Where The Wild Things Are: Brute-Force SSH Attacks In The Wild And How To Stop Them

Sachin Kumar Singh
University of Utah

Shreeman Gautam
University of Utah

Cameron Cartier
University of Utah
Black Hills Information Security

Sameer Patil
University of Utah

Robert Ricci
University of Utah

Abstract

SSH (Secure Shell) is widely used for remote access to systems and cloud services. This access comes with the persistent threat of SSH password-guessing brute-force attacks (BFAs) directed at `sshd`-enabled devices connected to the Internet. In this work, we present a comprehensive study of such attacks on a production facility (CloudLab), offering previously unreported insight. Our study provides a detailed analysis of SSH BFAs occurring on the Internet today through an in-depth analysis of `sshd` logs collected over a period of four years from over 500 servers. We report several patterns in attacker behavior, present insight on the targets of the attacks, and devise a method for tracking individual attacks over time across sources. Leveraging our insight, we develop a defense mechanism against SSH BFAs that blocks 99.5% of such attacks, significantly outperforming the 66.1% coverage of current state-of-the-art rate-based blocking while also cutting false positives by 83%. We have deployed our defense in production on CloudLab, where it catches four-fifths of SSH BFAs missed by other defense strategies.

1 Introduction

The Secure Shell [72] is widely used for remote administration and command execution. Due to this popularity, it is common for SSH servers to be targeted for password-guessing Brute-Force Attacks (BFAs). In such attacks, a malicious party attempts to connect to an SSH server using one or more `{username,password}` pairs, guessing values for both fields. Attempting to brute-force SSH may seem like an "outdated" attack: best practices recommend key-based authentication [29], many IPv4 devices are behind NAT [28], and scanning for hosts on IPv6 is notoriously difficult [36]. Yet, our experience shows that BFAs are still prevalent– in fact, increasing—on the IPv4 Internet. Such attacks are leveraged to exploit poorly configured and poorly secured SSH servers to build botnets [56]. If the attack against a machine succeeds, the attackers can use the machine to carry out further attacks as part of the botnet [11, 13, 19, 43, 46] or steal its computing resources (e.g., for cryptomining [24, 59]).

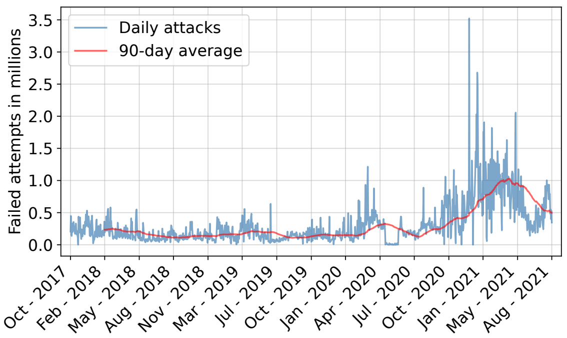


Figure 1: Failed SSH attempts on CloudLab (in millions).

While others have studied SSH attacks, existing studies involve smaller timescales [10, 30, 58, 71], examine fewer hosts [10, 58], use honeypots that do not include legitimate users [10, 30, 58, 71], or cover periods that are not recent [10, 58]. In contrast, we captured real-world BFAs mixed with the activities of legitimate users in a live production facility by capturing `sshd` logs for four years across ≈500 servers in CloudLab [27]. Due to this unique characteristic, our dataset provides a rare opportunity to understand changing attacker behavior over a longitudinal span and to compare it with the practices of legitimate users. Our analysis shows that SSH BFAs are evolving. As Figure 1 illustrates, BFA attackers are becoming more aggressive, with daily attempts increasing, particularly in recent years.

By "fingerprinting" usernames, we were able to track many attackers over time and across IP addresses. Our analysis revealed a wide range of attacker behaviors. Many attack quickly and then disappear, while others persist in their attacks for months or years; some focus on one username, while others attempt thousands. We found that attackers target a wide variety of devices (e.g., servers, Internet of Things [IoT] devices, routers) and software (e.g., databases, games, chat servers), shifting from typical administrator usernames (`root` and `admin`) to those associated with cloud images, network and IoT devices, and specific software. Based on observing that username fingerprints are a strong differentiator between attackers and legitimate users, we designed `Dictionary-Based Blocking (DBB)`—a novel technique for blocking SSH BFAs with high effectiveness. DBB blocks traffic based on *dictionaries* of usernames attempted by attackers. The low false positive rates of DBB ensure that legitimate users retain reliable access.

With our research, we make the following contributions:

- We present an analysis of the properties of the SSH brute-force attacks going on today, including insights about their methodology, inferences about their targets, investigations into their network sources, and analysis of attackers' persistence. (Sections 4, 5, 6)
- We develop a method for creating username dictionaries that allow us to track attacks across IP addresses and over time, even when data is incomplete or attackers make small adjustments to their username lists. (Section 7)
- We present a new method, Dictionary Based Blocking (DBB), for blocking SSH brute-force attacks using these dictionaries. (Section 8)
- We evaluate DBB using real-world data, showing it to be highly effective at blocking attackers while allowing legitimate users access. (Section 8.3)
- We present results from a production deployment of DBB, showing that it performs exceptionally well in practice. (Section 8.5)

We begin with an overview of related literature in Section 2 and data collection setting and method in Section 3.

2 Related Work

The literature most relevant to our work analyzes SSH BFAs and examines approaches to block them.

Analyzing SSH BFAs: An analysis of 103K login attempts from 271 IPs on three honeypots over 11 weeks revealed that the attacks target administrator as well as user accounts and can be thwarted with strong passwords and key-based logins [58]. More recently, researchers found that such attacks employ `root` and `admin` as popular usernames (95% and 3% of attempts, respectively) [10]. Complimentary research has focused on the use of stolen credentials and botnets when carrying out SSH BFAs [17,71].

SSH BFAs have been captured by honeypots deployed via IoT hardware and software as well and demonstrated that attackers who gain access engage in diverse activities, such as bitcoin mining, UDP/TCP flooding, SSH scans, and SSH port forwarding [24]. In a larger-scale deployment via 102 medium-interaction honeypots across three continents, researchers monitored 12 million connections originating from 38K unique IPs and examined how attacker behavior is influenced by the location of the honeypot, the difficulty of compromise, and the variety of files available on the honeypot [15]. Use of cipher suites and SSH version strings to fingerprint the mechanisms used in the attacks has identified that attackers use popular tools off-the-shelf software, such as Ncrack [6] and Hydra [4], as well as custom tools [30].

A recent deployment of medium-interaction honeypots for five protocols captured 73K IP addresses, noting an increasing frequency of SSH attacks throughout the period. More longitudinal honeypot deployments have observed a significant number of IP addresses engaged in multiple activities over 15 months [47] and two years [40]. The data covered various attacks, including SSH BFAs and non-standard port accesses, and the researchers observed that attacker preferences were relatively stable [40]. The studies mentioned above mostly use honeypots that collect only attacks. In contrast, we examined data from a production system, enabling us to compare attackers and legitimate users accessing the *same* system.

Blocking SSH BFAs: Researchers have proposed various approaches to detect and block BFAs, including network flow analysis [26,32–35,42,67] and machine learning/deep learning techniques [31,37,38,45,48,49,62]. However, employing network flow data to detect SSH BFAs can result in a high number of false positives.

Alternately, defenses against SSH BFAs can employ host-based approaches to track user/IP characteristics, such as failed attempts and interarrival time. Tools such as Fail2ban [3], denyhosts [1], and sshguard [9] use host-based blocking of suspicious traffic. They analyze authentication logs to compute relevant features, such as the number of failed login attempts, and block corresponding IP addresses with host-based firewalls such as `iptables` [7]. Fail2ban is one of the most widely used tools for stopping BFAs: it blocks IP addresses exceeding a threshold number of failed attempts within a specified period. Tuned time- and rate-based blocking mechanisms have been used to improve blocking strategies [63]. A simulation with synthetic data showed that a distributed active-response architecture can enable the sharing of relevant information—particularly attacking IP addresses—among trusted hosts [44]. However, such an approach is not privacy-preserving and requires a set of trusted hosts. By analyzing data from a production system, we were able to design and deploy a novel defense and compare its effectiveness (including false positives) with rate-based designs.

3 Data Collection

Our research is based on an analysis of `sshd` logs from CloudLab [27], a public facility used by academic researchers at institutions around the world. CloudLab has a cloud-like user model: the users are "tenants" who access servers *temporarily* assigned to them. Although CloudLab has some policy control, such as initial `sshd` and logging configurations, the users are not under direct control of CloudLab. Once users acquire control of their assigned nodes, they may, and occasionally do, alter the SSH settings without any supervision or regulation from CloudLab. Therefore, an additional layer of SSH security, beyond the defaults permitting only key-based authentication and prohibiting `username,password` authentication, is needed.

We used two sets of log files collected by CloudLab. The first (*Log1*) was collected on a single cluster over four years (October 2017 – August 2021) and contains a large number of attacks that enabled us to study general trends in SSH BFAs. The second (*Log2*) was collected at three different CloudLab sites over ten weeks (November 2022 – January 2023) to evaluate our proposed blocking mechanism. The three CloudLab sites are geographically dispersed and use unrelated IP addresses owned by different networks.

SSH Logs: Each host (also called a "node" or "server") in CloudLab has a persistent public IPv4 address and runs `sshd` for remote access—the primary way legitimate users interact with the host. By default, CloudLab hosts do not run other public-facing services, though users are permitted to start their own services if they wish. All CloudLab hosts are configured to log SSH login attempts to a central `syslog` server from which we obtained our datasets. We parsed the logs to extract various relevant features, such as source IP addresses, attempted usernames, authentication responses, etc. The log files did not contain passwords. To get a view approximating the network boundary (e.g., firewall or bastion host), we first removed any SSH attempts originating from within CloudLab itself. After removing the internal attempts, *Log1* included $\approx$ 840K unique source IP addresses attempting $\approx$ 277K unique usernames and making $\approx$ 427M login attempts *Log2* contained $\approx$ 91K unique source IPs attempting $\approx$ 98K unique usernames and $\approx$ 213M login attempts.

When analyzing the logs, we took into account that log messages can be lost due to central server overload, network congestion, misconfigurations on the hosts or the server, or intentional configuration changes by users. We further considered that CloudLab hosts are responsive to SSH requests only when in use, with usage periods of varying lengths distributed unevenly. As a result, log data pertaining to any given host can exhibit short-term gaps corresponding to periods in which it is not in use. To avoid incorrect conclusions on account of such short-term gaps, we used methods that support "fuzzy" matching and looked at long-term trends. Of the hosts included in *Log1*, 352 logged SSH connections every day of the logged period, and the three sites in *Log2* averaged connections from 1,322 hosts daily (616, 384, and 322 each).

Distinguishing Legitimate Users From Attackers: We considered an IP address as belonging to a legitimate user if it had at least one *successful* login. In other words, we assumed that IP addresses associated only with *failed* logins belonged to attackers. CloudLab's default `sshd` configuration allows only public-key authentication, and the CloudLab staff know of no attacks using stolen private keys. Therefore, we assumed that only legitimate users logged in successfully. On the other hand, attackers typically attempt to log in using passwords. Even though CloudLab is set to accept only key-based authentication, password-based login attempts are logged despite being disallowed. Although legitimate users may attempt (and fail) to log in with passwords, nearly all failed login attempts in our logs appear to be part of attacks rather than erroneous password-based attempts from legitimate users.

Advantages of Production Facility over Honeypots: Unlike honeypot logs that exclusively contain attacks, the CloudLab logs we analyzed contain actions of attackers along with those of legitimate users. As a result, analyzing the data allowed us to explore the practices of legitimate users in addition to those of attackers, thus facilitating comparisons between the two. One of the main challenges of honeypot logs is the inability to test blocking strategies because of the absence of legitimate users, resulting in a lack of false positives for the assessment. In contrast, our data can serve the dual purpose of aiding the development of blocking strategies *and* providing a means to assess their effectiveness.

Metadata Limitations: In parts of our analysis, we used data regarding network owners and geographic locations of the attacker IP addresses [5, 14]. Since we fetched this data in Summer 2022, after collecting *Log1*, it is possible that some IP addresses changed ownership after we had logged attacks from them. In addition, the use of NAT and dynamic address assignment on source networks may obscure the true number of attacking devices captured in the logs.

Ethical Considerations: Collecting logs is a routine operation in facilities like CloudLab. We have reported results using large aggregates without identifying individual users. In cases where mentioning specific users can provide illustrative value, we have anonymized the usernames. At the same time, we have assumed that attackers do not have a legitimate expectation of privacy. Still, we have not mentioned IP addresses since the devices may be compromised unbeknownst to their owners, who may have no malicious intent.

4 The Anatomy of SSH BFAs

In this section, we cover the basic features of SSH BFAs, attaching specific numbers and concrete behaviors from *Log1* to illustrate the concepts in the general descriptions.

Source IP Addresses: An SSH BFA originates from a *source* to a set of *targets* using a *guessing vector* of credentials. Such attacks can come from a large number of IP addresses. For instance, *Log1* contains attacks from over 800,000 IP addresses, with at least one attack from 90% of the 249 countries with ISO country codes [39]. However, few conclusions about the individual or entity controlling an attack can be drawn from source IP addresses alone. In many cases, the attacking devices do not belong to malicious actors themselves but to botnets composed of compromised machines [13, 46, 56] infected with self-replicating worms [25] or used to mask the locations of the actual attackers.

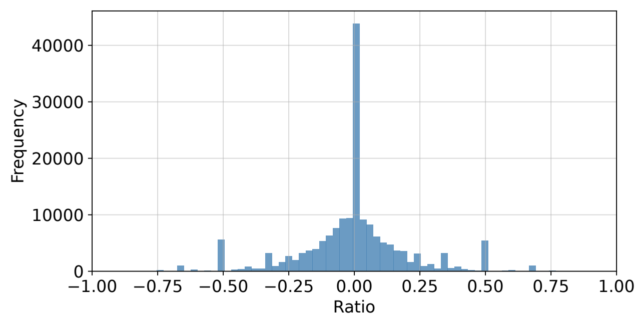


Figure 2: Histogram of sequence bias.

Target Host(s): Attackers may select hosts to target with several methods: scanning sequentially or randomly, checking for specific vulnerabilities on a host, etc. In practice, nearly all BFAs in *Log1* seemed to be based on random scanning of the IPv4 address space as seen in Figure 2, which is a histogram of the *sequence bias* for the sources in *Log1* that attacked at least 50 hosts. For each source IP address, we calculated the sequence bias as the fraction of successive target IP addresses higher than the one previously attacked. The sequence bias for a given sequence of IP addresses, $S_1, \ldots S_N$ is:

$$SB = \frac{1}{N-1}\sum_{i=1}^{N-2}\begin{cases}0 & \text{if } S_{i+1} = S_i\\ 1 & \text{if } S_{i+1} > S_i\\ -1 & \text{if } S_{i+1} < S_i\end{cases}$$

A sequence bias of 1 indicates that the source moves from lower-numbered addresses to higher ones 100% of the time, with -1 indicating the reverse movement order. A sequence bias of 0 means that each successive target is equally likely to be 'up' or 'down' in the IP space. The density of sources around zero—82% of the sources depicted in Figure 2 are in the range [-0.25,0.25]—indicates that most traverse the IP address space in random order.

Guessing Vectors: A BFA attacker attempts SSH login with a set of usernames and passwords. A username can be associated with multiple passwords and vice-versa. Each {`username, password`} combination is called a *credential vector*. Multiple such credential vectors are combined to construct a *guessing vector*. A study published in 2015 found that over 98% of BFAs contain the usernames `root` or `admin` [10]. Our more recent data shows that these usernames have become less dominant over time (see Figure 3). As seen in Figure 3, `root` has fallen dramatically, and `admin` is used only in a small fraction of attack attempts. Instead, as we present in more detail in Section 5, attackers are switching to usernames associated with cloud services and network devices.

5 Properties of SSH BFAs In Practice

After looking at the general structure of SSH BFAs, we used *Log1* to analyze specific aspects in more depth. We guided our analysis with a series of questions that arose from the general observations in the preceding section.

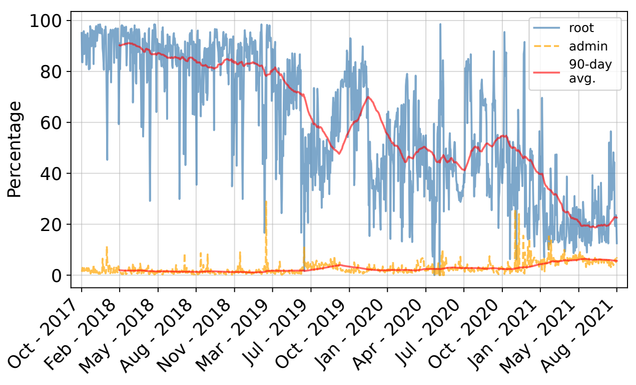


Figure 3: Percentage of BFAs with usernames `root` and `admin`.

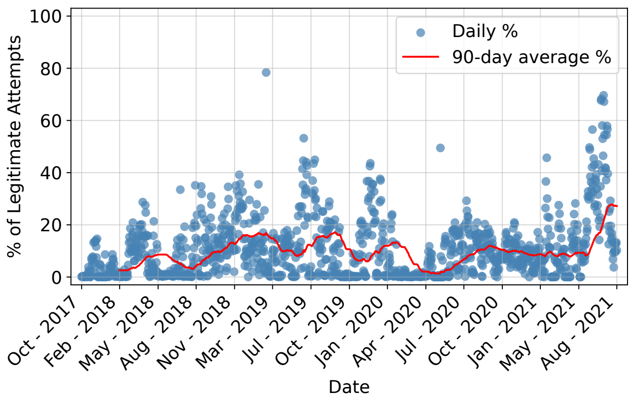


Figure 4: Percentage of accepted attempts at CloudLab.

How aggressive are individual attackers? A core reason behind blocking attacking IP addresses is that attackers attempt far more logins than real users. *Log1* shows that CloudLab is no exception, with only 10% of the login attempts being successful. On average, CloudLab experiences 25 successful SSH logins per minute as opposed to 211 failed ones.

Our data contains exceptions where the number of failed logins on a particular day did not outnumber successful ones. For instance, on a specific day, accepted attempts reached 621K—2.2 times the number of failed attempts. This case was mostly because of a single legitimate user, `river`,[1] who allocated numerous nodes and established thousands of SSH connections to each allocated node. Figure 4 shows the percentage of accepted attempts in the data: accepted SSH attempts were greater than failed ones on only 13 days in the four-year period covered by *Log1*. We speculate that the discrepant days may have resulted from the use of DevOps tools, such as Chef or Ansible [60], that make heavy use of SSH.

A majority of the failed attempts belonged to a small fraction of attacking IP addresses, with 1% accounting for 78%,

[1]Username changed for privacy.

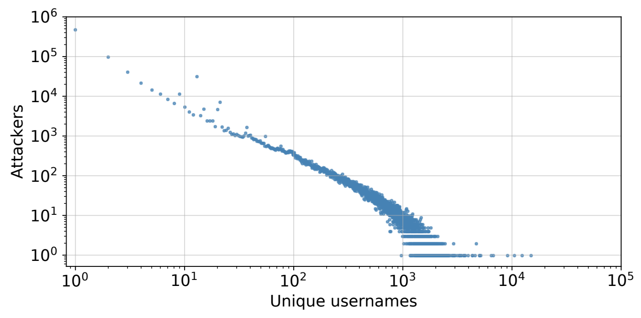


Figure 5: The number of usernames attempted vs. the number of attacking IP addresses using that many usernames. Both axes use a log scale to depict the extreme ranges.

2% for 85%, 4% for 90%, 8% for 95%, and 22% for 99% of all failed attempts. Over the four years represented in *Log1*, each attacking IP address performed 458 attempts on average. On the one hand, these numbers demonstrate that blocking by IP address is an attractive approach since blocking only a modest fraction of active attackers can greatly reduce attack volume. On the other hand, the numbers show that achieving perfect coverage by blocking specific IP addresses is difficult—moving from 99% to 100% coverage requires identifying and blocking nearly five times as many offending IP addresses.

Can we assume that all login failures are attacks? Of the ≈277K unique usernames in *Log1*, ≈99.995% were associated only with failed login attempts, but the remaining ≈0.005% had at least one accepted connection. For every legitimate username in *Log1*, there were about 199 associated only with failed attempts. Importantly, we found that *every* legitimate username logged at least one failure. We suspect failures associated with legitimate users are because of errors such as forgetting to add SSH keys to their agent, typos in hostnames, configuration errors on the servers, etc. The upshot is that administrators cannot deem an IP address to be malicious solely because of one, or even a few, failed login attempts. Since all legitimate users are likely to make the occasional error, effective identification of SSH attacks requires strategies more complex than simple failure-based blocking.

Do attackers try many usernames or focus on a small, high-value set? We uncovered numerous strategies regarding the number of usernames attackers tried. The number of usernames employed by each source IP address varied from one to ≈14K. More than half (54%) of the attacking IP addresses attempted only a single username in their guessing vectors. On average, attackers attempted 28 usernames, with 75% of the attackers using fewer than seven usernames and 90% using fewer than 39. A high variance (≈15170) in the distribution of usernames per source IP address shows that the number of usernames attempted by attackers is highly dispersed.

Figure 5 shows the number of usernames attempted against the number of attackers who attempted that many usernames. It can be seen in Figure 5 that a large number of attackers tried relatively few usernames (from one up to a few hundred). Only a small number of attaackers attempted a large number of usernames (in thousands). Those who tried the smallest number of usernames (i.e., 1 or 2) tended to go after administrator access with the usernames `root` and `admin`. The sets of usernames attempted in BFAs form the basis for our novel blocking strategy (described in Sections 7 and 8).

Are attackers successful at guessing legitimate usernames? We found a significant overlap between the usernames in guessing vectors used by attackers and those of CloudLabusers, indicating that attackers are somewhat successful at guessing legitimate usernames. Overall, 609 usernames belonging to legitimate CloudLab users appeared in the attacks. Although small, this is a non-trivial fraction (3%) of the ≈20,000 CloudLab users.[2] However, we saw no evidence that the guessing vectors targeted CloudLab specifically (e.g., via a leak of CloudLab's user database) since guessing vectors of substantial size were composed mostly of usernames *not* used by CloudLab's userbase. Instead, the presence of real usernames in the guessing vectors seems to be because of attackers attempting common names that are likely to be present in any sizable user base. Regardless, the overlap with legitimate usernames underscores the need to enforce good security practices (e.g., key-based login and/or strong passwords or passphrases).

What types of devices or software are targeted for attacks? Some Internet-connected devices have specific, known usernames, allowing us to infer that attackers trying those usernames were targeting devices of that type (regardless of whether such devices are present in CloudLab). Similarly, some usernames are commonly associated with specific software or operating system images intended for cloud platforms. Therefore, we split the usernames in *Log1* into three groups: non-administrator usernames, *generic* administrator usernames, and *device- or software-specific* usernames. We then manually classified the top 100 usernames, which collectively represented 83% of the failed logins. Of the top 100 usernames, 17% were non-administrator usernames, 67% were generic administrator usernames, and 16% targeted specific devices or software. Appendix E provides the full detail of these usernames and their classifications.

Non-administrator usernames consisted mostly of generic roles or common personal names containing no information about a specific targeted device or software. Most non-administrator usernames (66% of attacks in this username category) were based on specific user *roles*, such as `support`, `user1`, `profile1`, and `demo`. The rest were either personal names (e.g., `john`, `vivek`, `zhang`) or unattributable to particular attacker intentions (e.g., `default`, `ts`, etc.).

Generic administrator usernames were dominated by the username `root`, which accounted for 86% of all attacks in

[2]Not all CloudLab users logged in during the logged period.

this category of usernames, with admin being a distant second with 7%. Other generic administrator usernames included administrator, admin1, and sysadmin. Such generic administrator usernames are used broadly by UNIX-like systems, including servers, desktops, and even IoT devices, thus providing little information about specific targets.

Device- or software-specific usernames can be used for devices and software with default usernames (and sometimes default passwords). The largest set of such usernames (37%) targeted devices or software associated with network vendors Ubiquiti, Mikrotik, and Huawei. The nature of these attacks suggests that attackers attempt to gain access to entire networks by compromising routers and switches. Such an approach is particularly concerning because these vendors run the gamut from home to enterprise and from telecommunications to backbone networks. Another large set of attacks (30% of attacks in the category of device- or software-specific usernames) seemed to be leveraging usernames that match well-known software packages to target hosts with that software installed. While most of these usernames corresponded to typical server software such as oracle, mysql, hadoop, and nagios, there was a significant subset targeting gaming-related software, such as minecraft, csgoserver (Counter-Strike), and teamspeak3 (used for in-game voice chat). Additionally, 23% of attacks employing device- or software-specific usernames tried usernames associated with common server services (but not necessarily specific implementations of those services), such as web, ftp, and git.

Finally, 10% of attacks with device- or software-specific usernames appeared to target specific Linux distributions with usernames such as ubuntu, debian, and centos. Such usernames are often used by default in the disk images produced by these distributions for cloud use. Interestingly, the second-most-popular distribution-based username was pi, the default username for Raspberry Pi OS (formerly known as Raspbian), likely because, until 2022, this account had a well-known default username/password pair if sshd was enabled [61].

Do usernames become more popular when vulnerabilities are publicized? We noted that certain usernames in guessing vectors rose in volume with the release of corresponding Common Vulnerabilities and Exposures (CVEs) or breaking news about specific vulnerabilities. For instance, after a public disclosure that particular devices are affected by vulnerabilities, we found that a set of usernames associated with those devices surged among the attacks. Specifically, in mid and late 2019, it became known that vulnerabilities in products from video-centric IoT manufacturer Dahua [20] could be exploited for unauthorized remote access, device restart, or arbitrary code execution [21, 22, 52]. As Figure 6 shows, right after the public disclosure, there was a sharp spike in SSH attempts with two of Dahua's default usernames, 888888 and 666666 [23]. The two usernames follow a similar pattern, suggesting they belong to the same guessing vector. Notably, attacker interest in these usernames dwindled after the initial

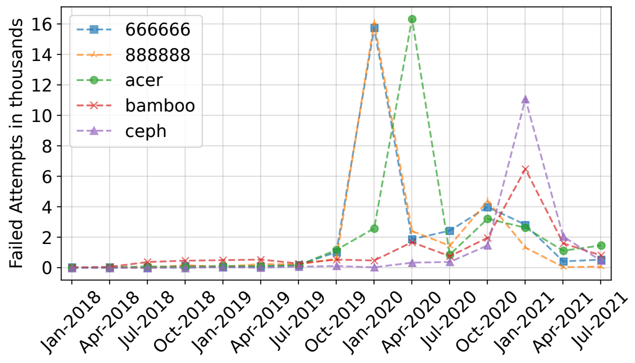


Figure 6: Spikes in usernames associated with CVEs.

spike, possibly because of lower-than- expected exploitability or the ban on these devices imposed by the US government due to security concerns [66]. Figure 6 also shows similar spikes in attacks for three other usernames acer, bamboo and ceph, when corresponding CVEs [51, 54, 55] were announced. Interestingly, none of these CVEs are directly related to SSH.

Sometimes, there was a direct or potential indirect connection between vulnerabilities and SSH access. In other cases, no such connection was obvious. It does appear that CVEs increase attacker use of related usernames, but it is not always clear *why*. Overall, however, attacker behavior shows the need to react relatively quickly when new usernames become more prevalent. An effective blocking strategy must be able to discover and block new usernames quickly.

6 Attacker Distribution and Persistence

SSH BFAs are highly distributed, but the persistence of attackers varies greatly. We examined trends in attacker distribution across countries and networks, along with their persistence across time. It is important to note that attackers often use compromised machines [24], so we cannot draw strong conclusions that the entity *controlling* an attack is based in the same country/network as the device *launching* the attack.

How widely are the attackers and legitimate users distributed? Only 1% of the IP addresses in *Log1* were used by legitimate users, with the other 99% (831K) used exclusively by attackers. These addresses were associated with 223 country codes and 18.5K network providers.

Most attacking IP addresses were from China (23%), followed by the United States (14.2%), Russia (5.8%), and Brazil (5.2%). Conversely, most legitimate IP addresses were from the United States (64%), followed by Indonesia (7.3%), Pakistan (5.2%), China (2.8%), and Brazil (2.8%). A majority of attacking IP addresses captured by honeypots have similarly been reported as originating from the United States and China [47, 64]. IP addresses from 21% of the country codes had at least one legitimate user, while those from the remaining 79% of the country codes were only sources of attacks. The number of IP addresses from a country was not always correlated with the number of *attacks* from that country. For instance, IP addresses from China were responsible for 44.7% of failed SSH login attempts, followed by those from the United States with 8.4%. IP addresses from China and the United States made up more than half of all failed login attempts. The distribution fell off quickly, with 99% of the failed logins coming from just 5.2% of all countries.

The number of attacks per network provider was similarly skewed. About half of the failed attempts were associated with just six network providers: China Telecom (26.8%), China Unicom (6.1%), DigitalOcean Cloud (6.0%), Tencent Cloud (5.5%), OVH Cloud (2.9%), and the Vietnam Posts and Telecommunications Group (2.7%). IP addresses from the top 100 network providers contributed 79.9% of the failed login attempts, with the top 13% contributing 99% of the failed attempts. Among the network providers seen in our data, 99 were used only by legitimate users, 431 by legitimate users as well as attackers, and the remaining 18,067 only by attackers. Such a distribution indicates that flagging entire networks as malicious is a poor strategy because most legitimate users are on networks that are also the sources of attacks.

Similar to prior research [16, 64], we observed a significant number of attacking IP addresses (at least 15.3%, responsible for 20% of the failed attempts) hosted by cloud providers. Of the top ten network providers based on the number of failed attempts, three were cloud providers. Six among the top 15 networks—and ten among the top 25—were cloud providers.

Further detail on the network providers and countries for IP addresses in *Log1* is included in Appendix F.

Are there instances of concentrated attacks from specific networks? The growth in the attacking IP addresses in CloudLab was generally steady over time. About 1–2% of the attacking IP addresses were new each month. At the high end, CloudLab experienced 4–6% new attackers in some months.

However; there were period during which there were large spikes in new addresses from specific networks or countries. To quantify this aspect, we calculated the month-to-month *growth ratio* for IP addresses from each country and network provider. We defined the growth ratio for a month as the number of new IP addresses seen that month divided by the average number of new IP addresses per month over the four-year logging period. The growth ratio enabled us to examine those network providers whose IP addresses exhibited high growth ratios at some point. Figure 7 shows growth ratios for four network service providers with Comcast (a large US ISP) as the baseline compared with Lumen/CenturyLink [69] (another large US ISP), Selectel [70] (a Russian cloud company that others have identified as an attack source [18]), and LG Uplus [68] (a South Korean mobile network operator). While there were many attackers from Comcast, they appeared at a fairly constant rate, with the growth ratio staying close to 1.0 as seen in Figure 7 In stark contrast, the IP addresses from the other three network providers exhibited large spikes, i.e.,

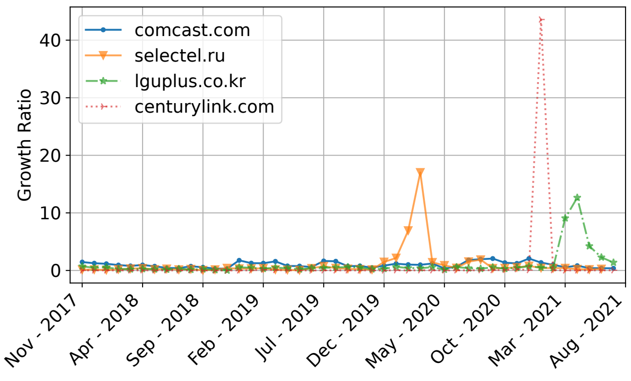


Figure 7: Growth ratios for four networks with Comcast as the baseline.

months during which there were far more new attackers from these networks than typical. We typically observed 82 new attacking IP addresses from Comcast per month, with a maximum of 170 new IP addresses in a month. In comparison, there were an average of 657 new attacking IP addresses per month from Lumen/CenturyLink, but the spike in January 2021 was composed of 28,640 new Lumen/CenturyLink IP addresses involved in attacks. Such concentrated attacks from a single network can have diverse causes, such as attackers forming botnets from vulnerable routers or cable modems specific to an ISP [50, 53], or gaining legal or illicit access to cloud or hosting services [12, 57]. Regardless of the reason, detecting spikes can help uncover and investigate anomalies.

How long do individual IP addresses persist? The IP addresses of most attackers and legitimate users had an active duration of less than one day, vanishing on the same day they first appeared. Figure 8 compares the IP addresses of attackers and legitimate users in terms of the active durations, i.e., the differences between the last and the first days on which the respective IP addresses were seen, in days. Although there were more IP addresses of legitimate users with short druations, the overall shapes of the distribution curves for the two groups are remarkably similar.

What metrics distinguish attackers and legitimate users? The ability to block attacks based on metrics, such as the number of attempts, relies mainly on finding metrics that clearly distinguish attackers from legitimate users. Figures 9 and 10 are scatter plots for three pairs of quantitative metrics, with each point in the plots representing one source IP address. Since there are far more attacking IP addresses than legitimate ones, the plots depicting the latter are sparser.

In all cases, legitimate users were "embedded" within the portions of the space covered by attackers. This can be clearly seen in Figure 9, which is a plot of the length of time between observing an IP address for the first and last times vs. the fraction of days within that span that the IP address was active. While legitimate users are clustered more around the axes of the plot, there are enough outliers such that few regions

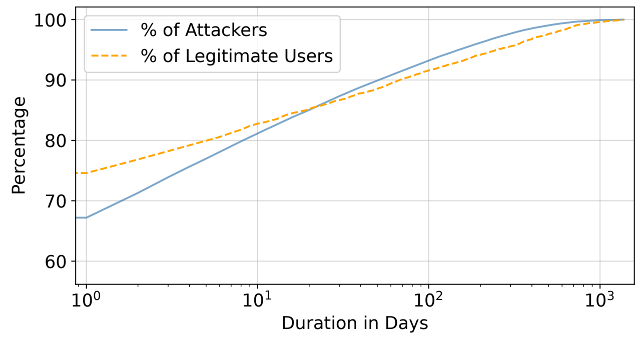


Figure 8: Cumulative Distribution Function (CDF) of the active durations of IP addresses used by attackers and legitimate users. The X axis is on a log scale.

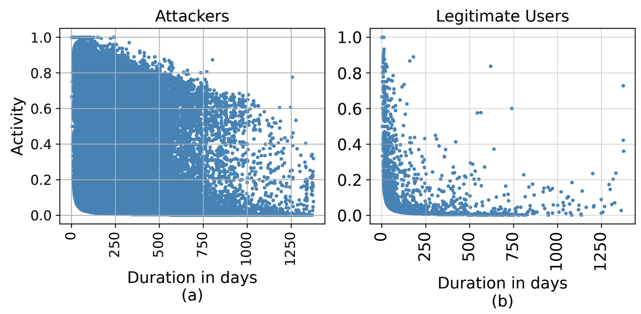


Figure 9: Duration (i.e., the span between the first and last connection) vs. activity ratio (i.e., the fraction of days in the span with connections) for IP addresses used by attackers and legitimate users.

of the plot are occupied exclusively by attackers. Practically speaking, using these metrics to mark IP addresses as attackers can catch a significant number of them, but at the expense of falsely flagging a number of legitimate users as attackers.

Plots (a) and (b) in Figure 10 show the total number of login attempts from each IP address vs. the number of usernames attempted. There is a clear region in each plot (above 100 usernames) containing a substantial number of attackers but only a single legitimate user. Still, these plots highlight the difficulty of trying to identify attackers with either metric because, like most attackers, a substantial number of IP addresses of legitimate users appear to use tens of usernames (likely due to NAT). In terms of the number of attempts, there is complete overlap on the X axis. Plots (c) and (d) in Figure 10 show similar observations for the number of hosts attacked by each source IP address vs. the number of usernames attempted.

7 Tracking Attacks With Dictionaries

As we saw in previous sections, BFAs come from a variety of locations, with attackers exhibiting a wide range of behavior. In addition, we showed that quantitative metrics alone cannot

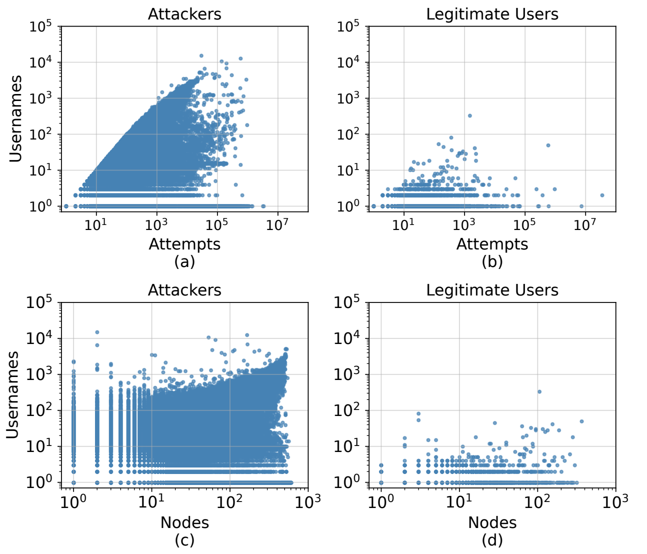


Figure 10: Plots (a) and (b) show the number of attempts vs. the number of usernames. Plots (c) and (d) show the number of hosts vs. the number of usernames for IP addresses used by attackers and legitimate users.

easily differentiate attackers from legitimate users. Therefore, we turned to identifying attacks by using the *sets of usernames* attempted by each source IP address as its *fingerprint*.

Constructing dictionaries and dictionary groups: We found that many individual source IP addresses had the same—or similar—fingerprints. We therefore compiled *dictionaries* of usernames, with each dictionary being a set of usernames used by a set of (more than one) source IP addresses. These dictionaries served two purposes. First, they enabled us to find the sets of usernames prevalent among attackers, forming the basis of DBB. Second, they helped us infer potential relationships between attacking IP addresses.

For an accurate count, we needed to find fingerprints that were not only identical but also *similar*. Such matching was required for two reasons. First, the data from production systems can be incomplete (as discussed in Section 3) so we needed a mechanism that can tolerate some amounts of missing data. Second, we hypothesized that attacker dictionaries would drift over time, with small additions of new usernames and/or removal of unfruitful ones, similar to dynamic credential fetching seen in recent malware [13]. Therefore, we developed a method for grouping similar dictionaries by determining the Jaccard Similarity (JS) [41] between username sets. The JS for two sets X and Y is defined as the ratio of $length|X \cap Y|$ to $length|X \cup Y|$. JS ranges from 0 to 1, being 0 when there is a null intersection between sets and 1 when the intersection and the union are the same.

Our method captures the transitive similarity between different dictionaries in two steps. First, we constructed an undirected graph `G = (V, E)` where each dictionary is represented as a vertex in V. An edge is added to E for every pair of vertices (v, v′) if $JS(v,v') \geq \hat{j}$, where JS(...) is the Jaccard similarity of the pair of dictionaries and $\hat{j}$ is a selected threshold. Second, we determined all *connected components* in G. i.e., groups of vertices $\tilde{V} \subset V$ for which every $v \in \tilde{V}$ can be reached from every other $v' \in \tilde{V}$ via the (undirected) edges. Note that individual dictionaries may form their own connected singleton components if they have no edges (i.e., they are not similar to any other dictionaries), and groups of dictionaries can become transitively included in larger components (e.g., if a dictionary A is similar to B, and B is in turn similar to C, the three dictionaries form one connected component). We selected a threshold value for $\hat{j}$ of 0.88 by considering a range of values and selecting the one that maximized the number of non-trivial (i.e., larger than size one) connected components. Intuitively, our choice of the threshold value is reasonable because it is large enough to group together only those dictionaries that are similar yet small enough for modes-sized dictionaries (e.g., with ten usernames) to pass the bar despite differing by one or two usernames.

Attacker Use of Dictionaries: Our construction of individual dictionaries from *Log1* resulted in 829 dictionaries covering 64% of the attacking IP addresses. After applying the similarity determination method above, we ended up with 48 non-trivial connected components, reducing the initial set of 829 dictionaries to a set of 682 Dictionaries *Groups* (DGs). As the numbers indicate, most DGs are singletons. A list of the most frequently encountered usernames in the DGs is included in Appendix D. Across the 682 DGs, the number of usernames per DG varied from one to ≈ 4.6K, with a median of four, and the number of attackers using the same DG varied from two[3] to ≈ 158K, with a median of 27.5. The numbers show that several DGs were large enough to make it highly unlikely that multiple sets of attackers independently stumbled upon them by coincidence. For dictionaries of non-trivial size, the use of the same dictionary intuitively suggests a potential connection between the attacking IPs—those employing the same set of usernames may be controlled by the same entity, using the same tool, following the same strategies, or sourcing the username lists from a common origin.

The DGs with a smaller number of usernames included mainly generic administrator or device/software-specific usernames as described in Section 5, while those with a larger number of usernames tended to contain guesses for usernames of regular users. Figure 11 depicts the number of IP addresses using a DG and the number of usernames in the DG. The pattern in Figure 11 shows that a large number of attackers use small DGs and vice versa. For instance, only two attackers used the largest DG with 4.6K usernames.

Attackers differed in the sequences in which they try usernames. The username sequences were random in some DGs, such as one with 18 usernames used by 3,447 attackers attempted in 3,342 unique sequences. The sequences were mostly the same in some other cases, such as a DG with just 41 unique sequences of 16 usernames used by 31,249 attackers. Such sequencing information can be used to fingerprint specific tools or scripts used by the attackers.

[3]The definition of a dictionary precludes the creation of dictionaries used by a single source.

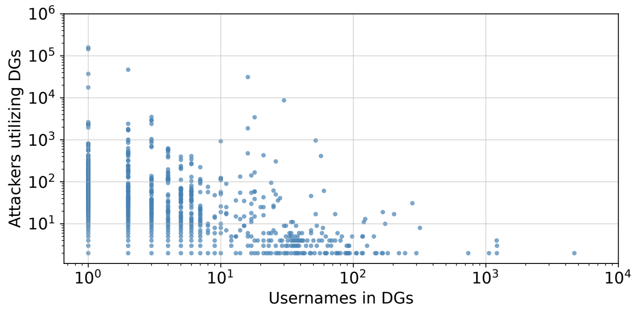


Figure 11: Usernames in a DG vs. attacking IP addresses using that DG. Both axes are on log scales.

Can dictionary-based blocking stop attacks? As we noted earlier, a small set of usernames dominated the BFAs in *Log1*. Most attackers used *at least one* username from a dictionary. We identified 9,819 unique usernames in the dictionaries. Blocking source IPs that used one of these usernames caught 100% of the attackers who used a complete dictionary and 84% of those who did not. We noted that a blocklist of around *10K usernames* (fewer than 4% of all usernames) effectively captured 94% of the attackers (the 64% of using a dictionary, plus 84% of the remaining 36%). Based on this observation, we designed a blocking approach called Dictionary-Based Blocking (DBB). As we describe in Section 8, we found the actual effectiveness of DBB to be even higher than expected: at timescales smaller than four years, only a few hundred usernames are needed at a time, and attackers employing these username lists are more active than average.

8 Dictionary-Based Blocking (DBB)

As we have shown in the previous sections, the usernames attempted by attackers are fundamental to BFAs. We leveraged this observation to devise DBB as an easy-to-apply technique to identify and block attackers and prevent SSH BFAs. We first evaluated DBB on *Log2*, which has no intersection with *Log1*, and later deployed it in production on a distinct cluster. By demonstrating the effectiveness of DBB over various sets of nodes, we highlight its general-purpose applicability for deployment on diverse machines over the Internet.

8.1 Threat Model

DBB is intended to counter attacks such as BFAs that target a large number of hosts and/or username/password sets. The threat model further assumes that an attacker may be aware of the presence of DBB and the dictionaries shared among the cooperating parties. Although we have described the use of DBB to protect individual hosts, it can be applied network-wide by using log files from individual hosts to generate rules for a border firewall. That said, DBB is not intended to defend against targeted attacks in which only a single facility is attacked for a short period. Further, DBB cannot defend against stolen credentials, which allow attackers to succeed at logging in on the first attempt. Defending against such attacks would require additional protective mechanisms [65, 71, 73].

8.2 Design of DBB

DBB involves a set of *collectors* that observe SSH login attempts. In the simplest cases, the collectors can be production machines that harvest their logs for usernames and source IP addresses associated with failed logins. Alternately, SSH honeypots or machines can be set up specifically as collectors. As described in Section 7, the collectors create a set of dictionaries by formulating a union of all usernames in their dictionaries. A Username Block List (*UBL*) is created by removing any locally valid usernames from the union. Each collector sends its *UBL* to a central *coordinator* that combines them with those received from other collections and distributes the union to any *defending hosts*, i.e., hosts that wish to defend themselves against SSH BFAs. DBB involves employing *UBL* to identify attacker IP addresses and blocking them at the defending host and/or at a network-wide firewall. When a defending host receives a failed login attempt for any username in the *UBL* that is not a local user[4], it blocks all further traffic from that source (e.g., by adding a local firewall rule [7]). We envision a deployment of DBB in which volunteers worldwide contribute *UBL*s from collectors on their networks to a trusted central coordinator that publishes a global *UBL* that any host can use for defense.

It is crucial to note a couple of important properties of DBB. First, no IP addresses are exchanged. As a result, unlike IP-based blocklists, DBB preserves the privacy of users and facilities and prevents “false accusations” against specific IP addresses. Second, DBB removes legitimate usernames when creating and applying *UBL*s since legitimate usernames can end up in dictionaries (as we showed in our analysis). Removal of legitimate usernames avoids information leakage from the collectors and prevents defending hosts from locking out unlucky users whose usernames end up in the *UBL*.

8.3 Effectiveness of DBB

We evaluated DBB using *Log2*, which was collected for nearly ten weeks—from November 2022 to January 2023—on three CloudLab sites: *Site-A*, *Site-B*, and *Site-C*. While *Site-B* and *Site-C* have characteristics similar to *Site-A*, they are geographically distributed and use distinct IP addresses on different networks. We simulated DBB on the data from these sites to derive and distribute new *UBL*s once a day using their log files as traces. In addition, we deployed DBB in production for three weeks at the *Emulab* [2] cluster in CloudLab.

Evaluating DBB: We simulated DBB at each site independently to determine its effectiveness by measuring the fraction of attack attempts blocked and the number of false positives (i.e., the number of blocked IP addresses of legitimate users).[5] We have reported false positives using absolute numbers rather than percentages because each false positive corresponds roughly to a single blocked user and/or one support ticket for the staff to resolve. For the same reason, we have reported false positives with respect to the number of blocked source IP addresses belonging to legitimate users.

We identified legitimate user IP addresses in *Log2* with the same method used for *Log1*, finding that *Site-A*, *Site-B* and *Site-C* had 2,952, 1,733, and 1,504 IP addresses of legitimate users, respectively. At *Site-A*, *Site-B* and *Site-C* DBB blocked 99.58%, 99.59%, and 99.39% of the BFAs with 17, 18, and 5 false positives, respectively. As we showed earlier, most attacks use a dictionary. Therefore, DBB achieved uniformly high block rates, with only 0.5% of BFAs going unblocked. The false positive rates for all sites were remarkably low, with an average of one false positive every five days.

Comparing DBB to Fail2ban: We ran simulations using *Log2* to compare DBB with Fail2ban [3], a widely deployed state-of-the-art tool for blocking SSH BFAs at the host. Fail2ban has three adjustable parameters: *Maxretry*—the number of failed attempts from the same IP address that activate blocking; *Findtime*—the period during which failed attempts are counted; and *Bantime*—the duration of the block. In comparison, DBB employs a single parameter equivalent to *maxretry*. Since we recommend blocking with DBB at the first failed login attempt, i.e. *maxretry* = 1, this can serve as the default configuration.

Trying every one of the numerous possible combinations of Fail2ban parameters is practically infeasible. Therefore, we tuned the parameters to conduct a fair comparison between Fail2ban and DBB without favoring either technique.We compared the performance of Fail2ban with DBB with three different settings variations: S_1**:** Default settings for DBB (*maxretry* = 1) and Fail2ban (*maxretry* = 5, *findtime* = 10 minutes, *bantime* = 10 minutes); S_2**:** Default settings for DBB (*maxretry* = 1) with Fail2ban adjusted to use the same *maxretry* value (*maxretry* = 1, *findtime* = 10 minutes *bantime* = 10 minutes); and S_3**:** No unblocking in Fail2ban (*bantime* = ∞) and varying values of *maxretry* from $[1, 45]$ for both techniques. All variations used the default Fail2ban *findtime* of 10 minutes.

[4]We recommend `root` be deemed an invalid username even if a facility permits its use.

[5]Even though defending hosts remove local usernames from the *UBL*, false positives can occur if a legitimate user tries to log in using a wrong username that happens to be in the *UBL* or shares an IP address with an attacker (e.g.. due to NAT).

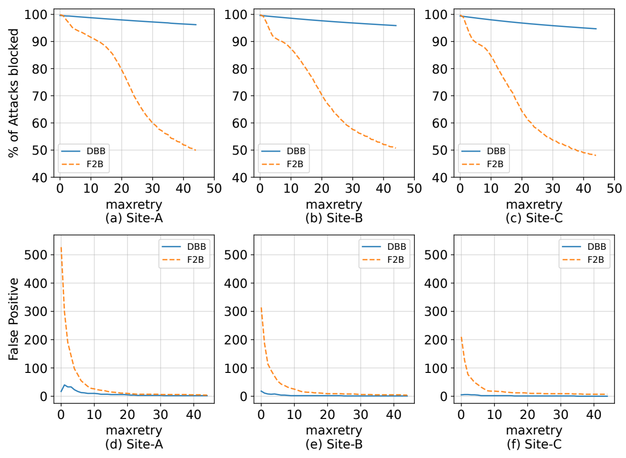


Figure 12: S_3: DBB and Fail2ban for *maxretry* ∈ [1, 45].

With default settings, DBB outperformed Fail2ban by a large margin. Across all sites, DBB blocked an average of 99.5% of the attacks, while Fail2ban blocked only 66.1%. One reason for this difference was the more aggressive default setting of DBB, with a lower value for *maxretry* compared to Fail2ban. Moreover, DBB generated only one-fifth as many false positives as Fail2ban (DBB_{FP}=40, F2B_{FP}=217). In other words, DBB locked far fewer legitimate users out of their accounts. When *maxretry* for Fail2ban was set to the DBB default of 1, the attacks blocked by Fail2ban increased from 66.1% to 85.20%. Appendix B illustrates site performance for exprement S_1 and S_2 . The tradeoff is in the number of false positives: the increase in blocked attacks was accompanied by a corresponding increase in false positives for Fail2ban from 217 to 1051, which was 26 times the rate for DBB.

To check if there is *any* setting for Fail2ban in which it meets or exceeds the performance of DBB, we varied *maxretry* for both from 1 to 45. In addition, we configured Fail2ban to mimic DBB by permanently blocking IP addresses. As Figure 12 shows, DBB achieved a better blocking rate than Fail2ban for all values of *maxretry* except 1, i.e., when Fail2ban was set to block an IP address on the first failed attempt and never unblock it. Moreover, DBB generated fewer false positives across all values of *maxretry* for all three sites. With increasing *maxretry*, the false positive rate of Fail2ban did progressively reduce to converge with that of DBB. However, the improvement came at the expense of reduced blocking effectiveness.

Overall, DBB's dictionary-based approach outperformed Fail2ban's rate-based approach. DBB can thwart nearly all incoming BFAs by blocking more aggressively without incurring many false positives because the approach is based on specific attacker behavior.

8.4 Number of Collectors

To examine the effect of the number of collectors on blocking performance, we computed dictionaries from C_n number of collectors $C_n \in \{[1,10] \cup \{20,30,40,50\}\}$ at each site in *Log2*. We then performed simulations using these reduced dictionaries as the *UBL*s of the hosts at the respective sites.

For each C_n, we ran the simulation nine times with randomly-selected collectors from all available nodes. Across these runs, the minimum and maximum blocked BFAs for *Site-A*, *Site-B*, *Site-C* were (98.2%, 99.5%), (97.7%, 99.5%) and (97.6%, 99.3%), respectively. All minima occurred at C_n = 1 and maxima at C_n = 50. DBB achieved most of its benefit with just one collector, with results within 1-2% of the performance when using *all* nodes as collectors. While there were slight increases in blocking with an increase in the number of collectors (full graphs are in Appendix C), the overall results show that DBB does not need a large number of collectors to achieve high performance. Over ten weeks, the *UBL* grew by only a few usernames per day, starting at ≈200 usernames per site and ending with 460 at the conclusion of the experiment.

Next, we checked whether *UBL*s created at one site are effective at other sites by testing all combinations in which each of the three sites was either a collector or deployment site. In all cases, DBB blocked at least 99.41% of the BFAs with at most 19 false positives (full details are provided in Appendix A). These results demonstrate that defending sites can implement DBB by simply using *UBL*s obtained from a trusted third party.

8.5 Deployment of DBB

We deployed DBB for three weeks on the *Emulab* [2] cluster of CloudLab (which consists of ≈400 nodes) with the three sites *Site-A*, *Site-B*, and *Site-C* as collectors. A single defending host removed locally valid usernames from the *UBL* and collected SSH logs from the other hosts in the cluster. Upon receiving a login attempt with any username in the local sanitized *UBL*, the defending host added the corresponding IP address to the blocklist. The blocklist was copied to the *Emulab*-wide firewall to block all further traffic from that IP address. For practicality, we updated the firewall blocklist on an hourly basis. The consequent gap between identifying attackers and subsequently blocking their access at the firewall can result in allowing attackers to keep trying SSH BFAs without restrictions for up to an hour in the worst case.

Before deploying DBB, we recorded the SSH traffic on *Emulab* for three weeks. Defense mechanisms already in place during this period consisted of two strategies: *lazy-fail2ban* and a firewall subscribing to a variety of public IP-address blocklists recommended by pfSense [8]. *lazy-fail2ban* adds an IP address to the blocklist at the firewall if the number of failed SSH login attempts from it crosses a threshold (i.e., *maxretry*=10). Unlike Fail2ban in standard configuration, *lazy-*

fail2ban operates without a finite *findtime*, instead using *findtime*=∞ and *bantime*=24 hours. When an IP address successfully logs in, the *maxretry* counter is reset to zero. *Emulab*'s firewall blocks *all* access from any IP address contained in the IP-address blocklists; this proactive measure is taken not only to counter SSH BFA but also to defend against attacks attempting to exploit other protocols and vulnerabilities.

Operational Effectiveness: It was possible to compute the precise percentage of BFAs blocked when assessing DBB performance in the simulation because we knew the total number of failed SSH attempts. However, blocking IP addresses in an operational system prevents subsequent failed attempts from that source, making it challenging to count the precise number of attack attempts that *would have been* logged without DBB. We therefore relied on the three-week record of failed attempts from *Emulab* to establish a baseline for estimating the proportion of attacks blocked. Importantly, these constraints mean that measurements of DBB performance blocking BFAs in operation provide lower bounds.

We found that DBB is significantly more effective than the existing blocking mechanism at *Emulab* during DBB deployment. DBB reduced failed SSH login attempts by 79.5%—from 80.6K to 16.5K per day—suggesting that it blocked four-fifths of the attacks not caught by the other defenses at *Emulab*. During the DBB deployment period, no legitimate user contacted the administrators about being blocked by DBB while two users contacted them because of blocking by *lazy-fail2ban*. One of the two blocked by *lazy-fail2ban* had mistakenly attempted to log in with a username resembling a university identification number, surpassing the *lazy-fail2ban* threshold and getting blocked. However, DBB did not block the user because the identification number used as the username was not in the *UBL*. The incident illustrates that the design of DBB has the advantage of minimizing the chances of blocking legitimate users because of inadvertent errors such as usernames with typos or those from other services.

8.6 Practical Considerations

DBB is a lightweight mechanism that involves negligible overhead for collectors or coordinators. The only requirement for DBB is that defending hosts have the ability to check username membership in a relatively small set. Moreover, the compact size of the *UBL* makes it well-suited for deployment on resource-constrained devices such as IoT devices. As a result, DBB can be easily deployed at larger scales by appropriately considering several practical factors.

Dictionaries Distribution: We propose hashing usernames before sharing with the coordinator. The approach permits easily testing membership in the *UBL* but avoids leaking valid usernames, thus making it difficult to use public *UBL*s to target high-value usernames. Moreover, hashed usernames prevent broader dissemination of newly discovered vulnerabilities pertaining to a username known only to a few attackers.

Dictionary Collection: Since dictionaries do not include IP addresses and locally valid usernames are removed before sharing, collectors can be set up anywhere on the Internet without raising privacy concerns. A collector need not reveal its own IP address and can be easily moved elsewhere to prevent adversaries from discovering it. However, DBB functions well even if the adversaries know the identities of some, or even *most*, collectors as long as there are a *few* collectors that are not known to the attackers. Since DBB does not place a high degree of trust in dictionary providers, attempting to deny service by inducing false inclusion of a username in a dictionary is ineffective. Therefore, a large network of volunteer collectors can operate with fairly light oversight.

Block Evasion: To evade DBB, an attacker must avoid high-value usernames in the public *UBL*, likely reducing the effectiveness of the attack. Attacking with a username not in the *UBL* requires that the username not be present in *any* dictionary. In other words, evading DBB requires avoiding *every* collector or avoiding using the same set of usernames from any source. The safest strategy for attempting to evade blocks is to mix high-value new usernames with unique "chaff" usernames in every attempt. Such an approach would slow the attacker down. Moreover, it would not help to have multiple attacking sources. In fact, it would require greater coordination or increased chaff to avoid reusing username sets.

9 Conclusion

While new cyberthreats emerge daily, attackers across the world continue to rely on simple, traditional approaches such as SSH BFAs. Yet, existing approaches face challenges in blocking such attacks with high accuracy without also blocking sizable numbers of legitimate users. The observed trends indicate an evolution of attacks on various accounts, software and devices, with attackers shifting from traditional *generic* attacks. Analyzing multi-year login attempt logs enables the effective identification of malicious activity by discerning differences in attacker and legitimate user behavior. We have shown that such insight can be applied for designing a lightweight blocking mechanism that can be deployed at scale with little overhead. Our approach outperforms the state-of-the-art in host-based SSH blocking, pointing the way to a new class of more effective defenses against BFAs.

Acknowledgements

We extend our sincere gratitude to the anonymous reviewers, our shepherd Bruce Davie, and the CloudLab administrators for their invaluable contributions and dedicated efforts in reviewing and supporting our research. Their insightful feedback have enriched the quality of our work. This material is based upon work supported by the National Science Foundation under Award Nos. 1743363, 1801446, and 2027208.

References

[1] Denyhosts. https://denyhosts.sourceforge.net/. Accessed 2024-03-03.

[2] Emulab. https://www.emulab.net/. Accessed 2024-03-03.

[3] Fail2ban. https://github.com/fail2ban/fail2ban. Accessed 2024-03-03.

[4] Hydra. https://www.kali.org/tools/hydra/. Accessed 2024-03-03.

[5] IPinfo. https://ipinfo.io/. Accessed 2024-03-03.

[6] Ncrack. https://nmap.org/ncrack/. Accessed 2024-03-03.

[7] The Netfilter.org "iptables" project. https://www.netfilter.org/projects/iptables/index.html.

[8] pfsense. https://www.pfsense.org/. Accessed 2024-03-03.

[9] SSH Guard. https://www.sshguard.net/. Accessed 2024-03-03.

[10] A. Abdou, D. Barrera, and P. C. van Oorschot. What lies beneath? Analyzing automated SSH bruteforce attacks. In F. Stajano, S. F. Mjølsnes, G. Jenkinson, and P. Thorsheim, editors, *Technology and Practice of Passwords*, pages 72–91, Cham, 2016. Springer International Publishing. DOI: 10.1007/978-3-319-29938_6.

[11] Akamai Security Research. Panchan's mining rig: New Golang peer-to-peer botnet says "Hi!". https://www.akamai.com/blog/security/new-p2p-botnet-panchan, Jun 2022. Accessed 2024-03-03.

[12] S. Alrwais, X. Liao, X. Mi, P. Wang, X. Wang, F. Qian, R. Beyah, and D. McCoy. Under the shadow of sunshine: Understanding and detecting bulletproof hosting on legitimate service provider networks. In *2017 IEEE Symposium on Security and Privacy*, IEEE S&P 2017, pages 805–823, 2017. DOI: 10.1109/SP.2017.32.

[13] 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 *26th USENIX Security Symposium*, USENIX Security 17, pages 1093–1110, Vancouver, BC, Aug 2017. USENIX Association. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[14] ARIN. American Registry for Internet Numbers. https://www.arin.net/. Accessed 2024-03-03.

[15] T. Barron and N. Nikiforakis. Picky attackers: Quantifying the role of system properties on intruder behavior. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC '17, pages 387–398, New York, NY, USA, 2017. Association for Computing Machinery. DOI: 10.1145/3134600.3134614.

[16] M. S. Bohuk, M. Islam, S. Ahmad, M. Swift, T. Ristenpart, and R. Chatterjee. Gossamer: Securely measuring password-based logins. In *31st USENIX Security Symposium*, USENIX Security 22, pages 1867–1884, Boston, MA, Aug. 2022. USENIX Association. https://www.usenix.org/conference/usenixsecurity22/presentation/sanusi-bohuk.

[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)*, pages 667–682, Boston, MA, Feb. 2019. USENIX Association. https://www.usenix.org/conference/nsdi19/presentation/cao.

[18] Check Point Research. Stopping serial killer: Catching the next strike. https://research.checkpoint.com/2021/stopping-serial-killer-catching-the-next-strike/, Jan 2021. Accessed 2024-03-03.

[19] P. J. Criscuolo. Distributed denial of service: Trin00, tribe flood network, tribe flood network 2000, and stacheldraht. *Department of Energy Computer Incident Advisory Capability (CIAC)*, UCRLID-136939, Rev. 1, Feb 2000.

[20] Dahua Technology. Dahua Intelligent Solutions. https://www.dahuasecurity.com/. Accessed 2024-03-03.

[21] Dahua Technology. Security advisory — Buffer overflow vulnerability found in some Dahua IP camera devices. https://www.dahuasecurity.com/support/cybersecurity/details/617, 2019. Accessed 2022-03-01.

[22] Dahua Technology. Security advisory — Some products of Dahua have security risks. https://www.dahuasecurity.com/support/cybersecurity/details/637, 2019. Accessed 2022-03-01.

[23] Dahua Wiki. Dahua default usernames and passwords. https://dahuawiki.com/UsernameandPassword. Accessed 2022-03-01.

[24] 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*, MobiSys '19, pages 482–493, New York, NY, USA, 2019. Association for Computing Machinery. DOI: 10.1145/3307334.3326083.

[25] Dr. Web. Linux.muldrop.14. https://vms.drweb.com/virus/?i=15391790, 2017. Accessed 2024-03-03.

[26] M. Drašar. Protocol-independent detection of dictionary attacks. In T. Bauschert, editor, *Advances in Communication Networking*, pages 304–309, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-40552-5_30.

[27] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb, A. Akella, K. Wang, G. Ricart, L. Landweber, C. Elliott, M. Zink, E. Cecchet, S. Kar, and P. Mishra. The design and operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, July 2019.

[28] K. Egevang and P. Francis. The IP Network Address Translator (NAT). *RFC 1631*, 1994. https://www.rfc-editor.org/rfc/rfc1631 Accessed 2024-03-03.

[29] D. Garn. Eight ways to protect SSH access on your system. https://www.redhat.com/sysadmin/eight-ways-secure-ssh, Oct 2020. Accessed 2024-03-03.

[30] V. Ghiette, 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, pages 61–71, Chaoyang District, Beijing, Sept. 2019. USENIX Association. https://www.usenix.org/conference/raid2019/presentation/ghiette.

[31] J. Hancock, T. M. Khoshgoftaar, and J. L. Leevy. Detecting SSH and FTP brute force attacks in big data. In *20th IEEE International Conference on Machine Learning and Applications*, ICMLA 2021, pages 760–765, 2021. DOI: 10.1109/ICMLA52953.2021.00126.

[32] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras. SSHCure: A flow-based SSH intrusion detection system. In R. Sadre, J. Novotný, P. Čeleda, M. Waldburger, and B. Stiller, editors, *Dependable Networks and Services*, pages 86–97, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-30633-4_11.

[33] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras. SSH compromise detection using NetFlow/IPFIX. *SIGCOMM Comput. Commun. Rev.*, 44(5):20–26, Oct 2014. DOI: 10.1145/2677046.2677050.

[34] R. Hofstede, M. Jonker, A. Sperotto, and A. Pras. Flow-based Web application brute-force attack and compromise detection. *Journal of Network and Systems Management*, 25(4):735–758, 2017. DOI: 10.1007/s10922-017-9421-4.

[35] R. Hofstede, A. Pras, A. Sperotto, and G. D. Rodosek. Flow-based compromise detection: Lessons learned. *IEEE Security & Privacy*, 16(1):82–89, 2018. DOI: 10.1109/MSP.2018.1331021.

[36] S. Hogg. IPv6 security vulnerability scanning. https://blogs.infoblox.com/ipv6-coe/ipv6-security-vulnerability-scanning/, Sep 2016. Accessed 2024-03-03.

[37] M. D. Hossain, H. Ochiai, F. Doudou, and Y. Kadobayashi. SSH and FTP brute-force attacks detection in computer networks: LSTM and machine learning approaches. In *5th International Conference on Computer and Communication Systems*, ICCCS 2020, pages 491–497, 2020. DOI: 10.1109/ICCCS49078.2020.9118459.

[38] K. Hynek, T. Beneš, T. Čejka, and H. Kubátová. Refined detection of SSH brute-force attackers using machine learning. In M. Hölbl, K. Rannenberg, and T. Welzer, editors, *ICT Systems Security and Privacy Protection*, pages 49–63, Cham, 2020. Springer International Publishing. DOI: 10.1007/978-3-030-58201-2_4.

[39] International Standards Organization. ISO 3166-1:2020: Codes for the representation of names of countries and their subdivisions — Part 1: Country codes. https://www.iso.org/iso-3166-country-codes.html, 2020.

[40] L. Izhikevich, M. Tran, M. Kallitsis, A. Fass, and Z. Durumeric. Cloud watching: Understanding attacks against cloud-hosted services. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, IMC '23, pages 313–327, New York, NY, USA, 2023. Association for Computing Machinery. DOI: 10.1145/3618257.3624818.

[41] P. Jaccard. The distribution of the flora in the Alpine zone. *New Phytologist*, 11(2):37–50, 1912. DOI: 10.1111/j.1469-8137.1912.tb05611.x.

[42] M. Jonker, R. Hofstede, A. Sperotto, and A. Pras. Unveiling flat traffic on the Internet: An SSH attack case study. In *2015 IFIP/IEEE International Symposium on Integrated Network Management*, IM 2015, pages 270–278, 2015. DOI: 10.1109/INM.2015.7140301.

[43] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. DDos in the IoT: Mirai and other botnets. *Computer*, 50(07):80–84, Jul 2017. DOI: 10.1109/MC.2017.201.

[44] J. Lane Thames, R. Abler, and D. Keeling. A distributed active response architecture for preventing SSH dictionary attacks. In *IEEE SoutheastCon 2008*, pages 84–89, 2008. DOI: 10.1109/SECON.2008.4494264.

[45] T.-H. Lee, L.-H. Chang, and C.-W. Syu. Deep learning enabled intrusion detection and prevention system over SDN networks. In *2020 IEEE International Conference on Communications Workshops*, ICC 2020 Workshops, pages 1–6, 2020. DOI: 10.1109/ICCWorkshops49005.2020.9145085.

[46] P. Muncaster. Massive Qbot botnet strikes 500,000 machines through WordPress. *Infosecurity Magazine*, Oct 2014. https://www.infosecurity-magazine.com/news/massive-qbot-strikes-500000-pcs/.

[47] C. Munteanu, S. J. Saidi, O. Gasser, G. Smaragdakis, and A. Feldmann. Fifteen months in the life of a honeyfarm. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, IMC '23, pages 282–296, New York, NY, USA, 2023. Association for Computing Machinery. DOI: 10.1145/3618257.3624826.

[48] M. M. Najafabadi, T. M. Khoshgoftaar, C. Calvert, and C. Kemp. Detection of SSH brute force attacks using aggregated netflow data. In *IEEE 14th International Conference on Machine Learning and Applications*, ICMLA 2015, pages 283–288, 2015. DOI: 10.1109/ICMLA.2015.20.

[49] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, and R. Zuech. Machine learning for detecting brute force attacks at the network level. In *2014 IEEE International Conference on Bioinformatics and Bioengineering*, pages 379–385, 2014. DOI: 10.1109/BIBE.2014.73.

[50] National Vulnerability Database. CVE-2014-8423 detail. Nov 2014. https://nvd.nist.gov/vuln/detail/CVE-2014-8423.

[51] National Vulnerability Database. CVE-2019-18670 detail. Dec 2019. https://nvd.nist.gov/vuln/detail/CVE-2019-18670.

[52] National Vulnerability Database. CVE-2019-9676 detail. June 2019. https://nvd.nist.gov/vuln/detail/CVE-2019-9676.

[53] National Vulnerability Database. CVE-2019-19494 detail. Jan 2020. https://nvd.nist.gov/vuln/detail/CVE-2019-19494.

[54] National Vulnerability Database. CVE-2020-8566 detail. Dec 2020. https://nvd.nist.gov/vuln/detail/CVE-2020-8566.

[55] National Vulnerability Database. CVE-2021-26067 detail. Jan 2021. https://nvd.nist.gov/vuln/detail/CVE-2021-26067.

[56] S. News. Poorly secured SSH servers targeted by Chalubo botnet. https://news.sophos.com/en-us/2018/10/24/poorly-secured-ssh-servers-targeted-by-chalubo-botnet/, Oct 2018. Accessed 2024-03-03.

[57] A. Noroozian, J. Koenders, E. van Veldhuizen, C. H. Ganan, S. Alrwais, D. McCoy, and M. van Eeten. Platforms in everything: Analyzing ground-truth data on the anatomy and economics of bullet-proof hosting. In *28th USENIX Security Symposium*, USENIX Security 19, pages 1341–1356, Santa Clara, CA, Aug. 2019. USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/noroozian.

[58] 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.

[59] S. Pastrana and G. Suarez-Tangil. A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth. In *Proceedings of the Internet Measurement Conference*, IMC '19, pages 73–86, New York, NY, USA, 2019. Association for Computing Machinery. DOI: 10.1145/3355369.3355576.

[60] A. Rahman, M. R. Rahman, C. Parnin, and L. Williams. Security smells in Ansible and Chef scripts: A replication study. *ACM Trans. Softw. Eng. Methodol.*, 30(1), Jan 2021. DOI: 10.1145/3408897.

[61] Raspberry Pi Documentation. Configure a user manually. https://www.raspberrypi.com/documentation/computers/configuration.html#configuring-a-user. Accessed 2024-03-03.

[62] G. K. Sadasivam, C. Hota, and B. Anand. Classification of SSH attacks using machine learning algorithms. In *6th International Conference on IT Convergence and Security*, ICITCS 2016, pages 1–6, 2016. DOI: 10.1109/ICITCS.2016.7740316.

[63] S. Schechter, Y. Tian, and C. Herley. StopGuessing: Using guessed passwords to thwart online guessing. In *2019 IEEE European Symposium on Security and Privacy*, EuroS&P 2019, pages 576–589, 2019. DOI: 10.1109/EuroSP.2019.00048.

[64] Z. Shamsi, D. Zhang, D. Kyoung, and A. Liu. Measuring and clustering network attackers using medium-interaction honeypots. In *2022 IEEE European Symposium on Security and Privacy Workshops*, EuroS&PW 2022, pages 294–306, 2022. DOI: 10.1109/EuroSPW55150.2022.00036.

[65] A. Sharma, A. Z. Kalbarczyk, Sharma, R. Iyer, and J. Barlow. Analysis of credential stealing attacks in an open networked environment. In *2010 Fourth International Conference on Network and System Security*, pages 144–151, 2010. DOI: 10.1109/NSS.2010.56.

[66] B. Toulas. Unpatched Dahua cams vulnerable to unauthenticated remote access. https://www.bleepingcomputer.com/news/security/unpatched-dahua-cams-vulnerable-to-unauthenticated-remote-access/, Oct 2021. Accessed 2024-03-03.

[67] J. Vykopal, T. Plesnik, and P. Minarik. Network-based dictionary attack detection. In *2009 International Conference on Future Networks*, pages 23–27, 2009. DOI: 10.1109/ICFN.2009.36.

[68] Wikipedia. LG Uplus. https://en.wikipedia.org/wiki/LG_Uplus. Accessed 2024-03-03.

[69] Wikipedia. Lumen Technologies. https://en.wikipedia.org/wiki/Lumen_Technologies. Accessed 2024-03-03.

[70] Wikipedia. Selectel. https://en.wikipedia.org/wiki/Selectel. Accessed 2024-03-03.

[71] Y. Wu, P. M. Cao, A. Withers, 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.

[72] T. Ylönen. SSH—Secure login connections over the Internet. In *Proceedings of the Sixth USENIX Security Symposium*, pages 37–42.

[73] W. Yurcik and C. Liu. A first step toward detecting ssh identity theft in hpc cluster environments: discriminating masqueraders based on command behavior. In *IEEE International Symposium on Cluster Computing and the Grid, 2005*, volume 1 of *CCGrid 2005*, pages 111–120, 2005. DOI: 10.1109/CCGRID.2005.1558542.

A Cross collector performance

Table A.1 shows the performance of DBB when the collector is not at the deployed site. We conducted simulations for all possible combinations of our sites.

Table A.1: Performance of DBB when collector and deployed site are different.

Deployed Site	Collector Site	% Attacks Blocked	False Positive
C	*A*	99.45	4
C	*B*	99.41	4
A	*C*	99.55	17
A	*B*	99.56	19
B	*C*	99.58	17
B	*A*	99.61	18

B Individual site performance in experiments S_1 and S_2

Figure B.1 presents the performance of DBB and F2B in experiments S_1 and S_2.

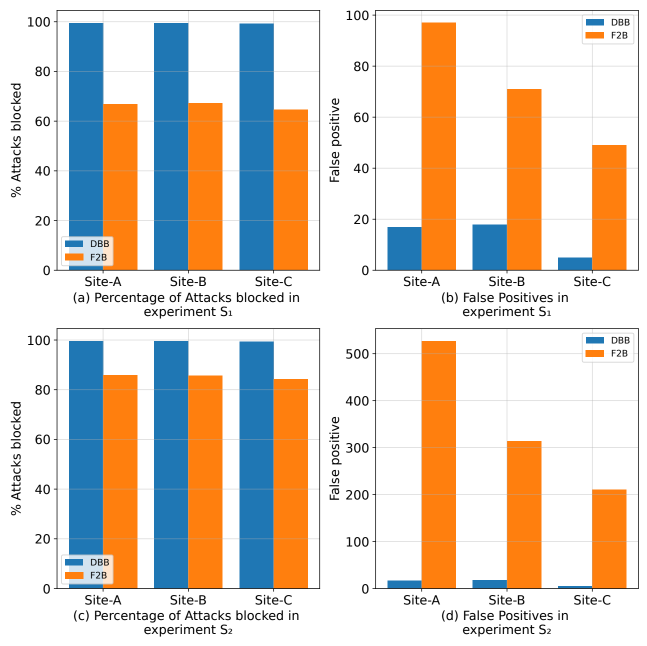


Figure B.1: DBB and F2B using settings S_1 and S_2.

C C_n : Number of Collectors

Figure C.1 depicts how the performance of DBB varies based on the number of collectors (C_n).

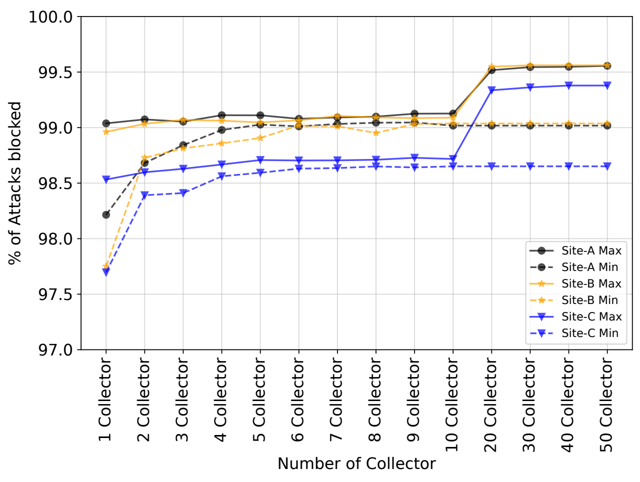


Figure C.1: Performance of DBB as the number of collectors increases.

D Top usernames in DG

Table D.1 shows the usernames present in at least 5% of the DGs.

Table D.1: Usernames in $\geq 5\%$ of all DGs.

Username	Percentage of DGs	Username	Percentage of DGs
root	69.94	debian	8.65
admin	55.87	centos	8.65
user	36.07	demo	8.65
test	30.21	minecraft	8.36
support	26.83	zabbix	7.92
ubnt	24.93	odoo	7.92
oracle	23.02	server	7.92
ubuntu	22.87	ts3	7.92
postgres	20.38	apache	7.77
ftp	20.09	teamspeak	7.77
pi	19.21	dev	7.62
guest	18.48	vagrant	7.33
git	16.42	web	7.33
service	15.54	mother	7.33
usuario	14.96	test1	7.04
mysql	14.22	administrator	6.89
nagios	13.49	system	6.74
hadoop	12.76	weblogic	6.30
tomcat	11.88	steam	6.16
jenkins	11.29	svn	5.43
user1	10.12	ansible	5.28
www	9.82	test2	5.28
student	9.38	kafka	5.13
supervisor	9.24	alex	5.13
deploy	8.80	webmaster	5.13

E Username Classification

Table E.1 contains our manual classification of the 100 most-attempted usernames. (Usernames colliding with real CloudLab usernames have been omitted for privacy.)

Username	Percentage of Attempts	Cumulative Percentage	Category	Sub-category
root	47.81	47.81	admin	–
admin	3.97	51.78	admin	–
support	2.66	54.44	non-admin	role
ubnt	1.82	56.26	specific	network
user1	1.79	58.06	non-admin	role
default	1.78	59.83	non-admin	misc
MikroTik	1.72	61.56	specific	network
administrator	1.71	63.26	admin	–
admin1	1.70	64.97	admin	–
profile1	1.70	66.67	non-admin	role
demo	1.70	68.37	non-admin	role

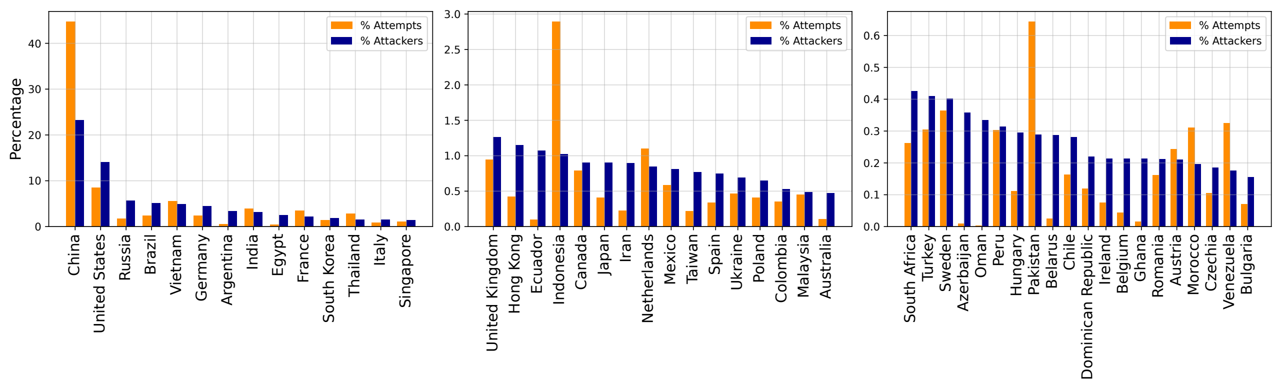
Username	Percentage of Attempts	Cumulative Percentage	Category	Sub-category
web	1.70	70.07	specific	service
tech	1.59	71.66	non-admin	role
telecomadmin	1.54	73.20	specific	network
oracle	0.78	73.98	specific	software
ubuntu	0.64	74.62	specific	distribution
ftp	0.58	75.20	specific	service
postgres	0.50	75.70	specific	software
pi	0.40	76.10	specific	distribution
git	0.35	76.45	specific	service
guest	0.31	76.75	non-admin	role
test1	0.29	77.05	non-admin	role
export	0.28	77.33	non-admin	role
usuario	0.28	77.61	non-admin	role
test2	0.26	77.86	non-admin	role
mysql	0.21	78.07	specific	software
hadoop	0.20	78.28	specific	software
deploy	0.18	78.45	non-admin	role
nagios	0.17	78.62	specific	software
jenkins	0.16	78.78	specific	software
dev	0.15	78.93	non-admin	role
www	0.14	79.07	specific	service
debian	0.13	79.20	specific	distribution
minecraft	0.12	79.32	specific	software
odoo	0.11	79.44	specific	software
ansible	0.11	79.55	specific	software
teamspeak	0.11	79.66	specific	software
student	0.11	79.76	non-admin	role
tomcat	0.10	79.87	specific	software
ts3	0.10	79.97	specific	software
server	0.10	80.07	non-admin	role
centos	0.09	80.16	specific	distribution
es	0.09	80.24	specific	software
zabbix	0.08	80.32	specific	software
weblogic	0.08	80.40	specific	software
steam	0.07	80.47	specific	software
vagrant	0.06	80.54	specific	software
elasticsearch	0.06	80.60	specific	software
elastic	0.06	80.66	specific	software
webadmin	0.06	80.72	specific	service
kafka	0.06	80.78	specific	software
ftpadmin	0.06	80.84	specific	service
webmaster	0.06	80.90	specific	service
vnc	0.06	80.96	specific	software
system	0.06	81.01	admin	–
contador	0.06	81.07	non-admin	role
ftptest	0.06	81.13	specific	service
service	0.06	81.18	non-admin	role
baikal	0.06	81.24	specific	software
ts	0.05	81.29	non-admin	misc
duni	0.05	81.34	non-admin	misc
temp	0.05	81.40	non-admin	misc
spark	0.05	81.45	specific	software

Username	Percentage of Attempts	Cumulative Percentage	Category	Sub-category
svn	0.05	81.50	specific	service
docker	0.05	81.55	specific	software
developer	0.05	81.60	non-admin	role
jira	0.05	81.66	specific	software
app	0.05	81.70	non-admin	misc
sinusbot	0.05	81.75	specific	software
apache	0.05	81.80	specific	software
sysadmin	0.05	81.84	admin	–
nexus	0.05	81.89	specific	software
uftp	0.04	81.93	specific	service
ec2-	0.04	81.98	non-admin	misc
bot	0.04	82.02	non-admin	misc
butter	0.04	82.06	specific	software
mcserver	0.04	82.10	specific	software
teamspeak3	0.04	82.14	specific	software
nginx	0.04	82.18	specific	software
csgo	0.04	82.22	specific	software
backup	0.04	82.25	non-admin	role
vbox	0.04	82.29	specific	software
csgoserver	0.04	82.33	specific	software
gpadmin	0.04	82.36	specific	software
info	0.03	82.40	non-admin	misc
hd	0.03	82.43	non-admin	misc
a	0.03	82.46	non-admin	misc
db	0.03	82.50	specific	service
teste	0.03	82.53	non-admin	misc
user2	0.03	82.56	non-admin	role
deployer	0.03	82.59	non-admin	role
daniel	0.03	82.63	non-admin	name
nvidia	0.03	82.66	specific	software
db2inst1	0.03	82.69	specific	software
ethos	0.03	82.72	specific	distribution
manager	0.03	82.75	non-admin	role
www-data	0.03	82.78	specific	service
wp	0.03	82.81	specific	software
redis	0.03	82.84	specific	software
testing	0.03	82.87	non-admin	role

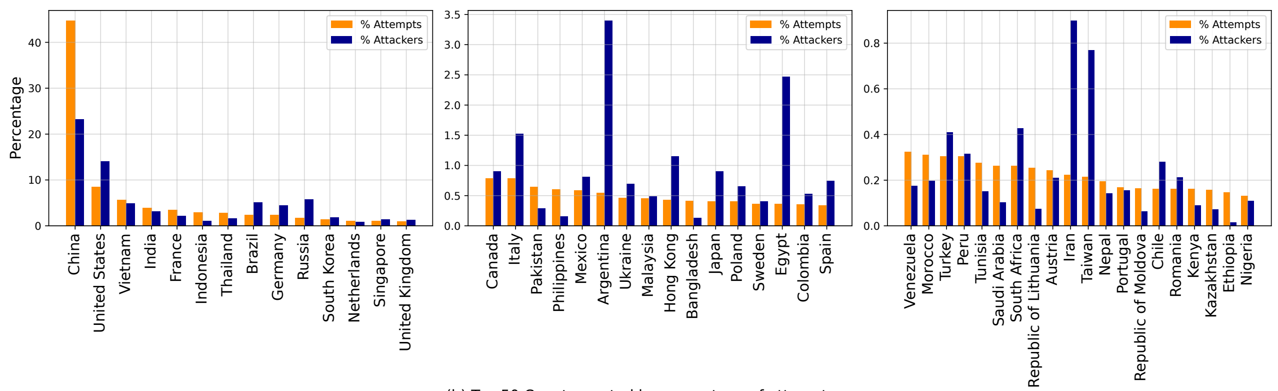
Table E.1: Username classification

F Top 50 Countries And Network Providers

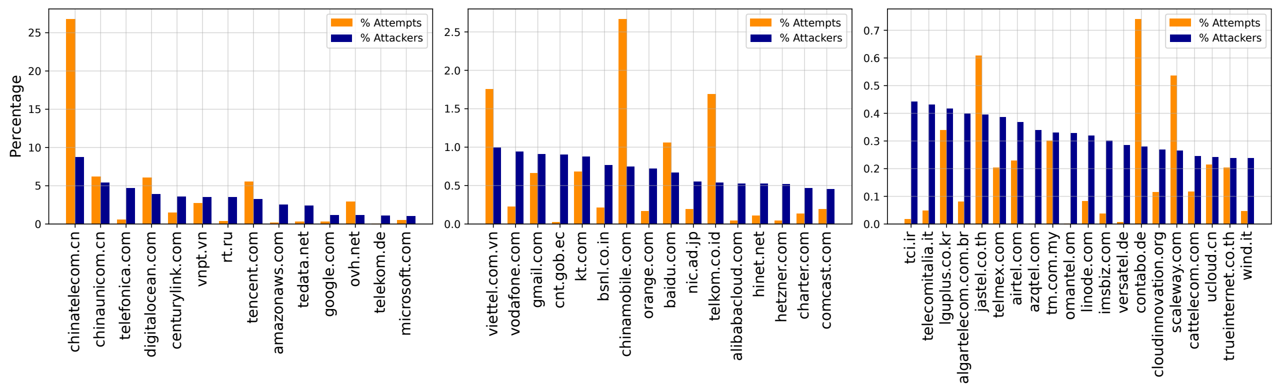
Figure F shows the top 50 countries and network providers based on the percentage of attackers and attempts.



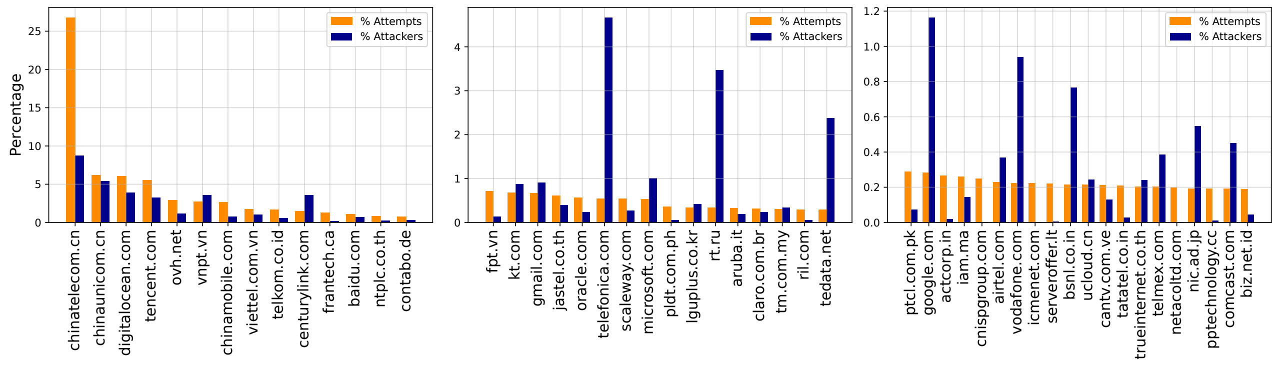
(a) Top 50 Country sorted by percentage of attackers



(b) Top 50 Country sorted by percentage of attempts



(c) Top 50 Network Providers sorted by percentage of attackers



(d) Top 50 Network Providers sorted by percentage of attempts