MINING THREAT INTELLIGENCE FROM BILLION-SCALE SSH BRUTE-FORCE ATTACKS

BY

YUMING WU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Advisers:

Professor Ravishankar K. Iyer
Research Professor Zbigniew T. Kalbarczyk

ABSTRACT

This thesis first presents Continuous Auditing of Secure Shell (SSH) Servers to Mitigate Brute-Force Attacks (CAUDIT), an operational system deployed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois. One of CAUDIT's key features includes a honeypot, which attracted and recorded 11 billion SSH brute-force attack attempts targeting the operational system at NCSA from February 2017 to November 2019. Based on the attack data, this thesis then presents a comprehensive study to characterize the attack nature of the 11 billion attack attempts. We report the nature of these attacks in terms of i) persistence (i.e., consecutively attacking over an entire year), ii) targeted strategies (i.e., using stolen SSH keys), iii) large-scale evasion techniques (i.e., using randomized SSH client versions) to bypass signature detectors, and iv) behaviors of human-supervised botnet.

The significance of our analyses for security operators include i) discerning cross-country attacks versus persistent attacks, ii) notifying cloud providers and IoT vendors regarding stolen SSH keys for them to verify the effectiveness of software patches, iii) deterring the above evasion techniques by using anomaly detectors/rate limiters, and iv) differentiating between fully automated attacks versus more sophisticated attacks driven by human.

The work in this thesis is completed in two stages along with two papers. The first paper is published in 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI'19), and the second paper is to be published in Workshop on Decentralized IoT Systems and Security (DISS) 2020.

We collaborated with NCSA, which provided us with the network operational system and attack data. The research and analysis were performed jointly with the co-authors in the two papers. My specific contribution is highlighted in this thesis is threat intelligence analysis.

To my parents and colleagues, for their love and support.

ACKNOWLEDGMENTS

I am grateful to my advisors Prof. Ravishankar K. Iyer and Prof. Zbigniew T. Kalbarczyk. The completion of this thesis would not have been possible without their advice and support. I am also thankful to my colleagues in the DEPEND research group, especially Phuong M. Cao, who has provided instrumental guidance throughout the research work. Also, the DEPEND Symphony cluster and its maintenance group offer indispensable technical support for the large-scale data analysis. I want to thank the NCSA security team, especially Alexander Withers, who shared the network operational system and attack data. I would also like to thank Ms. Frances Rigberg Baker and Ms. Jenny Applequist for proofreading.

The research-supporting infrastructure is part of a larger project: SDAIA at the National Center for Supercomputing Applications (NCSA). This project is supported by the National Science Foundation Awards “SI2-SSE: AttackTagger: Early Threat Detection for Scientific Cyberinfrastructure” and “CICI: Secure Data Architecture: Shared Intelligence Platform for Protecting our National Cyberinfrastructure” under Grant No. `1535070` and `1547249` respectively.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The Secure Shell (SSH) is the universal authentication protocol for managing remote servers. Critical devices often expose their remote access interfaces to the Internet via the SSH protocol [1], using default usernames/passwords [2]. The availability of stolen credentials [3] has imposed a new dimension to the problem: attackers can remotely masquerade as legitimate users and penetrate internal networks to misuse computational resources and leak sensitive data [4]. Recently, attacks targeting publicly exposed SSH servers experience an exponential growth due to the availability of leaked passwords and stolen keys [5–9]. An attacker typically logs into the server as a normal user first, then escalates to root privilege, thus enables persistent access for compromising the internal network, exfiltrating sensitive data [10–12] and causing financial losses. While only a small fraction of such attempts succeed, they have led to major misuses in over 50% of 2,100 surveyed systems administrators [13]. For example, when offered 50 Bitcoins by a hacker, a former server administrator at ShapeShift [14], a cryptocurrency company, gave away an SSH private key to the company's Bitcoin core server for accessing internal Bitcoin's wallets. This incident eventually led to $230,000 losses [15, 16].

To mitigate those arising problems from SSH brute-force attacks, this thesis contributes in the following two areas. The first area focuses on the production deployment of Continuous Auditing of SSH Servers to Mitigate Brute-Force Attacks (CAUDIT) [17, 18] at the National Center for Supercomputing Applications (NCSA) at the University of Illinois. CAUDIT provides the environment that set up the honeypot, which attacted and recorded 11 billion SSH brute-force attack attempts targeting the operational system at NCSA over 1,000-day operation from February 2017 to November 2019. Based on the attack data, the second part of this thesis focuses on a comprehensive study to characterize the attack nature of the 11 billion attack attempts. Therefore, we leverage CAUDIT, especially its honeypot, to gather the large-scale threat intelligence, from which we perform analytics on the nature of these attacks in terms of persistence (i.e., consecutively attacking over an entire year), targeted strategies (i.e., using stolen SSH keys), large-scale evasion techniques (i.e., using randomized SSH client versions) to bypass signature detectors, and behaviors of human-supervised botnet.

From the system design perspective, key contributions of CAUDIT include 1) a honeypot for attracting SSH-based attacks over a `/16` IP address range and extracting key metadata (e.g., source IP, password, SSH-client version, or key fingerprint) from these attacks; 2) executing audits on the live production network by replaying of attack attempts recorded by the honeypot; 3) using the IP addresses recorded by the honeypot to block SSH attack attempts at the network border by using a Black Hole Router (BHR) while significantly reducing the load on NCSA's security monitoring system; and 4) the ability to inform peer sites of attack attempts in real-time to ensure containment of coordinated attacks. The system is composed of existing techniques with custom-built components, and its novelty is its ability to execute at a scale that has not been validated earlier (with thousands of nodes and tens of millions of attack attempts per day).

From the threat-intelligence perspective, key contributions of the security data analysis consist of i) discerning cross-country attacks versus persistent attacks, ii) notifying cloud providers and IoT vendors regarding stolen SSH keys for them to verify the effectiveness of software patches, iii) deterring the above evasion techniques by using anomaly detectors/rate limiters, and iv) differentiating between fully automated attacks versus more sophisticated attacks driven by human.

The work in this thesis is completed in two stages with the following two papers: P. Cao et al., "CAUDIT: Continuous Auditing of SSH Servers To Mitigate Brute-Force Attacks" published in 16th USENIX Symposium on Networked Systems Design and Implementation [17] and Y. Wu et al., "Mining Threat Intelligence from Billion-scale SSH Brute-Force Attacks" to be published in Decentralized IoT Systems and Security (DISS) Workshop 2020 [19].

CHAPTER 2

APPROACH OVERVIEW

This chapter describes CAUDIT,[1] an operational system deployed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois, as a part of the Science DMZ Actionable Intelligence Appliance (SDAIA) project [18]. CAUDIT is a fully automated system that enables the identification and exclusion of hosts that are vulnerable to SSH brute-force attacks. Its key features include a lightweight, non-interactive honeypot for attracting SSH-based attacks over a `/16` IP address range simulating ∼65K machines. The honeypot extracts key metadata (e.g., source IP, username, password, SSH client version, and SSH key fingerprint) from these attacks.

2.1 SSH Authentication Logger (SAL)

The core component of our approach is a lightweight, non-interactive SSH server, i.e., the honeypot, that records SSH brute-force attacks (shown as `ssh-auth-logger` in Figure 2.1). Instead of exposing the internal servers of the existing infrastructure (dotted box) to the Internet, CAUDIT uses a virtual server farm of SALs that attracts attack attempts and uses an auditor to mimic such attempts on internal servers continuously. An SSH database of excessive scanners is used by both a controller that instructs a Black Hole Router to dynamically create null routes to block attackers at the network border, and an alert-sharing network.

Attracting attackers. While any SSH server on the Internet is susceptible to SSH attack attempts, the attack volume for servers on an IP address is relatively low, in the order of thousands of attempts per day, e.g., 27K attempts per day in a relevant SSH honeypot deployed by the Naval Postgraduate School [20] in 2017. To attract more attackers than existing

[1] `https://pmcao.github.io/caudit`

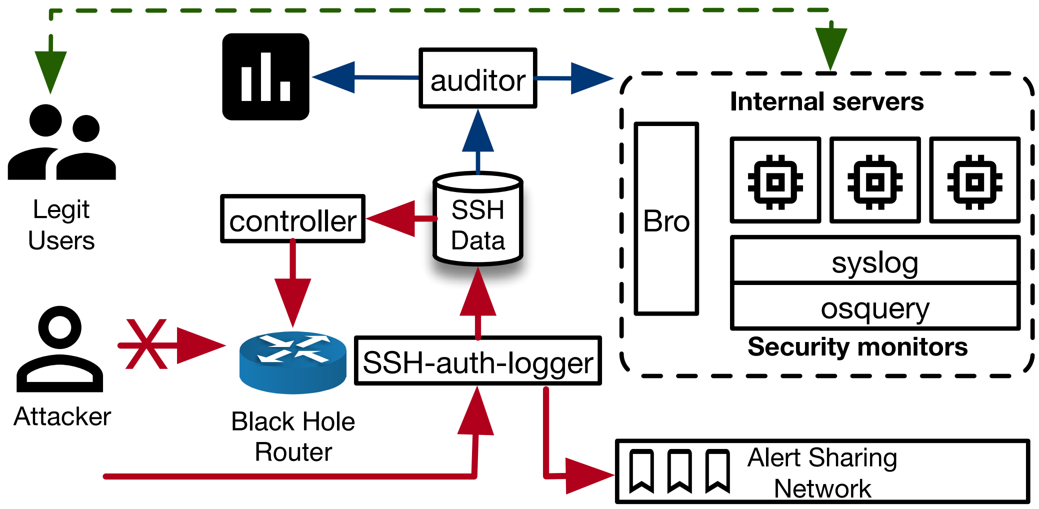


Figure 2.1: Overview of CAUDIT.

honeypots [21–24], we deployed SAL on an enticing classless inter-domain routing (CIDR) `/16` address space that mimics 65, 536 (2^{16}) realistic SSH servers.

By deploying our honeypot on an entire `/16` IP address space, it exposes our honeypot to a larger attack space because all incoming credential guesses targeting the CIDR address space, i.e., the darknet, are redirected to the single instance of the honeypot (Figure 2.2). Thus, our single instance collects attack traces equivalent to those targeting 65, 536 distinct SSH servers. Also, all connections targeting the CIDR address space of the honeypot can be automatically labeled as malicious attempts, because no legitimate server is assigned an IP address in the CIDR address space. Thus, one instance of our honeypot covers an entire `/16` IP address space with significantly fewer hardware resources (Figure 2.3). The disadvantage of this technique is that it puts all the attack loads on a single physical server. However, one can mitigate that issue with a load balancer to distribute traffic in front of the server that is hosting the honeypot.

As a result, our honeypot resulted in attracting an average of 875,491 attack attempts every day, i.e., 33× more than the honeypot in [20].

Deceiving attackers. Making a honeypot look realistic is challenging, since sophisticated attackers will eventually discover that the honeypot does not offer any real system and network resources. Our goal is not to completely fool attackers, but to make our honeypot realistic enough to attract a large number of attacks. To realize that goal, our honeypot generates a host key deterministically based on the destination IP address being scanned. Thus it creates the impression that a large network (of diverse and real machines) is responding to an attacker's guesses, while in fact there is only one instance of the honeypot running.

Memory footprint of interactive versus non-interactive honeypot. A traditional interactive honeypot [22, 23] provides a shell for each attack attempt. However, such an interactive method does not scale, since the more realistic a honeypot is, the more resources (e.g., memory) are needed for deceiving the attackers. For instance, Figure 2.3 compares the memory footprint of an interactive honeypot that provides a shell (based on Linux containers) for a number of concurrent connection attempts vs. our non-interactive honeypot on a commodity server. While the non-interactive honeypot maintains a constant memory usage of ∼98MB, the interactive-honeypot uses linearly more memory for new containers as each new connection is made. Thus, traditional interactive honeypots are hard to scale to millions or billions of attack attempts. The small scale of recorded attack traces most probably will miss a broad landscape of distributed, global coordinated attacks, not to mention advanced target strategies and evasion techniques on a global scale.

To address those challenges in terms of operational resources and scalability, our SSH authentication logger is implemented in Golang with only 159 lines of code [25]. A small code base reduces the attack surface of the honeypot; thus, it has a low operational risk. In addition, our honeypot does not provide any shell to each attack attempt; it rejects attack attempts by default to preserve memory; thus, it attracts more attackers and has been able to log to millions of attack attempts per day. Furthermore, a small code base eases its deployment, i.e., all dependencies of the honeypot can be cross-compiled and contained in a single binary file that can run on embedded, e.g., ARM devices such as Raspberry Pi.

```
for x in \$(seq 1 254); do
  ip route add local 143.x.\${x}.0/24 dev p5p4
  ip rule add from 143.x.\${x}.0/24 table darknet
done
ip rule add from 141.x.y.z dev p5p4 table darknet
```

Figure 2.2: Sample `ip route` rules to set up a darknet for a `/16` CIDR address space for an interface named `p5p4`.

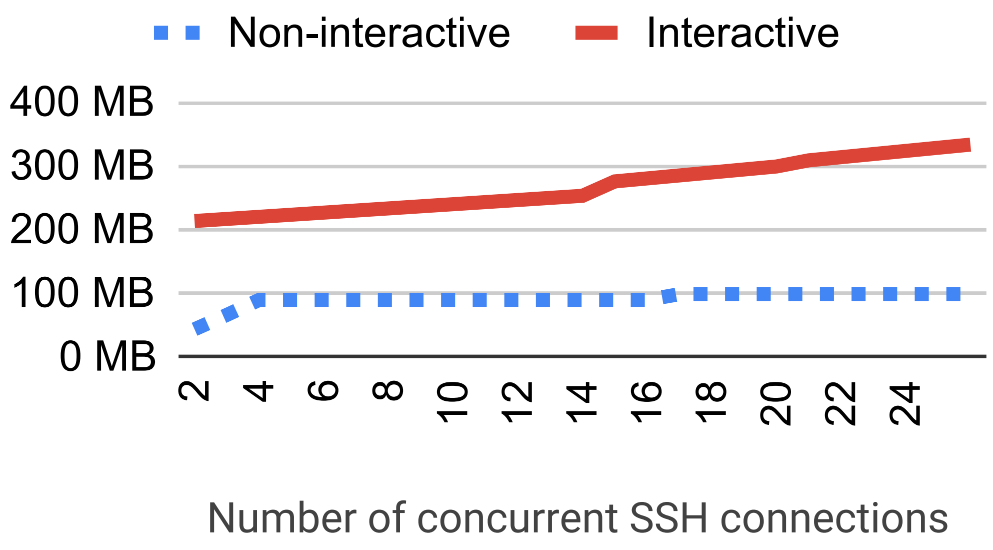


Figure 2.3: Comparison of memory footprint between an interactive honeypot and our non-interactive honeypot.

Attack attempts records. For each attack attempt, the SAL logs a 5-tuple record of the following data: SSH client version, SSH key fingerprint, source IP address, username, and guessed password (shown in Table 2.1). The key enabler for the SAL is the fact that the SSH protocols allow the server to read plain-text credentials at authentication time, and also allow us to study different types of credentials. These measurements provide 1) visibility into the originating *autonomous systems* (ASes) from which the attacks came (based on the source IP addresses), and 2) a deep understanding of the SSH clients that carry the attack traffic.

Identifying malicious scanners. Our goal is to have high precision in identifying malicious scanners (i.e., a low rate of false positives). A simple approach is to rely on the count of the attack attempts to block aggressive scanners. Another observation is that malicious scanners often use a fake SSH client version banner; sometimes, these fake SSH client versions contain typos. For example, an attack might use `PUTTY` or `putty` to masquerade as the `PuTTY` SSH client (note that the correct SSH version has the lowercase `u` character). Furthermore, in addition to fake versions, we have also discovered million-scale randomization of client versions, indicating a globally coordinated effort to evade detection and hide attack traces. We will disclose and illustrate more details in Section 4.3.

2.2 SSH Credential Auditor (SCA)

Despite that existing tools such as the UNIX `passwd` utility [26, 27] can give warnings for dictionary-based passwords and misconfigurations, none of the available tools can be used as a fire drill, i.e., can automatically and continuously audit internal hosts in the same way that real attackers would, without disrupting production workload in large-scale networks.

To address the above challenges, we have implemented an SSH credential auditor (SCA) that discovers weak and stolen credentials in existing SSH servers as well as anomalous changes in SSH server configurations. In contrast to existing tools that use default dictionaries, the SCA is driven by passwords that have been used by attackers who targeted the honeypot, collected by our SAL unit. Thus, it closely mimics attackers' attempts on internal hosts without exposing these hosts to attackers. SCA works as follows.

Discovery of internal SSH hosts. In a large-scale network, scanning an entire IP space, e.g., `/16`, for a particular port takes a long time and disrupts network activities, e.g., triggers false alerts. Similar to existing tools such as nmap [28], SCA periodically performs a full discovery on the entire network but only weekly, because a full discovery would be disruptive.

SCA minimizes the disruption of full scans on the production network by generating a list of suspected hosts based on the basic information provided by Zeek on the SSH protocol activities on the network. For example, Zeek can output a source/destination IP, SSH server banner, and client key fingerprint, based on the handshake of an SSH connection.

Table 2.1: Example records of SSH password- and key-based authentication.

Field	**Password-based record**	**Key-based record**
Source IP	`202.100.xxx.yyy`	`80.44.xxx.yyy`
Client Version [`SSH-2.0-`]	`sshlib-0.1`	`libssh-0.5.2`
SSH Key (SHA256)	N/A	`B6kr4...ORMeg`
Username	`admin`	`root`
Password	`7ujMko0admin`	N/A
Date	`2019-08-19`	`2018-12-25`

Audit of SSH hosts. The SSH credential auditor performs following the audit schedules.

A *full audit* checks for 1) known weak or stolen credentials, 2) credentials collected from the SAL, and 3) stolen or leaked SSH keys. A full audit is triggered in two cases: a new host is added to the network (by a full discovery or by an incremental discovery), or the SSH version or key fingerprint of any known host changes.

A *partial audit* only checks for new credentials, e.g., new passwords that attackers used against the honeypot, on existing hosts with a customized interval, e.g., every day.

Localization and isolation of weak/compromised SSH hosts. Our experiences with past incidents [29] have shown that an unexpected change in a server key fingerprint and server version is typically an indication that an attack is compromising the OpenSSH daemon. Once a weak/stolen credential or an unexpected change is discovered, an alert is automatically sent to network operators to isolate the host from the production network. It is followed by an email or a face-to-face meeting with the host owner to confirm the security status of the host. All network flows in/out of the host are reviewed for possible redirection into the BHR.

2.3 Black Hole Router (BHR)

Although the BHR seems logically similar to blacklisting of IP addresses, our BHR's goal is not to block 100% of the attack attempts, but to reduce the loads on existing network security monitors. There is only one global blacklist in the BHR, at the network border, that makes it easier to manage and to update the list (i.e., by removing inactive IP addresses from the list). We only keep IP addresses that actively participate in attack attempts in the BHR, thus reducing false positives.

Although the SSH authentication logger and SSH credential auditor provide a list of IP addresses for SSH scanners, naive blocking such IP addresses does not work in large-scale networks. Existing host-based blocking approaches such as `Fail2ban` [30] cannot manage a global list of blocking IP addresses in a large-scale network. Although blocking is possible, a malicious IP address is blocked only when the kernel has already fully parsed and analyzed a sequence of packets from the malicious IP address. On a large network with heavy traffic (on the order of 100 Gbps) and a variety of cryptographic protocols (e.g., IPSec, SSH, or TLS), network security monitors (NSMs) such as Zeek are often overloaded and thus drop packets. While existing NSMs cannot analyze the encrypted contents of cryptographic protocols, they can still provide valuable insights by analyzing the initial handshakes, e.g., alerting on the use of outdated SSL protocols, expired SSL certificates [31], or compromised SSH keys.

Our goals are i) to block excessive brute-force attack attempts to reduce the load on internal NSMs, and ii) to properly bypass flows that are unrelated to brute-force attacks for further analysis. To achieve these goals, we implemented our Black Hole Router (BHR) at the network perimeter by using the Border Gateway Protocol (BGP). The BHR works with the exit routers and manages a list of malicious IP addresses.

Null route. Immediately after a credential-guessing attempt is observed, the network flows carrying out excessive attacks (e.g., guessing of multiple usernames within a short period) from malicious IP addresses are redirected to a *null route*, which is a standard feature in BGP routers. The BHR discards the incoming network traffic without informing the source IP address of incompleteness of the network flow. In this way, the attackers are more likely to send more requests, intending to receive a proper response. As a result, the BHR reduces overall network load on the WAN border switch while allowing the honeypot to avoid excessive attacks.

Catch-and-release. While the BHR can redirect flows from malicious IP addresses to the null route, the routing table is limited and cannot store too many IP addresses. In our router configuration, the upper limit of the routing table is a million entries, and the number of malicious IP addresses makes up one-third of that (and could increase in the future). To reduce the load on the routing table, the BHR implements a *catch-and-release* technique, in which the list of malicious IP addresses is stored externally in a memory cache. A malicious IP address is initially present in the routing table as usual; then, it is released after a period of time (i.e., an hour) of inactivity. When there is a new flow from a new IP address, the flow is compared with the memory cache, and the malicious IP address is re-inserted into the routing table if it is present in the cache.

Flow shunting. We have implemented flow shunting for the Zeek IDS by using eXpress Data Path (XDP) [32], a programmable network data path in the Linux kernel. XDP preprocesses an incoming packet without early allocation of the *skbuff* [33] data structure in the networking stack in the Linux kernel or software queues. XDP works by looking at a specific offset in the packet, e.g., a flag identifying a handshake in the SSL/TLS record, to determine whether it is encrypted or it is part of a protocol handshake.

As a result, packets coming from malicious IP addresses and packets that contain encrypted data are shunted (discarded) before any further parsing (by kernel-level packet-capture mechanisms such as the Berkeley Packet Filter [32]) happens, except for that of the handshakes.

2.4 Alert-Sharing Network (ASN)

Large-scale networks employ a variety of monitoring tools and corresponding analysis techniques to provide comprehensive coverage. Network IDSs [34] perform deep packet inspection of network traffic for the detection of anomalous activity. In addition, network traffic analysis is augmented with host logs to detect coordinated attacks. While such alerts are often handled by a dedicated incident response team, recent attacks have happened across multiple institutions on a global scale. Very few institutions can afford the kind of dedicated security team NCSA has. Thus, a new coordination effort is needed to prevent such attacks. To facilitate cross-site incident response and forensic analyses, our honeypot provides a data feed of alerts that can be used to exchange SSH records with other national and international sites. Our honeypot is being used in the bidirectional exchange of security-alert-related information with one site, the Singapore University of Technology and Design. The major participating sites in the U.S. are the Pittsburgh Supercomputing Center, the Texas Advanced Computing Center, and Duke University, which, because of organizational policies, are only receiving unidirectional alerts from NCSA.

Site authentication and alert encryption. Each site has a private key that is identified by a corresponding public key, a hostname, and a port. Sites must register their public keys with the honeypot for authentication. Since sites exchange alerts that may contain sensitive personnel information and IP addresses, we encrypt alerts in transport by using NIST's recommended

Curve25519 cryptography [35]. To implement authentication and encryption, we utilize ZeroMQ [36], the high-performance message queue library that has been proven in financial applications with similar needs.

Site discovery. Sites use a simple gossip protocol for discovery and alert exchange [36]. First, a new site needs to be introduced to the network by a trusted peer. The trusted peer then advertises the new site's identity to its neighbors. Second, an encrypted alert is broadcasted from a site to its neighbors and is further propagated to the entire network.

Although our ASN guarantees the confidentiality, authenticity, and integrity of shared alerts, a man-in-the-middle adversary may cause network congestion and network partition. Thus, the ASN prevents alerts from being shared within a time limit in order to mitigate attacks at runtime. We will investigate techniques such as the use of redundancies [37] to send alert replicas across multiple network paths, thus maximizing the probability that the alert will be successfully delivered.

CHAPTER 3

DATA OVERVIEW

CAUDIT has attracted and collected a total of 11 billion attack attempts over 1,000-day operation from February 2017 to November 2019. The 11 billion attack attempts include 3.4 billion connections and 7.9 billion SSH password- and key-based brute-force attack records. In this chapter, we will present key findings from the longitudinal analysis of those SSH attack trends over this long period, in order to obtain summary statistics and provide insights into monthly trends of attack techniques.

3.1 Analysis Workflow

The main steps in our analyses are to discern the nature of attacks in terms of persistence (Chapter 3), identify coordination and evasion techniques (Chapter 4), and distinguish human-supervised and fully automated botnet attacks (Chapter 5). Figure 3.1 illustrates the logical flow of our approach with detailed descriptions as follow.

We first load our large dataset (3TB) into Clickhouse, a columnar database [38], for fast and distributed SQL queries on the dataset. Then, we present summary statistics and month-by-month trends of attack techniques, including SSH keys, passwords, usernames, SSH client versions, and IP sources. We focused on trend anomalies and persistent footprints over the entire honeypot operation. Second, we closely inspected attacker tactics (i.e., through SSH key and client version forensics) in terms of exploitation, coordination, and evasion. Finally, we quantify the degree of automation in attack strategies by comparing attack activities during the weekend and weekday. This analysis allows us to discern and distinguish human-supervised botnets with fully automated ones.

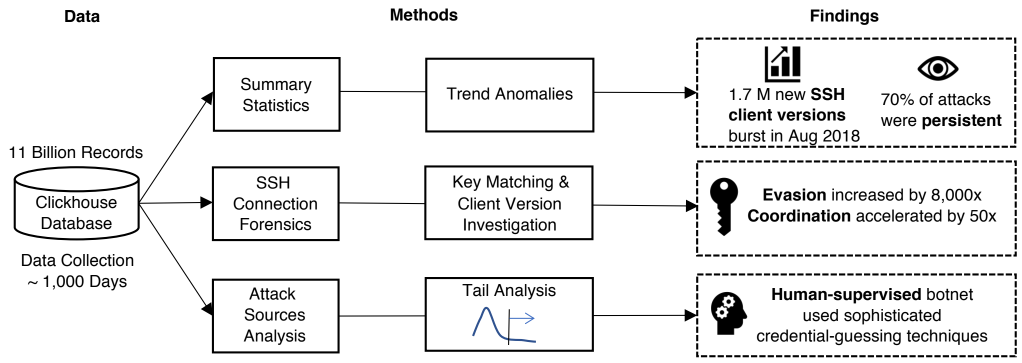

Data
Methods
Findings
11 Billion Records
Clickhouse Database
Data Collection ~ 1,000 Days
Summary Statistics
SSH Connection Forensics
Attack Sources Analysis
Trend Anomalies
Key Matching & Client Version Investigation
Tail Analysis
1.7 M new SSH client versions burst in Aug 2018
70% of attacks were persistent
Evasion increased by 8,000x
Coordination accelerated by 50x
Human-supervised botnet used sophisticated credential-guessing techniques

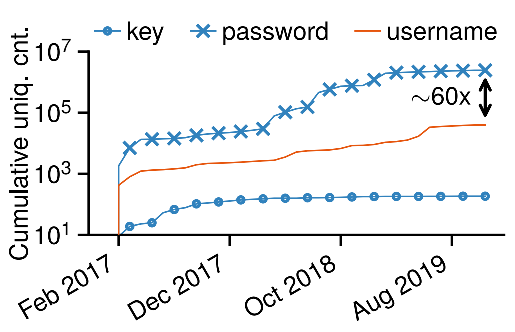
Figure 3.1: Analysis workflow to mine threat intelligence.

3.2 Trend Anomalies

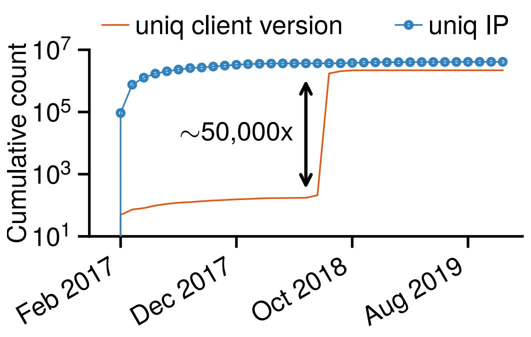
The anomalies in trend, including surgent client versions, dissipating active IPs versus rising active attack attempts, guided us to discover particular attacker tactics behind the scene.

Abrupt upsurge of unseen client versions from new attackers. The honeypot witnessed a dramatic increase of ~50,000× unique new-attempted client versions from August to October 2018. Eventually, as implied in Figure 3.2(b), aligned with the total 4 million unique IP sources, the final accumulative number of unique client versions also reached a scale of 2 million in the end. This number is abnormally high because the previous studies both reported just more than 100 version strings [8, 17]. Further inspection reveals attackers from a group of new-emerging, globally distributed IP sources were massively spoofing client versions in coordination. Contrary to the version trend, within the same month in August, increments of new IPs slowed down (Figure 3.2(b) and 3.2(c)). In Chapter 4.3, we present the detailed attack tactics underlying the million-scale SSH version string manipulation.

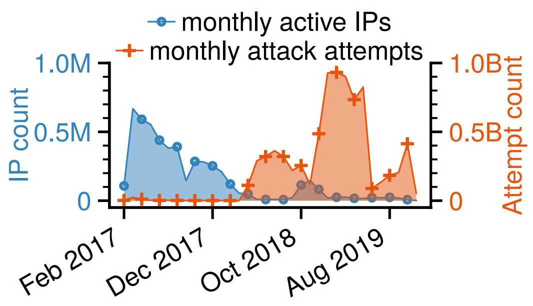
Increasing scale of attack attempts from fewer attackers. Figure 3.2(c) illustrates the monthly progression of active IPs and attack attempts in reverse trends. Due to changes in network policy to no longer block honeypot traffic, monthly attack attempts increased by 600 times in April 2018. On the other hand, while the honeypot did not filter certain IPs, the overall IPs were still decreasing. This implies that, after fruitless exploitation over a short period, most attackers in the wild were shifting targets rapidly. Combining the two reverse trends, we could conclude that number of attacks per IP were dramatically increasing over time. We spec-



(a) Monthly new-incoming unique usernames, passwords, and SSH keys.



(b) Monthly new-incoming unique IPs and client versions.



(c) Overlay of monthly active IPs (unique) and SSH brute-force attack attempts.

Figure 3.2: Cumulative trend plots depicting key features in more than 8 billion attack attempts spanning 1,000 days.

ulate that certain attackers remained persistent in exploiting our honeypot by repeatedly coming back or launching a large number of exploits at one time. We will characterize and present the activities and patterns of those persistent attackers below.

3.3 Persistent Attack Traces

We had observed four million unique IP addresses during the initial 463 days of honeypot deployment [17]. However, this number increased by only 0.5 million (12.5%) over the next 537 days. In contrast, total attack attempts increased by almost 20 times (from 405 million to 7.9 billion). We suspect certain persistent attackers were repeatedly coming back to attack the hon-

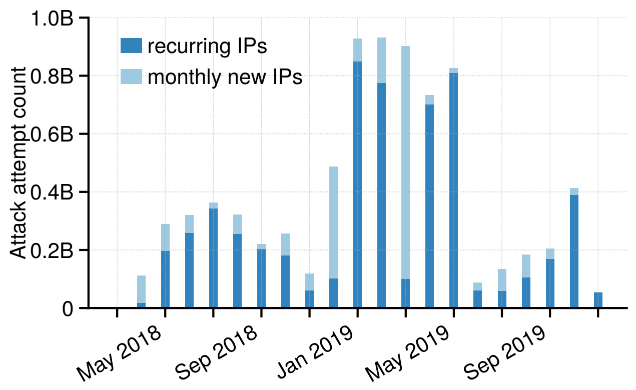


Figure 3.3: Monthly attack attempts comprising of monthly new-incoming IPs and cross-month recurring IPs.

eypot. Therefore, we present the statistics and patterns of attackers in terms of persistence as follow.

Persistent attackers constituted over 70% of all attacks. We categorize two types of IPs in Figure 3.3: recurring IPs and monthly new IPs. In contrast with monthly new IPs (with the first occurrence at the current month), recurring IPs repeatedly appear to attack the honeypot across different months. Hence their occurrences in the current month are recurrences. For recurring IPs, the maximum value of monthly average attempts per IP is 0.2 million, which is 5× that of new IPs. As Figure 3.3 implies, starting from May 2018, the trend of actively recurring IPs started to catch up with monthly new IPs. In general, total attacks from the recurring IPs are 2.5× the overall contribution from monthly new IPs, and those recurring IPs constituted more than 70% of all the attack attempts in the dataset.

Persistent attackers continuously attacked for over one year. To quantify recurring IP's degree of persistence, we investigated three metrics: attack time span, effective attack days, and the longest span of consecutive attack days. For each IP, We define "attack time span" as the time between the dates of that IP's first and last attack, "effective attack days" as the number of days with at least one attack attempt by that IP, and "longest consecutive days" as the highest number of consecutive effective attack days for that IP.

Even within a long time span, attackers may not actively attack every day. Our results show that four IPs have the longest attack time span of 1,000 days, which is as long as the time span of the honeypot's operation. However,

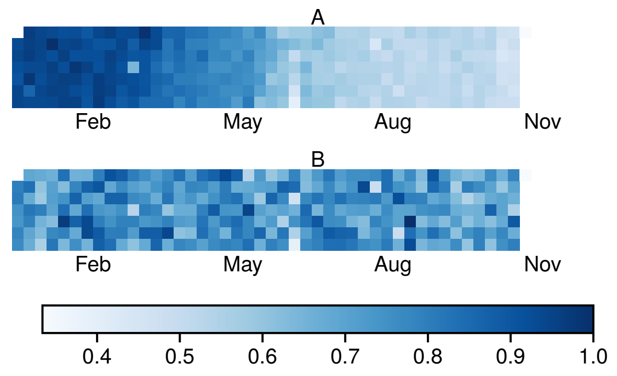


Figure 3.4: Normalized attack activities between two persistent IPs (A&B) in 2019 [M: Monday, T: Thursday, S: Sunday].

their effective attack days range from 62 to 221 days, which means these four IPs were only effectively attacking at most 22% of their time spans.

We then switched focus to attackers with a large number of effective attack days. The most significant value is 555 days, which was achieved by two IPs. Moreover, both IPs were actively attacking for 384 consecutive days – over an entire year. The top five IPs with the most effective attack days were all attacking consecutively over an entire year. In particular, these five IPs came from four different ISPs. In Figure 3.4, we select two persistent IPs from the top five to illustrate the different patterns in their normalized attack activities. Both IPs adopted naive password strategies: A rotated over 42 unique passwords almost every day; while B alternated almost equally between only two passwords every single day.

Implications. Persistent IPs are more prone to access a range of Internet-connected sites and devices [39]. Therefore, we advocate sharing the long-term threat intelligence with peer sites, so that corresponding network operators can preempt the aggressive threats by monitoring or flagging these persistent attackers in advance, saving both defense resources and time.

CHAPTER 4

EXPLOITATION, COORDINATION, EVASION

Chapter 3 provides an overview of the attack trend from longitudinal perspective. The findings including trend anomalies and persistence nature further motivate us to explore deeper into more detailed and advanced attack strategies, to be present in this chapter, in terms of exploitation (Section 4.1), collaborating (Section 4.2), and evasion (Section 4.3).

4.1 Exploitation of Leaked SSH Keys

In total, 185 unique SSH public key fingerprints (in the SHA-256 hash) found their way into our honeypot. By matching each of the keys with a public database and online files of bad keys [40, 41], we discovered and recovered the identities of seven keys that were publicly leaked due to vulnerabilities.

Further investigations implied that cybercriminals were trying to gain root permission to vulnerable production appliances and devices in the wild using

Table 4.1: Details of the seven leaked SSH keys (sorted by public disclosure year).

SSH Key (SHA256)	**Key Owner**	**Appliance Type**	**Public Disclosure Year**	**1st Attack Attempt Year**	**User-name**
`1M4Rz...`	Vagrant [42]	Base box for development environments	2010	2018	`root`
`9prMb...`	F5 [43]	BigIP appliances	2012		
`MEc4H...`	Loadbalancer [44]	Virtual load balancer	2014		
`VtjqZ...`	Quantum [45]	Virtual deduplication backup appliance			
`/JLp6...`	Array Networks [46]	Virtual application delivery controllers			`sync`
		Secure access gateways			
`Z+q4X...`	Ceragon [47]	IP traffic router	2015		`mateidu`
`f+1oG...`	VMware [48]	Data Protection appliances	2016		`admin`

these leaked keys, even years after the key-pertinent vulnerability disclosure. Their exploitation strategies are revealed in more detail below.

Attackers were targeting production devices using leaked keys. The seven leaked keys belonged to seven different enterprises (see Table 4.1). All these keys granted attackers with root permission in the targeted systems eventually. Among those keys in Table 4.1, the top four have direct root privileges. With the bottom three keys, attackers gain access as non-root users initially. However, either by exploiting local vulnerabilities (e.g., Array Networks [46] and Ceragon [47]) or becoming a sudoer without a password (e.g., VMware [49]), attackers eventually escalate privilege to root.

The attackers used the privilege level related to each corresponding leaked key when targeting our honeypot. In Table 4.1, the top four keys were all attempted with root permission, while the bottom three were attempted using the exact user names issued from key owners and related to disclosed vulnerabilities. For example, username `admin` from VMware can escalate to root without a password [50]. This observation indicates, instead of randomly using leaked keys to brute-force, the attackers have adequate details about pertinent vulnerabilities when plotting the targeted attacks.

Attackers were rapidly exploiting the leaked keys. Attacks that originated from Google LLC (Google) [51], Charter Communications [52], and Portlane [53] participated in exploiting the seven leaked keys. Table 4.2 presents an overview. As can be inferred from Table 4.2, two of the seven keys were used by all three autonomous systems (ASs); four of the seven keys were used by both Google and Charter Communications. In particular, attackers from Google tried all seven identified, leaked keys. In addition to the seven leaked keys, Google-owned IPs also exploited four other keys with unknown identities. We suspect that those four keys, though not identified yet, also

Table 4.2: A summary of the ASes that exploited the seven leaked SSH keys.

AS	**Client Version** [SSH-2.0-]	**SSH Key (SHA256) & Key Owner**						
		`1M4...`	`9pr...`	`MEc...`	`Vtj...`	`/JL...`	`Z+q...`	`f+1...`
		Vagrant	F5	Load--balancer	Quan--tum	Array Networks	Ceragon	VMware
Google	`libssh_0.7.0`	✓	✓	✓	✓	✓	✓	✓
Charter Comm.	`Ruby/Net...`		✓	✓	✓	✓	✓	✓
Portlane	`libssh-0.6.1`			✓	✓			

Google refers to Google LLC;
Charter Comm. refers to Charter Communications;
`Ruby/Net::...` refers to `Ruby/Net::SSH_5.0.2 x86_64-linux-gnu`.

belong to production devices with disclosed vulnerabilities. Coincidentally, all seven keys were attempted on the same day (Dec 14, 2018) by Google-originated attackers. In addition, IPs owned by Charter Communications attempted to exploit the five known leaked keys, together with three other unidentified keys, over two days (July 28-29, 2018). IPs from Portlane used two and only two known keys on another day (Mar 24, 2018). Therefore, we speculate that these attackers were rapidly switching targets for massive exploits of (outdated) vulnerabilities in order to access compromised devices in the wild by collecting and reusing old, sensitive SSH keys.

Challenges and limitations. Previous work [17] claimed no evidence of leaked keys. We suspect that was either due to a mismatched comparison between different hashing algorithms in SSH protocol (e.g., SHA256, DSS, RSA) or insufficient keys for analysis. In our work, we did the exact SHA256 conversion (`ssh-keygen -lf id_rsa.pub`) of the publicly available keys in online database and files. As a result, we have successfully pinpointed matches between public datasets and recorded attempts at our honeypot. Therefore, these matches further allowed us to trace attack origins, behaviors, and strategies.

In addition to a proper conversion method, the challenges in SSH key matching also lie in the limited availability of a public, formatted database of leaked SSH keys. We matched our keys with one public bad SSH key database [40], which only contained nine keys with direct access to devices. To expand the search scope, we also manually looked for leaked keys listed in the files at Packet Storm [41]. The entire search, however, is far from automated and complete, and thus leaves the remaining 178 keys unidentified.

4.2 Key-based Collaboration

We inspected the diversity of attack sources using SSH keys in general, from which we uncovered global coordination.

An SSH key was exploited by 20 countries. We sorted key-based attempts characterized by the number of the originating IP address, presenting the top ten in Table 4.3. Each of the ten keys originated from more than 15 distinct IP addresses, with the highest number being 71. However, most attackers originated from just a single country or AS. The only exception

Table 4.3: Top ten SSH key fingerprints from most diverse IP sources (reversely sorted by number of originating countries).

SSH Key (SHA256)	**# Countr(y/ies)**	**# AS(es)**	**# IPs**	**Client Version** [SSH-2.0-]
qlIN/...	**20**	**38**	**64**	Go
B6kr4...	1	2	25	libssh-0.5.2
mumiE...	1	1	49	libssh-0.5.2
jSCqa...	1	1	42	libssh-0.5.2
V6OOC...	1	1	28	libssh-0.5.2
zPA6Y...	1	1	23	libssh-0.5.2
NH5Y7...	1	1	19	libssh-0.5.2
OyHmn...	1	1	17	libssh-0.5.2
8blLD...	1	1	16	libssh-0.5.2
+UJNI...	1	1	71	kthrssh__x00

is the first key in Table 4.3, which was used by 64 IPs scattered over 20 countries from four continents (including Asia, Europe, North America, and Oceania).

A persistent, single-country botnet versus a rapid, globally colluding botnet. Further inspection revealed that this globally coordinated botnet exploited a single SSH key 90 times within only four days (Dec. 11 to Dec. 14, 2017). On the other hand, the last key in Table 4.3 originated from 71 IPs, yet all from a single country and AS. In contrast with the global botnet, this botnet persistently used one key for 2,700 times spanning five months (Feb. 2017 to July 2017). Compared with this single-origin bot, we can conclude that the globally coordinated bot wrapped up its fruitless attacks and shifted targets 50× faster.

4.3 Client Version-based Coordination and Evasion

Starting from August to October in 2018, the honeypot witnessed an unprecedented emergence of unseen client versions. More than 1.7 million new client versions sprang up in August alone, which was over 8,000× more than the total number of unique client versions in the previous 18 months. Further, inspection revealed that only several hundred IPs spoofed client versions by randomizing over one million OpenSSH version banners. This is unusual because, among all attackers, about 90% of IPs advertised only one client version. We speculate these randomizations were the attackers' mimic technique responding to our honeypot's deceitful defense mechanism.

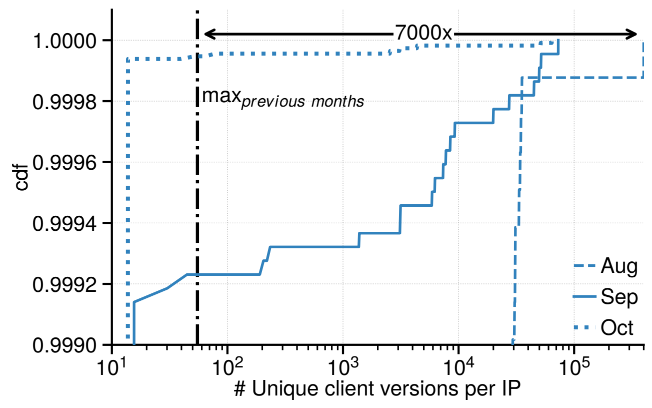


Figure 4.1: CDFs indicate over million unique client versions in total during August - October 2018.

Attackers randomized SSH client version banners at high frequencies. We sorted those unique client versions by the number of occurrences in August. The distribution of occurrences turned out to have an extremely long tail: 88 out of 1.7 million client versions occurred over 2,000 times each; ten out of 1.7 million occurred over ten times, while the rest 1.7 million distinct client versions just occurred at most nine times each. Moreover, the top-spoofing IP advertised 400,000 unique client versions during its 200-hour attack campaign, implying varying an average of 2,000 client versions per hour. This attacker advertised `SSH-2.0-OpenSSH_7` within first several days of attack, then switched to massive spoofing by appending `SSH-2.0-OpenSSH_` with five-character random strings (e.g., `+qLfH`). Plus, contributions from other attackers adopting the same technique spoofed random strings had more than a million permutations in total.

A globally-coordinated botnets were involved in forging a million permutations of client versions. As illustrated in Figure 3.2(b), the pace of new-incoming IPs even slowed down in August 2018. We initially suspected a large number of recurring botnets were scheming the large-scale randomization. However, only several hundred IPs were involved, and over 85% of them were new-incoming IPs in August. On the other hand, the earliest IPs involved in spoofing launched its first attack in February 2017 – as early as the honeypot's initial operation, yet these IPs did not start spoofing client versions until August.

Around 90% IPs advertised only one client version. In the long tail of the unique number of client version per IP distribution, further investigation showed that less than 300 IPs, yet globally coordinated from over 30 countries across six continents (all excluding Antarctica), actually accounted for the million-scale random permutations of client versions to masquerade their attack traces. Figure 4.1 presents the CDFs of client versions per IP tail distribution during Aug – Oct 2018, with a maximum number of 400,000 in August, which was over 7,000× its counterpart in all previous 18 months, with the maximum number of unique client versions per IP being 55.

Defense-targeting evasion. The honeypot deceives attackers by randomizing key fingerprints for each of the 65,536 servers on the entire `/16` IP address space [17]. We, therefore, suspect that the attackers were mimicking our honeypot's defense mechanism responsively. Besides, attackers were massively hiding essential attack signatures, which would generally invalidate signature-based detection. As a result, it calls for needs to deploy new defense strategies such as rate-limiting or anomaly-based detection.

CHAPTER 5

HUMAN-SUPERVISED ATTACK TECHNIQUES

This chapter analyzes human working activity patterns embedded in a large-scale of time-series attack data (Chapter 5.1). After discovering working patterns of human attackers on a weekly basis, we further provide case studies to compare and contrast the distinctive behavior patterns and strategies between fully automated botnets and human-supervised botnets (Chapter 5.2).

5.1 Human-supervised attacks

The human factor is an essential contribution to launching sophisticated cybersecurity attacks, e.g., ransomware propagation, advanced tool development at targeted victims [54, 55]. Revealing human attacker evidence aids in the detection of sophisticated underlying strategies that automated bots alone cannot accomplish. However, it is non-trivial to distinguish both techniques in large-scale security data.

Current work [5, 11, 56, 57] implemented additional features to capture human-generated activities, e.g., keyboard/mouse typing/clicking, window resizing, typing pauses, and mistakes. However, these methods introduced overhead to networking system design. Instead of modifying or adding features to the current design, the billion-scale attack attempts motivate us to come up with a data-driven methodology for mining human activity patterns.

Tail analysis of attack distributions. We take regional time differences into account when investigating evidence of routine human activities. In practice, we pinpointed one specific time zone and focused only on IP addresses originating from this time zone. Then we chose a month with the most attack attempts, to increase the chance of finding evidence of human activities among source IPs.

After grouping by IP, we computed average weekday and weekend attempts

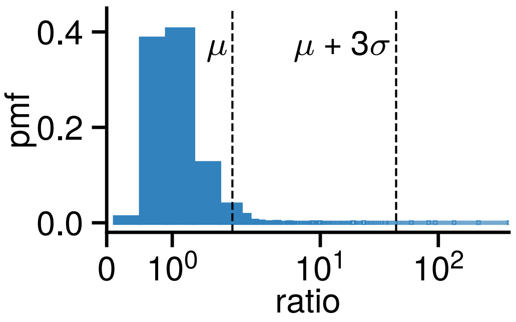


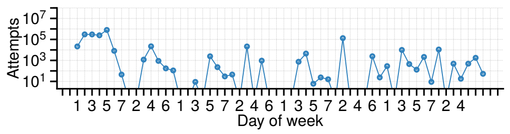
Figure 5.1: Distribution for ratio of weekday to weekend average attempts in June 2019.

for each IP during the selected month. To quantitatively capture routine human evidence, we calculated a ratio of a weekday to weekend average attempts for each IP. Since we aimed to find relatively long-term (4–6 weeks) evidence, we filtered out IPs with the number of active weekdays lower than 15. Figure 5.1 presents the ratio distribution after filtering, which drew our attention to the tail. Specifically, we then focused on IPs with the ratio Z-score [58] greater than three standard deviations (3σ) from the mean (μ), as shown by the tail on the rightmost part of the distribution.

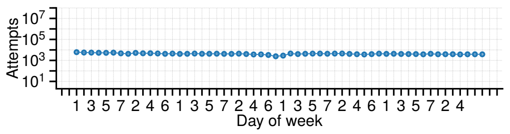
Activity patterns of the human-supervised botnet. It turned out that all IPs in the tail, with similar activity patterns, originated from the same `/8` subnet, indicating organized routine management of botnet by the human attacker(s). These IPs also used the same client versions, passwords, and usernames. The daily intensity of these bots indeed aligned with human social routine on a weekly basis: periodic variations with decreasing activities on weekends (especially Sundays). As an example, we illustrate daily activities of one such IP over eight weeks in Figure 5.2(a), a representative of a human-supervised bot. Further inspection revealed that its average attack attempts during working hours (i.e., 9 am–5 pm) are over $2\times$ the counterpart outside working hours.

5.2 Case Studies of Two Botnet Types

To fairly compare between two distinct botnet behaviors, over the same duration, we selected another IP with a weekday to weekend average attempt ratio equaling to one. Its daily activities are illustrated in Figure 5.2(b), a



(a) Human-supervised attack attempts declined on weekends.



(b) Fully automated attack patterns were almost unvarying over different days of the week.

Figure 5.2: An illustration of daily attempts for human-supervised and fully automated botnet [May 27 – July 21, 2019 (8 weeks)].

representative of a fully automated bot. In general, Table 5.1 provides an overview of feature-wise comparisons in terms of main authentication credentials. The duration we have selected is representative of the entire attack campaign for both IPs. Furthermore, more detailed case studies to distinguish the attack strategies adopted by both botnet types are provided as follow.

Human-supervised botnet is more resourceful in terms of attack devices. The entire human-supervised botnet shared the four client versions listed in Table 5.1. All bots iterated over these four client versions with equal distribution for each. There were cases when these four client versions were used at the same time by one bot. On the other hand, the fully automated bot advertised one and only one commonly used client ver-

Table 5.1: Human-supervised versus fully automated botnet.

Type	**List of Unique Client Versions** [SSH-2.0-]	**List of Unique Username(s)**	**Number of Unique Passwords**

Human-supervised	`PuTTY`
	`OpenSSH_5.3`
	`OpenSSH_6.2p2 Ubuntu-6`
	`nsssh2_4.0 NetSarang Computer, Inc`
		`root`	35,952

Fully automated	`sshlib-0.1`	`root`, `user`, `admin`, `ubnt`, `usuario`, `pi`, `supervisor`, `support`, `service`, `mother`	42

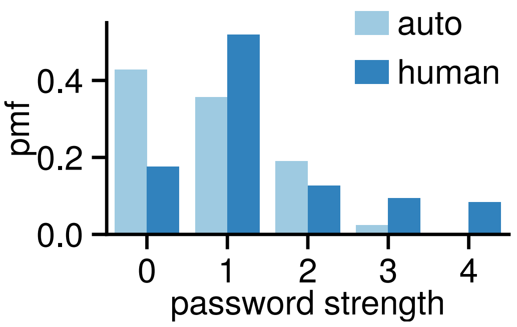


Figure 5.3: Password score distribution from two types of botnets.

sion. Therefore, human-supervised botnet employed a more diverse handful of devices to launch attacks.

Human-supervised botnet is more ambitious and strategic in terms of credential brute-forcing. We used Dropbox zxcvbn [59] to measure password strength. Figure 5.3 summarizes the score distribution for both botnets. For a fully automated bot, only one password (`7ujMko0admin`) scores 3, which is the highest among all 42 unique passwords it attempted, with the majority scoring 0. On the other hand, around 3,000 passwords used by the human-supervised botnet score 4.

Notably, one password used by human-supervised botnet begins with `Branch:masterFindfileCopypath`, and ended with a path in a Github repository [60]. This Git repo contains a wide range of passwords collected from backdoors, web shells, mail servers, WebLogic, Linux OS, dictionaries, etc. In addition to passwords, we also found collections of database and backdoor file paths, plus a script for brute-forcing enterprise mail servers, including Exchange [61], etc.

On the other hand, fully automated bot rotated all 42 passwords every day over the entire attack campaign. Most passwords are commonly used default passwords in Linux OS, IoT devices, routers, and firewalls.

CHAPTER 6

RELATED WORK

This chapter discusses prior work in honeypot design, security auditing, malicious traffic routing, and alert sharing networks.

Honeypots. Table 6.1 presents a historical overview of various honeypots over time in terms of characteristics, utilities, and limitations. HoneyStat [62] has been deployed for local worm detection. However, 1) it is deployed on local networks whereas ours is deployed on NCSA's global peer-to-peer sharing infrastructure; and 2) it only carries out logit analysis for worm detection, and thus lacks a mechanism for protecting inner network infrastructure from honeypot intrusions, while NCSA applies auditing tools to preempt compromises based on honeypot intrusion data. John et al. [63] deployed heat-seeking honeypots in a university network, which attracted $\sim 44,000$ attacker visits from $\sim 6,000$ distinct IPs, which inspired NCSA's honeypot deployment in a similar campus deployment environment. However, NCSA's honeypot traffic is at least 1,000 times greater. With that relatively limited attack surface, these Web server-based honeypots rely on other Web pages with high page ranks and dynamic linking of search engines to attract up-to-date or zero-day attacks. Therefore, our non-interactive honeypot is scalable: it can handle an order of magnitude more attack attempts compared to interactive honeypot such as Kippo [22], which is more expensive to maintain. Such interactive honeypots allow attackers to interact with a shell: thus, they require more resources and need careful network configuration (blocking of new outgoing connections) to isolate attackers.

Provos [23] presented a framework that simulates virtual honeypots and Vrable et al. [64] built a prototype of virtual honeyfarm system, both in opposition to a physical one, with Vrable's honeyfarm system inspired NCSA's honeypot deployment. Provo's work was driven by the IP space limitations placed on traditional physical honeypots, and that is not an issue for the NCSA honeypot currently. The effectiveness of building deceptive honeypots

Table 6.1: An overview of historical honeypots.

Publication Year	Honeypot	Char-acteristics	Utility	Critiques
2004	HoneyStat [62]	Low false positives	local worm detection	Deploy on local network No mechanism for inner network protection
2004	Honeyd [23]	Virtual framework	Network decoy, worm detection, spam prevention	No physical infrastructure Only cover thousands of IP spaces
2005	Potemkin [64]	Improved emulation scability of virtual honeyfarm	Large- -scale detection with high fidelity	Limited scope of attracted attacks Easy to infer for attackers
2007	VoIP [65]	Voice over IP SIP specific	Defend against VoIP threats	Lack of feed-back decision mechanism
2011	Heat-seeking honeypot [63]	Web server-based	Analyze attacker behaviors towards vulnerable servers	Deploy in a small scale with limited attack surface Rely on high page ranks and dynamic linking Not easy to scale
2011	Artemisa [66]	Back-end to VoIP domains	Early stage detection SPIT mitigation	No real-world deployment No attack trace collected
2016	Kippo [22]	Medium-interaction	Collect exploits and malware	Require more resources Expensive to maintain No longer active
2017	Deceptive honeypots [67]	Controlled experiment between Linux containers, bare metal, and virtual machines	Study deceptive virtualization capabilities of Linux containers	Easy to infer from attacker perspective Fundamental flaws in the design of system architecture [68]
2019	Honeycloud [11]	Hardware and specifically-designed software IoT honeypots	Analyze fileless attacks	Impose overhead by implementing additional features to capture human attacker activities

SIP refers to Session Initiation Protocol;
SPIT refers to SPAM over Internet Telephony.

within a virtual environment was further investigated using Linux containers [67]. However, [68] illustrated the limitations of these virtual honeypots from both attackers and system architecture's points of view. From the attackers' viewpoint, there is the ease of detection without any privileges; and in the underlying system architecture, there were fundamental flaws in virtualization.

There have also been studies in VoIP honeypots [65, 66]. The main limitation of [65] is a lack of decisions in reaction to attackers, compared to the NCSA honeypot's deployment of real-time decision infrastructure based on the collected data from the honeypots. Artemisa [66] has only been implemented in a preliminary stage; it has not been deployed on a large scale, and its honeypots do not maintain interactions with the rest of the security components, leading to delayed enforcement of security policies in response to real-time dynamic attacks.

Security auditing. Table 6.2 presents an overview of various auditing and scanning techniques. Prior work on security auditing has focused on preventing SSH brute-force attacks [20, 34, 69] using multi-factor authentication [70], deceiving attackers using honeypots [21, 23, 64], Internet-scale scanning of vulnerable hosts [71], mimic real attacks to audit internal hosts [72], and use of BHR to block blacklisting-based malicious IP addresses [73]. Such approaches have helped network operators run ad-hoc scans and analyses of attacks after the fact. Participants in *red teams* and *bug bounty programs* emulates malicious behavior of attackers to uncover security vulnerabilities and bugs [74]. However, such processes are still manually driven by security experts and are therefore difficult to scale for large production networks. The above techniques are often ineffective in practice, e.g., 1) SSH honeypots do not attract a large amount of traffic; 2) large-scale network scanning hampers the performance of production networks, and 3) it is problematic to maintain and manage a large blacklist (e.g., because of false-positive filtering). In addition, coordinated attacks can be thwarted using security intelligence sharing and analysis between geo-distributed sites [75]. Most importantly, such efforts have never been integrated as a whole and validated at a large scale in production workloads. Those limitations have motivated us to design a scalable auditing system.

Routing malicious traffic. Yu et al. [76] proposed a precise network instrumentation framework to mitigate malicious traffic, e.g., by forcing the

Table 6.2: An overview of auditing and scanning techniques.

Release/ Publication Year	Auditing & Scanning Tool	Utility	Similarities with CAUDIT	Comments
1994	Zeek [34]	Network Security monitor	Deep packet inspection	Often overloaded and drop packets
1997	nmap [28]	Large-scale network scanning	Full discovery of the entire IP space	Motivate CAUDIT's design
2009	uRPF [73]	Blocking malicious IP addresses by Black Hole Routers	Keeping track of malicious attackers	Hard to scale and maintain for a large blacklist
2011	Chaos Monkey [72]	Resilience tool to randomly terminate production instances	Mimic real attacks to audit internal hosts	Disrupt production services
2013	ZMap [71]	IPv4 address space scanning	New protocol analyses	Random address space scanning
2013	Distributed SSH brute-forcing detector [69]	Detecting distributed and stealthy SSH brute-forcing	Preventing and studying SSH brute-forcing	Detection manually driven by security experts

user to change the default password of an IoT device. However, in our approach, the user is redirected to the null route instead. Wu et al. presented a packet filter [77] with low filter update latency and high-speed packet processing. 007 [78] is an application deployed to diagnose, detect, and trace source causes for TCP connection packet drops. APUNet integrated GPU in APU platforms to accelerate packet processing in network applications [79], while, on the other hand, Netmap enables rapid network packet delivery without requiring customized hardware or modified applications [80].

Sarma et al. broadened the availability of hardware switches for network resource allocation algorithms, thus making the implementations of network protocols more flexible [81]. To achieve easier and more efficient network flow processing development in stateful middleboxes, Jamshed et al. designed and implemented a reusable network stack [82].

Alert-sharing network. R-cisc is a cybersecurity sharing center for the retail ecosystem [75]. However, unlike NCSA's sharing network, that sharing center shares security incident data among retail sites in a manner that is neither real-time nor encrypted. The publish of new threats in Facebook ThreatExchange [83] is not automated. On the other hand, it is promising to integrate NCSA's alert-sharing network with Facebook ThreatExchange and IBM X-Force to make use of the APIs for threat intelligence sharing [83, 84].

CHAPTER 7

FUTURE WORK AND CONCLUSION

In this chapter, we discuss potential future work and conclude the thesis.

7.1 Future Work

Future work spans advanced strategies involving SSH key identification, client version detection, and threat intelligence sharing.

SSH keys. The identified SSH keys (especially disclosed in recent years) alert key owners and IoT appliance operators to i) investigate the coverage of patches for outdated vulnerabilities and ii) examine whether affected devices and users still exist in the wild. On the other hand, for unknown keys, we can speculate their identities or targeted devices based on associated usernames (e.g., `raspberry`) and client versions, so that we can accordingly broadcast unknown keys to in advance as precautions for zero-day exploits toward targeted devices.

Client versions. Attackers' massive evasion techniques to bypass signature detection motivate rate-limiting or anomaly-based detection. Moreover, we plan to detect fake and spoofed SSH version banners with [8] and [85]. Both methods require additional SSH information (e.g., key exchange algorithms, encryption methods), which is not available at present. On the other hand, apart from fake banners, attackers also advertised over 50 seemingly legitimate client versions (the vertical line in Figure 4.1). A further inspection into the landscape and dynamics of such resourceful attackers will benefit the design of effective defense strategies and mitigation operations.

Privacy-preserving insight sharing. In terms of threat intelligence sharing across different sites, we suggest employing Private Intersection-Sum [86] to 1) preserve the privacy of sensitive information to be shared and 2) establish anonymous hacker identifiers. In the long term, the intelligence gathered from the honeypot can be leveraged in the automatic learning and upgrading of defense systems, thus building up to AI-driven intrusion detection that can respond automatically to unknown threats based on past evidence.

7.2 Conclusion

In this thesis, we first present the operational experiences with the proposed framework CAUDIT deployed in operation at NCSA. Then, we investigated a broad scope of attack strategies in billion-scale SSH brute-force attacks targeting at the operational system. Key findings and implications from the longitudinal study of SSH brute-force attacks include:

- **Persistent attacks versus cross-country attacks.** Persistent attackers constituted over 70% of total attack attempts; some of them have been brute-forcing consecutively for over an entire year. While the total number of unique attack sources (4M IPs) does not increase significantly, we saw a 20× increase in attack attempts since the public disclosure of the honeypot [17]. On the other hand, attackers from 20 countries across four continents rapidly exploited a particular SSH key over four days – 50× faster than a single-country botnet even with more IP sources, implying a global coordination effort. The finding indicates that most attackers were rapidly shifting targets after fruitless exploitation within a short period. At the same time, a few persistent attackers were relentlessly coming back to the honeypot and posing aggressive threats to production systems.
- **Leaked SSH keys exploitation.** Attackers from the major cloud providers and ISPs (e.g., Google or Charter Communications) used leaked SSH keys of IoT appliances (e.g., Enterprise VA load balancer [44] or VMware data protection appliances [48]). Specifically, we have identified seven leaked SSH keys. In particular, attackers originated from Google-owned IPs rapidly exploited all the seven leaked keys in one day. Further investigations implied that cybercriminals were trying to gain root permission to vulnerable production appliances and devices in the wild using these leaked keys, even years after

the key-pertinent vulnerability disclosure.

- **Large-scale evasion techniques.** A globally coordinated botnet (from 30 countries across six continents) spoofed millions of unique client versions for three months, which was over 8,000$\times$ the counterpart in the previous 18 months. This finding implies that new bot blocking techniques such as rate limiting or anomaly-based blocking should be deployed instead of signature-based blocking based on known client versions.

- **Human-supervised botnets.** We found a group of human-supervised bots, operating from the same /8 subnet, that only executed the attack during working days. Compared to the fully automated botnet that attacks constantly, the human-supervised botnet employed more diverse devices and planned more strategically in credential brute-forcing. Security operators can integrate our approach into defense strategies to distinguish human-driven botnets and build bot-specific defense models.

Therefore, we discover the great potential in attackers to launch large-scale, persistent, and evasion attacks that are accompanied by strategic human supervision. Also, we contribute methodology to reveal bot collaboration campaign in the wild, offer a scientific data-driven approach to differentiate between human-supervised versus fully automated botnet, as well as motivate privacy-preserving threat intelligence sharing for AI-driven intrusion detection in the ultimate goal.

REFERENCES

[1] N. DeMarinis, S. Tellex, V. Kemerlis, G. Konidaris, and R. Fonseca, "Scanning the internet for ROS: A view of security in robotics research," *arXiv preprint arXiv:1808.03322*, 2018.

[2] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks.* ACM, 2015, p. 5.

[3] "Have I Been Pwned," 2020. [Online]. Available: https://haveibeenpwned.com/

[4] P. Cao, E. Badger, Z. Kalbarczyk, R. Iyer, and A. Slagell, "Preemptive intrusion detection: Theoretical framework and real-world measurements," in *Proc 2015 Symposium and Bootcamp on the Science of Security.* ACM, 2015, p. 5.

[5] D. X. Song, D. A. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on SSH," in *USENIX Security Symposium*, vol. 2001, 2001.

[6] J. Owens and J. Matthews, "A study of passwords and methods used in brute-force SSH attacks," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[7] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving SSH-style host authentication with multi-path probing," in *USENIX Annual Technical Conference*, vol. 8, 2008, pp. 321–334.

[8] V. Ghiëtte, H. Griffioen, and C. Doerr, "Fingerprinting tooling used for SSH compromisation attempts," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*, 2019, pp. 61–71.

[9] "Expert found a hardcoded SSH key in Fortinet SIEM appliances," 2020, https://securityaffairs.co/wordpress/96649/security/hardcoded-ssh-key-fortinet.html.

[10] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis et al., "Understanding the Mirai botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[11] F. Dang, Z. Li, Y. Liu, E. Zhai, Q. A. Chen, T. Xu, Y. Chen, and J. Yang, "Understanding fileless attacks on Linux-based IoT devices with HoneyCloud," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2019, pp. 482–493.

[12] P. Muncaster, "Poorly secured SSH keys exposing firms to breaches." [Online]. Available: https://www.infosecurity-magazine.com/news/poorly-secured-ssh-keys-exposing/

[13] J. Vijayan, "Poorly managed SSH keys pose serious risks for most companies." [Online]. Available: https://www.computerworld.com/article/2488012/poorly-managed-ssh-keys-pose-serious-risks-for-most-companies.html,year=2014

[14] "ShapeShift." 2020. [Online]. Available: https://shapeshift.io/#/coins

[15] E. Voorhees, "Looting of the fox: The story of sabotage at ShapeShift," 2016. [Online]. Available: https://news.bitcoin.com/looting-fox-sabotage-shapeshift/

[16] S. Higgins, "ShapeShift lost $230k in string of thefts, report finds," 2016. [Online]. Available: https://www.coindesk.com/digital-currency-exchange-shapeshift-says-lost-230k-3-separate-hacks

[17] P. M. Cao, Y. Wu, S. S. Banerjee, J. Azoff, A. Withers, Z. T. Kalbarczyk, and R. K. Iyer, "CAUDIT: Continuous auditing of SSH servers to mitigate brute-force attacks," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19).* Boston, MA: USENIX Association, Feb. 2019. [Online]. Available: https://www.usenix.org/conference/nsdi19/presentation/cao pp. 667–682.

[18] "SDAIA." 2019. [Online]. Available: https://wiki.ncsa.illinois.edu/display/cybersec/SDAIA

[19] Y. Wu, P. Cao, A. Wither, Z. T. Kalbarczyk, and R. K. Iyer, "Mining threat intelligence from billion-scale SSH brute-force attacks," in *Workshop on Decentralized IoT Systems and Security (DISS)*, 2020.

[20] R. J. McCaughey, "Deception using an SSH honeypot," Naval Postgraduate School Monterey United States, Tech. Rep., 2017.

[21] P. Wang, L. Wu, R. Cunningham, and C. C. Zou, "Honeypot detection in advanced botnet attacks," *International Journal of Information and Computer Security*, vol. 4, no. 1, pp. 30–51, 2010.

[22] "Kippo - SSH honeypot." [Online]. Available: https://github.com/desaster/kippo

[23] N. Provos et al., "A virtual honeypot framework," in *USENIX Security Symposium*, vol. 173, 2004, pp. 1–14.

[24] C. J. B. Abbas, L. J. G. Villalba, and V. L. López, "Implementation and attacks analysis of a honeypot," in *International Conference on Computational Science and Its Applications.* Springer, 2007, pp. 489–502.

[25] "A low/zero interaction SSH authentication logging honeypot." [Online]. Available: https://github.com/ncsa/ssh-auth-logger

[26] "passwd(1) - Linux manual page." [Online]. Available: http://man7.org/linux/man-pages/man1/passwd.1.html

[27] "SSH server auditing." [Online]. Available: https://github.com/arthepsy/ssh-audit

[28] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning.* Insecure, 2009.

[29] A. Sharma, Z. Kalbarczyk, R. Iyer, and J. Barlow, "Analysis of credential stealing attacks in an open networked environment," in *Network and System Security (NSS), 2010 4th Intl. Conf on.* IEEE, 2010, pp. 144–151.

[30] "Fail2ban." 2016. [Online]. Available: https://www.fail2ban.org/wiki/index.php/Main_Page

[31] L. Zhang, D. Choffnes, D. Levin, T. Dumitras, A. Mislove, A. Schulman, and C. Wilson, "Analysis of SSL certificate reissues and revocations in the wake of Heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference.* ACM, 2014, pp. 489–502.

[32] "BPF and XDP reference guide." [Online]. Available: https://cilium.readthedocs.io/en/latest/bpf/

[33] G. Insolvibile, "Kernel korner: Inside the Linux packet filter," *Linux Journal*, vol. 2002, no. 94, p. 7, 2002.

[34] "The Zeek network security monitor." 2020. [Online]. Available: https://zeek.org

[35] E. Barker, L. Chen, S. Keller, A. Roginsky, A. Vassilev, and R. Davis, "Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography," National Institute of Standards and Technology, Tech. Rep., 2017.

[36] P. Hintjens, *ZeroMQ: messaging for many applications.* "O'Reilly Media, Inc.", 2013.

[37] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using redundancy to cope with failures in a delay tolerant network," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 109–120.

[38] "Clickhouse DBMS." 2020. [Online]. Available: https://clickhouse.yandex/

[39] D. Mashima, Y. Li, and B. Chen, "Who's scanning our smart grid? Empirical study on honeypot data," in *2019 IEEE Global Communications Conference (GLOBECOM).* IEEE, 2019, pp. 1–6.

[40] "SSH bad keys." [Online]. Available: https://github.com/rapid7/ssh-badkeys

[41] "Packet Storm." [Online]. Available: https://packetstormsecurity.com/

[42] "Insecure keypair." [Online]. Available: https://github.com/hashicorp/vagrant/tree/master/keys

[43] H. Moore, "Scanning for vulnerable F5 BigIPs with Metasploit." 2012. [Online]. Available: https://blog.rapid7.com/2012/06/11/scanning-for-vulnerable-f5-bigips-with-metasploit/

[44] "Loadbalancer.org Enterprise VA 7.5.2 static SSH key." 2014. [Online]. Available: https://packetstormsecurity.com/files/125754/Loadbalancer.org-Enterprise-VA-7.5.2-Static-SSH-Key.html

[45] "Quantum DXi V1000 2.2.1 - static SSH key." 2014. [Online]. Available: https://www.exploit-db.com/exploits/32372

[46] "Array Networks vxAG / xAPV privilege escalation." 2014. [Online]. Available: https://packetstormsecurity.com/files/125761/Array-Networks-vxAG-xAPV-Privilege-Escalation.html

[47] "Ceragon FibeAir IP-10 SSH private key exposure (CVE-2015-0936)." [Online]. Available: https://gist.github.com/todb-r7/5d86ecc8118f9eeecc15

[48] "VMware VDP known SSH key." 2017. [Online]. Available: https://packetstormsecurity.com/files/143883/VMware-VDP-Known-SSH-Key.html

[49] "CVE-2016-7456." 2017. [Online]. Available: https://packetstormsecurity.com/files/cve/CVE-2016-7456

[50] "Vmware VDP known SSH key." 2016. [Online]. Available: https://www.rapid7.com/db/modules/exploit/linux/ssh/vmware_vdp_known_privkey

[51] "AS15169 Google LLC." [Online]. Available: https://db-ip.com/as15169

[52] "AS20001 Charter Communications Inc." [Online]. Available: https://db-ip.com/as20001

[53] "AS42708 GleSYS AB." [Online]. Available: https://db-ip.com/as42708

[54] "Cisco 2018 annual cybersecurity report," 2018. [Online]. Available: https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf

[55] "More than 99% of cyberattacks rely on human interaction." 2019. [Online]. Available: https://www.helpnetsecurity.com/2019/09/10/cyberattacks-human-interaction/

[56] R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy, "Not-a-bot: Improving service availability in the face of botnet attacks," in *NSDI*, vol. 9, 2009, pp. 307–320.

[57] S. Dowling, M. Schukat, and H. Melvin, "A ZigBee honeypot to assess IoT cyberattack behaviour," in *2017 28th Irish Signals and Systems Conference (ISSC)*. IEEE, 2017, pp. 1–6.

[58] A. Hayes, "Z-score definition." 2020. [Online]. Available: https://www.investopedia.com/terms/z/zscore.asp

[59] D. Wheeler, "zxcvbn: Realistic password strength estimation," *Dropbox Tech Blog, Apr*, 2012.

[60] "Blasting_dictionary." [Online]. Available: https://github.com/rootphantomer/Blasting_dictionary

[61] "Enterprise email service for business - MS Exchange email." 2020. [Online]. Available: https://products.office.com/en-us/exchange/email

[62] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen, "Honeystat: Local worm detection using honeypots," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2004, pp. 39–58.

[63] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi, "Heat-seeking honeypots: Design and experience," in *Proceedings of the 20th International Conference on World Wide Web*. ACM, 2011, pp. 207–216.

[64] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage, "Scalability, fidelity, and containment in the Potemkin virtual honeyfarm," in *ACM SIGOPS Operating Systems Review*, vol. 39, no. 5. ACM, 2005, pp. 148–162.

[65] M. Nassar, R. State, and O. Festor, "VoIP honeypot architecture," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2007, pp. 109–118.

[66] R. Do Carmo, M. Nassar, and O. Festor, "Artemisa: An open-source honeypot back-end to support security in VoIP domains," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 361–368.

[67] A. Kedrowitsch, D. D. Yao, G. Wang, and K. Cameron, "A first look: Using Linux containers for deceptive honeypots," in *Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense*. ACM, 2017, pp. 15–22.

[68] M. Dornseif, T. Holz, and S. Müller, "Honeypots and limitations of deception," *"Heute schon das Morgen sehen", 19. DFN-Arbeitstagung über Kommunikationsnetze in Düsseldorf*, 2005.

[69] M. Javed and V. Paxson, "Detecting stealthy, distributed SSH brute-forcing," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013, pp. 85–96.

[70] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, "Hotp: An hmac-based one-time password algorithm," *The Internet Society, Network Working Group. RFC4226*, 2005.

[71] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast Internet-wide scanning and its security applications," in *USENIX Security Symposium*, vol. 8, 2013, pp. 47–53.

[72] "Chaos Monkey is a resiliency tool that helps applications tolerate random instance failures." [Online]. Available: https://github.com/Netflix/chaosmonkey

[73] W. Kumari and D. McPherson, "Remote triggered black hole filtering with unicast reverse path forwarding (uRPF)," *Network Working Group*, vol. 5635, 2009.

[74] A. Kuehn and M. Mueller, "Analyzing bug bounty programs: An institutional perspective on the economics of software vulnerabilities," in *2014 TPRC Conference Paper*, 2014.

[75] "RH-ISAC." 2020. [Online]. Available: https://r-cisc.org/#homeResources

[76] T. Yu, S. K. Fayaz, M. P. Collins, V. Sekar, and S. Seshan, "PSI: Precise security instrumentation for enterprise networks," in *NDSS*, 2017.

[77] Z. Wu, M. Xie, and H. Wang, "Swift: A fast dynamic packet filter," in *NSDI*, vol. 8, 2008, pp. 279–292.

[78] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. Liu, J. Padhye, B. T. Loo, and G. Outhred, "007: Democratically finding the cause of packet drops," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. USENIX Association, 2018.

[79] Y. Go, M. A. Jamshed, Y. Moon, C. Hwang, and K. Park, "APUNet: Revitalizing GPU as packet processing accelerator," in *NSDI*, 2017, pp. 83–96.

[80] L. Rizzo, "Netmap: A novel framework for fast packet I/O," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 101–112.

[81] N. K. Sharma, A. Kaufmann, T. E. Anderson, A. Krishnamurthy, J. Nelson, and S. Peter, "Evaluating the power of flexible packet processing for network resource allocation," in *NSDI*, 2017, pp. 67–82.

[82] M. A. Jamshed, Y. Moon, D. Kim, D. Han, and K. Park, "mOS: A reusable networking stack for flow monitoring middleboxes," in *NSDI*, 2017, pp. 113–129.

[83] "Getting started with ThreatExchange." 2020. [Online]. Available: https://developers.facebook.com/docs/threat-exchange/getting-started/v3.1

[84] "IBM X-Force threat intelligence." [Online]. Available: https://www.ibm.com/security/xforce

[85] "'HASSH' - A profiling method for SSH clients and servers." [Online]. Available: https://github.com/salesforce/hassh#hassh---a-profiling-method-for-ssh-clients-and-servers

[86] M. Ion, B. Kreuter, E. Nergiz, S. Patel, S. Saxena, K. Seth, D. Shanahan, and M. Yung, "Private intersection-sum protocol with applications to attributing aggregate ad conversions," *IACR Cryptology ePrint Archive*, vol. 2017, p. 738, 2017.