testing are summarized in Table 5.1. We did not encounter any stability issues during our tests. However, OpenSSH versions ($\leq$2.9) only support cryptographic algorithms no longer supported by current cryptographic libraries, which is why we left them out-of-scope.

One notable issue involved the handling of `ssh-rsa` keys. Since OpenSSH version 8.2, `ssh-rsa` has been deprecated[1], meaning that all OpenSSH servers running version 8.2 or higher reject `ssh-rsa` keys, even if they are installed.

To address this, we implemented a workaround similar to the approach used by the OpenSSH client. If an `ssh-rsa` key is denied due to its deprecation (indicated by a specific error message), our instrument automatically attempts to reconnect using the `rsa-sha-256` algorithm instead. Specifically, if the key '`ssh-rsa AAAABBBCCCDD...`' fails, our tool retries with '`rsa-sha-256 AAAABBBCCCDD...`'.

5.2.3 Controlled Environment Tests

To further evaluate our instrument, we also ran tests over the Internet against two consenting ASes. In each of the ASes, several hosts were deployed and SSH public keys installed for various users. Team members not informed about which hosts and users on these hosts had which keys installed then utilized our instrument to scan the ASes. Furthermore, we conducted high-throughput scans, again with consent of the operators, to assess whether unexpected issues may occur. These initial runs were invaluable for identifying and resolving minor bugs, as well as for uncovering unexpected behaviors.

For example, one issue our tool encountered was the 'max-startups' error, which can occur on smaller SSH servers that limit the number of simultaneous connection attempts. To address this, we implemented workarounds such as temporarily delaying the re-scan and retrying later, ensuring that the server could still be successfully scanned without being overwhelmed.

[1]https://www.openssh.com/txt/release-8.2

	Year	**Censys** (all ports) 2024-04-16	**Our Scan** (22, 2222) 2024-04-05	**Censys** (22, 2222) 2024-04-24	Deployment	Zmap	Zgrab2	PubKey Login
OpenSSH:								
9.4	2024	<0.01%	<0.01%	<0.01%	✓	✓	✓	✓
...	2016-2024	29.21%	22.7%	46.1%	✓	✓	✓	✓
7.4	2016	23.01%	17.00%	18.08%	✓	✓	✓	✓
...	2001-2016	22.28%	12.3%	6.17%	✓	✓	✓	✓
3.0	2001	<0.01%	<0.01%	<0.01%	✓	✓	✓	✓
2.9	2001	<0.01%	<0.01%	<0.01%	✓	✓	✗	✗
...	2000-2001	<0.01%	<0.01%	<0.01%	✓	✓	✗	✗
2.1.1	2000	<0.01%	<0.01%	<0.01%	✓	✓	✗	✗
Dropbear:								
0.84	2024	<0.01%	<0.01%	<0.01%	✓	✓	✓	✓
...	2019-2024	2.42%	0.91%	0.77%	✓	✓	✓	✓
0.78	2019	8.36%	10.01%	12.74%	✓	✓	✓	✓
...	2005-2019	3.51%	1.23%	3.80%	✓	✓	✓	✓
0.47	2005	<0.01%	<0.01%	<0.01%	✓	✓	✓	✓
0.46	2005	<0.01%	<0.01%	<0.01%	✗	✗	✗	✗
...	2004-2005	<0.01%	<0.01%	<0.01%	✗	✗	✗	✗
0.44	2004	<0.01%	<0.01%	<0.01%	✗	✗	✗	✗
0.43	2004	<0.01%	<0.01%	<0.01%	✓	✓	✓	✓
...	2003-2004	<0.01%	<0.01%	<0.01%	✓	✓	✓	✓
0.39	2003	-	-	-	✓	✓	✓	✓
0.38	2003	<0.01%	<0.01%	-	✗	✗	✗	✗
...	2003	<0.01%	<0.01%	<0.01%	✗	✗	✗	✗
0.23	2003	<0.01%	-	-	✗	✗	✗	✗
BitviseSSH:								
9.31	2023	<0.01%	-	<0.01%	✓	✓	✓	✓
9.29	2023	<0.01%	-	<0.01%	✓	✓	✓	✓
8.49	2021	<0.01%	-	<0.01%	✓	✓	✓	✓
7.46	2018	<0.01%	-	<0.01%	✓	✓	✓	✓
6.51	2018	<0.01%	-	<0.01%	✓	✓	✓	✓
WolfSSH:								
1.4.14	2023	<0.01%	<0.01%	<0.01%	✓	✓	✓	✓

Table 5.1: SSH server implementation, in-lab testing.

5.2.4 False-Positive Detection

A limited number of SSH servers observed in the wild consciously and consistently reply to all authentication requests with a challenge, regardless of the key actually being installed. These may either be honeypots, or–as we observed during our measurements–specific SFTP server implementations, see Chapter 6. Similarly, we encountered SSH servers that would send a challenge for any public key when it uses an algorithm they did not support, e.g. a prototype SSH server with support for federated authentication accepting all ED25519 keys [193].

To mitigate this issue, we use 'Canary' keys. These keys are fully valid SSH keys, but have been newly generated by us. If a remote server responds with a challenge for this key, we can be reasonably sure that it will do so for all keys. Furthermore, to mitigate the issue of some servers only showing this behavior for specific algorithms, the canary key must always be of the same key-type as other keys being tested at that time, e.g., only using an ED25519 canary key provides information regarding ED25519 keys under test. If RSA or DSS keys are to be tested as well, an additional canary for these algorithms must be created and tested.

In case a canary key solicits a challenge on a remote host, we classify this system as not being testable with our methodology. Over the course of our measurements, roughly 0.2% of hosts fell into this category.

Algorithm 1: Public Key Scanning Protocol.

```
Data: PublicKeys, TargetedPort, UsernameList, Blocklist
Result: Scan results for open ports and associated public keys
Split PublicKeys into 11 pubkey_sets;
foreach port in TargetedPort do
    foreach User in UsernameList do
        for round ← 1 to 11 do
            IPList ← Check blocklist;
            targetIPs ← Zmap(IPList, port);
            Generate 'Canary Key' with the same algorithm as the keys in pubkey_set;
            potential_ips ← Check targetIPs with 'Canary Key';
            foreach pubkey in pubkey_set do
                output ← Check potential_ips with pubkey;
                Sleep for 1 hour;
            end
        end
    end
end
```

5.2.5 Malicious Keys

To execute our methodology, we need to obtain a dataset of known malicious public keys. We received a set of 52 malicious keys from Bitdefender [37]—the threat intelligence company with which we are collaborating. These keys were encountered by the company during the investigation of incidents, during malware analysis, and in honeypots they operate.

We also used the available data from GCA honeyfarm (see Chapters 3 and 4). However, no additional keys were found, i.e., all keys found were also included in the data shared

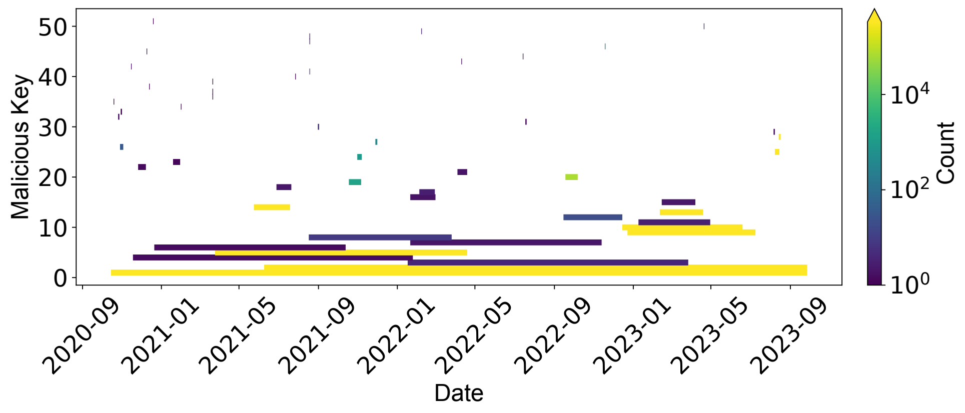


Figure 5.2: SSH Scan: Activity periods of malicious key recorded by the threat intelligence company.

by the threat intelligence company. The dataset also includes the first and last time the threat intelligence company observed each malicious key, with just a few keys being seen consistently over several years. Furthermore, we received the number of events observed in relation to a key, and–if available–the user name in which it was found. While we did receive some attribution information to known groups for individual keys, Bitdefender was unable to publicly share background information on most keys.

For our analysis, we ranked keys based on the number of events and their activity period, see Figure 5.2. This ranking is used in our subsequent analysis, with Malicious Key 01 (*MK01*) being associated with the highest number of attacks, and Malicious Key 52 (*MK52*) corresponding to the fewest attacks recorded in the threat intelligence company's dataset.

5.2.6 Tested Usernames

A caveat of our methodology is that we need to initiate an authentication attempt for each username and public key combination we want to test. Naturally, this means that we have to limit the usernames we are testing for the keys obtained from Bitdefender, as each tested username leads to 52 authentication attempts.

We hence decided to limit our measurements to three user names. Here, we include 'root' and 'admin' based on findings in prior work that consistently see these two accounts as the most targeted ones [83, 194, 195, 196]. Additionally, we included 'udatabase', given that this was the third most common username in the public key dataset where this information was available, after 'root' and 'admin'. The dataset we received from the threat intelligence company contains a total of 23 unique usernames, with 'root', 'admin', and 'udatabase' being the most frequent. The three selected usernames account for 83% of compromises in the threat intelligence dataset.

5.2.7 Ethical Considerations

Our work entails several ethical implications. As such, we tightly followed best-practices, see, e.g., the Menlo report [36], during the preparation and execution of our measurements.

IRB Clearance: Our work was submitted under No. 24-02-1 to our institution's responsible IRB. The IRB evaluated our experimental plan and instigated procedures, and ultimately granted permission to execute the work. The IRB was again contacted under the same case concerning a continuous execution of our measurements, see below. Similarly, this was also granted by the IRB.

Scanning Best-Practices: We followed best-practices for Internet wide scans, i.e., ensured a responsive reverse DNS entry, hosted a web-page explaining the experiments on our measurement systems, and actively handled messages sent to our network's abuse contact. This included opting out remote parties as soon as they requested it, and also actively notifying each person writing to the abuse contact that they can opt out.

Harm-Benefit Analysis: In addition to following best practices, we conducted a harm benefit analysis concerning our study. We identified the following potential for harm:

- Performing non-authorized connections to third parties.
- Accidentally overloading individual remote systems.
- Causing additional workload, as the scans are, see Section 5.2.1, indistinguishable from brute-force attempts.

We acknowledge that the harm of non-authorized connections cannot be easily mitigated. Similarly, the additional workload on operators cannot–despite best efforts in terms of transparency–be easily mitigated. Even though our traffic should not stand out in contrast to standard Internet background noise, these points will have to be weighted against the potential benefits of our research.

Concerning harm due to accidentally overloading systems, we took precautions in terms of rate-limiting, see Section 5.3. However, in practice these might be ineffective in individual cases. Issues we encountered were, for example, large sets of IPv4 addresses mapped to a single host [197], e.g., for SNI, and NAT64 setups accumulating significant traffic, as they map the whole IPv4 Internet into a single /96 IPv6. If we encountered such cases and either our monitoring or external notifications alerted us to them, we implemented additional precautions to ensure a further spread of requests.

In addition to the identified potential harm, we also identified several potential benefits of our work:

- The identification of compromised hosts allows remediation, especially if attackers did not yet become active on a system and remains undetected. This is specially crucial for compromised critical infrastructure systems.
- Characterizing how and where malicious actors compromise systems may provide further insights regarding individual malicious groups, allowing further root-cause oriented mitigation.

Weighting these potential benefits against the aforementioned potential harm, we concluded that our measurements *could* be ethically feasible, *if* we notify affected parties.

Notification: To ensure that affected operators are notified, we–prior to starting our measurements–reached out to the Shadowserver Foundation. The Shadowserver foundation is a non-profit focused on making the Internet secure, and operates notification channels where they distribute information about, e.g., compromised systems to responsible CSIRTs and operators. We opted against a general individual notification campaign, given that the expected number of compromised hosts would exceed what could reasonably be notified individually[2]. As such, special reports have been released by the Shadowserver Foundation for our results [3].

In addition to the Shadowserver foundation, CERT-Bund–the national CSIRT for Germany located at the Federal Office for Information Security (BSI)–reached out to us concerning our scans, initially assuming a compromise due to the–seemingly–SSH bruteforce attempts. After an explanation of our work, they also offered to participate in the notification of affected parties.

Furthermore, given the success after the first round, we obtained ethical clearance for *continuous* scans from our IRB. These scans are now running, and data-processing and notification have been automated with both, the Shadowserver foundation and our national CSIRT.

5.3 Experiment

Here, we briefly describe our scanning setup and dataset, i.e., we provide information on the machines used for our measurements, and the implemented scanning schedule.

5.3.1 Measurement Infrastructure

Our measurement systems are four physical machines with 16 cores/32 threads and 64GB of memory each. The machines are connected to a dedicated network segment that does not pass through stateful middle-boxes before reaching the Internet. Each machine had an informative reverse DNS set and ran a website with information on our study, see Section 5.2.7.

5.3.2 Scan Execution Methodology

For our scans, we first ingest lists of available IP addresses. For IPv4, we utilize the CAIDA Routing Data which collects BGP (Border Gateway Protocol) routing information and use a snapshot of all routable prefixes from April 2024[4]. For IPv6, we leverage the IPv6 Hitlist[5].

We then pre-process these inputs by filtering the prefixes against our blocklist, which is maintained and updated over time by our group to honor opt-out requests. Subsequently, we generate all IPs from the prefixes and randomly distribute them into 16 sets of similar size. Each machine is assigned to scan four of these sets.

For each set, one per machine at a time, we then process chunks of up to five keys. We first perform **Port Discovery**, where we utilize Zmap [198] to identify IP addresses that have the TCP port for that run open (Port 22 or 2222) [6]

[2]We conducted individual notifications for high-profile cases, e.g., government infrastructure, or a compromised back-bone router of a major ISP in the corresponding country.

[3]`https://www.shadowserver.org/what-we-do/network-reporting/compromised-ssh-host-special/`

[4]`https://publicdata.caida.org/datasets/routing/routeviews-prefix2as/2024/04/`

[5]`https://ipv6hitlist.github.io/`

[6]We do this multiple times, as we significantly reduced our scan volume, by ensuring that each IP only receives no more than one authentication attempt per hour. Hence, we need to periodically re-run zMap to prevent the impact of IP address churn influencing our results. See Section 5.2.7.

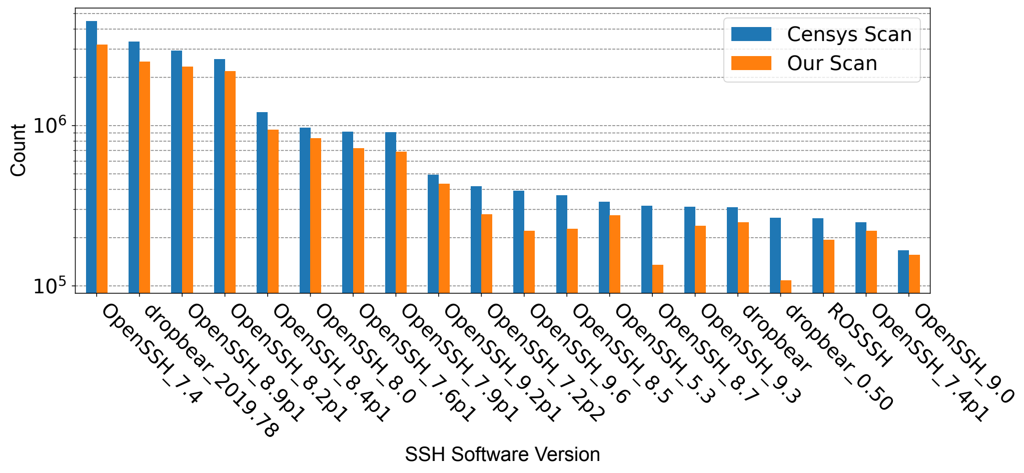


Figure 5.3: Censys vs. Our Scan (port 22 only)

Next, we execute **Public Key Verification** for all hosts with open ports. There, we employ our patched version of Zgrab2 [199], see Section 5.2, to check for the presence of a public key on each IP address with an open port identified in the first phase. We limit the number of Zgrab2 runs to one per hour. Before testing any malicious keys, we first run a newly generated canary key of the same algorithm as the other keys in the set, see Section 5.2.4. If the canary key solicits a challenge or results in a timeout/error, we do not test the remaining keys in the set against that specific host.

The total time to complete a full (all users) scan for one port on IPv4 is 37 days due to the rate limits we set. Specifically, for three users and four sets per scanning machine, Zgrab2 has to be run for 52 malicious keys and 11 canary keys (maximum of five keys per iteration), and we also run zMap 11 times. For IPv6 we use only one IP set per machine, i.e., can complete a run in 9–10 days.

5.3.3 Scanning Schedule

The data collection for the dataset started on 2024-04-04, beginning with the IPv4 address space and subsequently progressing to the IPv6 address space. The scanning of usernames was conducted in the following sequence: ‘root’, ‘admin’, and ‘udatabase’. All 52 keys were checked. The scans for port tcp/22 completed on 2024-06-01, and the one for the subsequently started port tcp/2222 measurements on 2024-07-31.

5.3.4 Descriptives and Comparison to Censys

In the collected dataset, we find around 25 million (25M ± 110K) servers responding on port 22, while 4.5 million (4.5M ± 31.2K) servers responded on port 2222. Please note that the listed deviation is due to IP address churn over time, i.e., when hosts became (un)reachable in between multiple key set iterations, or across different usernames. For port tcp/22, 23.9 million (23.9M ± 192K), i.e., nearly all actually implemented the SSH protocol, while the fraction on port tcp/2222 is lower with only 643K ± 5.7K servers confirmed as running SSH. These findings align with Censys scans conducted during the same period. On 2024-04-16, Censys reported 25.2 million servers on port 22 and 655K servers on port 2222 actually running SSH.

We also compare the SSH server implementations and versions identified in our scans with those from Censys scans, see Figure 5.3. Especially for major versions our results (orange) closely align with those of Censys (blue), see also Table 5.1. Our comparably lower rate of identified systems aligns with our use of a blocklist, see Section 5.2.7.

Our methodology was executed successfully on 16.9 million (16.9M ± 89K) hosts, while 8.3 million (8.3M ± 23K) hosts returned an error, e.g., not being configured to support public key authentication, or timed out. Additionally, 44K ± 1.3K hosts triggered our false-positive detection. Please note that the 8.3M failed hosts include hosts that were incomplete, e.g., had a timeout for individual keys, possibly including the canary key. However, with such partial results, we can not make definitive statements on whether these hosts are compromised.

5.4 Summary

In this chapter, we present an Internet-scale method to identify compromised SSH servers leveraging SSH's behavior of only sending a challenge for installed public keys during authentication. Our method neither requires privileged access nor does it allow us to access compromised systems, limiting the ethical implications of our approach.

We begin with a series of controlled in-lab experiments across various SSH server implementations and versions. These experiments confirmed that our tool behaves as expected in diverse scenarios. To further validate our findings, we compared the results of our initial port scan with data from Censys, which provided additional confidence in the reliability of our scanning tool.

Following the lab tests, we conducted a small-scale scan on a live network. This phase allowed us to identify and resolve several issues and bugs in our tool-chain, as well as to calibrate the scan rate for the optimal balance between obtaining successful responses and avoiding undue disruption to system administrators. In collaboration with Bitdefender, we compiled a set of known malicious public keys and commonly targeted usernames for use in the broader scanning effort.

We then address the ethical implications of conducting such scans, emphasizing the value of the insights gained in the context of Internet security research. Finally, we present the details of our full-scale, continuous scanning deployment, carried out using the methodology developed throughout this chapter.

Chapter 6
Compromised SSH Hosts

6.1 Introduction

In this chapter, we present the first study on the data collected by the previously described mechanic to identify compromised systems at Internet scale. The information gathered by the experiment described in Chapter 5, enables us to identify more than 21K compromised systems worldwide, including critical systems in core Internet infrastructure, government agencies, research institutions, and critical civil infrastructure. Furthermore, our collected data on *who* compromises *which* systems *where* allows us to characterize and analyze attacker behavior in so far unprecedented detail. Finally, we integrated our measurements with a notification campaign via the Shadowserver Foundation and–later–the national CSIRT for Germany located at the Federal Office for Information Security (BSI). We currently perform continuous regular scans with automated notifications generated and distributed via our partners.

Our main contributions are outlined as follows:

- We identify more than 21,700 compromised hosts in 1,649 ASes for 52 SSH public keys attributed to malicious activity on the most common SSH ports (tcp/22 and tcp/2222) for IPv4 and IPv6.
- We describe attackers' modus-operandi (identified by the used malicious key (*MK*)) in unprecedented detail, including persistent IoT botnets, as well as likely nation-state level attackers and individual attack specific keys.
- We track changes to compromised servers over consecutive measurements, and find that individual high-profle cases are quickly cleaned.
- We cross-reference our active measurement data with results from a large honeypot network, establishing that 24 of the compromised hosts we identified are actually used to perform attacks and other two are used as malware storage location, i.e., show ongoing attacker activity.

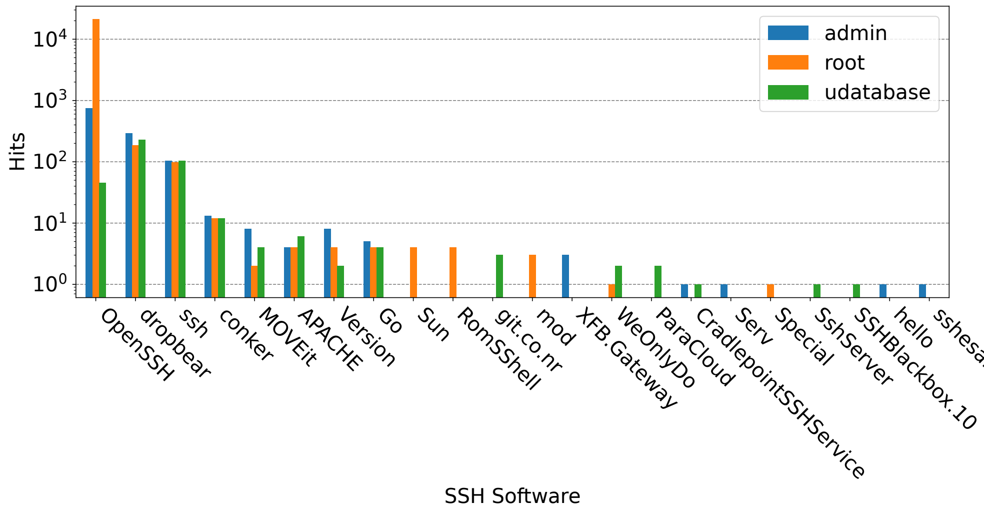


Figure 6.1: SSH server implementation vs. username, port 22

- We also demonstrate how it allows tracking malicious activity to better understand ongoing attacks.

In this chapter we present our results of our experiment. Initially focusing on findings for the standard SSH port, i.e., tcp/22, we first discuss how different types of SSH implementations show up among compromised hosts. Subsequently, we focus on the AS types where systems are affected, and then take a more global perspective, before delving into the behavior of individual actors. Finally, we present our results for non-standard SSH ports, tcp/2222 and we provide insights gained from our large-scale scanning campaign.

6.2 Compromised Host Overview Port 22

Overall, we find 22,938 instances of malicious keys being installed, 22,691 of these being for IPv4 (21,061 hosts) and 247 for IPv6 (163 hosts). Filtering by server host key, this number reduces to 16,753 unique servers. Please note that this likely includes host key collisions, e.g., due to poorly engineered IoT devices sharing host keys [200]. Furthermore, the host keys of 125 found via IPv6 can also be found via IPv4, with the other 38 being unique to IPv6.

SSH Server Versions: SSH servers usually sent their version and/or general information (banner) after a connection has been established. Investigating these, see Figure 6.1, we find that the most frequently encountered SSH server implementations are–as expected–OpenSSH and Dropbear, due to their general popularity. Beyond that, we find a plethora of other banners, including IoT specific software (RomSSHell), and servers obscuring their banner (simply returning, e.g., 'ssh'). Relating this information to the users that were compromised, 'root' is more frequent for OpenSSH than 'admin' or 'udatabase', likely due to OpenSSH being more commonly found on servers in comparison to, e.g., the embedded focused Dropbear. Notably, we find 320 cases of hosts were multiple users have been compromised.

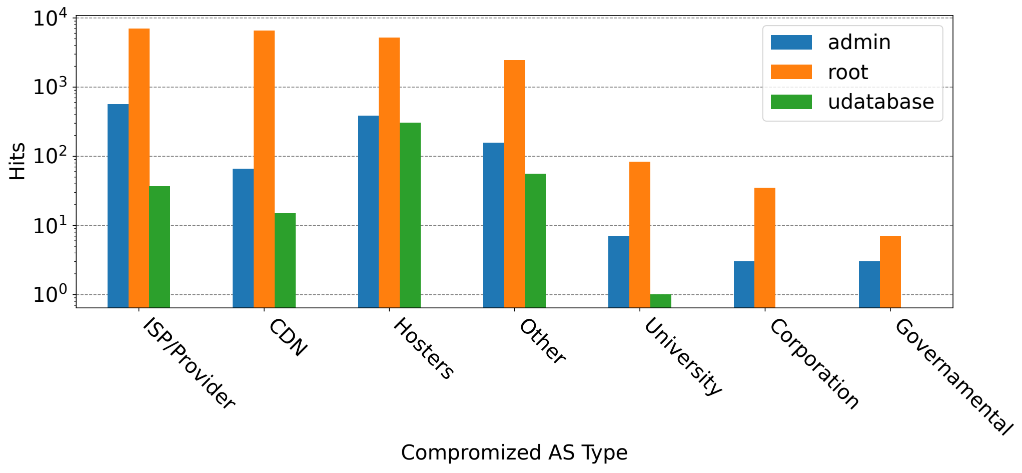


Figure 6.2: AS type vs. username, port 22

From the perspective of compromised users, there is an order of magnitude more compromised root accounts compared to the other two usernames. Within OpenSSH, there is a clear hierarchy in the prevalence of compromised users, with 'root' being the most common, followed by 'admin' and then 'udatabase'. In other SSH implementations, the distribution of compromised users appears relatively uniform. However, towards the tail of the plot, less popular server implementations tend to have only one compromised user. The analysis also reveals that 101 servers have all three users compromised, 219 servers have two users compromised, and 20904 servers have only one compromised user.

Affected AS Types: Next, we assess what types of ASes contain compromised systems. To categorize ASes, we use labels from PeeringDB [152] and data from BGP.tools [151], creating seven categories: CDN, corporate, government, hosting providers, ISPs, universities, and 'other', see Figure 6.2 for an overview.

Figure 6.3 illustrates the distribution of different SSH server implementations among the compromised hosts across various AS categories. The X-axis represents the SSH server implementations, while the primary Y-axis shows the ratio of hits for each SSH server implementation distributed across the different AS categories, with each category represented by a distinct color. The secondary Y-axis features a black dashed line indicating the total number of hits. Our analysis reveals that the majority of compromised hosts are found in ASes categorized as ISP providers, CDN, hosting providers, or others. `OpenSSH`, the most prevalent SSH server implementation, appears evenly distributed across these categories (see Figure 6.4). In contrast, certain implementations, such as `Dropbear` and `ssh`, are predominantly located in hosting provider ASes, while `APACHE`, `Version`, and `Sun` are more frequently found in ISP provider ASes. The `conker` implementation is unique in that it is observed exclusively in university/educational ASes, whereas `MOVEit` and `XBF.Gateway` are primarily associated with corporate ASes. The few instances of compromised hosts within government ASes are exclusively `OpenSSH` implementations.

We find that the majority of compromised hosts can be found in end-users ISPs, again hinting towards a high number of non-traditional servers among compromised hosts. In contrast, we find a limited number of compromised machines in University networks, Corporations, and Government networks. However, please note that especially giving

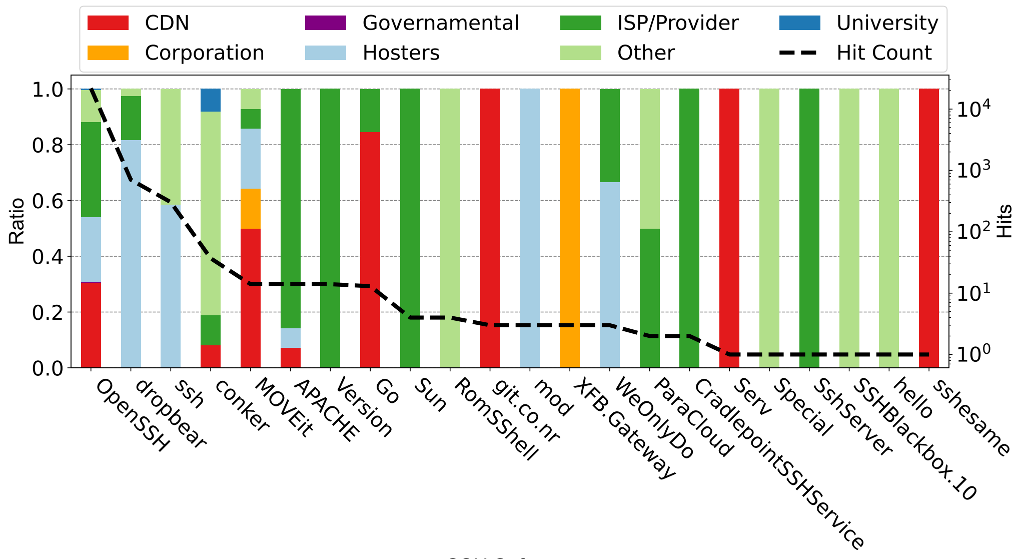


Figure 6.3: SSH server implementation vs. AS type, port 22

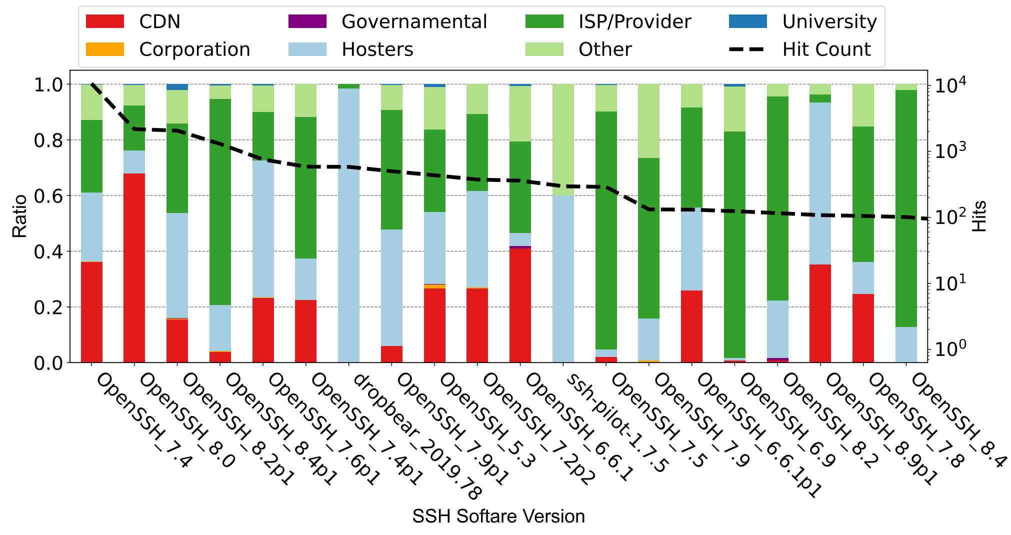


Figure 6.4: Top 20 SSH server versions vs. AS type, port 22

the progressing flattening of the Internet [201], a system in a hoster or other type of AS may still be operated by, e.g., another company or a government agency [202].

Summary: The majority of compromised systems runs OpenSSH, and these systems are concentrated within end-user ISPs, CDNs, and hosting ASes, with the 'root' user being the most frequently compromised across all categories.

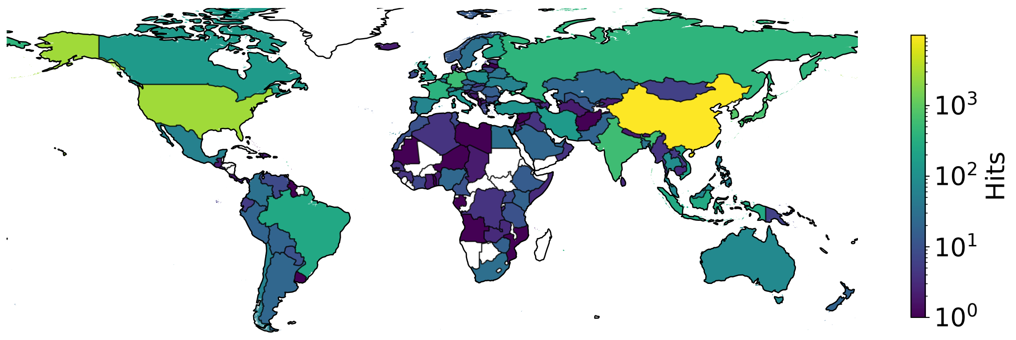


Figure 6.5: Compromised hosts port 22—world distribution

6.3 Geographic Distribution

Next, we investigate *where* compromised hosts are located. To determine the geolocation of each compromised server, we use the host IP address in conjunction with the IPinfo [203] geolocation tool, pinpointing the location at the country level. Figure 6.5 illustrates the distribution of compromised hosts across various countries.

Country-Level Distribution: We identify at least one compromised host in 144 different countries. On a high level, the distribution of compromised hosts follow, roughly, the 'size' of the Internet in corresponding countries, i.e., the highest numbers of compromised hosts are located in countries with a large population and/or a traditionally strong IT industry. Hence, the majority of these compromised hosts are concentrated in North America, Europe, and Asia, with fewer instances observed in South America, Australia, and Africa.

Location vs. AS type: Next, we study the affected AS types per country, see Figure 6.6 for the distribution of compromised servers across AS types in the 30 countries with most hits.

Globally, the majority of compromised machines is found in end-user ISPs and hosters. However, for–especially–China and France, a large portion of compromised machines is also found in CDNs. We attribute this to an unclear classification of the Cloud/Hosting business of major platforms in these countries (Huawei Cloud, Baidu, OVH) as CDNs in our data sources. Notably, for Iran, we find a higher number of machines having attacker-attributed SSH keys installed in universities.

Summary: Overall, we find compromised SSH servers to be globally evenly distributed along traditional metrics, with expectable concentrations in North America, Europe, and Asia, particularly in countries with large populations or robust IT infrastructure.

6.4 Malicious Keys & Actors

Our initial scan set included 52 verified malicious SSH public keys. Out of these, we found 41 in the wild. However, some of these keys are found on a notably larger portion of compromised machines than others. While not all keys could be attributed to specific attackers, our collaborator provided information that attributed six keys to 'teamtnt' (*MK23*, *MK30*, *MK31*, *MK32*, *MK33*, *MK34*), and three to the actor 'mexalz' (*MK25*, *MK26*,

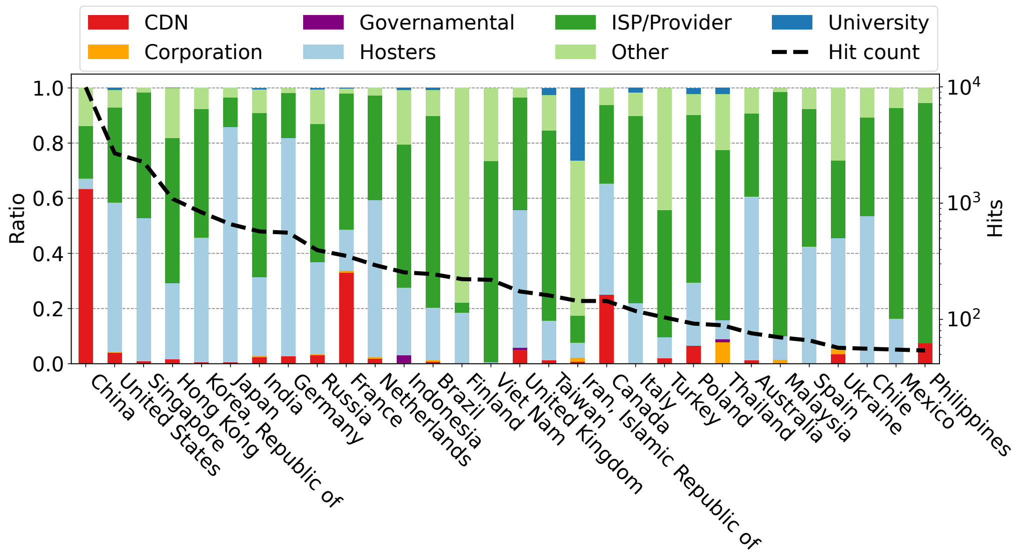


Figure 6.6: Top 30 countries based on number of compromised servers vs. AS type, port 22

MK27). Additionally, one key was attributed to the 'fritzfrog' P2P botnet (*MK01*), two to the 'coinminer' botnet (*MK40*, *MK43*), and one each to the 'mozi' (*MK28*), 'hehbot' (*MK24*), and 'muhstick' (*MK29*) botnets. Furthermore, two keys have been seen in relation to the persona 'Jia Tan', involved in the XZ backdoor [204]. Beyond that, we labeled *MK06* as 'mdrfckr', as–contrary to other keys–it is regularly installed on honeypots with that string in the SSH key's comment field. We did not receive specific attribution for the remaining keys, beyond their involvement in various malicious activities.

Expected vs. Measured Frequency Based on the number of hits observed by the threat intelligence company, there were underlying expectations that the most observed keys by them would also be the most observed keys by our methodology. This, however, is not the case, see Figure 6.7. The top malicious keys identified in our scans are *MK06* (16535 hits), *MK35* (2050 hits), and *MK32* (727 hits). We attribute this to different exploitation methodologies, i.e., keys observed more frequently in our dataset being either deployed via different methods[7], or the actors behind them being more adapt in discovering honeypots, not deploying keys there.

'teamtnt' and the 'ssh' Banner: 'teamtnt' emerged as a significant threat in early 2020, primarily targeting cloud environments, including misconfigured Kubernetes clusters, Docker APIs, and Redis servers [8]. This attribution supports the hypothesis that these keys may be part of an attack campaign aimed at CDNs.

For 'teamtnt', we identified 757 hosts compromised by one of their six keys. These keys are also over-proportionally found in hosting ASes, as well as South-East-Asian networks classified as 'CDN' by our data, i.e., most likely misclassified cloud networks, see Figure 6.9. This maps to the modus-operandi of 'teamtnt', i.e., their focus on compromising systems via misconfigurations and vulnerabilities associated with common cloud

[7]'teamtnt', e.g., focused on exploiting Kubernetes setups.
[8]`https://aquasec.com/blog/new-malware-in-the-cloud-by-teamtnt`

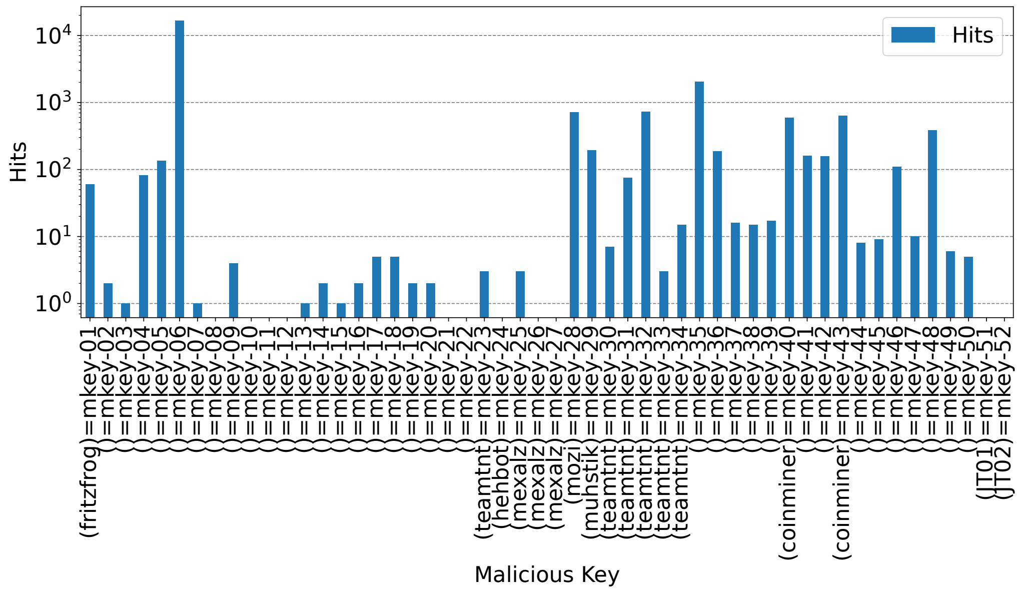


Figure 6.7: Malicious key count, port 22

software stacks, e.g., Redis [205] and Kubernetes [206].

Notably, though, one key attributed to 'teamtnt', *MK32*, is one of two keys found on servers featuring a version banner of simply '`ssh`'. The other key found on such hosts is *MK32*, and usually overlaps with *MK06* and *MK36*. Closer investigation revealed that the overlapping IP addresses–either in continuous netblocks in the Baidu cloud or in a hoster from Hong Kong–belong to a Chinese Git SaaS solution `gitee.com` operating similar to Github.

The most likely explanation for this is that the platform is being used as the code repository by 'teamtnt' and the unknown threat actor behind *MK36*, and the *custom* SSH implementation of the Git hoster replies with a challenge for any *known* key, i.e., key uploaded by a user for use with their repository. This is also supported by these keys (and no other keys, including the canary) having been found for all three users we tested for on those IPs, i.e., the gitee.com platform is *not* compromised. Additionally, revisiting file samples in relation to 'teamtnt' compromises, we also find these tool-chains to explicitly reference gitee.com [207, 208, 209]. We hence conclude that 'teamtnt' as well as the unknown actor behind *MK06* and *MK36*–possibly overlapping with 'teamtnt'–are likely using `gitee.com` for hosting their source code.

IoT Botnets: When looking at Dropbear, as a software more likely to be found on embedded devices, we only find *MK01* (7 hits), *MK06* (610 hits), *MK46* (2 hits), *MK48* (74 hits), and *MK49* (1 hit). Following up on the ASes we find these keys in, we, e.g., find *MK01* ('fritzfrog') to be most prevalent in end-user ISPs, i.e., exactly where one would expect a large number of IoT devices. Interestingly, our–by far–most prominent key, *MK06*, is also wide-spread among IoT devices. Unfortunately, aside from the label associated with this public key—'mdrfckr'—we lack additional information regarding the identity or origin of the attacker behind this activity. Still, we reached out to the authors of a related study investigating SSH honeypot data [31], and they confirmed that this key is among the most aggressive attacks identified in their research as well. Other

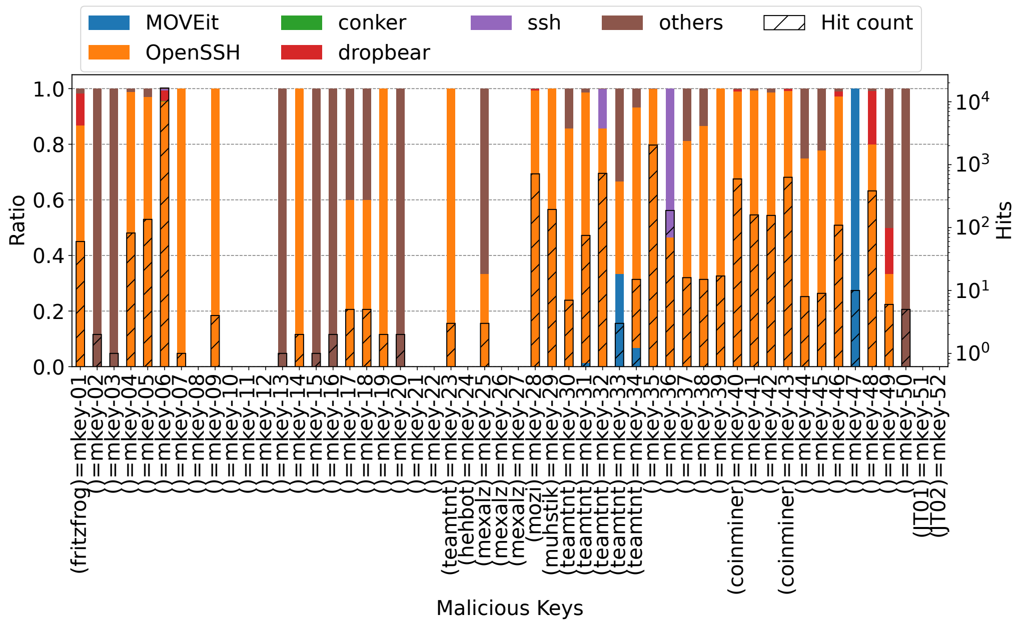


Figure 6.8: Malicious key vs. SSH server, port 22

keys found in notable frequency on Dropbear servers, i.e., likely IoT devices, are *MK48* and *MK49*, see Figure 6.8, even though those are less constrained to ISPs, see Figure 6.9.

Turning our attention to 'muhstick' botnet, we again find a high prevalence, i.e., more than 95% of hits in end-user ISPs. Incidentally, though, 'muhstick' is not prevalent among hosts running Dropbear. This is likely due to its distribution mechanic focusing on a critical Apache RocketMQ vulnerability (CVE-2023-33246) [9] to achieve remote code execution, targeting Linux servers and IoT devices for distributed denial-of-service (DDoS) attacks and cryptocurrency mining. As such, it is more likely to compromise home-servers beyond embedded machines, likely to also run OpenSSH instead of Dropbear.

Finally, we examine the correlation between the servers compromised by specific malicious keys and the usernames that are compromised. Figure 6.10 illustrates this correlation. The plot follows a similar format to the previous ones, with the exception that the bars represent the ratio of compromised usernames.

As observed, the username 'root' (orange) is the most frequently targeted by malicious keys. However, there are notable exceptions, such as *MK47*, where the 'admin' username is the most compromised. Similarly, in the cases of *MK17*, *MK18*, and *MK33*, the majority of compromised users are 'udatabase'.

Another significant finding is that botnets like 'mozi', 'muhstick', 'coinminer', and 'fritzfrog' predominantly target 'root' users, while keys associated with malicious actors such as 'teamtnt' and 'mexalz' appear to compromise servers with multiple usernames.

Potential State-Actor Deployment (*MK48*): The diverse prevalence of *MK48* throughout various AS types, see Figure 6.11, triggered a further investigation of this key. Specifically, the key can mostly be found in:

- Several European hosters generally renown for their affordable offerings.

[9] `https://thehackernews.com/2024/06/muhstik-botnet-exploiting-apache.html`

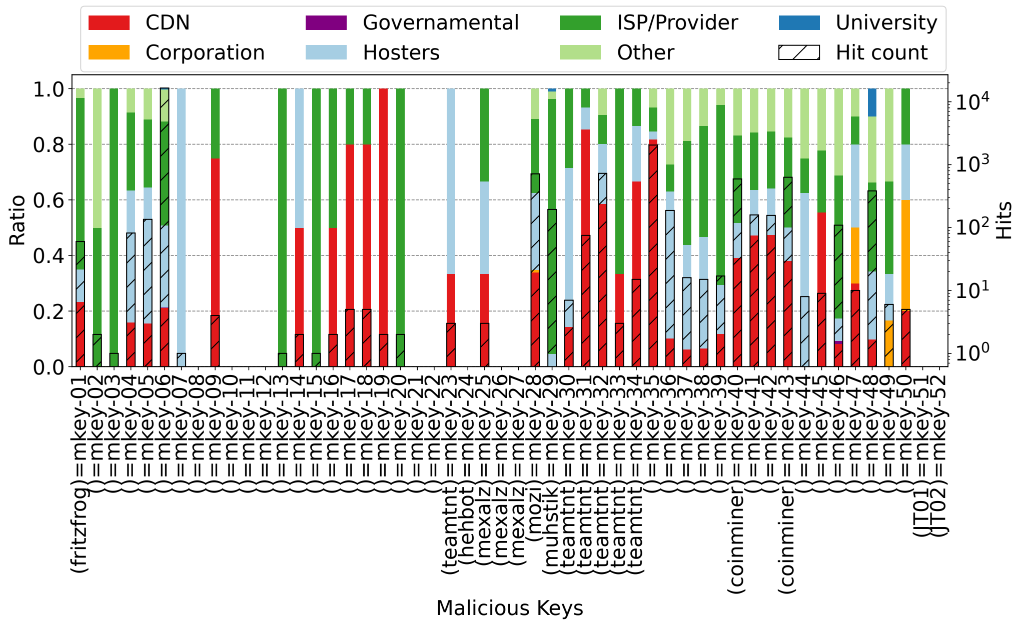


Figure 6.9: Malicious key vs. AS type, port 22

- Several ASes associated with a middle-eastern state (MES) with tense relationships with most European countries.
- Individual hosts in a diverse set of ASes, often associated to IoT devices (see above).

Notably, for the ASes where we find *MK48* within the MES, *MK48* is the *only* key we observe. Furthermore, across all systems where this key has been found with a high frequency (MES ASes and large European Hosters), we find a generally high rate of `OpenSSH 8.2p1`, i.e., the version found on Ubuntu 20.04 LTS (Focal Fossa). This version consistency contributes to an assumption that these systems may be associated with the actor behind *MK48* and not compromised.

A further investigation of the affected systems within European hosters provided several indications of them being–at least on the outside–being used for censorship evasion. This included Tor setups [210], V2Ray [211], HTTPS Proxies [212, 213], and domain-fronting [214, 215]. Incidentally, MES is known for a comparatively tight censorship regime.

Given the overall circumstances, and due to limited information available on the origin of *MK48*, several viable explanations for this pattern emerged:

- Actual censorship evasion infrastructure, also leveraging residential proxies (see Dropbear hosts).
- Censorship evasion infrastructure compromised by a state-actor from MES.
- Operated by a state-actor from MES as censorship-evasion 'honeypots' in use against their citizens.
- Operated and/or compromised by a 3^{rd}party state actor.

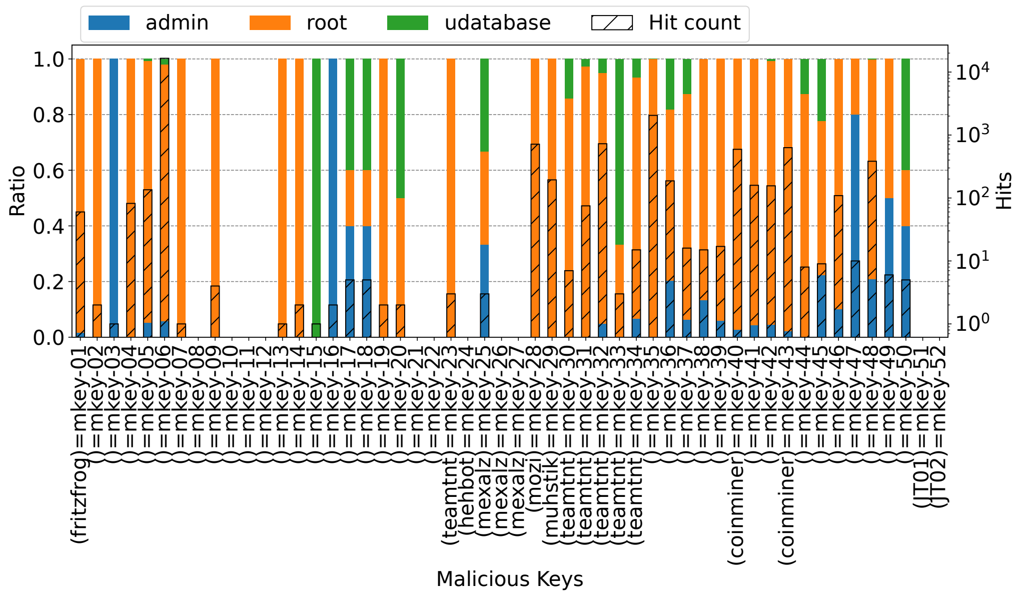


Figure 6.10: Malicious key vs. username, port 22

Given the potential ethical implications of any possible explanation, we immediately coordinated with our national CSIRT. The national CSIRT handled communication with relevant national authorities, who requested further information and asked for *MK48* to not be notified temporarily. After several weeks, we received the information that the event was investigated, and we could include *MK48* in future notifications. No further information was shared from the authorities.

KillNet Cross-Reference: Additionally, we conducted a cross-analysis with the KillNet DDoS Blocklist [10], which is a list of IP addresses or networks associated with KillNet, a pro-Russian hacking group known for launching Distributed Denial of Service (DDoS) attacks. KillNet has been actively targeting various organizations, particularly those in countries opposing Russian policies. The blocklist is instrumental in protecting networks by identifying and blocking traffic from IPs linked to KillNet's malicious activities.

Our motivation for using this specific list is twofold: First, ongoing research within our group suggests a connection between KillNet and IP addresses involved in installing public keys. This behavior makes the IP list particularly relevant to our study. Second, the KillNet list, unlike other blocklists, offers a high level of attribution confidence.

Our analysis revealed that five compromised hosts are listed on the KillNet DDoS Blocklist. All of these servers run OpenSSH, and the keys that matched are *MK06* and *MK48*, with the compromised user being 'root'. These servers are located in ISP or hosting provider ASes in Japan, Argentina, China, Germany, and the Netherlands.

The presence of *MK48* here is notable, as it adds further options regarding the MES case around *MK48* above. Given the observation of *MK06* along with *MK48*, it is possible that KillNet may be the party who compromised the MES' systems. Alternatively, the activity in and around the MES involving *MK48* may have been constructed to be able to led to false attribution of activity by KillNet to the MES. Furthermore, the MES

[10]`https://github.com/securityscorecard/SSC-Threat-Intel-IoCs/blob/master/KillNet-DDoS-Blocklist/ipblocklist.txt`

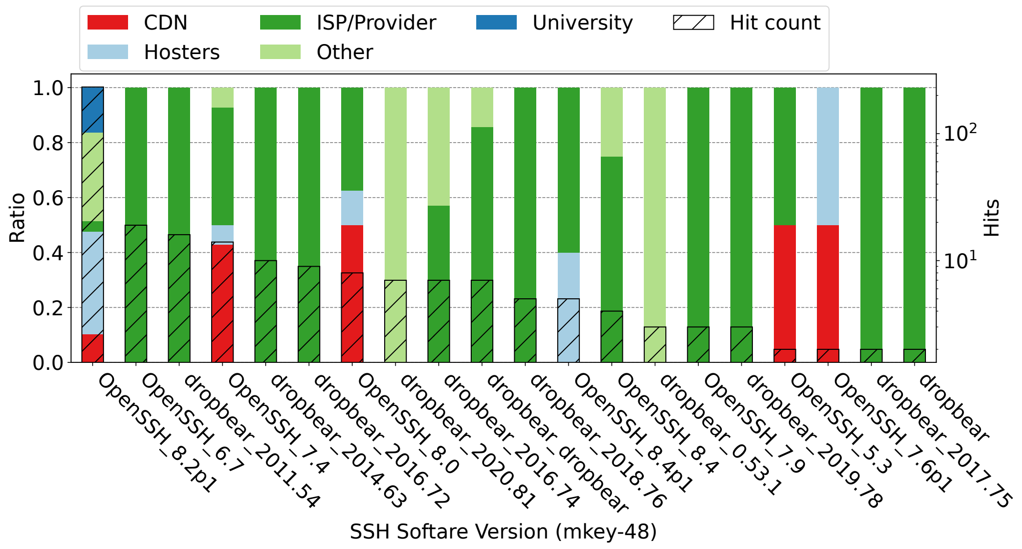


Figure 6.11: *MK48*: SSH server and AS type, port 22

and the organization behind KillNet may be collaborating, for example, by creating and maintaining censorship evasion honeypots to identify citizens trying to attain uncensored information as indicated above. However, as before, a conclusive assessment is not possible without further forensic evidence from affected machines.

Summary: Using our methodology, we could illustrate the operation of several attributed malware families. Furthermore, we were able to identify, e.g., code repositories as well as coincidences of malicious key co-location. Given further forensic evidence, these observations may allow conclusive attribution of specific groups to countries or regions of origin.

Overall our scan identified 41 out of 52 targeted malicious SSH keys, revealing that these keys are concentrated in ISP and CDN ASes, with certain keys linked to specific botnets like ‘fritzfrog’ and ‘muhstick’ or actors like ‘teamtnt’, indicating targeted attack campaigns within these network environments.

6.5 Port 2222 Findings

The scan of non-standard SSH port 2222 revealed a significantly lower number of servers and subsequently identified compromised hosts compared to the standard SSH port 22. Among the 643k ± 5.7k servers confirmed as running SSH on port tcp/2222, we find 547 hits corresponding to 480 unique IP addresses and 395 unique server host keys. Again, this indicates that some of the identified servers are likely configured with multiple IP addresses or that there are shared configurations across different servers. For IPv6, we only observed 8 hits for port tcp/2222.

Furthermore, we find 10 compromised IP addresses running SSH on both port 22 and port 2222, with the same server host key present on both ports. This suggests a possible configuration pattern where certain servers are accessible through multiple SSH ports, e.g., during a transition phase to an off-port, or due to a misconfiguration like not commenting out the standard ssh port. In addition to these ten cases, we identified

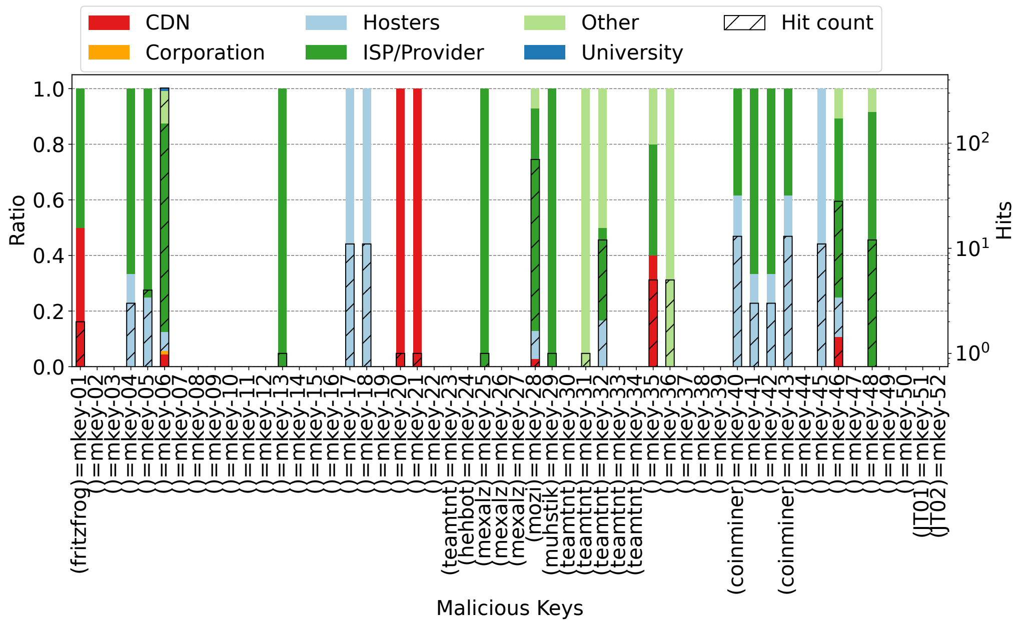


Figure 6.12: Malicious key vs. AS type, port 2222

another 15 IP addresses that share a server host key with IPs detected in the port 22 scan, again, likely due to IP churn.

The results from the port 2222 scan, despite containing less hosts are largely consistent with those from the port 22 scan with a few notable exceptions. Apart from the–naturally–generally smaller number of different keys found, we actually observe a key–*MK21*–*only* on port tcp/2222. The affected machine was running in Amazon EC2 and has since been disconnected. This was the only reported hit for *MK21* in our data. Additionally, we find less compromised IoT devices and systems in end-user ISPs on port tcp/2222, see Figure 6.12. We attribute this to IoT devices usually using standard ports, while using an off-port commonly requires manual intervention.

Summary: Overall, the data suggests that while port 2222 is less commonly used for SSH than port 22, using an off-port for SSH does not necessarily prevent compromises.

6.6 Notable Events

In this section, we describe several notable circumstances during our study, reaching beyond the collected data itself.

Opt-Back Hoster: A major European hoster has been opted out of measurements from our group for several years. As such, we initially did not plan to include this hoster, who currently holds around 2.8 million IPv4 addresses and–likely–a corresponding amount of VMs and dedicated servers.

However, given that this hoster is, due to the high number of end-user controlled systems, likely to also harbor a large amount of potentially compromised systems, we reached out to the hoster to obtain explicit consent. After a brief discussion, explicit consent was granted.

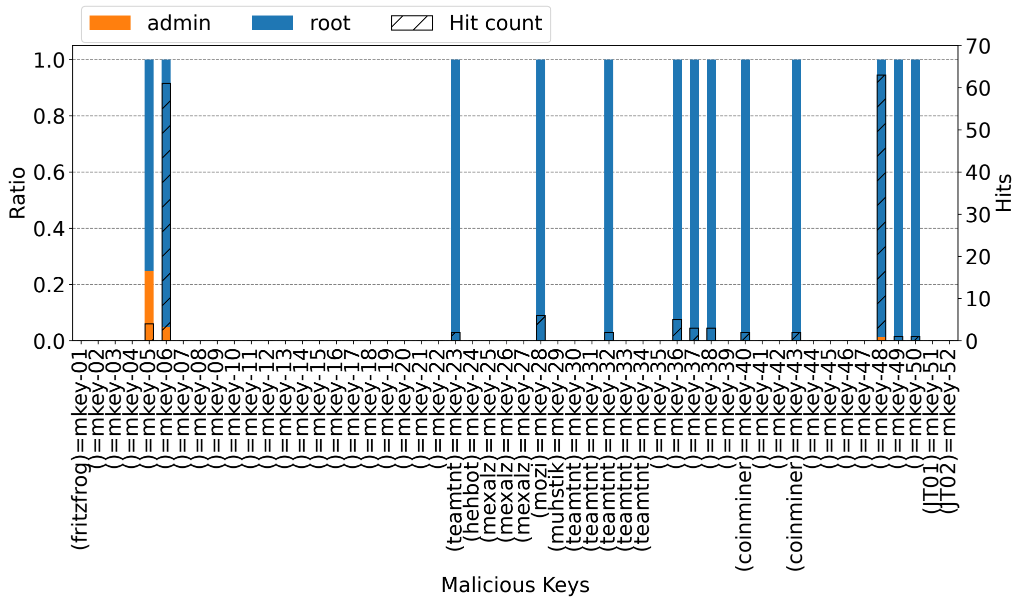


Figure 6.13: European Hoster case: Malicious key distribution on port 22. First scan.

In our subsequent measurements, we find that almost 1% of all compromised hosts identified by our methodology are hosted in the networks of this hoster. Notably, the identified compromised keys are dominated by two distinct keys: *MK06*, which is the generally most prevalent key, and *MK48* (see Figure 6.13). Note that *MK48* is the key related to the MES in Section 6.4. Furthermore, especially the high prevalence of *MK48* in this specific hoster triggered our deeper investigation of hosts found to have this key. Hence, we likely would not have detected the MES case without the hoster opting back in.

As part of our continuous scanning efforts, we revisited the set of compromised hosts belonging to the hosting provider. Figure 6.14 shows the distribution of detected malicious keys within the provider's network. It is evident that our notifications reached the system administrators, and appropriate remediation actions were taken. Some keys, such as *MK06*, have been completely removed and are no longer present on any of the hosts. Others, like *MK48*, still persist but in significantly reduced numbers. Overall, the number of compromised hosts has decreased from 144 to 31. This clearly indicates that our scans and subsequent notifications had a positive impact on cleaning up compromised systems within the network.

GitHub Key Identification: Following our observations on `gitee.com`, we also verified our full key list against `github.com`. Unlike with `gitee.com`, GitHub's SSH implementation does not trigger challenges on any installed key regardless of the username. Hence, we ran a single test for all 52 keys against a GitHub SSH endpoint with the username `git`.

In total, 5 keys (*MK05*, *MK17*, *MK36*, *MK51* and *MK52*) were found to be installed, with the latter two being expected to be present, as the keys have not been removed after the account was disabled. Notably, *MK36*, might be related to 'teamtnt', see Section 6.4. Apart from GitHub, these keys can be found on 328 compromised hosts world-wide. The information was shared for mitigation with a GitHub contact.

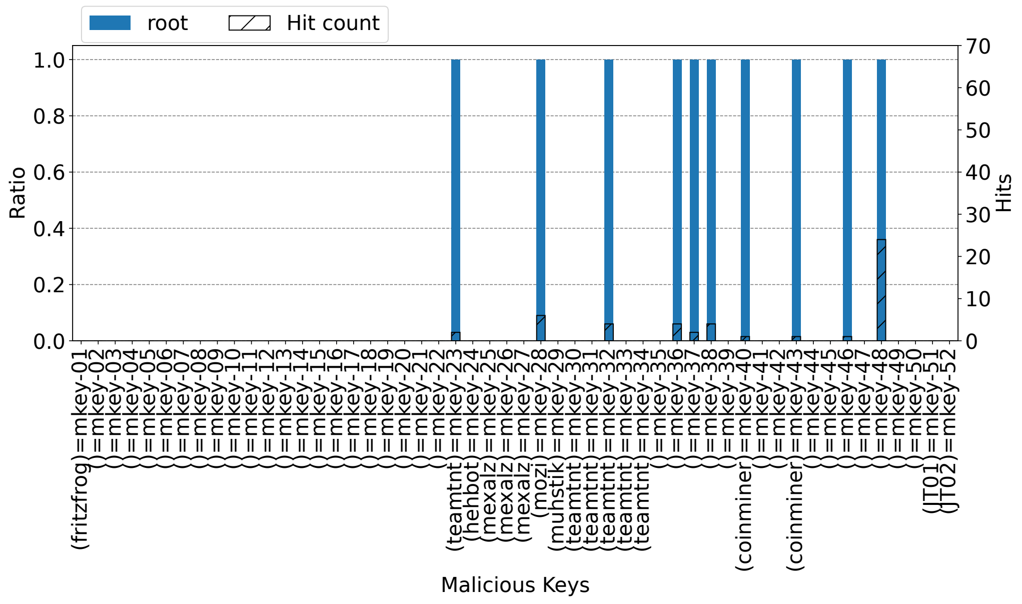


Figure 6.14: European Hoster case: Malicious key distribution on port 22. Second scan.

Compromised Internet Core Router: When cross-referencing our results with data on Internet core routers from a study going on in parallel, we found 66 IPv4 addresses that are *both* compromised and have port tcp/179, i.e., BGP, open to the Internet. After examining the server host keys, we found that these correspond to 19 different hosts. 18 of these were non-critical edge devices, e.g., exposed Mikrotik routers that do not necessarily participate in the Internet core.

However, one device was one of the core-routers of a leading ISP in a large central European country, where we detected *MK06*, i.e., a potential compromise by a nation-state associated attack group. In fact, the device in question was between one and two network hops away from the ASes border when attempting to reach hosts within the network. We reached out to a contact with another major ISP of which the affected one is a subsidiary. The contact relayed the information, and incident investigation was initiated. Due to the critical nature of this procedure, no information was communicated back to the status of the incident investigation.

We also cross-referenced the compromised host IPs against the dataset of GCA honeyfarm. During the period of our scans, we identified 26 IPs that appeared in the honeynet dataset. Of these, 24 IPs were observed connecting to the honeypots with malicious intent, executing commands, and downloading 'coinminers', 'bots', or 'malware'. The remaining 2 IPs were not used to connect to the honeypots but served as storage locations for malicious files.

Summary: Our measurements identified additional attacker activity on GitHub, highlighted the impact large networks can have when opting out of measurements, and led to the clean-up of compromised Internet core infrastructure.

6.7 Discussion

Here, we further discuss and contextualize implications of our findings and responses we received during our measurements.

6.7.1 Internet Measurement Opt-Out and Ethics

In Section 6.6, we discuss the case of a hoster who had previously opted out of measurements from our group. However, after getting in contact with the hoster, we were able to convince them to opt back in.

In our results, around 1% of compromised hosts is located within this hoster. Furthermore, we might have missed events around *MK48*, which was noticeably pronounced in this hoster. Hence, this single hoster not being included could have significantly changed our findings. Furthermore, missing detection and remediation of these systems *could* have been causing harm to other Internet participants.

Current discussions around ethical feasibility of Internet scanning usually circle around an implicit assumption that there are no negative implications, i.e., *potential to cause harm*, by opting out [36]. However, this case, we argue, demonstrates that this is not *always* the case. This highlights an interesting tension for assessing ethical feasibility:

- Depending on a measurement's purpose, an operator opting out may, unintentionally, occlude malicious activity. This may not only impact research results, but also–as in this case–led to *potential harm* for third parties.
- Researchers usually are, in all likelihood, convinced that their measurements, especially security focused ones, are uniquely important, and will be more inclined to *assume* that opt-out by operators may hold potential for harm.
- Operators may be skeptical of network measurements, and hence prefer to opt out over 'enduring' measurements, including those preventing *them* causing harm.

While we believe that this discussion–as also greatly advanced by Kohno et al. [216]–is already ongoing, we note a considerable uni-polar perspective on *researcher's* actions here. However, researcher's action can not necessarily be seen in a vacuum, and instead the responsibility of other parties, e.g., in the context of Internet scanning, needs to be more thoroughly considered by the community when evaluating ethics.

6.7.2 Legal Challenges

Given that our scans, even though *they were not*, did cause the *impression* of brute-force attacks in system logs, we also received several more stern messages concerning our measurements, partially involving threats of legal action. However, consulting with legal scholars for our institution's jurisdiction, we learned that there is a limited exploration of these topics within the legal scripture. This mostly relates to the issue that, at least within our jurisdiction, cases related to circumstances as above rarely become sufficiently publicly documented, as most end in private litigation.

However, this creates a realm of legal uncertainty for researchers. Given the underlying potential benefits, see Section 5.2.7, we argue that there is a need for official structures to evaluate and support such research, possibly in collaboration with, e.g., law enforcement or national security centers [217, 218].

6.7.3 Feedback on IPv6 Measurements

During our measurements, we received several opt-out requests, as well as automated and manual complaints. A few noteworthy cases emerged during our IPv6 scans. In one instance, we received an abuse report for a host where the logs only showed IPv4 addresses. This was due to the log analyzer's inability to process IPv6 addresses.

Moreover, we encountered several instances where we received reports related to our IPv6 scanning activity, particularly from network administrators who included log messages that resembled those generated by routers from vendors like Cisco and Juniper. Interestingly, these reports often came from parties that did not experience any IPv4 traffic from our scans on those–apparently–network infrastructure systems. This indicates that their IPv6 firewall configurations were more permissive than their IPv4 counterparts, as we would have otherwise probed the associated IPv4 addresses as well. This observation suggests that IPv6 networks may sometimes be less stringently secured, potentially exposing critical infrastructure to increased risks, aligning with observations by Czyz et al. [219].

These experiences highlight the importance of robust configuration and monitoring practices in IPv6 networks, especially when NAT64 gateways and firewall settings are involved. Network administrators should ensure that their IPv6 configurations are as secure and well-managed as their IPv4 counterparts to prevent unintended consequences during scanning activities. Additionally, clear communication between network operators and external scanning entities is crucial to mitigate potential issues and misunderstandings.

In another case, our scans unintentionally generated a high number of requests per second to a couple of hosts, which were later identified as NAT64 gateways. These gateways were translating IPv6 traffic into IPv4, leading to an unexpectedly high load on the IPv4 servers behind them. This incident underscores the importance of carefully managing NAT64 configurations to avoid overloading servers when large-scale IPv6 scans are conducted.

6.7.4 Evasion Tactics and Scalability Challenges

Attackers could potentially install a non-root or non-admin SSH key to an account (with sudo capabilities) that they create, as a means to evade detection. Similarly, they could use one unique key per host or patch the SSH server to always send a challenge. While these tactics are theoretically feasible, they pose significant logistical challenges. Attackers would need to meticulously track user/key combinations for each compromised host, adding complexity and scalability issues. As observed in our ongoing research, attackers typically aim to make compromised systems as fungible as possible to streamline their operations. Additionally, if attackers switch to a non-unique user, it can be quickly detected and added to our scan list.

Patching SSH servers presents similar difficulties. Attackers would need to reliably patch a diverse range of software across various platforms without causing system failures, a task that requires significant effort and precision.

6.8 Summary

In this chapter, we present the results of an Internet-scale scan to identify compromised SSH servers leveraging SSH's behavior of only sending a challenge for installed public keys during authentication.

Using our method, we find more than 21,700 compromised accounts on systems in 144 countries, compromised by at least one of the 52 verified malicious keys provided by a threat intelligence company. Our results uncovered compromised servers in critical infrastructure such as core Internet routers of a European ISP, and allowed us to uniquely characterize behavior of known threat actors, like 'teamtnt', or 'KillNet'.

In addition to one-time notifications for the data presented in this study, we implemented continuous scanning with automated reporting via our national CSIRT and the Shadowserver Foundation for the foreseeable future, to ensure that compromised machines–as well as potentially threat-actor *operated* systems can be remediated. We have also released our scan-tool as an Artifact [39] and collaborate with the CSIRT and threat intelligence community on further remediation of compromised systems.

Chapter 7 Conclusion

The topic of this thesis is to analyze, investigate, and evaluate current landscape of SSH attacks on the Internet. As such, we tackle four research questions, with the goal of investigating how has SSH attack activity changed over time, and how compromised hosts can be effectively detected through non-intrusive Internet measurements at scale.

7.1 Summary

We start by analyzing the traffic of a large honeyfarm that has collected SSH attacks for almost 3 years. This helps us to answer the first question:

Can we understand unwanted SSH traffic on a global scale?

We analyze data collected from a newly deployed honeyfarm composed of 221 honeypots distributed across 55 countries and 64 networks, operated continuously over a period of 32 months. Our analysis, performed both on individual honeypots and the honeyfarm as a whole, reveals substantial variation in unsolicited and unwanted Internet activity. By taxonomizing the incoming connections, we identify diverse scanning, scouting, and intrusion behaviors. Interestingly, honeypots that collect most files do not necessarily receive the most traffic overall, but they tend to detect new hashes earlier. These findings can guide the strategic deployment of honeypots based on specific goals, such as maximizing early malware detection or increasing visibility of scanning activity. We also observed that although attackers generate a large variety of files, no single honeypot recorded more than 5% of all unique files, highlighting the importance of scale and geographical diversity in capturing rare and potentially more sophisticated intrusions.

Furthermore, many attacks persisted over long periods, suggesting that they could be detected with relatively simple monitoring. Some attacks consistently targeted a small subset of honeypots using only a few client IPs—making them potentially easy to disrupt—while others leveraged large botnets with extensive IP distribution, contacting nearly all honeypots. These large-scale, distributed attacks are harder to mitigate but offer valuable insights for tracking and analyzing botnet activity. Together, these observations underscore the critical role of global, long-term honeypot deployments in understanding the dynamics of unsolicited Internet traffic.

Naturally, SSH attacks and intruder behavior are not static–they evolve over time. To capture these temporal dynamics, we conducted a longitudinal analysis of the honeyfarm data to address our second research question:

How does attacker behavior change over extended periods?

To answer this question, we performed an in-depth longitudinal analysis of SSH-based attacks over a three-year period. By analyzing intrusive sessions, command execution patterns, malicious file transfers, and the underlying infrastructure, we observed notable changes in attacker tactics–such as increased reconnaissance activity and shifts in methods used to deploy files onto compromised systems. Our analysis also uncovered a growing use of newly registered ASes for hosting malicious content, along with evidence that some attackers are capable of identifying and even misusing honeypots for their own ends.

These findings underscore the need to continuously adapt defensive strategies in response to the evolving nature of attacks. The study highlights the value of ongoing monitoring and detailed analysis to detect emerging trends and behavioral shifts in malicious activity. While the integration of various data sources proves effective in enhancing visibility into attacker behavior, deeper collaboration and continued research are essential to further strengthen defensive capabilities in an increasingly dynamic threat environment.

With a clearer understanding of SSH intruder behavior, we shift toward a more proactive analytical approach. Leveraging the design of the SSH protocol, we explore a way to address our third research question:

Can we design a non-intrusive methodology to identify compromised hosts at scale?

By exploiting a characteristic of the SSH protocol–specifically, that servers only issue authentication challenges for public keys that are already installed–we use an Internet-scale method for identifying compromised SSH servers. Our approach does not require privileged access to target systems and does not grant us entry into compromised machines, thereby minimizing ethical concerns. We began by conducting a series of controlled in-lab experiments across a variety of SSH server implementations and versions. These tests confirmed that our tool performs reliably under diverse conditions. To further verify its accuracy, we compared the results of our initial port scans with publicly available data from Censys, which reinforced the reliability of our scanning methodology.

Building on these results, we performed a small-scale scan on a real network, which enabled us to improve the toolchain. This phase also allowed us to fine-tune the scan rate to strike a balance between collecting useful data and minimizing disruption for system administrators. In collaboration with a threat intelligence partner, we assembled a list of known malicious public keys and frequently targeted usernames to guide our broader scanning effort. We then reflect on the ethical considerations of conducting such scans, emphasizing their contribution to Internet security research. Lastly, we detail the deployment of our full-scale, continuous scanning operation based on the developed methodology.

With the toolchain in place, we now turn to our fourth and final research question:

Can we successfully identify and characterize compromised SSH hosts in the wild?

We present the results of an Internet-scale scan designed to identify compromised SSH servers by leveraging the protocol's behavior of only sending authentication challenges for installed public keys. Applying this method, we discovered over 21,700 compromised accounts across systems in 144 countries, each affected by at least one of 52 verified malicious keys provided by a trusted threat intelligence partner. Our findings revealed compromised systems within critical infrastructure, including core Internet routers operated by an European ISP, and enabled us to uniquely profile the behavior of known threat actors such as `teamtnt` and `KillNet`.

Beyond the initial, one-time notifications related to these findings, we established a continuous scanning framework with automated reporting through our national CSIRT and the Shadowserver Foundation. This ongoing effort aims to support the timely remediation of both compromised machines and those potentially operated by threat actors. Additionally, we have released our scanning tool as an open-source artifact [39] and continue to collaborate closely with CSIRTs and the broader threat intelligence community to facilitate long-term mitigation and response.

7.2 Future Directions

In this section, we discuss possible future directions of our work.

A New Generation of Honeypots. As shown in Chapter 4, attacker behavior has evolved toward more stealthy and deliberate techniques that increasingly bypass detection by commonly used honeypot implementations. These legacy systems often lack critical capabilities such as support for advanced command emulation, state preservation across sessions, and awareness of known default credentials. These limitations significantly hinder not only the detection of sophisticated attacks but also the ability to infer the true intent and strategy of the intruders.

Given this, we argue that a new generation of honeypots is needed. These updated systems should be capable of emulating a broader range of commands frequently used by attackers–such as *rsync* for file transfer–and should maintain state over time to account for delayed execution behavior, a pattern we observed in which malicious files are uploaded but not immediately executed. Stateless honeypots are becoming less effective in revealing attacker motivations and methodologies. As the offensive landscape becomes more refined, our defensive tools must also evolve. This ongoing arms race necessitates continuous improvement to avoid falling behind.

Expanding Proactive Scanning. Our Internet-scale scans, discussed in Chapters 5 and 6, reveal the value of proactive scanning in identifying compromised systems. However, our findings also suggest untapped potential in enriching this approach. Once a host is flagged as compromised, follow-up scans can be tailored to collect more detailed information about the system. This includes port scanning to discover active services, fingerprinting the underlying operating system, identifying the device type, and potentially even the manufacturer.

Such extended profiling will allow us to build detailed behavioral and structural fingerprints of compromised systems. This will not only improve our understanding of the types of targets attackers prefer but also help predict future targets based on observed characteristics. A more intelligent and responsive scanning strategy–triggered by initial compromise detection–can turn reactive security into a proactive tool for threat anticipation.

Enhancing Attribution Through Collaboration. As discussed in Chapters 4 and 6, correlating data from multiple sources significantly enriches the analytical value of our findings. One of the most challenging aspects of threat intelligence remains attribution–identifying the actors behind specific attacks. Attribution requires assembling many pieces of a complex puzzle, often spanning behavioral patterns, infrastructure use, and forensic traces.

Although our standalone analysis may not be sufficient to reliably attribute attacks, there is strong potential for collaborative efforts to bridge this gap. Combining our work with malware behavioral analysis and threat intelligence feeds from security companies offers a promising path forward. These partnerships can help fill in missing links by cross-referencing file hashes, infrastructure details, and attacker signatures with broader threat intelligence databases. In doing so, we can move closer to not only identifying attackers but also understanding their motivations, capabilities, and long-term strategies.

Appendix A
Commands classification Table

Table A.1 presents the list of labels and regex rules that were used to categorize the sessions based on commands.

Label	Regular Expresion
mdrfkr	r"mdrfckr"
echo_ok	r"\\\\x6F\\\\x6B"
echo_ok_txt	r"echo ok"
echo_ssh_check	r"SSH check"
echo_os_check	r"\becho\b\s+[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}"
uname_a	r"'uname\s+-a'"
uname_svnrm	r"'uname\s+-s\s+-v\s+-n\s+-r\s+-m'"
uname_svnr	r"'uname\s+-s\s+-v\s+-n\s+-r'"
uname_a_nproc	^(?=.*nproc)(?=.*\buname\s+-a\b).*
uname_snri_nproc	^(?=.*nproc)(?=.*\buname\s+-s\s+-n\s+-r\s+-i\b).*
bbox_5_char_v2	^(?=.*\/bin\/busybox\s+([a-zA-Z0-9]{5}\'))(?=.*tftp;\s+wget).*
bbox_scout_cat	r"/bin/busybox\s+cat\s+/proc/self/exe\s*\\|\\|\s*cat\s+/proc/self/exe"
bbox_loaderwget	r"loader\.wget"
bbox_echo_elf	r"\\\\x45\\\\x4c\\\\x46'"
bbox_unlabelled	/bin/busybox\s|busybox\s
juicessh	r"juicessh"
passwd123_daemon	^(?=.*Password123)(?=.*daemon).*
pattern_7	r"cd\s+/tmp\s*;\s*rm\s+-rf\s+/tmp/\*\s*\\|\\|\s+cd\s+/var/run\s*\\|\\|\s+cd\s+/mnt\s*\\|\\|\s+cd\s+/root\s*;\s*rm\s+-rf\s+/root/\*\s*\\|\\|\s+cd\s+/"
rapperbot	r"ssh-rsa\s+AAAAB3NzaC1yc2EAAAADAQABA"
root_17_char_pwd	r"root:[A-Za-z0-9]{15,}\\|chpasswd"
pattern_5	^(?=.*rm\s+-rf\s+\*;\s*cd\s+/tmp\s*;\s*rm\s+-rf\s+\*)(?=.*xoxox0\\\\nxoxox0).*
curl_maxred	r"–max-redir"
lenni_0451	r"lenni0451"
binx86	^(?=.*CPU\(s\):)(?=.*bin\.x86_64).*
export_vei	r"export VEI"
clamav	r"\bclamav\b"
g*****_echo	r"\\\\x67\\\\x61\\\\x79\\\\x66\\\\x67\\\\x74"
dget_4	^(?=.*wget\s+-4)(?=.*dget\s+-4).*
openssl_passwd	r"openssl passwd -1 \S{8}"
cloud_print	r"cloud\s+print"
shell_fp	r"(?=.*\$\bSHELL\b)(?=.*bs=22)"
root_12_char_capscout	r"(?=.*root:[A-Za-z0-9]{12})(?=.*awk\s+'{print\s+\$4,\$5,\$6,\$7,\$8,\$9;}')"
perl_dred_miner	r"(?=.*perl)(?=.*dred)"
stx_miner	r"(?=.*stx)(?=.*LC_ALL)"
f******_attack	r"fuckjewishpeople"
ohshit_attack	r"ohshit"
onions_attack	r"onions1337"
sora_attack	r"sora"
heisen_attack	r"Heisenberg"
zeus_attack	r"Zeus"
update_attack	r"update\.sh"
ak47_scout	r"(?=.*\\\\x41\\\\x4b\\\\x34\\\\x37)(?=.*writable)"
uname_svnr	r"(?=.*'uname\s+-s\s+-v\s+-n\s+-r')(?=.*model\s+name)"
gen_curl_wget	r"(?=.*curl)(?=.*wget)"
gen_echo	r"(?=.*echo)"
gen_echo_ftp_wget	r"(?=.*echo)(?=.*ftp)(?=.*wget)"
gen_echo_wget	r"(?=.*echo)(?=.*wget)"
gen_ftp	r"(?=.*ftp)"
gen_wget	r"(?=.*wget)"
gen_curl_echo_wget	r"(?=.*curl)(?=.*echo)(?=.*wget)"
gen_echo_ftp	r"(?=.*echo)(?=.*ftp)"
gen_curl_ftp	r"(?=.*curl)(?=.*ftp)"
gen_curl_ftp_wget	r"(?=.*curl)(?=.*ftp)(?=.*wget)"
gen_ftp_wget	r"(?=.*ftp)(?=.*wget)"
gen_curl_echo_ftp_wget	r"(?=.*curl)(?=.*echo)(?=.*ftp)(?=.*wget)"
gen_curl	r"(?=.*curl)"
gen_curl_echo	r"(?=.*curl)(?=.*echo)"

Table A.1: Regular expressions used to categorise the commands. For reproducibility, slurs used in regular expressions are not redacted here, see Section 4.2. Reader discretion is advised.

List of Figures

3.1 Honeyfarm analysis: Presence of 221 Cowrie honeypots in 55 countries. . 18

3.2 Honeyfarm analysis: Number of sessions collected by each Cowrie honeypot in honeyfarm. 18

3.3 Honeyfarm analysis: Activity across time for top 5% of Cowrie honeypots with most sessions. 19

3.4 Honeyfarm analysis: Activity across time for all Cowrie honeypots. 19

3.5 Honeyfarm analysis: Session classification diagram. 20

3.6 Honeyfarm analysis: Number of sessions per session type over time. . . . 22

3.7 Honeyfarm analysis: ECDF of session duration by session type. 22

3.8 Honeyfarm analysis: Activity split by session type across time showing the number of sessions per honeypot. 23

3.9 Honeyfarm analysis: Activity by session type across time for the top 5% honeypots showing the number of sessions per honeypot. 24

3.10 Honeyfarm analysis: World map illustrating the geographic distribution of client IPs that established connections with the honeypots. 26

3.11 Honeyfarm analysis: Number of unique client IPs connecting to the honeyfarm per session type across time. 26

3.12 Honeyfarm analysis: World map illustrating the distribution of client IPs that established connections with the honeypots split by session type. . . . 27

3.13 Honeyfarm analysis: ECDF of number of honeypots contacted per client IP by session type. 28

3.14 Honeyfarm analysis: ECDF of number of days that a client IP is observed by session type. 29

3.15 Honeyfarm analysis: Number of unique clients connecting to the honeyfarm split by session type per honeypot. 30

3.16 Honeyfarm analysis: Number of clients connecting to the honeyfarm per combinations of session types across time. 30

3.17 Honeyfarm analysis: Regional diversity of the client vs. the honeypot they connect to across time. 31

3.18 Honeyfarm analysis: Regional diversity of the client vs. the honeypot they connect to across time. Separate by session type. 32

3.19 Honeyfarm analysis: Activity of the hashes collected by the honeyfarm across time and fraction of fresh (newly seen) hashes. 36

3.20 Honeyfarm analysis: Number of unique hashes and client IPs seen per honeypot. 37

3.21 Honeyfarm analysis: Number of unique hashes and sessions seen per honeypot. 38
3.22 Honeyfarm analysis: Distribution of unique hashes over client IPs and unique client IPs over hashes. 38
3.23 Honeyfarm analysis: ECDF of attack campaign duration per different attack type. 39
4.1 Attack characterization: Distribution of command sessions that change and don't change the state of the honeypot across time. 45
4.2 Attack characterization: Normalized DLD Matrix heatmap of the bot categories. 46
4.3 Attack characterization: Distribution of sessions that do not change the state of the honeypot across time. 47
4.4 Attack characterization: Distribution of sessions that do change the state of the honeypot across time. 48
4.5 Attack characterization: Snippet of a code containing the curl with "cookie attack". 49
4.6 Attack characterization: Distribution of sessions that attempt to execute files on the honeypot across time. 50
4.7 Attack characterization: Distribution of sessions that attempt to execute files separately for the cases when the files are recorded and not recorded on the honeypot. 51
4.8 Attack characterization: Heatmap of the bot categories and file types. . . . 53
4.9 Attack characterization: Distribution of the Mirai related bots over time. . 54
4.10 Attack characterization: Mirai related bot categories over time. 55
4.11 Attack characterization: Sankey plot # of client IPs by AS type vs. # malware storage IP by AS type. 56
4.12 Attack characterization: AS age and size analysis of malware storage locations. 57
4.13 Attack characterization: AS Types of malware storage location seen over time. 58
4.14 Attack characterization: Malware storage location activity recorded over time. 59
4.15 Attack characterization: Sessions that record intrusions for username 'root' using one of the Top 5 passwords recorded by the honeyfarm. 60
4.16 Attack characterization: Login attempts using default Cowrie usernames, "richard" and "phil", observed over time. 61
4.17 Attack case study: Behavior change in the "*mdrfckr*" bots and correlation to "3245gs5662d34" credential attack. 63
4.18 Attack case study: A temporal view of all sessions involving "*mdrfckr*" actor recorded by the honeyfarm. 64
5.1 SSH Scan: Step-by-Step SSH authentication diagram. 68
5.2 SSH Scan: Activity periods of malicious key recorded by the threat intelligence company. 73

5.3 SSH Scan: Comparison between Censys and our Scan on port 22. 76
6.1 Compromised Hosts: SSH server implementation vs. username, port 22 . 79
6.2 Compromised Hosts: AS type vs. username, port 22. 80
6.3 Compromised Hosts: SSH server implementation vs. AS type, port 22. . . 81
6.4 Compromised Hosts: Top 20 SSH server versions vs. AS type, port 22. . . 81
6.5 Compromised Hosts: Global distribution, port 22. 82
6.6 Compromised Hosts: Top 30 countries based on number of compromised servers vs. AS type, port 22. 83
6.7 Compromised Hosts: Malicious key count, port 22. 84
6.8 Compromised Hosts: Malicious key vs. SSH server implementation, port 22. 85
6.9 Compromised Hosts: Malicious key vs. AS type, port 22. 86
6.10 Compromised Hosts: Malicious key vs. compromised user, port 22. 87
6.11 Compromised Hosts with *MK48*: SSH server implementation and AS type, port 22. 88
6.12 Compromised Hosts: Malicious key vs. AS type, port 2222. 89
6.13 European Hoster case: Malicious key distribution on port 22. First scan. . 90
6.14 European Hoster case: Malicious key distribution on port 22. Second scan. 91

List of Tables

2.1 Background: Comparison of Interaction Type, Scope, and Time Analysis Across Related Studies 15
3.1 Honeyfarm analysis: Percentage of all sessions per session type and per protocol (SSH/Telnet). 21
3.2 Honeyfarm analysis: Top 10 most used passwords for username 'root'. . . 22
3.3 Honeyfarm analysis: Top 20 commands collected by the honeyfarm. . . . 33
3.4 Honeyfarm analysis: Top 20 hashes collected by the honeyfarm, sorted by the number of sessions. 34
3.5 Honeyfarm analysis: Top 20 hashes collected by the honeyfarm sorted by the number of client IPs. 35
3.6 Honeyfarm analysis: Top 20 hashes collected by the honeyfarm sorted by the number of active days. 35
5.1 SSH scan: Tool testing across multiple SSH server implementations. 71
A.1 Attack characterization: Regular expressions used to categorise the commands collected by the honeyfarm. 100

Bibliography

[1] Tatu Ylonen. SSH–secure login connections over the Internet. In *USENIX Security Symposium*, 1996.

[2] Z. Durumeric, D. Adrian, A. Mirian, M.Bailey, and J. A. Halderman. A search engine backed by Internet-wide scanning. In *ACM CCS*, 2015.

[3] M. Solomon Zemene and P.S. Avadhani. Implementing high interaction honeypot to study SSH attacks. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1898–1903, 2015.

[4] Pratibha Khandait, Namrata Tiwari, and Neminath Hubballi. Who is trying to compromise your SSH server? An analysis of authentication logs and detection of bruteforce attacks. In *Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking*, pages 127–132, 2021.

[5] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the Mirai Botnet. In *USENIX Security Symposium*, 2017.

[6] Harm Griffioen, Georgios Koursiounis, Georgios Smaragdakis, and Christian Doerr. Have you SYN me? characterizing ten years of Internet scanning. In *ACM Internet Measurement Conference*, 2024.

[7] Jan Vykopal. Flow-based brute-force attack detection in large and high-speed networks. *Masaryk University (Brno, Czech Republic), PhD Thesis*, 2013.

[8] Jan Vykopal, Tomas Plesnik, and Pavel Minarik. Network-Based Dictionary Attack Detection. In *2009 International Conference on Future Networks*, pages 23–27, 2009.

[9] Martin R Albrecht, Kenneth G Paterson, and Gaven J Watson. Plaintext recovery attacks against SSH. In *IEEE Symposium on Security and Privacy*, pages 16–26, 2009.

[10] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on {SSH}. In *USENIX Security Symposium*, 2001.

[11] Fabian Bäumer, Marcus Brinkmann, and Jörg Schwenk. Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 7463–7480, 2024.

[12] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Longman Publishing Co., Inc., USA, 2002.

[13] Iyatiti Mokube and Michele Adams. Honeypots: concepts, approaches, and challenges. In *ACMSE*, New York, NY, USA, 2007. ACM.

[14] Lance Spitzner. The Honeynet Project: trapping the hackers. *IEEE Security & Privacy*, 1(2):15–23, 2003.

[15] Niels Provos. A Virtual Honeypot Framework. In *USENIX Security Symposium*, 2004.

[16] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. *NDSS*, 2014.

[17] Marcin Nawrocki, John Kristoff, Raphael Hiesgen, Chris Kanich, Thomas C. Schmidt, and Matthias Wählisch. SoK: A Data-driven View on Methods to Detect Reflective Amplification DDoS Attacks Using Honeypots. In *IEEE Euro S&P*, 2023.

[18] Johannes Krupp, Michael Backes, and Christian Rossow. Identifying the Scan and Attack Infrastructures Behind Amplification DDoS Attacks. In *ACM CCS*, 2016.

[19] Marcin Nawrocki, Matthias Wählisch, Thomas C. Schmidt, Christian Keil, and Jochen Schönfelder. A Survey on Honeypot Software and Data Analysis. *CoRR*, 2016.

[20] Harm Griffioen and Christian Doerr. Examining Mirai's Battle over the Internet of Things. In *ACM CCS*, 2020.

[21] Niels Provos. Developments of the Honeyd Virtual Honeypot. `https://www.honeyd.org/`. Last accessed: 2025-06-26.

[22] Cowrie. Cowrie on GitHub. `https://github.com/cowrie/cowrie`. Last accessed: 2025-06-26.

[23] P. Baecher, M. Koetter, and G. Wicherski. Nepenthes on GitHub. `https://github.com/jrwren/nepenthes`. Last accessed: 2025-06-26.

[24] DutchSec B.V. Honeytrap on GitHub. `https://github.com/honeytrap/honeytrap`. Last accessed: 2025-06-26.

[25] Justin Varner. Contextualize honeypot alerts automatically with GreyNoise, runZero, Thinkst Canary, and Tines. `https://www.runzero.com/blog/contextualize-honeypot-alerts/`. Last accessed: 2025-06-26.

[26] NetScout ASERT Team. Dipping Into The Honeypot. `https://www.netscout.com/blog/asert/dipping-honeypot`. Last accessed: 2025-06-26.

[27] Global Cyber Alliance. GCA AIDE – Automated IoT Defense Ecosystem. `https://www.globalcyberalliance.org/`. Last accessed: 2025-06-26.

[28] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, page 635–647, New York, NY, USA, 2009. Association for Computing Machinery.

[29] Timothy Barron and Nick Nikiforakis. Picky attackers: Quantifying the role of system properties on intruder behavior. In *Annual Computer Security Applications Conference*, 2017.

[30] Tobias Fiebig. Getting back at Trudy: SSH Botnet Member Credential Collection using Connect Back Honeypots. 2013. Universiteit van Amsterdam.

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

[32] Antonio Villalón-Huerta, Hector Marco-Gisbert, and Ismael Ripoll-Ripoll. A taxonomy for threat actors' persistence techniques. *Computers & Security*, 121:102855, 2022.

[33] Artem Golubin. Public SSH keys can leak your private infrastructure. `https://web.archive.org/web/20240817101658/https://rushter.com/blog/public-ssh-keys/`. Last accessed: 2025-06-26.

[34] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Authentication Protocol. RFC 4252, IETF, Jan 2006.

[35] Manfred Kaiser. CVE-2016-20012 - Publickey Information leak - Only allow SSH_MSG_USERAUTH_REQUEST with signature #270. `https://web.archive.org/web/20240723013154/https://github.com/openssh/openssh-portable/pull/270`. Last accessed: 2025-06-26.

[36] D. Dittrich, E. Kenneally, et al. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. *U.S. Department of Homeland Security*, 2012.

[37] Bitdefender. Bitdefender. `https://www.bitdefender.com/`. Last accessed: 2025-06-26.

[38] The Shadow Server Foundation. Compromised SSH Host Special Report. `https://www.shadowserver.org/what-we-do/network-reporting/compromised-ssh-host-special/`. Last accessed: 2025-06-26.

[39] Cristian Munteanu. Artifact: Catch-22: Uncovering Compromised Hosts using SSH Public Keys. `https://edmond.mpg.de/dataset.xhtml?persistentId=doi%3A10.17617%2F3.LVPCS6`, 2025.

[40] Cristian Munteanu, Yogesh Bhargav Suriyanarayanan, Georgios Smaragdakis, Anja Feldmann, and Tobias Fiebig. Attacks come to those who wait: Long-term observations in an ssh honeynet. In *Proceedings of the 2023 ACM on Internet Measurement Conference*. ACM, 2025.

[41] Cristian Munteanu, Georgios Smaragdakis, Anja Feldmann, and Tobias Fiebig. Catch-22: Uncovering Compromised Hosts using SSH Public Keys. In *USENIX Security Symposium*, 2025.

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

[43] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, IETF, Jan 2006.

[44] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Transport Layer Protocol. RFC 4253, IETF, Jan 2006.

[45] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Connection Protocol. RFC 4254, IETF, Jan 2006.

[46] M. Baushke. More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH). RFC 8268, IETF, Dec 2017.

[47] D. Bider. Extension Negotiation in the Secure Shell (SSH) Protocol. RFC 8308, IETF, Mar 2018.

[48] D. Bider. Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol. RFC 8332, IETF, Mar 2018.

[49] B. Harris and L. Velvindron. Ed25519 and Ed448 Public Key Algorithms for the Secure Shell (SSH) Protocol. RFC 8709, IETF, Feb 2020.

[50] L. Velvindron. Deprecating RC4 in Secure Shell (SSH). RFC 8758, IETF, Apr 2020.

[51] M. Baushke. Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH). RFC 9142, IETF, Jan 2022.

[52] D. Bider and M. Baushke. SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol. RFC 6668, IETF, Jul 2012.

[53] Fail2Ban. Fail2Ban. `https://github.com/fail2ban/fail2ban`. Last accessed: 2025-06-26.

[54] OpenSSH. OpenSSH. `https://www.openssh.com/`. Last accessed: 2025-06-26.

[55] Dropbear. Dropbear. `https://matt.ucc.asn.au/dropbear/dropbear.html`. Last accessed: 2025-06-26.

[56] Oliver Gasser, Ralph Holz, and Georg Carle. A deeper understanding of SSH: Results from Internet-wide scans. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9, 2014.

[57] Hilarie Orman. The Morris worm: A fifteen-year perspective. *IEEE Security & Privacy*, 1(5):35–43, 2003.

[58] Sachin Kumar Singh, Shreeman Gautam, Cameron Cartier, Sameer Patil, and Robert Ricci. 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)*, pages 1731–1750, Santa Clara, CA, Apr 2024. USENIX Association.

[59] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. IoTPOT: A novel honeypot for revealing current IoT threats. *Journal of Information Processing*, 24(3):522–533, 2016.

[60] Tongbo Luo, Zhaoyan Xu, Xing Jin, Yanhui Jia, and Xin Ouyang. Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices. *Black Hat*, 1:1–11, 2017.

[61] Pamela Beltrán-García, Eleazar Aguirre-Anaya, Ponciano Jorge Escamilla-Ambrosio, and Raúl Acosta-Bermejo. IoT botnets. In *Telematics and Computing: 8th International Congress, WITCOM 2019, Merida, Mexico, November 4–8, 2019, Proceedings 8*, pages 247–257. Springer, 2019.

[62] Majda Wazzan, Daniyal Algazzawi, Omaima Bamasaq, Aiiad Albeshri, and Li Cheng. Internet of Things botnet detection approaches: Analysis and recommendations for future research. *Applied Sciences*, 11(12):5713, 2021.

[63] Simon Nam Thanh Vu, Mads Stege, Peter Issam El-Habr, Jesper Bang, and Nicola Dragoni. A survey on botnets: Incentives, evolution, detection and current trends. *Future Internet*, 13(8):198, 2021.

[64] European Commision. Europe's Internet of Things Policy. `https://digital-strategy.ec.europa.eu/en/policies/internet-things-policy`. Last accessed: 2025-06-26.

[65] UK Commision. UK Internet of Things Policy. `https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf`. Last accessed: 2025-06-26.

[66] S. Frankel, R. Glenn, and S. Kelly. The AES-CBC Cipher Algorithm and Its Use with IPsec. RFC 3602, IETF, Sep 2003.

[67] Y. Nir and A. Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC 7539, IETF, May 2015.

[68] P. Gutmann. Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7366, IETF, Sep 2014.

[69] Martin Drašar. Protocol-Independent Detection of Dictionary Attacks. In Thomas Bauschert, editor, *Advances in Communication Networking*, pages 304–309, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[70] Laurens Hellemons, Luuk Hendriks, Rick Hofstede, Anna Sperotto, Ramin Sadre, and Aiko Pras. SSHCure: A Flow-Based SSH Intrusion Detection System. In Ramin Sadre, Jiří Novotný, Pavel Čeleda, Martin Waldburger, and Burkhard Stiller, editors, *Dependable Networks and Services*, pages 86–97, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[71] Rick Hofstede, Mattijs Jonker, Anna Sperotto, and Aiko Pras. Flow-based web application brute-force attack and compromise detection. *Journal of network and systems management*, 25:735–758, 2017.

[72] Rick Hofstede, Aiko Pras, Anna Sperotto, and Gabi Dreo Rodosek. Flow-based compromise detection: Lessons learned. *IEEE security & privacy*, 16(1):82–89, 2018.

[73] B. Claise, B. Trammell, and P. Aitken. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, IETF, Sep 2013.

[74] Mattijs Jonker, Rick Hofstede, Anna Sperotto, and Aiko Pras. Unveiling flat traffic on the internet: an SSH attack case study. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 270–278. IEEE, 2015.

[75] John Hancock, Taghi M Khoshgoftaar, and Joffrey L Leevy. Detecting SSH and FTP brute force attacks in big data. In *2021 20th IEEE international conference on machine learning and applications (ICMLA)*, pages 760–765. IEEE, 2021.

[76] Md Delwar Hossain, Hideya Ochiai, Fall Doudou, and Youki Kadobayashi. SSH and FTP brute-force Attacks Detection in Computer Networks: LSTM and Machine Learning Approaches. In *2020 5th international conference on computer and communication systems (ICCCS)*, pages 491–497. IEEE, 2020.

[77] Karel Hynek, Tomáš Beneš, Tomáš Čejka, and Hana Kubátová. Refined detection of SSH brute-force attackers using machine learning. In *ICT Systems Security and Privacy Protection: 35th IFIP TC 11 International Conference, SEC 2020, Maribor, Slovenia, September 21–23, 2020, Proceedings 35*, pages 49–63. Springer, 2020.

[78] Tsung-Han Lee, Lin-Huang Chang, and Chao-Wei Syu. Deep learning enabled intrusion detection and prevention system over SDN networks. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2020.

[79] Maryam M Najafabadi, Taghi M Khoshgoftaar, Clifford Kemp, Naeem Seliya, and Richard Zuech. Machine learning for detecting brute force attacks at the network level. In *2014 IEEE International Conference on Bioinformatics and Bioengineering*, pages 379–385. IEEE, 2014.

[80] Maryam M Najafabadi, Taghi M Khoshgoftaar, Chad Calvert, and Clifford Kemp. Detection of ssh brute force attacks using aggregated netflow data. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 283–288. IEEE, 2015.

[81] Christian Kreibich, Nicholas Weaver, Chris Kanich, Weidong Cui, and Vern Paxson. GQ: Practical Containment for Measuring Modern Malware Systems. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 397–412, 2011.

[82] Phuong M. Cao, Yuming Wu, Subho S. Banerjee, Justin Azoff, Alex Withers, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. CAUDIT: Continuous Auditing of SSH Servers To Mitigate Brute-Force Attacks. In *NSDI*, 2019.

[83] Jim Owens and Jeanna Neefe Matthews. A Study of Passwords and Methods Used in Brute-Force SSH Attacks. `https://api.semanticscholar.org/CorpusID:15340523`, 2008.

[84] Zain Shamsi, Daniel Zhang, Daehyun Kyoung, and Alex Liu. Measuring and Clustering Network Attackers using Medium-Interaction Honeypots. In *IEEE European Symposium on Security and Privacy Workshops*, 2022.

[85] Matteo Boffa, Giulia Milan, Luca Vassio, Idilio Drago, Marco Mellia, and Zied Ben Houidi. Towards NLP-based Processing of Honeypot Logs. In *IEEE European Symposium on Security and Privacy Workshops*, 2022.

[86] Vincent Ghiette, Harm Griffioen, and Christian Doerr. Fingerprinting Tooling used for SSH Compromisation Attempts. In *RAID*, 2019.

[87] Yuming Wu, Phuong M Cao, Alexander Withers, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. Mining Threat Intelligence from Billion-scale SSH Brute-Force Attacks. In *DISS*, 2020.

[88] Shreya Udhani, Alexander Withers, and Masooda Bashir. Human vs Bots: Detecting Human Attacks in a Honeypot Environment. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–6, 2019.

[89] Gabriel Salles-Loustau, Robin Berthier, Etienne Collange, Bertrand Sobesto, and Michel Cukier. Characterizing Attackers and Attacks: An Empirical Study. In *17th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 174–183, 2011.

[90] Farhan Sadique and Shamik Sengupta. Analysis of Attacker Behavior in Compromised Hosts During Command and Control. In *ICC 2021 - IEEE International Conference on Communications*, pages 1–7, 2021.

[91] Liz Izhikevich, Manda Tran, Michalis Kallitsis, Aurore Fass, and Zakir Durumeric. Cloud Watching: Understanding Attacks Against Cloud-hosted Services. In *Internet Measurement Conference*, pages 313–327, 2023.

[92] Zhenxin Zhan, Maochao Xu, and Shouhuai Xu. Characterizing Honeypot-Captured Cyber Attacks: Statistical Framework and Case Study. *IEEE Transactions on Information Forensics and Security*, 8(11):1775–1789, 2013.

[93] Gokul Kannan Sadasivam, Chittaranjan Hota, and Bhojan Anand. Classification of SSH Attacks Using Machine Learning Algorithms. In *2016 6th International Conference on IT Convergence and Security (ICITCS)*, pages 1–6, 2016.

[94] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of Grey: On the effectiveness of reputation-based "blacklists". In *3rd International Conference on Malicious and Unwanted Software (MALWARE)*, pages 57–64. IEEE, 2008.

[95] Vector Guo Li, Gautam Akiwate, Kirill Levchenko, Geoffrey M Voelker, and Stefan Savage. iClairvoyance: Inferring blocklist use on the internet. In *Passive and Active Measurement*, pages 57–75. Springer, 2021.

[96] Andreas Pitsillidis, Chris Kanich, Geoffrey M Voelker, Kirill Levchenko, and Stefan Savage. Taster's Choice: A Comparative Analysis of Spam Feeds. In *Internet Measurement Conference*, pages 427–440, 2012.

[97] Matthew L Bringer, Christopher A Chelmecki, and Hiroshi Fujinoki. A survey: Recent advances and future trends in honeypot research. *International Journal of Computer Network and Information Security*, 4(10):63, 2012.

[98] Xinwen Fu, Wei Yu, Dan Cheng, Xuejun Tan, Kevin Streff, and Steve Graham. On Recognizing Virtual Honeypots and Countermeasures. In *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 211–218. IEEE, 2006.

[99] Michael Vrable, Justin Ma, Jay Chen, David Moore, Erik Vandekieft, Alex C Snoeren, Geoffrey M Voelker, and Stefan Savage. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 148–162, 2005.

[100] Joni Uitto, Sampsa Rauti, Samuel Laurén, and Ville Leppänen. A survey on anti-honeypot and anti-introspection methods. In *Recent Advances in Information Systems and Technologies: Volume 2 5*, pages 125–134. Springer, 2017.

[101] John P John, Fang Yu, Yinglian Xie, Arvind Krishnamurthy, and Martín Abadi. Heat-seeking honeypots: design and experience. In *Proceedings of the 20th international conference on World wide web*, pages 207–216, 2011.

[102] Binbin Wang, Yilian Zhang, Minjie Zhu, and Yan Chen. Design and Implementation of Web Honeypot Detection System Based on Search Engine. In *2020 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pages 130–135. IEEE, 2020.

[103] Matthias Wählisch, Sebastian Trapp, Christian Keil, Jochen Schönfelder, Thomas C Schmidt, and Jochen Schiller. First insights from a mobile honeypot. *ACM SIGCOMM Computer Communication Review*, 42(4):305–306, 2012.

[104] David Dagon, Xinzhou Qin, Guofei Gu, Wenke Lee, Julian Grizzard, John Levine, and Henry Owen. Honeystat: Local worm detection using honeypots. In *Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings 7*, pages 39–58. Springer, 2004.

[105] Kippo. Kippo on GitHub. `https://github.com/desaster/kippo`. Last accessed: 2025-06-26.

[106] Melike Başer, Ebu Yusuf Güven, and Muhammed Ali Aydın. SSH and Telnet Protocols Attack Analysis Using Honeypot Technique: Analysis of SSH AND TELNET Honeypot. In *6th International Conference on Computer Science and Engineering (UBMK)*, pages 806–811, 2021.

[107] Ioannis Koniaris, Georgios Papadimitriou, and Petros Nicopolitidis. Analysis and Visualization of SSH Attacks using Honeypots. In *Eurocon 2013*, 2013.

[108] Iyatiti Mokube and Michele Adams. Honeypots: Concepts, Approaches, and Challenges. In *Annual Southeast Regional Conference*, 2007.

[109] The Honeynet Project. The Honeynet Project. `https://www.honeynet.org/`. Last accessed: 2025-06-26.

[110] Fan Dang, Zhenhua Li, Yunhao Liu, Ennan Zhai, Qi Alfred Chen, Tianyin Xu, Yan Chen, and Jingyu Yang. Understanding Fileless Attacks on Linux-Based IoT Devices with HoneyCloud. In *ACM MobiSys*, 2019.

[111] Thomas Favale, Danilo Giordano, Idilio Drago, and Marco Mellia. What Scanners do at L7? Exploring Horizontal Honeypots for Security Monitoring. In *IEEE European Symposium on Security and Privacy Workshops*, 2022.

[112] Hwanjo Heo and Seungwon Shin. Who is Knocking on the Telnet Port: A Large-Scale Empirical Study of Network Scanning. In *ACM ASIACCS*, 2018.

[113] Philipp Richter and Arthur Berger. Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope. In *ACM Internet Measurement Conference*, 2019.

[114] Alexander Vetterl and Richard Clayton. Bitter Harvest: Systematically Fingerprinting Low- and Medium-interaction Honeypots at Internet Scale. In *USENIX Workshop on Offensive Technologies*, 2018.

[115] Shun Morishita, Takuya Hoizumi, Wataru Ueno, Rui Tanabe, Carlos Gañán, Michel J.G. van Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. Detect Me If You... Oh Wait. An Internet-Wide View of Self-Revealing Honeypots. In *IFIP/IEEE Symposium on Integrated Network and Service Management*, 2019.

[116] Alexander Vetterl, Richard Clayton, and Ian Walden. Counting Outdated Honeypots: Legal and Useful. In *IEEE Security and Privacy Workshops*, 2019.

[117] Shodan. Honeypot Or Not? Shodan, `ttps://honeyscore.shodan.io`. Last accessed: 2025-06-26.

[118] Daniel R. Thomas, Richard Clayton, and Alastair R. Beresford. 1000 days of UDP amplification DDoS attacks. In *Symposium on Electronic Crime Research (eCrime)*, 2017.

[119] Harm Griffioen, Kris Oosthoek, Paul van der Knaap, and Christian Doerr. Scan, Test, Execute: Adversarial Tactics in Amplification DDoS Attacks. In *ACM CCS*, 2021.

[120] Lukas Krämer, Johannes Krupp, Daisuke Makita, Tomomi Nishizoe, Takashi Koide, Katsunari Yoshioka, and Christian Rossow. Amppot: Monitoring and defending against amplification ddos attacks. In *RAID*, 2015.

[121] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. Going Wild: Large-Scale Classification of Open DNS Resolvers. In *ACM Internet Measurement Conference*, 2015.

[122] Anton O Prokofiev and Yulia S Smirnova. Counteraction against Internet of Things botnets in private networks. In *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 301–305. IEEE, 2019.

[123] Igor Kotenko, Alexey Konovalov, and Andrey Shorov. Agent-based modeling and simulation of botnets and botnet defense. In *Conference on Cyber Conflict. CCD COE Publications. Tallinn, Estonia*, pages 21–44. Citeseer, 2010.

[124] Jeremy Kepner, Jonathan Bernays, Stephen Buckley, Kenjiro Cho, Cary Conrad, Leslie Daigle, Keeley Erhardt, Vijay Gadepally, Barry Greene, Michael Jones, et al. Zero Botnets: An Observe-Pursue-Counter Approach. *arXiv preprint arXiv:2201.06068*, 2022.

[125] Rapid7. Project Heisenberg. Rapid7, `https://www.rapid7.com/research/project-heisenberg/`. Last accessed: 2025-06-26.

[126] Honeynet Project. The Honeynet Project. `https://www.honeynet.org/about/`. Last accessed: 2025-06-26.

[127] SANS Internet Storm Center. DShield Honeypot. DShield Honeypot, `https://isc.sans.edu/tools/honeypot/`. Last accessed: 2025-06-26.

[128] T-Systems. The T-Sec Radar shows cyber attacks happening worldwide on our and our partners' honeypot infrastructure. . T-Sec Radar, `https://www.sicherheitstacho.eu/start/main`. Last accessed: 2025-06-26.

[129] David Moore, Colleen Shannon, Geoffrey M Voelker, and Stefan Savage. Network Telescopes: Technical Report. Technical report, Cooperative Association for Internet Data Analysis (CAIDA), July 2004.

[130] Alberto Dainotti, Alistair King, kc Claffy, Ferdinando Papale, and Antonio Pescapè. Analysis of a "/0" Stealth Scan from a Botnet. In *ACM Internet Measurement Conference*, 2012.

[131] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. In *USENIX Security Symposium*, 2001.

[132] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford-Chen, and Nicholas Weaver. Inside the Slammer Worm. *IEEE Security and Privacy*, 1(4), 2003.

[133] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C. Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of Country-wide Internet Outages Caused by Censorship. In *ACM Internet Measurement Conference*, 2011.

[134] Karyn Benson, Alberto Dainotti, kc claffy, Alex C. Snoeren, and Michael Kallitsis. Leveraging Internet Background Radiation for Opportunistic Network Analysis. In *ACM Internet Measurement Conference*, 2015.

[135] Daniel Wagner, Sahil Ashish Ranadive, Harm Griffioen, Michalis Kallitsis, Alberto Dainotti, Georgios Smaragdakis, and Anja Feldmann. How to Operate a Meta-Telescope in your Spare Time. In *ACM Internet Measurement Conference*, 2023.

[136] Raphael Hiesgen, Marcin Nawrocki, Alistair King, Alberto Dainotti, Thomas C. Schmidt, and Matthias Wählisch. Spoki: Unveiling a New Wave of Scanners Through a Reactive Network Telescope. In *USENIX Security Symposium*, 2022.

[137] Philipp Richter, Oliver Gasser, and Arthur Berger. Illuminating Large-Scale IPv6 Scanning in the Internet. In *ACM Internet Measurement Conference*, 2022.

[138] RIPE. RIPE Stat. `https://stat.ripe.net/`. Last accessed: 2025-06-26.

[139] MaxMind. MaxMind. `https://www.maxmind.com/`. Last accessed: 2025-06-26.

[140] VirusTotal. VirusTotal. `https://www.virustotal.com/`. Last accessed: 2025-06-26.

[141] ClamAV. ClamAV. `https://www.clamav.net/`. Last accessed: 2025-06-26.

[142] FileScan.IO. FileScan.IO. `https://www.filescan.io/`. Last accessed: 2025-06-26.

[143] InQuest. InQuest. `https://inquest.net/`. Last accessed: 2025-06-26.

[144] Cert.PL. Cert.PL. `https://mwdb.cert.pl/`. Last accessed: 2025-06-26.

[145] YOROI YOMI. YOROI YOMI. `https://yomi.yoroi.company/`. Last accessed: 2025-06-26.

[146] Shane Huntley. Fog of war: how the Ukraine conflict transformed the cyber threat. `https://blog.google/threat-analysis-group/fog-of-war-how-the-ukraine-conflict-transformed-the-cyber-threat-landscape/`. Last accessed: 2025-06-26.

[147] abuse.ch. abuse.ch: Fighting Malware and Botnets. `https://abuse.ch/`. Last accessed: 2025-06-26.

[148] Cymru Team. IP to ASN mapping. *`http://www.team-cymru.org/IP-ASN-mapping.html`*. Last accessed: 2025-06-26.

[149] ArmstrongTechs. ArmstrongTechs Indicators-of-compromise-IOCs. `https://github.com/ArmstrongTechs/Indicators-of-compromise-IOCs?tab=readme-ov-file`. Last accessed: 2025-06-26.

[150] Florian Streibelt, Martina Lindorfer, Seda Gürses, Carlos H Gañán, and Tobias Fiebig. Back-to-the-Future Whois: An IP Address Attribution Service for Working with Historic Datasets. In *International Conference on Passive and Active Network Measurement*, pages 209–226. Springer, 2023.

[151] Basil Fillan, Ben Cartwright-Cox, Cynthia Revström, Elimalko Saado, eraters, Igloo22225, Jeroen Massar, Job Snijders, Molly Miller, Puck Meerburg, Roelf Wichertjes, Tim Stallard, and Tommy Bowditch. BGP.tools. https://bgp.tools/. Last accessed: 2025-06-26.

[152] PeeringDB. https://www.peeringdb.com. Last accessed: 2025-06-26.

[153] Microsoft. An overview of Russia's cyberattack activity in Ukraine. https://www.microsoft.com/en-us/security/security-insider/intelligence-reports/special-report-ukraine. Last accessed: 2025-06-26.

[154] Jiawei Zhou, Zidong Zhang, Lingyun Ying, Huajun Chai, Jiuxin Cao, and Haixin Duan. Hey, Your Secrets Leaked! Detecting and Characterizing Secret Leakage in the Wild. In *IEEE Symp. on Security and Privacy*, 2025.

[155] Alessandro Cantelli-Forti and Michele Colajanni. Adversarial fingerprinting of cyber attacks based on stateful honeypots. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 19–24. IEEE, 2018.

[156] Nicholas Wells. Busybox: A swiss army knife for linux. *Linux Journal*, 2000(78es):10–es, 2000.

[157] Aohui Wang, Ruigang Liang, Xiaokang Liu, Yingjun Zhang, Kai Chen, and Jin Li. An Inside Look at IoT Malware. In Fulong Chen and Yonglong Luo, editors, *Industrial IoT Technologies and Applications*, pages 176–186, Cham, 2017. Springer International Publishing.

[158] Joseph Khoury, Morteza Safaei Pour, and Elias Bou-Harb. A Near Real-Time Scheme for Collecting and Analyzing IoT Malware Artifacts at Scale. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ARES '22, New York, NY, USA, 2022. Association for Computing Machinery.

[159] Andrei Costin and Jonas Zaddach. IoT malware: Comprehensive survey, analysis framework and case studies. *BlackHat USA*, 1(1):1–9, 2018.

[160] Dreambox Configuration. https://dreamboxedit.com/en/workshop-2/konfiguration/. Last accessed: 2025-06-26.

[161] Dreambox Enigma 1. https://dreambox.de/board/index.php?board/15-enigma-1-alle-themen/. Last accessed: 2025-06-26.

[162] Nguyen Quang Minh. Dasan H660DW. https://www.minhng99.cloud/Exploring-router-Dasan_H660DW/. Last accessed: 2025-06-26.

[163] Zetong Zhao, Shreyas Srinivasa, and Emmanouil Vasilomanolakis. SweetCam: an IP Camera Honeypot. In *Proceedings of the 5th Workshop on CPS&IoT Security and Privacy*, pages 75–81, 2023.

[164] Hewlett Packard. HP Poly acquisition. https://www.hp.com/us-en/newsroom/press-releases/2022/hp-inc-completes-acquisition-of-poly.html. Last accessed: 2025-06-26.

[165] Microsoft. Polycom Lync Update 2017. https://support.microsoft.com/en-us/topic/april-2017-cumulative-update-for-microsoft-lync-phone-edition-for-polycom-cx500-polycom-cx600-and-polycom-cx3000-telephones-kb4019529-7c85b54c-e4b1-44a8-b07b-e61d3eb92359. Last accessed: 2025-06-26.

[166] Michel Oosterhof. Cowrie Docs. https://readthedocs.org/projects/cowrie/downloads/pdf/latest/. Last accessed: 2025-06-26.

[167] Yoroi. Yoroi Outlaw Hacking Group. https://yoroi.company/research/outlaw-is-back-a-new-crypto-botnet-targets-european-organizations/. Last accessed: 2025-06-26.

[168] Trendmicro. Trendmicro, Outlaw Hacking Group. https://www.trendmicro.com/de_de/research/18/k/outlaw-group-distributes-botnet-for-cryptocurrency-mining-scanning-and-brute-force.html. Last accessed: 2025-06-26.

[169] Git, Crypto Miner Hack. https://github.com/dangoldin/crypto-miner-hack. Last accessed: 2025-06-26.

[170] Blog Port22. mdrfckrs – part one. https://blog.port22.dk/mdrfckrs-part-one/. Last accessed: 2025-06-26.

[171] Blog Port22. mdrfckrs – part two. https://blog.port22.dk/mdrfckrs-part-two/. Last accessed: 2025-06-26.

[172] Malware News. Mdrfckr: How to stay safe online this festive season. https://malware.news/t/dota-campaign-analyzing-a-coin-mining-and-remote-access-hybrid-campaign/30326. Last accessed: 2025-06-26.

[173] WorkMiner Bot. https://www.a1ee.cn/medium/workminer/. Last accessed: 2025-06-26.

[174] Alex.Turing, Hui Wang, and Genshen Ye. The death of Mozi. https://blog.netlab.360.com/the_death_of_mozi_cn. Last accessed: 2025-06-26.

[175] C2-Daily-Feed. https://github.com/criminalip/C2-Daily-Feed. Last accessed: 2025-06-26.

[176] Stephen Mondiguing. KillNet DDoS Blocklist. https://github.com/securityscorecard/SSC-Threat-Intel-IoCs/tree/master/KillNet-DDoS-Blocklist. Last accessed: 2025-06-26.

[177] Killnet: Inside the World's Most Prominent Pro-Kremlin Hacktivist Collective. https://flashpoint.io/intelligence-101/killnet/. Last accessed: 2025-04-15.

[178] ZNC. https://wiki.znc.in/ZNC. Last accessed: 2025-06-26.

[179] Ukrainska Pravda. US charges 6 Russians with cyberattack on Ukraine and NATO before 2022 full-scale invasion. https://www.pravda.com.ua/eng/news/2024/09/5/7473672/. Last accessed: 2025-06-26.

[180] US Department of Justice. Russian National Charged for Conspiring with Russian Military Intelligence to Destroy Ukrainian Government Computer Systems and Data. https://www.justice.gov/opa/pr/russian-national-charged-conspiring-russia-military-intelligence-destroy-ukrainian. Last accessed: 2025-06-26.

[181] Ken Proska, John Wolfram, Jared Wilson, Dan Black, Keith Lunden, Daniel Kapellmann Zafra, Nathan Brubaker, Tyler McLellan, and Chris Sistrunk. Sandworm Disrupts Power in Ukraine Using a Novel Attack Against Operational Technology. https://cloud.google.com/blog/topics/threat-intelligence/sandworm-disrupts-power-ukraine-operational-technology/. Last accessed: 2025-06-26.

[182] James Pearson. Russian spies behind cyber attack on Ukraine power grid in 2022 - researchers. https://www.reuters.com/technology/cybersecurity/russian-spies-behind-cyberattack-ukrainian-power-grid-2022-researchers-2023-11-09/. Last accessed: 2025-06-26.

[183] Daryna Antoniuk. Ukraine energy facility took unique Sandworm hit on day of missile strikes, report says. https://therecord.media/sandworm-attack-ukraine-energy-facility-missile-strikes. Last accessed: 2025-06-26.

[184] Jason Firch. Russian Hacktivists, Killnet, Take Down US Airport Websites. https://purplesec.us/breach-report/killnet-ddos-airport-websites/. Last accessed: 2025-06-26.

[185] Daryna Antoniuk. Russian hackers infiltrated Ukrainian telecom giant months before cyberattack. https://therecord.media/russians-infiltrated-kyivstar-months-before. Last accessed: 2025-06-26.

[186] Cyber Peace Institute. Cyber Dimensions of the Armed Conflict in Ukraine. https://cyberconflicts.cyberpeaceinstitute.org/report. Last accessed: 2025-06-26.

[187] CERT-EU: The Cybersecurity Service for the Union institutions, bodies, offices and agencies. Cyber Security Brief 24-02 - January 2024; APT29 cyber-attacks. https://cert.europa.eu/publications/threat-intelligence/cb24-02/. Last accessed: 2025-06-26.

[188] Jessica Lyons. Kremlin's Sandworm blamed for cyberattacks on US, European water utilities. https://www.theregister.com/2024/04/17/russia_sandworm_cyberattacks_water/. Last accessed: 2025-06-26.

[189] SOC Radar. Major Cyberattack April 2024: Sandworm. https://socradar.io/major-cyber-attacks-in-review-april-2024/. Last accessed: 2025-06-26.

[190] Chris Siebenmann. Your SSH Keys are a (potential) Information Leak. https://web.archive.org/web/20240827115338/https://utcc.utoronto.ca/~cks/space/blog/tech/SSHKeysAreInfoLeak. Last accessed: 2025-06-26.

[191] Bitvise. Bitvise. https://bitvise.com/. Last accessed: 2025-06-26.

[192] WolfSSH. WolfSSH. https://www.wolfssl.com/products/wolfssh/. Last accessed: 2025-06-26.

[193] Wayf.dk. Wayf.dk. https://www.wayf.dk/en/wayf-part-intl-workshop-federated-access-ssh. Last accessed: 2025-06-26.

[194] Ioannis Koniaris, Georgios Papadimitriou, and Petros Nicopolitidis. Analysis and visualization of SSH attacks using honeypots. In *Eurocon 2013*, pages 65–72, 2013.

[195] E. Kheirkhah, Sayyed Amin, H.A. Sistani, and H. Acharya. An Experimental Study of SSH Attacks by using Honeypot Decoys. *Indian Journal of Science and Technology*, 6:5567–5578, 12 2013.

[196] AbdelRahman Abdou, David Barrera, and Paul C. van Oorschot. What Lies Beneath? Analyzing Automated SSH Bruteforce Attacks. In Frank Stajano, Stig F. Mjølsnes, Graeme Jenkinson, and Per Thorsheim, editors, *Technology and Practice of Passwords*, pages 72–91, Cham, 2016. Springer International Publishing.

[197] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 3546, IETF, Jun 2003.

[198] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-Wide Scanning and its Security Applications. In *USENIX Security Symposium*, 2013.

[199] Z. Durumeric. ZGrab2. `https://github.com/zmap/zgrab2`. Last accessed: 2025-06-26.

[200] Jonathan Kilgallin and Ross Vasko. Factoring RSA keys in the IoT era. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 184–189. IEEE, 2019.

[201] T. Arnold, J. He, W. Jiang, M. Calder, I. Cunha, V. Giotsas, and E. Katz-Bassett. Cloud Provider Connectivity in the Flat Internet. In *ACM Internet Measurement Conference*, 2020.

[202] Bernardus Jansen, Natalia Kadenko, Dennis Broeders, Michel van Eeten, Kevin Borgolte, and Tobias Fiebig. Pushing boundaries: An empirical view on the digital sovereignty of six governments in the midst of geopolitical tensions. *Government Information Quarterly*, 40(4):101862, 2023.

[203] IPInfo. IPInfo. `https://ipinfo.io/`. Last accessed: 2025-06-26.

[204] Mario Lins, René Mayrhofer, Michael Roland, Daniel Hofer, and Martin Schwaighofer. On the critical path to implant backdoors and the effectiveness of potential mitigation techniques: Early learnings from XZ. *arXiv preprint arXiv:2404.08987*, 2024.

[205] Tobias Fiebig, Franziska Lichtblau, Florian Streibelt, Thorben Krüger, Pieter Lexis, Randy Bush, and Anja Feldmann. Learning from the past: designing secure network protocols. *Cybersecurity Best Practices: Lösungen zur Erhöhung der Cyberresilienz für Unternehmen und Behörden*, pages 585–613, 2018.

[206] Kubernetes.io. Kubernetes.io. `https://kubernetes.io`. Last accessed: 2025-06-26.

[207] teamtnt. Web Reseouce 1. `https://web.archive.org/web/20240904145504/https://documents.hubaoquan.cn/HS6SqV7w.txt`. Last accessed: 2025-06-26.

[208] teamtnt. Web Reseouce 2. `https://web.archive.org/web/20240904144948/https://www.cnblogs.com/sexiszero/p/14775793.html`. Last accessed: 2025-06-26.

[209] teamtnt. Web Reseouce 3. `https://web.archive.org/web/20240904145120/https://www.52pojie.cn/thread-1283815-1-1.html`. Last accessed: 2025-06-26.

[210] Tor Project. Tor Project. `https://community.torproject.org/`. Last accessed: 2025-06-26.

[211] V2Ray. V2Ray. `https://www.v2ray.com/`. Last accessed: 2025-06-26.

[212] Shane Miller, Kevin Curran, and Tom Lunney. Securing the internet through the detection of anonymous proxy usage. In *2015 World Congress on Internet Security (WorldCIS)*, pages 153–158. IEEE, 2015.

[213] Mandeep Pannu, Bob Gill, Robert Bird, Kai Yang, and Ben Farrel. Exploring proxy detection methodology. In *2016 IEEE International Conference on Cybercrime and Computer Forensic (ICCCF)*, pages 1–6. IEEE, 2016.

[214] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies*, 2015.

[215] Zeyu Li, Meiqi Wang, Xuebin Wang, Jinqiao Shi, Kexin Zou, and Majing Su. Identification domain fronting traffic for revealing obfuscated C2 communications. In *2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*, pages 91–98. IEEE, 2021.

[216] Tadayoshi Kohno, Yasemin Acar, and Wulf Loh. Ethical frameworks and computer security trolley problems: Foundations for conversations. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5145–5162, 2023.

[217] Ben Stock, Giancarlo Pellegrino, Frank Li, Michael Backes, and Christian Rossow. Didn't You Hear Me?—Towards More Successful Web Vulnerability Notifications. 2018.

[218] Ben Stock, Giancarlo Pellegrino, Christian Rossow, Martin Johns, and Michael Backes. Hey, you have a problem: On the feasibility of {Large-Scale} web vulnerability notification. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1015–1032, 2016.

[219] Jakub Czyz, Matthew Luckie, Mark Allman, and Michael Bailey. Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy. In *NDSS*, 2016.