What Lies Beneath? Analyzing Automated SSH Bruteforce Attacks

AbdelRahman Abdou[1], David Barrera[2], and Paul C. van Oorschot[1]

[1] Carleton University, Canada
[2] ETH Zürich, Switzerland

Abstract. We report on what we believe to be the largest dataset (to date) of automated secure shell (SSH) bruteforce attacks. The dataset includes plaintext password guesses in addition to timing, source, and username details, which allows us to analyze attacker behaviour and dynamics (*e.g.,* coordinated attacks and password dictionary sharing). Our methodology involves hosting six instrumented SSH servers in six cities. Over the course of a year, we recorded a total of ∼17M login attempts originating from 112 different countries and over 6K distinct source IP addresses. We shed light on attacker behaviour, and based on our findings provide recommendations for SSH users and administrators.

1 Introduction

Internet accessible secure shell (SSH [18]) servers are consistently flooded with credential guessing attempts originating from a wide range of globally-distributed hosts. Every login attempt generates an entry in a log file which conscientious system administrators painstakingly monitor. While some of these guesses target specific systems and user accounts, most aim to identify weak authentication configurations on the entire public IPv4 address space every minute of every day. This status quo of seemingly endless login attempts has led to a cottage industry of software tools, recommended service configuration, and a folklore for throttling the frequency (and corresponding log entries) of such attacks.

In principle, sensible configurations (*e.g.,* rate-limiting login attempts, disabling remote logins for privileged accounts) of remotely accessible services in conjunction with strong password selection by users should suffice in limiting the success of such attacks. However, the low cost of executing these attacks from geographically-diverse datacenters or from compromised end user systems (*e.g.,* as part of a botnet) has apparently made these guessing attacks viable for attackers. Administrators also appear hesitant to deploy restrictive network access control (*e.g.,* IP address range-blocking) configurations which may lead to increased service desk call volume or poor user experience.

Version: November 20, 2015. The official Passwords 2015 proceedings version will appear in Springer LNCS (DOI to be available).

Well known software tools exist (many are free and open source) to monitor and limit the number of login attempts that can be performed on a given server. These tools usually maintain state measuring login attempts per source IP address, or login attempts from any source within a defined period of time. Once some pre-defined threshold is met, future login attempts are blocked or throttled, either for a specified time period. Popular SSH server implementations (*e.g.,* OpenSSH and Dropbear) ship with default configurations that limit the number of allowed authentication attempts per session (*e.g.,* disconnecting the remote system after three unsuccessful attempts) and disable remote logins for privileged accounts. These settings can be easily configured by administrators. Despite the widespread availability of tools and configurations, SSH bruteforce login attacks are on the rise [4,1], supporting the theory that some of these login attempts must be succeeding.

While academic work in the field has identified attacker use of common usernames and passwords [14], and also focused on detection of highly distributed or stealthy attacks [13], to our knowledge, there is little academic work in understanding the anatomy of run-of-the-mill automated SSH bruteforce attacks. Understanding these attacks can shed light on how attackers create and use password dictionaries, guessing strategies, and adaptive techniques. Insight on attacker behaviour can help design and improve defensive techniques, and demonstrate the ineffectiveness of others. This paper makes the following contributions:

- We describe a data collection methodology for SSH login attempts that allowed us to record over 17 million guesses including over 1.4 million plaintext passwords and nearly 28 thousand usernames. The vast majority of these login attempts were performed by automated software tools.
- We perform an in-depth analysis of these login attempts by analyzing two datasets; the first contains log entries collected on a single system over a one-year period; the second includes logs from five servers over a 10-week period. Among others, our analysis offers insights into guessing strategies, password dictionaries used, and data sharing amongst attackers.

The sequel is organized as follows. Section 2 presents related work. Section 3 describes our data collection methodology and initial dataset observations. In Section 4, we detail characteristics of attacking sources including network location and number of attacker IPs per network block. Section 5 analyzes password guesses, including composition, re-use, and password sharing among sources. Section 6 covers distribution of usernames and password guesses per username. Section 7 provides details on timing dynamics. We discuss recommendations for users and administrators in Section 8, and conclude in Section 9.

2 Related Work

Host-based detection of bruteforce attacks. By default, SSH server software generates log entries when authentication requests are received. An entry includes timestamp, source IP address, source port, username, and the result

of the authentication request (*e.g.,* success, incorrect password, invalid user, *etc.*). These logs can be locally monitored by client software, such as Blockhosts (`www.aczoom.com/blockhosts`), Fail2Ban (`www.fail2ban.org`), or DenyHosts (`www.denyhosts.sourceforge.net`), to detect attacks by observing recent login attempts and blocking sources that exceed a threshold of login attempts per time period. Source blocking can be enforced at the network level (*i.e.,* by adding network firewall rules) or at the application level (*e.g.,* by using `tcpwrappers`). Certain tools allow submitting failed attempts to centralized services for aggregation and analytics [1,17].

Analysis of Passwords used in SSH Attacks. Most closely related to our work, Owens and Matthews [14] collected around 103,000 login attempts on three honeypots over an 11 week period in 2007-2008. The authors report an overwhelming majority of attempts targeting the `root` account, and nearly 49% of all attempted guesses had the username equal to the password. Compared to the size of the dataset of Owens and Matthews, the one collected herein is an order of magnitude larger. In addition, our attacker labelling methodology is more comprehensive by considering usernames, passwords, and timing independently, which allows us to identify dictionary re-use regardless of dictionary sizes or the order of password guesses within.

Network-based detection of SSH bruteforce attacks. Javed *et al.* [13] propose a methodology to detect distributed and potentially stealthy SSH guessing activity. Hofstede *et al.* [11] propose a large-scale system to detect SSH compromises (*i.e.,* successful guessing attempts) using NetFlow data. Satoh *et al.* [15] similarly use network flow data to identify the source's authentication type (interactive, key-based, *etc.*) in an effort to remove non-automated logins from their set. We achieve the same server behaviour by configuring the server to exclusively accept password authentication (see Section 3). Sperotto *et al.* [16] built Hidden Markov Models from an SSH bruteforce attack recorded at their university, and used this model to generate synthetic SSH network traces. The authors found that the generated traces can be used to simulate a ground truth for the evaluation of defense systems. We believe our dataset to be a ground truth of SSH dictionary attacks due to its construction.

3 Data Collection Methodology

All logging was performed on Ubuntu 12.04 or 14.04 virtual machines (VM) running on a popular cloud server platform, which assigned public and persistent IPv4 addresses to each VM at creation. IPv6 connectivity was not enabled. No publicly facing network services were enabled on the VMs other than two SSH daemons (one for data collection and one for administration), and software firewalls were configured to allow all inbound and outbound traffic.

The pre-installed SSH daemon served as the management interface for the VMs. To prevent guessing attempts against this interface, we moved the daemon to a non-standard TCP port and we disabled password-based authentication (requiring key-based logins). In Appendix A, we describe a small-scale experiment to determine if attackers target SSH servers on non-standard ports.

We installed a second instance of the OpenSSH server (version 6.5p1) on all VMs to log guessing attempts. To additionally record the password guesses, we modified the OpenSSH server by inserting a log function in the password authentication module (`auth-passwd.c`). This modified server was configured to start at boot and listen for incoming authentication requests on the default TCP port 22. The server configuration was also changed to allow up to 50 (from the default 10) concurrent unauthenticated connections to the daemon. Only password authentication was allowed on the modified daemon. All login attempts were logged to the `syslog` logging facility.

Preventing accidental human logins. Our paper focuses on the analysis of automated login attempts. Thus, we wish to minimize the probability of recording login information from non-automated sources. To prevent accidental non-automated logins (by either curious users or by accidentally typing the wrong IP address or other misconfiguration), we displayed the following SSH banner to all incoming authentication requests prior to password entry:

```
*WARNING*
This OpenSSH server has been modified to STORE USERNAMES AND PASSWORDS.
This server does not have any valid user accounts, so no attempted logins
will succeed. The sole purpose of this server is to collect (for research
purposes) login information used in automated SSH brute-force attacks. If
you are human, you should not attempt to log in to this server.
```

Data collection. We began collecting login attempts on a single VM running in Ottawa (OTT), Canada on Mar 1, 2014. On Jan 4, 2015, we instrumented and enabled five additional VMs, each in a separate geographical region: San Francisco (SFO), New York (NY), London (LON), Amsterdam (AMS), and Singapore (SGP). These five VMs collected data for only 66 days, but broadened our geographical scope allowing us to make location-specific observations. Since these additional VMs were on distinct networks, they allowed us to detect sources which target multiple destination IPs. All data collection was halted on Mar 8, 2015, giving a total of 373 days of aggregate data collection. Throughout the paper, we report on the aggregate set of login attempts performed on all VMs, clarifying VM-specific observations where necessary.

4 Characteristics of Attacking Systems

Table 1 summarizes the collected login attempts. Collectively, the VMs received ∼17M attempts from ∼6.2K IP addresses located in 112 countries.[3] A total of ∼27K distinct usernames and ∼1.4M distinct password guesses were observed.

A single /24 subnet (`103.41.124.0`/24) based in Hong Kong (HK) was observed guessing credentials aggressively on the OTT VM beginning Nov 15, 2014, and later on all five VMs of the short-term study within two days of the VMs

[3] We used the `http://ipinfo.io` IP geolocation database [12] to obtain geographic location and Autonomous System (AS) information of these IP addresses.

Table 1. A summary of the collected data.

VM	Attempts	AS	IP addresses				Regions		Distinct	
			/8	/16	/24	/32	Country	City	Usernames	Passwords
OTT	9,925,706	934	164	1,735	2,934	4,233	100	580	25,551	1,209,851
LON	2,609,164	357	125	573	826	1,134	66	197	2,742	618,869
NY	1,661,628	245	123	426	508	707	61	137	1,637	464,456
SFO	1,491,949	250	125	444	548	760	58	152	2,160	492,340
AMS	1,107,247	353	126	547	803	1,065	64	199	2,505	314,384
SGP	421,982	333	132	525	834	1,114	62	185	4,520	135,015
*	17,217,676	1,235	171	2,338	4,187	6,297	112	744	27,855	1,449,146

$*$ = Combined statistics.

coming online (Jan 4, 2015). This subnet alone was responsible for 8,757,604 (or ∼51%) of all recorded guessing attempts, from 65 observed host addresses.

Because the OTT VM collected attempts for the longest time period, it received the largest number of login attempts. The corresponding source IP addresses belong to 934 ASes and 100 countries.

For the short-term study, the five VMs experienced different login attempt rates. The LON VM received the highest rate, logging ∼2.6M attempts in 10 weeks. The SGP VM received less than one-sixth of those attempts, albeit from almost the same number of IP addresses (1,114 vs. 1,134). The reason for this discrepancy is unclear; the aggressive HK subnet discovered both VMs on the same day, Jan 5 (*i.e.,* the day we started observing login attempts from that subnet). Of the 65 host addresses belonging to the HK subnet, 64 made attempts on the LON VM, and 62 similarly on SGP. The two hosts not showing interest in the SGP VM were among the least aggressive in the subnet, responsible only for 0.3% and 0.9% of attempts made by the entire subnet.

Attackers may seek to compromise systems in specific regions, and the discrepancy between attacks seen on LON and SGP VMs suggests that attackers may actually be distributing their resources unevenly, perhaps for higher benefits (or smaller risk of being shut down).

4.1 Number of IPs per /24

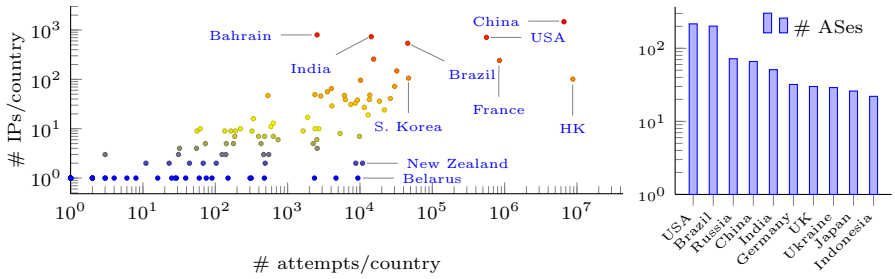
Recall from Table 1 that we logged ∼4.2K distinct /24 subnets. About 84% (3,528) of those had only one host IP address recorded in our logs. On the other hand, we observed 76 hosts in `116.10.191.0`/24, making it the subnet with the largest number of *malicious*[4] hosts. This subnet was responsible for 7.5% (1,298,912) of all guessing attempts. We observed 626 subnets with between 3 and 10 malicious hosts, and 33 subnets with 11 or more malicious hosts.

[4] Although a logged IP address may not necessarily belong to a user with deliberate malicious intent (*e.g.,* it could be remotely exploited by a malicious third party) we refer to the IP as such for simplicity.

4.2 Countries with the most aggressive sources

Figure 1a shows the relationship between the number of IP addresses observed per country, and the number of attempts thereof. As expected, the chart reveals a positive correlation between these factors. Together, Hong Kong and China (two right most points) constitute ∼90% of all guessing attempts observed.

China also comes first in the list of countries by number of observed IP address (highest point), followed by Bahrain despite the country's relatively small IP address allocation—∼450K IPs [3] (see Section 4.3 below). Additionally, analyzing the collective number of attempts per /24 subnets, 7 of the 10 most aggressive (by number of attempts) subnets are Chinese; the remaining three are from Hong Kong, USA and France.



(a) (b)

Fig. 1. (a) The number of attempts, per country, with respect to IP addresses. For example, the point (2552, 796) indicates that a country (which is Bahrain) originated 2,552 attempts from 796 IP addresses. (b) The number of ASes per top 10 countries.

The distribution in the number of ASes observed per country is shown in Fig. 1b. UK, Ukraine, Japan and Indonesia show up in the top 10 countries with highest number of ASes, while absent from the list of top 10 by number of IP addresses. This suggests that malicious machines in those countries were more scattered amongst varying ASes, rather than concentrated in a few. Countries in which malicious hosts reside in a small number of ASes can more easily coordinate patching efforts. For example, although Bahrain comes second in the list of countries by number of IP addresses (796 addresses), all such addresses belonged to three ASes (six /8 subnets). On the other hand, we saw 538 Brazilian addresses belonging to 201 ASes, requiring involvement from more ISPs to fix.

4.3 IP addresses as a ratio of the total allocation per country

We analyze each country's proportion of malicious IP addresses relative to the total addresses allocated to the country. For example, Panama is allocated a total

of 1,909,212 IP addresses [3] of which we observed only 6 in our logs, giving a ratio of 0.31 address per 100,000.

Bahrain tops the list of countries with the largest proportion of malicious addresses, at ∼177 IP per 100K allocated. Figure 2 shows a geo-chart with the number of IP addresses observed per 100K allocated for each country. Note that Bahrain is not represented in this chart, as it overshadows the rest of the data. Additionally, we exclude from the chart data of 10 countries allocated less than 100K addresses in total.

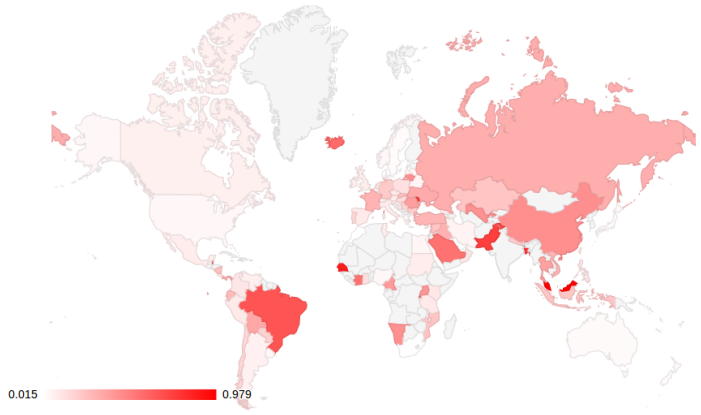


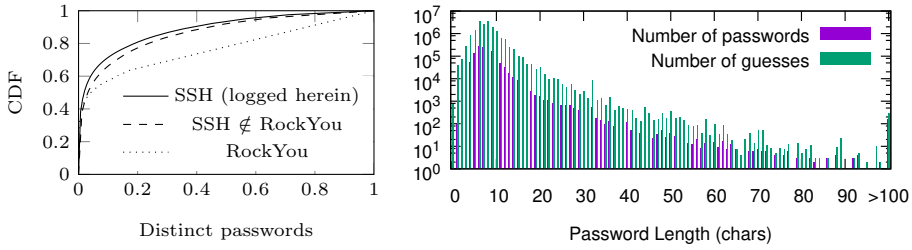
Fig. 2. Ratio of malicious addresses to 100K allocated. *Created using the Google Chart libraries, according to terms described in the Creative Commons 3.0 License.*

5 Password Analysis

Figure 3a shows the Cumulative Distribution Function (CDF) of all observed password guesses, and the CDF of those not appearing in the RockYou (`www.rockyou.com`) dataset. Both distributions are highly skewed. Under the hypothesis that these distributions reflect previously successful SSH guesses, their skewness suggests the potential existence of accounts with poorly- (or carelessly-) chosen passwords. Passwords of such accounts would require less guess work [6] to crack than those in the RockYou dataset—CDF also plotted in Fig. 3a.

5.1 Password Length

Figure 3b plots the distribution of password lengths next to the number of passwords seen for each lengths. The distributions resemble each other, with more passwords of a given length being reflected as more login attempts for passwords of that length. Passwords varied in length from 0 (no password received) to 270. We observed several hosts in a single network attempting passwords that appear to be Unix shadow file entries (118 characters long), indicating that attackers may not always inspect the correctness of their dictionary entries.



(a) (b)

Fig. 3. (a) CDF of distinct password guesses; a point (x, y) means the proportion x of distinct passwords accounted for the proportion y of all observed. (b) Password length distribution. For each password length, the bar on the left shows the number of unique passwords of that length in our set. The bar on the right shows the number of login attempts seen with a password of that length.

5.2 Password composition compared to known dictionaries

In Table 2, we show the password composition of the 1.4M passwords in our set. Over 22% of password guesses observed were constructed as one or more letters followed by one or more numbers (*e.g.,* `test123` or `admin6`). More than half of all passwords contained only lowercase characters. With the exception of only lowercase passwords,[5] the password composition of passwords in our set resembles that of the RockYou dataset, where the vast majority of passwords include only alphanumeric characters (*i.e.,* no special characters), and a large set of passwords was composed by lowercase characters followed by numbers.

Table 2. Password composition: observed per experiments herein versus RockYou.

Password Type	SSH•		RockYou dataset	
	Count	%	Count	%
Only lowercase	771,101	53.2	3,783,103	26.4
Only uppercase	5,883	0.406	234,913	1.64
Only numbers	140,074	9.67	2,348,128	16.4
Letters then numbers	325,547	22.5	5,340,129	37.2
Have no special characters	1,372,858	94.7	13,395,174	93.4
Have special characters	76,288	5.26	949,217	6.62
Total	1,449,146	100	14,344,391	100

•per experiments herein

[5] Attackers may guess only lowercase passwords more frequently in expectation that system administrators pick these types of passwords more often.

We noticed more than 3.2K distinct passwords in the form of URLs (*e.g.,* ending in `.com`, or `.edu`), which were collectively tried ∼140K times. The most common of those ending in `.com` and `.net` respectively were `123.com` (attempted 7,014 times) and `nowtop.net` (971 times).

Many of the attempted passwords have not been seen in the commonly studied leaked dictionaries (*e.g.,* RockYou and phpbb). For example, out of the ∼1.4M distinct passwords, 876,012 (60%) passwords were not present in the RockYou dataset. Those passwords were collectively attempted 7,136,356 times out of the ∼17M total attempts (or ∼41%). Examples of those passwords include `http://managers.at` and `CactiEZ`. See Appendix B for more details on the top ten passwords and their counts.

Note that the number of guesses of many passwords unique to our dataset was relatively high. For example, the password `idcidc` (likely short for "I don't care") was attempted 5,272 times (0.03% of all ∼17M attempts) from 533 IP addresses (of all 6,297) belonging to 27 ASes and 11 different countries. This information increases the likelihood of those ∼5K attempts originating from different attackers using shared lists. Note also that those attempts were (almost) evenly distributed over the course of 13 months. Table 3 shows five examples of passwords that did not appear in any of the following leaks: RockYou, GawkerPasswords, myspace, phpbb, SonyPasswords, and YahooVoicePasswords.

Table 3. Examples of passwords observed in our experiments but not appearing in common leaked dictionaries (see inline for details). The column to the right shows the number of days the password appeared in our logs.

Password	# of attempts	# of sources			# of days (months)
		IPs	ASes	Countries	
`o12nu27`	5,425	512	45	17	369 (12)
`idcidc`	5,272	533	27	11	368 (12)
`rkqldk`	4,457	381	9	5	349 (11.4)
`\\001`	3,837	26	3	1	341 (11.1)
`zxm10`	3,168	465	32	13	26 (0.85)

5.3 Dictionary Sharing and Splitting Among Sources

Owens and Matthews identified password sharing among bruteforcers [14]. Their work used a narrow definition of sharing that required distinct attackers to attempt the same username-password pairs in the same order to qualify as sharing. While we also observed same-order guesses as described by Owens and Matthews, we also noticed random guessing from shared dictionaries (described below).

One possible way to detect dictionary sharing is to identify overlap between dictionaries. Figure 4 shows a heatmap of the percentage overlap between pairs of dictionaries in our set. For this graph, dictionaries are built as the aggregate set of distinct passwords used by any source within a particular /24 network. We take the top 1000 largest (by password count) per-/24 dictionaries, and

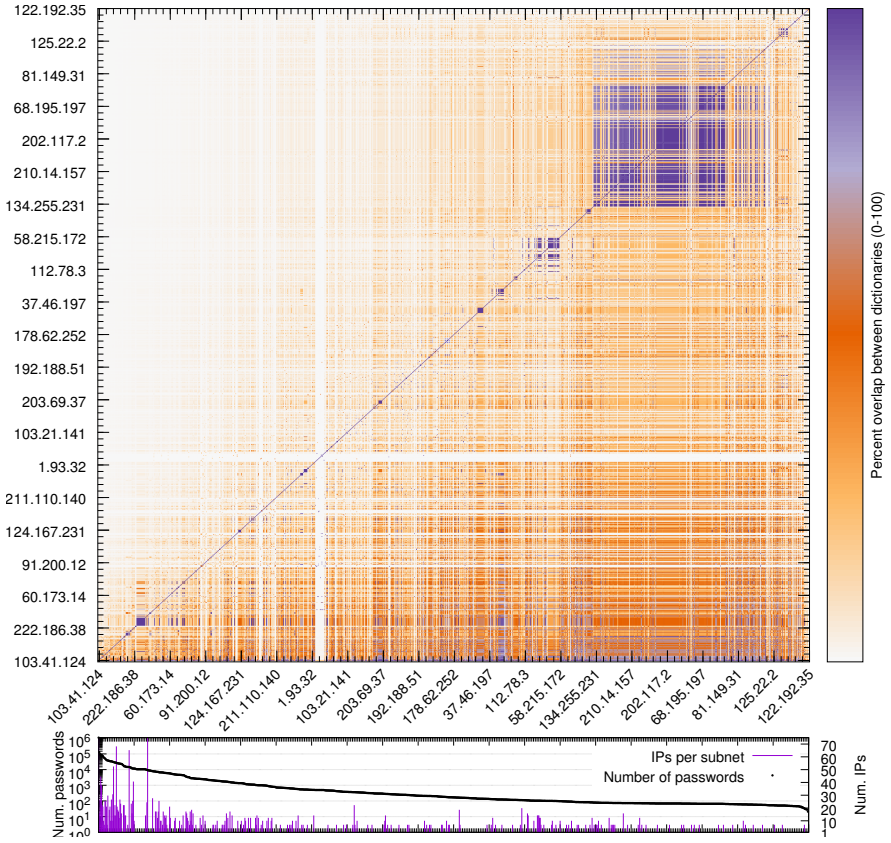


Fig. 4. Overlap between per-subnet dictionaries. See inline. Best viewed in colour.

sort them from largest to smallest on the x and y axes.[6] For reference, the largest subnet dictionary (*i.e.,* passwords used by `103.41.124.0`/24) contains ∼1.09M distinct passwords. The 1000^{th} largest dictionary (seen used by sources in the `122.192.35` subnet) contains 21 distinct passwords. As the colour gradient shows, white indicates no overlap (*i.e.,* no common entries) between a dictionary on the x-axis and that on the y-axis, while purple indicates 100% overlap.

Coordinated dictionary split. The horizontal and vertical white bands near the `1.93.32` subnet indicate that passwords guessed were mostly unique compared to all other passwords. Upon further investigation, we identified 13 sources each guessing from a set of 500 lowercase alphabetical passwords. 11 of the 13 sources showed no overlap between their 500 passwords and other sets in the group. This lack of overlap and equally sized dictionaries shows with high prob-

[6] In Fig. 8, we show a similar heatmap for overlap between the largest 1000 per-IP dictionaries (*i.e.,* passwords seen used by each IP).

ability that a single attacker split a large dictionary into smaller sets of 500 entries, distributing these small sets to bots under his control.

Dictionary sharing. Purple squares near the $x = y$ line highlight (possibly non-contiguous) /24 subnets using the same set of passwords. We identified many such instances, with the most notorious being the large purple region near the top right of the heatmap. Here, we identified 404 distinct sources from 54 countries all guessing from a dictionary of around 90 alphanumeric strings. These guesses also came from a set of around 90 distinct username-password combinations, indicating coordination among sources.

Another example is the largest purple region near the bottom left. Here 12 distinct networks were seen using the same dictionary of 10,546 passwords. Sources in this set also each performed the same number of guesses overall (10,852), and each guessed each password in their dictionary the same number of times (*e.g.,* `123456` 23 times, `password` 14 times, `123` 11 times, *etc*). While this behaviour may hint at coordination or centralized control, it could also reflect distinct attackers using software configured the same way (*e.g.,* by not modifying the default configuration or password list).

The predominant purple and orange shading suggests that dictionary sharing or distribution takes place among attackers.[7] Small sets of passwords, likely bundled with bruteforcing tools or found online, are used by many attackers. However, as we discuss in Section 5.2, the overlap of these passwords with common web-based leaked dictionaries (*e.g.,* RockYou) is not significant. One possible explanation is that attackers may be running SSH servers on their botnets to log guessing attempts made by other attackers. Such a strategy would allow attackers to effortlessly and quickly expand their dictionaries.

5.4 Reattempting username-password combination

Recall that none of our VMs had any valid accounts, so all login attempts failed for all guesses. Despite this, we noticed that some username-password guesses were made by the same source on the same destination VM more than once. For example, the username `nano` and password `bl3rand,.` were tried four times against the LON VM from a machine in Baotou, China between Jan and Feb 2015. About one-third of all source IP addresses (2,019 of 6,297) manifested that behavior (77 addresses were particularly responsible for ∼50% of all repeated attempts). Those addresses belong to 1,189 of all 4,187 /24 IP addresses. In total, such *repeated attempts* account for 25% (or ∼4.3M of ∼17M) of all guessing attempts, with time between pairs of repeated attempts ranging from less than one second to ∼11 months. Attempts repeated more than twice also occurred, as evidenced by an address that guessed username `root` and password `\\001` 1,220 times against the OTT VM in only 19 minutes.

[7] We believe it is unlikely that all such highly overlapping dictionaries belong to a single attacker since many of their bruteforcing behaviors were different, *e.g.,* timing dynamics, rate of attempts, *etc.* Even dictionary pairs with extreme overlap had different guessing order.

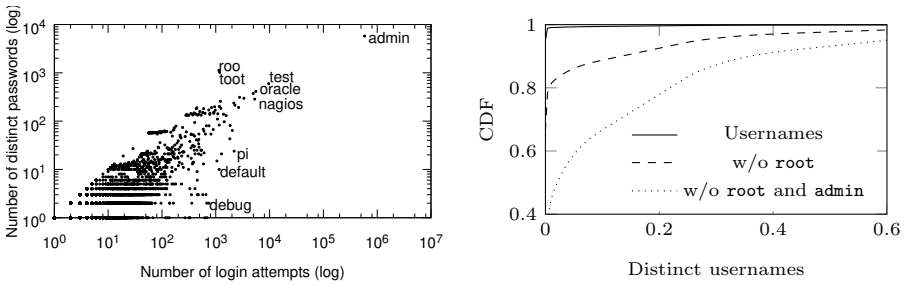


Fig. 5. (a) Number of login attempts and distinct passwords per username (excluding root). (b) A CDF of usernames; a point (x, y) means the proportion x of distinct usernames accounted for the proportion y of all ∼17M observed.

One possible explanation for repeating attempts is that a machine could be controlled by multiple attackers simultaneously. This may happen when attackers, upon compromising a new system, fail to patch the vulnerability they exploited.[8] Attackers may also be repeating attempts to account for password aging policies [7], expecting previously-failed attempts to succeed due to a password change. Although both hypotheses above could explain attempts repeated within months or weeks, they fail to explain repeated attempts within shorter intervals (*e.g.,* minutes or seconds). We found that the time between 59,514 (1.4%) pairs of repeated attempts was one second or less, and between 251,305 (5.7%) was one minute or less, hinting at attack software misconfiguration.

Another explanation is that attackers may not have enough resources to store which username-password pairs were attempted on which SSH server. Thus, they operate a herd of bots arbitrarily, where any bot guesses a combination (from the same fixed list of candidate guesses) on any server. Machines showing extremely large average time (*i.e.,* multiple months) between repeated attempts could have gone through three phases: compromised, patched, then re-compromised (either by the same attacker, or another one with overlapping dictionary). The source may also have completed a full cycle through a list of targets and passwords, arriving at the starting point once again.

6 Username Analysis

Root and Admin. The most commonly attempted usernames were `root` (∼95% of all attempts) and `admin` (∼3%). Interestingly, only 63% of source IP addresses guessed passwords for `root` or `admin`, meaning that the remaining 37% specifically targeted non-administrative accounts, though at a much slower rate. Figure 5a plots the number of login attempts observed for a specific username

[8] Attackers may be unwilling to change the password or patch the vulnerability used to compromise to avoid detection by the legitimate user of that system.

on the x-axis, and the number of distinct passwords for that username on the y-axis. Based on this behaviour, defensive strategies that block the IP addresses attempting `root` will fail to block a large portion of sources, but will succeed in blocking the vast majority of attempts (until attack behaviour changes).

Figure 5b shows a CDF of username distribution. We note that the skew in the CDF even when excluding `root` and `admin` indicates that attackers target a narrow set of usernames. We discuss the implications of this in Section 8.

Other Usernames. The remaining set of usernames were tried comparatively fewer times with fewer distinct passwords. Out of 27,855 total usernames, 7,612 (27.33%) saw only a single login attempt with a single password; 1,369 usernames (4.91%) saw 2 attempts with a single password; and 7,340 (26.35%) usernames recorded 2 attempts with 2 distinct passwords. This trend does not continue, since only 119 usernames received 3 attempts with 3 distinct passwords.

In 2008, Owens and Matthews [14] reported that attackers frequently use the username as the password (*e.g.,* user `david`, password `david`) in SSH guessing attacks. We recorded 50% of non-root and non-admin guesses attempting usernames as the password, confirming that attackers continue using this strategy.

7 Timing Analysis

Table 4 summarizes the rates of attempts received on each VM per day, and per hour. The last row shows combined statistics—minimum of the six minimums, median of medians, and maximum of maximums respectively.

Table 4. Summary of the rate of attempts.

VM	Rate of attempts					
	Per day			Per hour		
	Min	Median	Max	Min	Median	Max
OTT	349	17,805	273,120	0	754	85,770
LON	836	35,445	116,935	0	1,204	15,375
NY	354	15,036	192,351	0	53	69,770
SFO	180	15,993	119,981	0	47	60,026
AMS	281	9,655	91,901	0	101	7,742
SGP	516	3,755	38,290	0	55	10,451
*	180	15,515	273,120	0	55	85,770

On Jun 14, 2014, between 3pm and 4pm EST, the OTT VM received more than 85K attempts, averaging ∼24 per second. Collectively, those originated from 55 IP addresses in 13 /24s; however, about half of these attempts originated from a single /16. For the short term study (the other 5 VMs), we noticed that the two European-based VMs had relatively less skewed distribution of attempts over time. This can be inferred from the *Median Per Hour* (relatively large medians) and *Max Per Hour* columns (relatively small), suggesting the possible presence

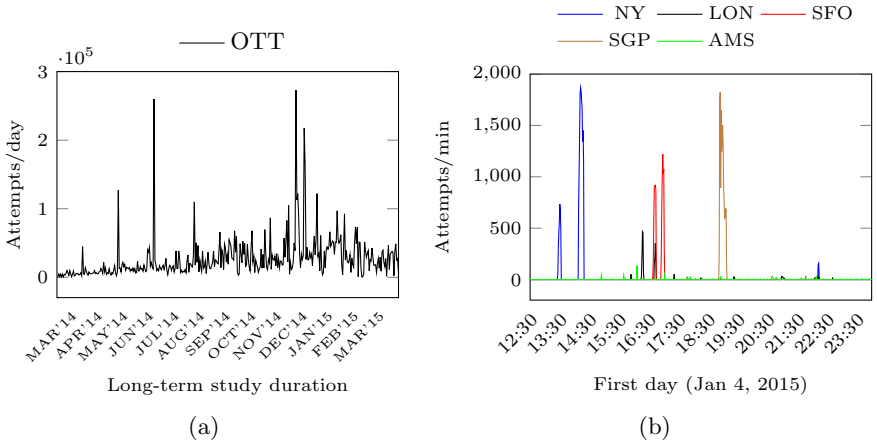


Fig. 6. (a) Daily attempts during the long-term study. (b) Attempts per minute during the first day of making the SSH servers public; all five VMs were started at the same time—the first point on the x-axis is 12:30pm UTC. Best viewed in colour.

of IDS-like systems in common backbone European networks that enforce some form of rate-limiting.

At no single day did any of the six VMs receive zero attempts. The median attempts per day varied from 3.7K (SGP) to 17.8K (OTT). Statistical confidence from this data found a 90% chance the median number of (unthrottled) login attempts received per day by any SSH server is between 6.7K and 24.3K. The maximum number of attempts per day varied considerably across VMs. On Nov 16, 2014, the OTT VM experienced 273K attempts—the largest rate per day of all VMs. Similarly, the NY VM had the highest rate per day in the short term study, scoring 192K attempts on Jan 24, 2015.

Figure 6a shows the number of attempts received by the OTT VM in each day of the long term study (373 days). The spikes on May 6 and Jun 14 (126,917 and 260,046 attempts, respectively) occurred when an attacker first discovers the SSH server. A single source IP address was responsible for 94% of attempts on May 6, but that machine did not make further attempts on any VM after May 6. Likewise, a single IP was responsible for 98% of the attempts made on Nov 16; and `62.210.128.0`/18 was responsible for 79% of those on Nov 25.

Figure 6b shows the number of attempts in the first 12 hours (beginning at 12:30pm UTC) after starting the SSH servers of the five short-term study VMs. The NY VM was the first to receive guessing attempts only 51 minutes after it was started, versus 4 hours and 33 minutes for the SGP VM which was the latest. The number of sources attempting guesses on the first day varied as well, from 4 (SFO) to 15 (LON). Only a single source (located in Ukraine) made guesses on all five VMs on the first day within a two hour window (about four hours after starting the SSH service). This machine seemed not aggressive, making only one guess on each VM on that day. In fact, we found that this machine

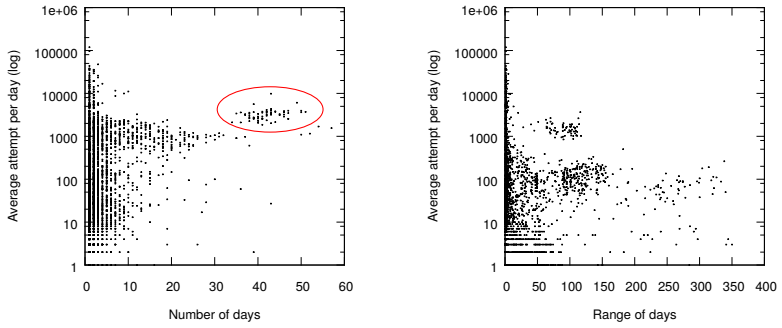


Fig. 7. A point (x, y) represents a machine (IP address) that made an average of y attempts per day and appeared in our logs (a) for x possibly non-consecutive days (b) for a range of x days, *i.e.,* difference between the first and the last day.

guesses a username-password combination on the five VMs then proceeds to the next combination; attempts on the VMs were in the same order (which is AMS, LON, SFO, NY, SGP), and the time between VMs was constant (*e.g.,* 7 minutes between AMS and LON, 25 between LON and SFO, *etc.*). It is thus likely that this machine performs reconnaissance concurrently to guessing passwords.

Next, we analyze the density of guessing attempts per source IP address over time. Figure 7 shows scatter plots where each point represents a source IP address. In Fig. 7a, the average number of attempts per day made by an IP address is plotted against the number of days the address appeared in our logs. Most IP addresses showed up for ten days or less. The highest point is a source machine that made 119,267 guessing attempts within a two hour period. Similar behaviour was noticed from five other hosts in the same /24, and there was a high degree of overlap between password guesses.

The point $(x, y) = (39, 2)$ is a source machine that collectively made 79 attempts in 39 non-consecutive days, averaging $\sim$2 attempts per day. All guesses were attempted on the SFO VM. The concentration of points between 35 and 50 (circled in Fig. 7a) belonged to the aggressive HK subnet `103.41.124.0`/24 (see Section 4), highlighting the persistence of this subnet.

Figure 7b similarly shows the average number of attempts but over a duration of consecutive days, from the first to the last day an IP address has appeared in the logs. The concentration of points is now closer to the bottom of the y-axis, compared to the previous case, since the average is taken over a longer period of time. Although the maximum number of (possibly non-consecutive) days an IP address have appeared in the logs did not exceed 60 (*i.e.,* from Fig. 7a), a range of up to 350 days was noticed. This could be due to an attacker not fully utilizing its computing resources, or perhaps machines being fixed then re-exploited. In conclusion, if IP-blacklisting was employed, system administrators should not

haste into whitelisting addresses shortly after guessing attempts discontinue, since attacking machines may turn dormant for days before resuming activity.

8 Recommendations

Our data analysis allows us to propose concrete suggestions to administrators for reducing the probability of successful guesses of SSH credentials. We note that these recommendations do not preclude additional security mechanisms such as IDS, public key authentication, *etc.*

Network blocking. We observed that certain IP addresses, often working closely with other IP addresses in the same network block can exhibit aggressive behaviour when guessing credentials. Opportunistic detection and blocking of these sources could reduce the number of guesses by a significant fraction. For this purpose, tools such as DenyHosts and Fail2Ban (see Section 2), in addition to rate-limiting login attempts can prevent login attempts from polluting the system logs. We note, however, that any mechanism that blocks an IP address should periodically (*e.g.,* after three to six months) remove the block, as attacking systems tend to persist their attack for short time periods.

Username whitelisting/blacklisting. The overwhelming majority of login attempts target the `root` account. Thus, disabling remote logins for `root` could be an effective method of preventing successful guesses. A more proactive approach involves explicitly whitelisting usernames that are allowed to login via SSH, thus reducing the attack surface. Moreover, from our dataset, blacklisting the top 20% of observed usernames could prevent 80% of successful attacks, which aligns with suggestions by Florencio *et al.* [9]. This strategy may impose less cognitive load on the user (as compared to blacklisting passwords), since usernames must not be secret.

Proactive password checks. Proactive password checking is believed to improve the security of a system [5]. Florencio *et al.* further suggest that blacklists of as many as 10^6 passwords seems to be an effective operating point to prevent online guessing without causing substantial user inconvenience [10]. In addition to identifying frequently targeted accounts and passwords, we suggest proactive checking to disallow passwords equal to usernames.

9 Conclusion

We presented a methodology for collecting automated SSH login attempts, and analyzed the largest set of attack logs to date. By analyzing plaintext password guesses, we were able to identify broad dictionary sharing across all attackers, and note that many of the passwords used during SSH bruteforce campaigns are independent from those of common web-based password leaks. We also observed highly coordinated and highly distributed guessing attacks both within single subnets and distributed across multiple regions. Our analysis shows that attacks target all publicly accessible systems, but some systems may be of more interest to attackers due to their physical location or connectivity.

Acknowledgements. We thank Hala Assal, Elizabeth Stobert, Mohamed Aslan, Raphael Reischuk, and the anonymous referees for insightful comments which have improved this paper.

References

1. Internet Storm Center - SSH Scanning Activity. `https://isc.sans.org/ssh.html` Accessed September 13, 2015.
2. Nagios. `https://www.nagios.org` Accessed Sep. 13, 2015.
3. Country IP Blocks - Allocation of IP addresses by Country. `https://www.countryipblocks.net/allocation-of-ip-addresses-by-country.php`, Accessed September 13, 2015, 2015.
4. M. Alsaleh, Mohammad Mannan, and P. C. van Oorschot. Revisiting Defenses against Large-Scale Online Password Guessing Attacks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 9(1):128–141, 2012.
5. F. Bergadano, B. Crispo, and G. Ruffo. High dictionary compression for proactive password checking. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):3–25, 1998.
6. J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy*, 2012.
7. S. Chiasson and P. C. van Oorschot. Quantifying the security advantage of password expiration policies. *Designs, Codes and Cryptography*, 77(2-3):401–408, Springer 2015.
8. Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *USENIX Security*, Aug. 2013.
9. D. Florêncio, C. Herley, and B. Coskun. Do Strong Web Passwords Accomplish Anything? In *USENIX HotSec*, pages 10:1–10:6, 2007.
10. D. Florêncio, C. Herley, and P. C. van Oorschot. An Administrator's Guide to Internet Password Research. In *USENIX LISA*, pages 35–52, 2014.
11. R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras. SSH Compromise Detection using NetFlow/IPFIX. *ACM SIGCOMM CCR*, 44(5):20–26, 2014.
12. IPinfo. IP Address Details - ipinfo.io. `http://ipinfo.io` Accessed Sep. 13, 2015.
13. M. Javed and V. Paxson. Detecting Stealthy, Distributed SSH Brute-Forcing. In *ACM CCS*, 2013.
14. J. Owens and J. Matthews. A Study of Passwords and Methods Used in Brute-Force SSH Attacks. In *USENIX LEET*, 2008.
15. A. Satoh, Y. Nakamura, and T. Ikenaga. Identifying User Authentication Methods on Connections for SSH Dictionary Attack Detection. In *IEEE Annual Computer Software and Applications Conference Workshops (COMPSACW)*, 2013.
16. A. Sperotto, R. Sadre, P.-T. d. Boer, and A. Pras. Hidden Markov Model Modeling of SSH Brute-Force Attacks. In *Integrated Management of Systems, Services, Processes and People in IT*, pages 164–176. Springer Berlin Heidelberg, 2009.
17. J. L. Thames, R. Abler, and D. Keeling. A Distributed Active Response Architecture for Preventing SSH Dictionary Attacks. In *IEEE Southeastcon*, pages 84–89, 2008.
18. T. Ylonen. SSH - Secure Login Connections over the Internet. In *USENIX Security*, 1996.

Appendix A SSH servers on non-standard ports

A commonly suggested strategy to reduce the chances of guessing attacks succeeding is to change the default listening port (TCP 22) to a non-standard port. Using a port other than the default requires client-side changes, so easy to remember ports (such as 2222 or 2022) are sometimes suggested by administrators and users [14]. To investigate the validity of this suggestion, we created a network daemon that accepts incoming connections on all TCP ports (except port 22). When a new connection is received, the daemon looks for an initial SSH handshake and immediately closes the connection. We recorded the source IP, port number, and timestamp of all incoming SSH protocol connections. We ran this daemon on a separate VM over a 12 day period, and recorded 30,169 incoming connections to 522 distinct ports, out of which 77 were SSH connection. These incoming connections originated from 9 distinct source IP addresses. The full list of ports which received SSH connections are listed in Table 5. Notice that all incoming SSH connections have the digit 2 in the port number, many with two or more occurrences. Also of interest is that none of the 9 sources was seen making connections in the long-term or short-term study, hinting at the possibility that attackers dedicate some bots to non-standard scans/attacks. Despite the short duration of this study above (port numbers), the results show that attackers do not simply ignore non-standard ports. With availability of modern (and extremely fast) network scanning tools [8], attackers can quickly scan all open ports on sets of systems typically identified by an IPv4 address. Thus, moving an SSH daemon to a port other than 22 may not provide a comprehensive solution.

Table 5. List of non-standard ports on which incoming SSH connections were received.

Port	Count	Port	Count	Port	Count	Port	Count	Port	Count	Port	Count
2222	29	55022	3	22002	3	8822	1	2101	1	2233	1
1202	7	2312	3	21002	3	8022	1	2011	1	10022	1
2111	5	2266	3	7022	2	22203	1	2001	1		
22088	4	221	3	22201	2	2212	1	122	1		

Appendix B Top usernames and passwords (non-root)

Table 6 shows the top ten usernames and passwords, including the top ten passwords that did not appear in the RockYou dataset (RockYou's top ten are also included for reference). From the second column, we notice that the top ten passwords in our set are likely relevant to conventions among system administrators, due to the nature of SSH. The most common attempted password, `toor`, is the mirror of *root*. Nagios [2] is an open-source monitoring software.

For reference, in Table 7 we list the most frequent username-password combinations in our set. Note that most of these combinations follow the "username as password" strategy described in Section 6.

Table 6. Top ten passwords

SSH•			SSH• Not in RockYou			RockYou		
Password	Count	%	Password	Count	%	Password	Count	%
admin	20657	0.120	toor	7204	0.101	123456	290729	0.892
123456	17592	0.102	root@123	6771	0.095	12345	79076	0.243
password	14981	0.087	r00t	6593	0.092	123456789	76789	0.236
root	12122	0.070	data	6275	0.088	password	59462	0.182
1234	11515	0.067	root00	6269	0.088	iloveyou	49952	0.153
test	10091	0.059	p@ssw0rd1	5947	0.083	princess	33291	0.102
12345	9963	0.058	nagios	5908	0.083	1234567	21725	0.067
123	9371	0.054	admin@123	5806	0.081	rockyou	20901	0.064
abc123	9113	0.053	root123!@#	5581	0.078	12345678	20553	0.063
12345678	8747	0.051	shisp.com	5543	0.078	abc123	16648	0.051

•per experiments herein

Table 7. Top ten username-password combination, where the username is neither `root` nor `admin`. The length of this set is 294,694, of which 69,110 are unique.

Username	Password	Count	%
ubnt	ubnt	2098	0.712
guest	guest	1933	0.656
test	test	1932	0.656
oracle	oracle	1893	0.642
support	support	1516	0.514
info	info	1412	0.479
nagios	nagios	1203	0.408
pi	raspberry	1199	0.407
postgres	postgres	1103	0.374
ftp	ftp	1089	0.370

Appendix C Overlap of per-IP dictionaries

Figure 8 shows percentage overlap between per-IP dictionaries. For this graph, the per-IP dictionary contains the set of all passwords tried by that IP address during the full collection period. We sort the IP addresses numerically on the x and y axes, which allows us to identify large contiguous IP space exhibiting similar behaviour. For example, the subnet 103.41.124.0/24 contains many hosts with similarly sized dictionaries that have very little overlap (white vertical banding) with other dictionaries in the set.

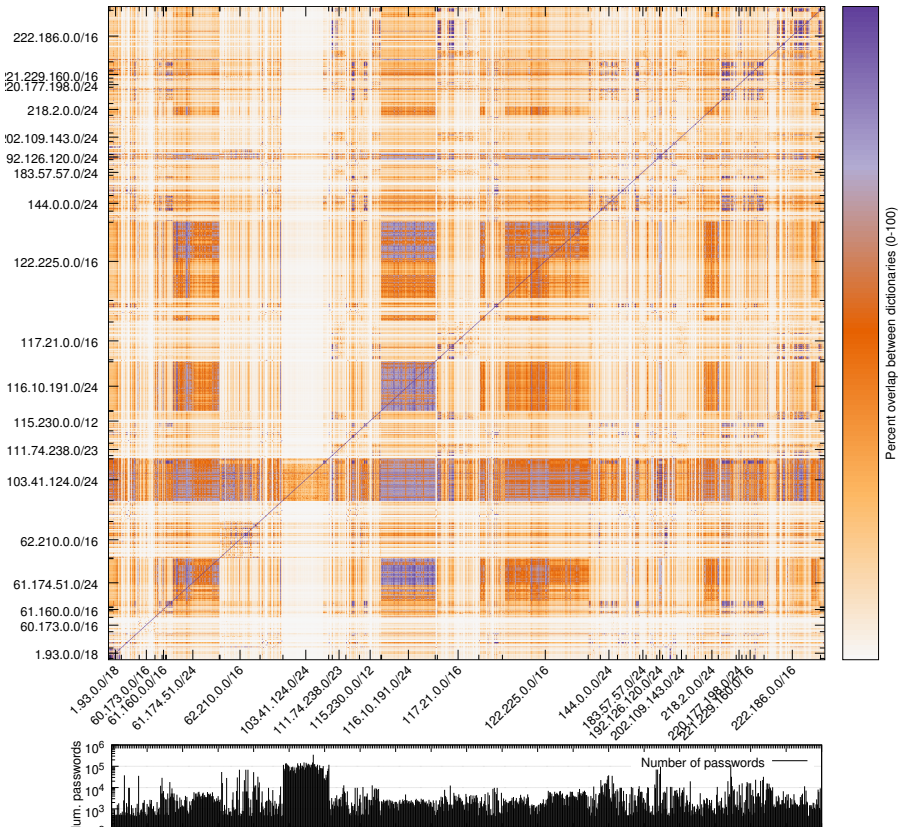


Fig. 8. Overlap between per-IP dictionaries. This figure plots overlap between the 1000 largest per-IP dictionaries. Dictionaries are sorted by IP address. The histogram below the heatmap shows the number of passwords in the per-IP dictionary for the IP immediately above.