Catch-22: Uncovering Compromised Hosts using SSH Public Keys

Munteanu, Cristian; Smaragdakis, G.; Feldmann, Anja; Fiebig, Tobias

Publication date
2025

Document Version
Final published version

Published in
Proceedings of the 34th USENIX Security Symposium

USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Catch-22: Uncovering Compromised Hosts using SSH Public Keys

Cristian Munteanu, *Max Planck Institute for Informatics;*
Georgios Smaragdakis, *Delft University of Technology;*
Anja Feldmann and Tobias Fiebig, *Max Planck Institute for Informatics*

https://www.usenix.org/conference/usenixsecurity25/presentation/munteanu

This paper is included in the Proceedings of the
34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

Catch-22: Uncovering Compromised Hosts using SSH Public Keys

Cristian Munteanu
Max Planck Institute for Informatics

Georgios Smaragdakis
Delft University of Technology

Anja Feldmann
Max Planck Institute for Informatics

Tobias Fiebig
Max Planck Institute for Informatics

Abstract

Attackers regularly use SSH (Secure SHell) to compromise systems, e.g., via brute-force attacks, establishing persistence by deploying SSH public keys. This ranges from IoT botnets like Mirai, over loader and dropper systems, to the back-ends of malicious operations. Identifying compromised systems at the Internet scale would be a major break-through for combatting malicious activity by enabling targeted clean-up efforts.

In this paper, we present a method to identify compromised SSH servers at scale. For this, we use SSH's behavior to *only* send a challenge during public key authentication, to check if the key is present on the system. Our technique *neither* allows us to access compromised systems (unlike, e.g., testing known attacker passwords), *nor* does it require access for auditing.

With our methodology used at an Internet-wide scan, we identify more than 21,700 unique systems (1,649 ASes, 144 countries) where attackers installed at least one of 52 verified malicious keys provided by a threat intelligence company, including critical Internet infrastructure. Furthermore, we find new context on the activities of malicious campaigns like, e.g., the 'fritzfrog' IoT botnet, malicious actors like 'teamtnt', and even the presence of state-actor associated keys within sensitive ASes. Comparing to honeypot data, we find these to under-/over-represent attackers' activity, even underestimating some APTs' activities. Finally, we collaborate with a national CSIRT and the Shadowserver Foundation to notify and remediate compromised systems. We run our measurements continuously and automatically share notifications.

1 Introduction

Since its introduction in 1995, SSH has become the de facto standard for secure remote access to shell-enabled systems. Originally developed for Unix and Linux systems, SSH servers have now permeated the digital landscape, being available even for Windows systems. This popularity is also reflected in the up to 40 million SSH servers [27] seen to be publicly accessible on the Internet. Furthermore, with the advent of the Internet-of-Things (IoT), especially Linux found its way onto millions more devices, with SSH being commonly available and enabled by default on IoT systems [78].

The most common implementations of SSH support a variety of authentication mechanisms, including public/private key-based authentication and, traditionally, password-based authentication. With the general prevalence of password authentication, attackers commonly run brute-force campaigns to compromise systems with weak or default passwords [50, 102]. Similarly, many systems attackers compromise may also run SSH servers, meaning a convenient way of access after the system has been compromised [89].

After a compromise, it may be difficult for an attacker to *change* the password of a system, as being suddenly unable to log in would likely alert the benign owners to the compromise [30]. Hence, attackers have been observed in several incidents [7], but also via honeypots [64], to install their own public SSH keys into accounts on compromised systems.

Interestingly, the behavior of most SSH implementations differs depending on a public key *actually being installed* when a user provides the fingerprint of a public key to initiate authentication via the associated key-pair: Usually, a challenge will *only be sent if the key is installed for the user* [36]. This is a conscious design choice in the SSH protocol [98], and as such, not considered a security issue by, e.g., OpenSSH developers [48]. In turn, this means that systems compromised by attackers can be remotely identified, if one gets a hold of the attackers *public* SSH key.

In this paper, we present the first study leveraging this mechanic to identify compromised systems at the Internet scale. We received a set of 52 keys through a collaboration with Bitdefender [15]. Bitdefender is a threat intelligence company that provides antivirus software, Internet security, and other security solutions to protect devices and networks. These keys were encountered in honeypots, or during malware analysis and incident investigations. We then design a measurement methodology that 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.

Contributions: In summary, our contributions are:

- We design and implement a methodology to accurately identify compromised SSH servers by leveraging unique characteristics of the SSH protocol.
- 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 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.
- We also demonstrate how it allows tracking malicious activity to better understand ongoing attacks.

Structure: The remainder of this paper is structured as follows. In Section 2, we introduce the necessary background on attacker activities and intricacies of the SSH protocol. Subsequently, we document our methodology in Section 3, including an extensive ethics discussion, and a documentation of our lab experiments to ensure the reliability of our method. In Section 4, we then document the parameters and schedule of our measurements. We present and analyze our results in Section 5, before discussing notable events and cases in more detail in Section 6. Finally, we compare to related work in Section 7, before concluding in Section 8.

2 Background

Here, we first discuss the functionality of the SSH protocol as per RFC4252 [98], and then focus on the mechanic we leverage for our methodology. Additionally, we discuss common attacks on SSH, and attackers' post-compromise behavior.

2.1 The Secure SHell (SSH) 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 [98–101], were only published in 2006, and later updated by several documents [8, 9, 12–14, 38, 88], mostly to update algorithms used for encryption or authentication.

SSH is a layered protocol, with an underlying transport layer (RFC4253 [101]) over which sessions for individual purposes, e.g., sFTP, X11 forwarding, port forwarding, or–commonly–an interactive terminal session can be started (RFC4254 [99]). For each session, different authentication and authorization requirements may be set, with authentication being defined in RFC4252 [98].

These days, the most prevalent SSH implementation is OpenSSH [68] by the OpenBSD project. In addition, especially on embedded devices, Dropbear–a more resource conscious implementation–is widely used [25], see also Table 1.

2.2 Public-Key Based Authentication in SSH

In addition to, e.g., password based authentication, SSH supports a public key based authentication method, see RFC4252, Section 7 [98]. 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 1. After the TCP handshake the SSH Client and SSH Server exchange version information. Subsequently, the client initiates a Key Exchange.

Only now, 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 [76]. 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 [36]. In 2021, Kaiser submitted a pull-request to the OpenSSH project [48] 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 [48].

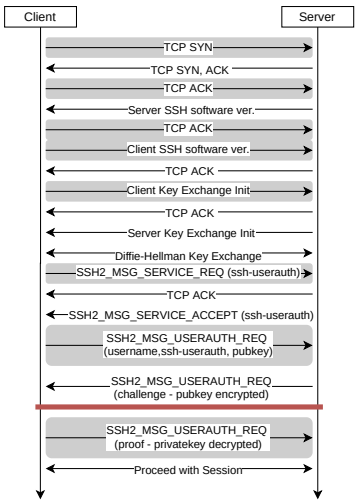

Figure 1: Step-by-Step SSH authentication. The red line indicates when a client knows if a public key is present or not.

2.3 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 [69]. 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 [7, 30, 50, 77, 91, 92].

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 [62, 71]. Accordingly, several botnets of IoT devices used for, e.g., (d)DoS attacks, including Mirai, have been created using mostly brute-force attacks [4, 11, 82, 96].

While industry and policy actions, especially in the context of IoT [29, 86], are increasingly enforced, the underlying issue remains. Similarly, newer versions of SSH, by default, do not permit password authentication for the `root` user, or–as in newer Ubuntu versions–a remote `root` login all-together.

Technically, the best method to evade brute-force attacks on SSH is disallowing password authentication in favor of, e.g., token-backed public key authentication. Nevertheless, especially for embedded devices and systems operated by less experienced users, passwords remain common.

2.4 Attackers' Post-Compromise Behavior

Once attackers compromise a system, usually the first order of action is ensuring persistence [89]. From honeypot studies [64], 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 [64].

3 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 4.

3.1 Public-Key Presence Identification

The core of our methodology is the behavior of the SSH protocol during public key authentication, see Section 2.2. Using the method described there, see also Figure 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 3.4.

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 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 3.7.

3.2 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 [68], Dropbear [25], BitviseSSH [16], and WolfSSH [97]. We selected these implementations based on the observed distribution of implementations in scan-data from censys.io [27], see Table 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 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 AAAABBBCCCCCDDDD...`' fails, our tool retries with '`rsa-sha-256 AAAABBBCCCCCDDDD...`'.

3.3 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.

[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 1: SSH server implementation, in-lab testing.

3.4 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 Section 5. Similarly, we encountered SSH servers that would send a challenge for any public key when it uses and algorithms they did not support, e.g. a prototype SSH server with support for federated authentication accepting all ED25519 keys [95].

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
```

3.5 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—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 are also collaborating with the Global Cyber Alliance (GCA) [35], a non-governmental organization that operates a large-scale honeynet comprising 221 SSH/Telnet honeypots powered by Cowrie [20]. These honeypots are uniformly configured and distributed across 55 countries and 65 ASes, primarily deployed within residential ISPs.

However, no additional keys were shared with us, i.e., all keys from these sources were also included in the data shared 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 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.

3.6 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.

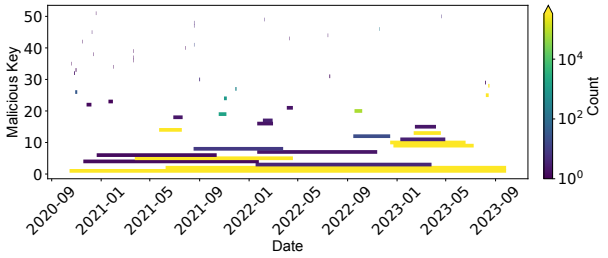

Figure 2: Activity periods of malicious key recorded by the threat intelligence company.

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 [3, 51, 55, 70]. 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.

3.7 Ethical Considerations

Our work entails several ethical implications. As such, we tightly followed best-practices, see, e.g., the Menlo report [23], 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 3.1, 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 4. 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 [17], 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 brute-force 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.

[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 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.

4.1 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 3.7.

4.2 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 [28] to identify IP addresses that have the TCP port for that run open (Port 22 or 2222) [6]

Next, we execute **Public Key Verification** for all hosts with open ports. There, we employ our patched version of Zgrab2 [26], see Section 3, 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 3.4. 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.

[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 3.7.

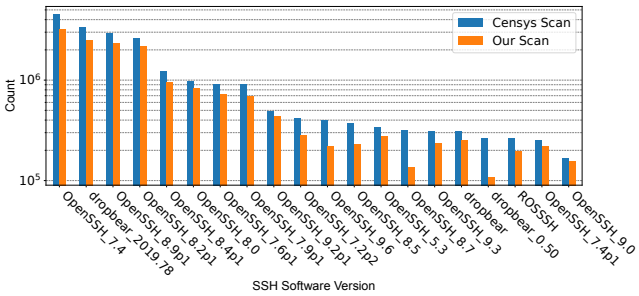

Figure 3: Censys vs. Our Scan (port 22 only)

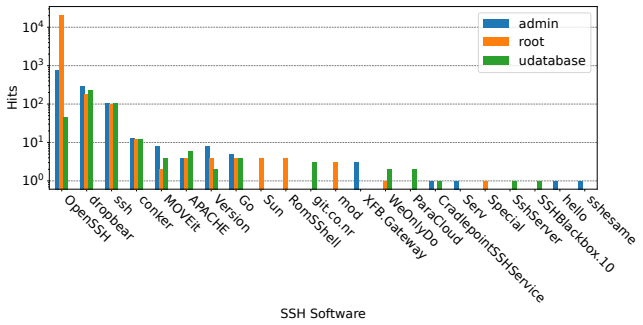

Figure 4: SSH server implementation vs. username, port 22

4.3 Scanning Schedule

The data collection for the dataset in this paper 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.

4.4 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 3. Especially for major versions our results (orange) closely align with those of Censys (blue), see also Table 1. Our comparably lower rate of identified systems aligns with our use of a blocklist, see Section 3.7.

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 Results

In this section 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.

5.1 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 [52]. 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 4, 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.

Affected AS Types: Next, we assess what types of ASes contain compromised systems. To categorize ASes, we use labels from PeeringDB [2] and data from BGP.tools [1], creating seven categories: CDN, corporate, government, hosting providers, ISPs, universities, and 'other', see Figure 5 for an overview.

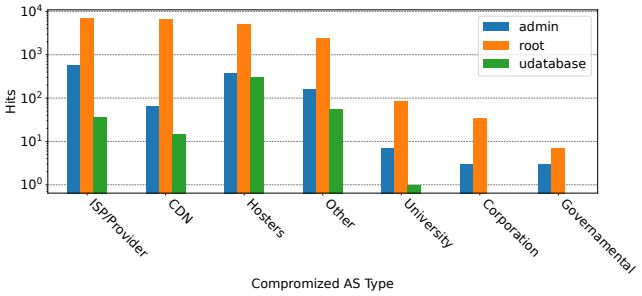

Figure 5: AS type vs. username, port 22

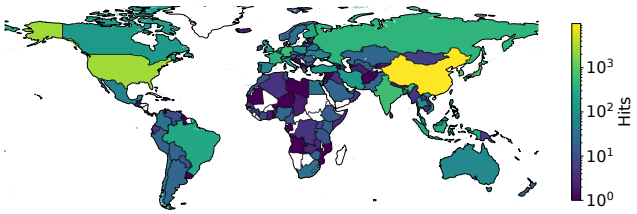

Figure 6: Compromised hosts port 22—world distribution

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 the progressing flattening of the Internet [5], a system in a hoster or other type of AS may still be operated by, e.g., another company or a government agency [45].

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.

5.2 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 [44] geolocation tool, pinpointing the location at the country level. Figure 6 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 7 for the distribution of compromised servers across AS types in the 30 countries with most hits.

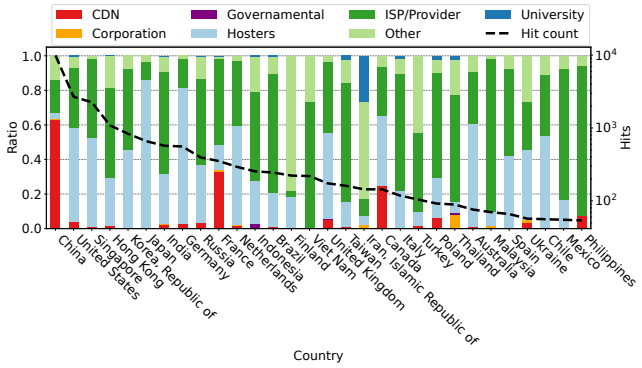

Figure 7: Top 30 countries based on number of compromised servers vs. AS type, port 22

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.

5.3 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*, *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 [61]. 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.

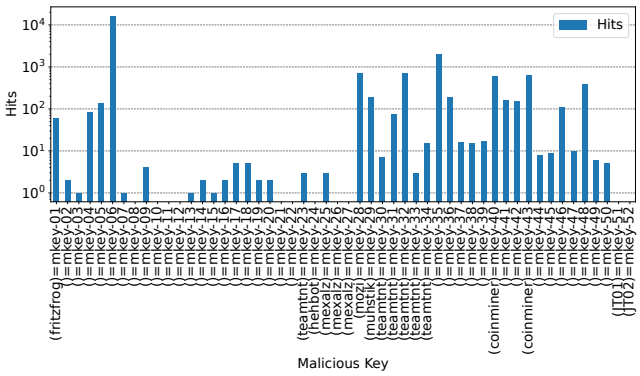

Figure 8: Malicious key count, port 22

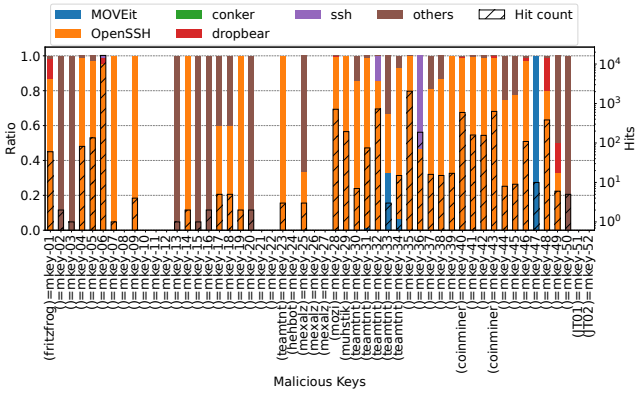

Figure 9: Malicious key vs. SSH server, port 22

This, however, is not the case, see Figure 8. 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 10. This maps to the modus-operandi of 'teamtnt', i.e., their focus on compromising systems via misconfigurations and vulnerabilities associated with common cloud software stacks, e.g., Redis [31] and Kubernetes [58].

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 toolchains to explicitly reference gitee.com [79–81]. 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 [64], and they confirmed that this key is among the most aggressive attacks identified in their research as well. Other keys found in notable frequency on Dropbear servers, i.e., likely IoT devices, are *MK48* and *MK49*, see Figure 9, even though those are less constrained to ISPs, see Figure 10.

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.

[7] 'teamtnt', e.g., focused on exploiting Kubernetes setups.

[8] https://aquasec.com/blog/new-malware-in-the-cloud-by-teamtnt muhstik-botnet-exploiting-apache.html

[9] https://thehackernews.com/2024/06/

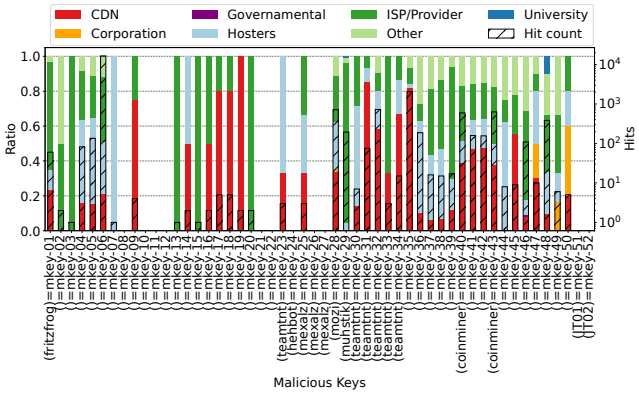

Figure 10: Malicious key vs. AS type, port 22

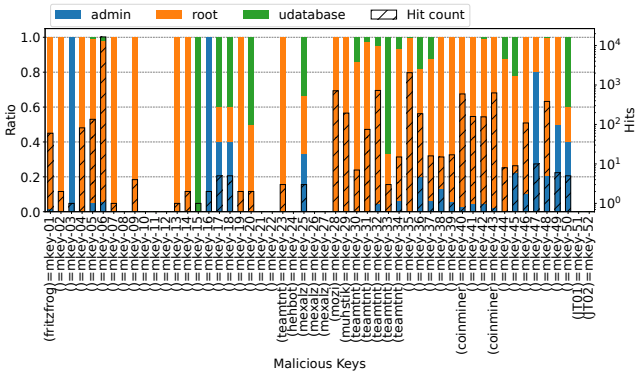

Figure 11: Malicious key vs. username, port 22

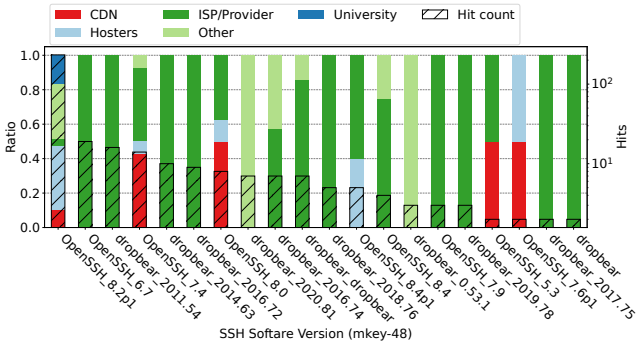

Figure 12: *MK48*: SSH server and AS type, port 22

Potential State-Actor Deployment (*MK48*): The diverse prevalence of *MK48* throughout various AS types, see Figure 12, triggered a further investigation of this key. Specifically, the key can mostly be found in:

- Several European hosters generally renown for their affordable offerings.
- 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 [83], V2Ray [87], HTTPS Proxies [63,72], and domain-fronting [32, 60]. 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.

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.

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

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 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.

5.4 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 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 13. 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.

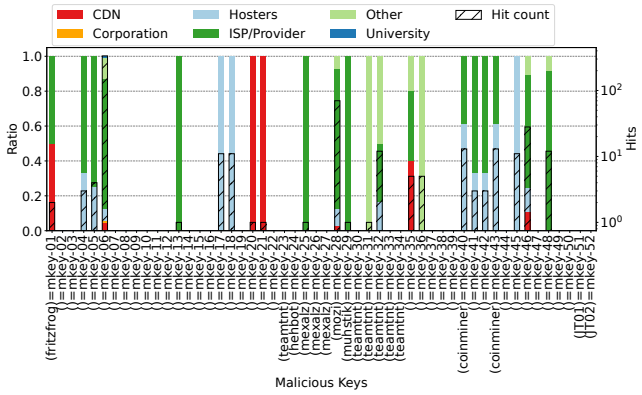

Figure 13: Malicious key vs. AS type, port 2222

5.5 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.

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*. Note that *MK48* is the key related to the MES in Section 5.3. 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.

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 5.3. Apart from GitHub, these keys can be found on 328 compromised hosts world-wide. The information was shared for mitigation with a GitHub contact.

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.

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 Discussion

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

6.1 Internet Measurement Opt-Out and Ethics

In Section 5.5, 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 [23]. 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. [54]–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.

6.2 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. [21].

6.3 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.

7 Related Work

In this section, we briefly compare to related work on (i) honeypots, (ii) Scanning/SSH Compromises and Attacks, (iii) Botnet Counteraction, and (iv) Threat-Actor Characterization.

Honeypots: Honeypots act like viable targets, waiting for attackers to compromise them [18, 67]. They can generally be implemented as VMs or on physical hardware [33, 85, 90], and come in various specialized forms, e.g., as web [46, 94] or mobile [93] honeypots, or with a focus on worm detection [22]. Furthermore, in 2022, Hiesgen et al. used 'Spoki', a mix between a traditional network telescope and a honeypot to study malicious activity, finding over 10K executables, including variants of attacks such as 'Mozi' and 'Mirai'.

Regarding SSH honeypots, Kippo [53] and Cowrie [20] are among the most widely used today [67], and commonly form the foundation of studies of malicious activity [10, 56]. For instance, Munteanu et al. [64] use data from a honepot setup to analyze attacks captured over a fifteen-month period. Their work focuses on SSH brute-force attacks, malware distribution campaigns, and botnet activity.

IoT honeypots have proven to be invaluable in identifying various malware families targeting IoT devices [62, 71], contributing information for botnet counteraction [49, 57, 73]. These studies emphasize the necessity of proactive detection and mitigation strategies to effectively combat botnets.

Scanning, SSH Compromises & Brute-Force attacks: SSH, being around since the 1990's, has been frequently studied from different perspectives. On a descriptive level, a study by Gasser et al. [34] presents a comprehensive analysis of SSH deployments in the wild, examining the prevalence, configuration, and security practices of SSH servers across the Internet. Several studies also investigated the behavior of brute-force attacks against SSH [7, 30, 50, 70, 91, 92, 102]. Several of these studies also suggest high-interaction honeypots to solicit further information on attackers and enhance mitigation.

Other related work focuses on SSH brute force detection and countermeasures, utilizing techniques such as network flow analysis [24, 39–41, 47] and machine learning [37, 42, 43, 59, 65, 66]. However, especially machine-learning based approaches often show a high number of false positives. A more recent study by Sachin et al. [77] 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%.

In 2019, Cao et al. [19] 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.

Threat-Actor Characterization: Attribution and characterization are some of the most difficult aspects of defending against organized or state-sponsored threat actors [6, 75, 84]. Still, these studies note that specific system properties, such as configuration, vulnerabilities, and the perceived value of targets, play a crucial role in shaping attacker behavior. These factors influence the strategies attackers employ, the persistence of their attacks, and the likelihood of specific types of attacks occurring. Each study also highlights the importance of distinguishing between human-driven attacks and those carried out by automated bots.

As before, honeypots can be instrumental in this, as for example shown by Sadique et al. [74], who analyze botnet behavior, focusing on the actions and strategies employed by attackers after compromising hosts.

Summary: Related work heavily leans on honeypots for the identification of attacks and characterization of threat actors. Contrary to these studies, our technique allows us to identify threat actor compromised systems at Internet scale, even if the systems are not in active use by attackers. Furthermore, due to the exhaustive nature of our method, we are able to find correlations between–in our case–public SSH keys employed by threat actors, indicating the possibility of an overlap between so-far assumed distinct groups. Overall, our approach lifts the state-of-the-art from *reactive* characerization of attackers to a *proactive* characterization.

8 Conclusion

In this paper, 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.

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 forseeable future, to ensure that compromised machines–as well as potentially threat-actor *operated* systems can be remidiated. We also plan to release our code and collaborate with the CSIRT and threat intelligence community on further remidiation of compromised systems.

Open Science & Artifact Availability

Our measurement instrument, including the patched version of ZGrab2, has been released under an Open Source license and is publicly available. You can access it here: https://edmond.mpg.de/dataset.xhtml?persistentId=doi%3A10.17617%2F3.LVPCS6

Acknowledgments

The authors would like to thank the Shadowserver Foundation and CERT-BUND for their collaboration in the notificaiton campaign for the results presented in this work. This work would not have been possible without the tireless efforts of technical and administrative staff at the authors' institutions supporting the authors by providing infrastructure and handling administrative tasks.

This work was supported by the European Commission under the Horizon Europe Programme as part of the projects SafeHorizon (Grant Agreement #101168562) and RECITALS (Grant Agreement #101168490).

Any opinions and conclusions expressed in this article are those of the authors and do not necessarily reflect the official opinion of the European Union, the Shadowserver Foundation, the CERT-BUND/BSI, the German Government, or the authors' host institutions.

References

[1] bgp.tools. https://bgp.tools/.

[2] PeeringDB. https://www.peeringdb.com.

[3] 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.

[4] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. Understanding the Mirai Botnet. In *USENIX Security Symposium*, 2017.

[5] 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 on Internet Measurement Conference*, 2020.

[6] Timothy Barron and Nick Nikiforakis. Picky attackers: Quantifying the role of system properties on intruder behavior. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 387–398, 2017.

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

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

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

[10] 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.

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

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

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

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

[15] Bitdefender. Bitdefender. https://www.bitdefender.com/, 2024.

[16] Bitvise. Bitvise. https://bitvise.com/, 2024.

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

[18] 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.

[19] Phuong Cao, Yuming Wu, Subho S Banerjee, Justin Azoff, Alexander Withers, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. CAUDIT: Continuous Auditing of SSH Servers To Mitigate Brute-Force Attacks. In *NSDI*, volume 19, pages 667–682, 2019.

[20] Cowrie. Cowrie on GitHub. https://github.com/cowrie/cowrie, 2019.

[21] 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.

[22] 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, Sophia Antipolis, France, September 15-17. Proceedings 7*, pages 39–58. Springer, 2004.

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

[24] 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.

[25] Dropbear. Dropbear. https://matt.ucc.asn.au/dropbear/dropbear.html, 2024.

[26] Z. Durumeric. ZGrab2. https://github.com/zmap/zgrab2, 2018.

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

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

[29] European Commision. Europe's Internet of Things Policy. https://digital-strategy.ec.europa.eu/en/policies/internet-things-policy, 2024.

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

[31] 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.

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

[33] 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, 2006.

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

[35] Global Cyber Alliance. GCA AIDE – Automated IoT Defense Ecosystem. https://gcaaide.org, 2023.

[36] Artem Golubin. Public SSH keys can leak your private infrastructure, 2019.

[37] John Hancock, Taghi M Khoshgoftaar, and Joffrey L Leevy. Detecting SSH and FTP brute force attacks in big data. In *20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 760–765, 2021.

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

[39] 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.

[40] 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.

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

[42] 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 *5th IEEE International Conference on Computer and Communication Systems (ICCCS)*, pages 491–497, 2020.

[43] 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 , Maribor, Slovenia, September 21–23, 2020, Proceedings 35*, pages 49–63. Springer, 2020.

[44] IPInfo. IPInfo. https://ipinfo.io/, 2024.

[45] 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.

[46] 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.

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

[48] Manfred Kaiser. CVE-2016-20012 - Publickey Information leak - Only allow SSH_MSG_USERAUTH_REQUEST with signature #270, 2021.

[49] 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.

[50] 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.

[51] 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.

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

[53] Kippo. Kippo on GitHub. https://github.com/desaster/kippo, 2019.

[54] Tadayoshi Kohno, Yasemin Acar, and Wulf Loh. Ethical frameworks and computer security trolley problems: Foundations for conversations. In *32nd USENIX Security Symposium*, pages 5145–5162, 2023.

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

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

[57] 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.

[58] Kubernetes.io. Kubernetes.io. https://kubernetes.io, 2024.

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

[60] 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, 2021.

[61] 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.

[62] 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.

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

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

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

[66] 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 *IEEE International Conference on Bioinformatics and Bioengineering*, pages 379–385, 2014.

[67] Marcin Nawrocki, Matthias Wählisch, Thomas C. Schmidt, Christian Keil, and Jochen Schönfelder. A Survey on Honeypot Software and Data Analysis. *CoRR*, abs/1608.06249, 2016.

[68] OpenSSH. OpenSSH. https://www.openssh.com/, 2024.

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

[70] Jim Owens and Jeanna Neefe Matthews. A Study of Passwords and Methods Used in Brute-Force SSH Attacks. 2008.

[71] 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.

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

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

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

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

[76] Chris Siebenmann. Your SSH keys are a (potential) information leak, 2016.

[77] 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.

[78] Satyajit Sinha. State of IoT 2024: Number of connected IoT devices growing 1318.8 billion globally. https://iot-analytics.com/number-connected-iot-devices/, 2024.

[79] teamtnt. Web Resource 1. https://web.archive.org/web/20240904145504/https://documents.hubaoquan.cn/HS6SqV7w.txt, 2024.

[80] teamtnt. Web Resource 2. https://web.archive.org/web/20240904144948/https://www.cnblogs.com/sexiszero/p/14775793.html, 2024.

[81] teamtnt. Web Resource 3. https://web.archive.org/web/20240904145120/https://www.52pojie.cn/thread-1283815-1-1.html, 2024.

[82] 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.

[83] Tor Project. Tor Project. https://community.torproject.org/, 2024.

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

[85] 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.

[86] UK Commision. UK Internet of Things Policy. https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf, 2024.

[87] V2Ray. V2Ray. https://www.v2ray.com/, 2024.

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

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

[90] 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.

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

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

[93] 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.

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

[95] Wayf.dk. Wayf.dk. https://www.wayf.dk/en/wayf-part-intl-workshop-federated-access-ssh, 2024.

[96] 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.

[97] WolfSSH. WolfSSH. https://www.wolfssl.com/products/wolfssh/, 2024.

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

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

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

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

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