A deeper understanding of SSH: Results from Internet-wide scans

Oliver Gasser, Ralph Holz, Georg Carle
Technische Universität München
Faculty of Informatics
Chair for Network Architectures and Services
Email: {gasser,holz,carle}@net.in.tum.de

***Abstract*—Until recently, relatively little was known about the characteristics of the SSH protocol on the Internet, until two larger studies analysed the cryptographic properties of SSH host keys and identified weaknesses in a number of SSH devices. However, there is no succinct comprehensive image yet how the SSH landscape looks like from the point of view of deployment practices, especially with respect to key management. In this paper, we present the results of Internet-wide SSH scans that we carried out over a period of 7 months, which resulted in the largest data set to date. We enriched our data set with large-scale mappings obtained from DNS scans, AS and WHOIS lookups, and a geo-IP database. We analysed the distribution of server and protocol versions, and found that while SSH 2 has displaced SSH 1, the rate of software updates seems to be slow. We analysed the mentioned cryptographic weaknesses and found they have become fewer, but continue to persist one year after the disclosure. Finally, we investigated the reasons for duplicate yet cryptographically strong keys. We found these are used in very different setups at varying degrees of security. Some are indeed dangerous weaknesses, others are the result of a careful and centralised setup. By example of the ten most common keys, we show the circumstances in which they occur and assess the security of each deployment. Finally, we analysed the deployment of ciphers and associated key lengths and found good results in terms of security. As our scans are of a sensitive nature, we also document the ethical considerations that guided us.**

I. Introduction

The Secure Shell (SSH) protocol has been a popular tool since its conception in 1995. Its versatility and omnipresence make it indispensable for many power users, especially system administrators. Two recent publications by Heninger *et al.* [1] and Lenstra *et al.* [2] have addressed the cryptographic security of TLS/SSL and SSH keys and shown that a surprising number of Internet hosts use either cryptographically weak keys or reuse the same (cryptographically strong) keys as other hosts.

It is an interesting and mostly unanswered question where, and due to which causes, the latter phenomenon occurs. One problem lies in the difficulty to decide whether specific key occurrences (especially duplicate keys) are the result of poor practices like default keys, or stem from inadequate network management – or whether they're actually used on purpose and in a safe way. These research questions are naturally linked to others: what are the SSH configurations that we find deployed with respect to server and protocol versions, and what cryptography is used? In order to shed some light on these questions, we carried out three Internet-wide scans over the course of seven months, and enriched our result set with data from other sources like DNS, WHOIS and geo-IP mappings. In this paper, we present the results of our scans which, to our knowledge, constitute the largest available data set to date. In doing so, we hope to provide a better understanding of the ways SSH is deployed and used.

978-1-4799-0913-1/14/$31.00 © 2014 IEEE

Contributions and organisation. We first provide background on SSH to highlight features that are essential for security (Section II). Related work is summarised in Section III. In Section IV, we elaborate on our scanning approach and describe how we enriched our data sets with data obtained from other active scans and databases, specifically DNS scans using a fast custom scanner, AS and WHOIS lookups, as well as a geo-IP database. Section V contains our main contributions. These are threefold. First, we document SSH properties like protocol and server versions in use, key lengths, and cryptographic algorithms, as used across the IPv4 space. Second, we were interested to which degree the weaknesses described by Heninger *et al.* persist a year after the authors' disclosure. Finally, we chose to investigate the issue of duplicate *yet non-weak* keys in more detail. We could link the top 10 occurrences to certain networks. We show that the underlying reasons are very different. Some do indeed point at problematic deployments, albeit to varying degrees: some keys occur across the globe; others are restricted to certain ASes only. However, we also found that some cases of key reuse are in fact the result of secure centralised setups. Our DNS scans also allowed us to investigate the use of the SSHFP resource record. As our scans span 7 months, we are able to document the slow rate of change in the SSH landscape. In Section VI, finally, we address the ethical considerations that guided us in carrying out the scans as well as issues related to publishing our data sets.

II. Background

SSH provides mutual authentication, encryption, and message integrity. It is primarily used as a secure remote shell, to access files (`SCP` and `SFTP`), and to establish tunnels for other application layer protocols. There are two major versions: 1.x and 2.0. Version 1 has serious security issues and should not be used anymore. SSH version 2 is a redesign that removes these flaws. It is defined in RFCs 4250–4256.

An SSH session begins with the client establishing a connection to the server. Both client and server send messages with their SSH identification strings, which include protocol version and server version. Subsequently, they send each other their supported cryptographic algorithms and perform the key exchange. As part of this key exchange, the server sends a public key to the client, the so-called host key. In the optimal case, the client can verify the authenticity of the host key because it has been made known to him out-of-band, e.g., by an administrator. If not, the client may proceed with an unauthenticated session and save the host key in a local database for comparison in future connections. This principle is called *Trust-On-First-Use*. The idea is that an attacker has to compromise the first connection – otherwise, any later key checks will reveal the attacker. The user will then authenticate to the server. Several methods are possible, most prominently password login or a challenge-response protocol based on a public key that the client must have made known to the server before the connection. Password authentication suffers from the usual drawbacks such as dictionary attacks or passwords being lost or forgotten. The public key method is more favourable for clients and has an important advantage: if the client's public key has been securely placed on the server, Man-in-the-Middle attacks are not possible anymore. The reason is that SSH is cleverly designed such that an attacker can either swap the Diffie-Hellman secrets or be able to forward the correct session ID to its victims, but not both.

III. Related Work

In 2001, Provos and Honeyman [3] presented results of an SSH scan of 2 months' duration. Their scanner downloaded the SSH identification string of 2 million public IP addresses plus their local university network. They found that the adoption of SSH 2 had risen to almost 50%, a favourable value at the time. In comparison, our scans exceed theirs by a huge margin.

Scans of other security protocols have been carried out before. Yilek *et al.* [4] carried out scans of TLS/SSL to determine the number of weak public keys that were caused by the now-infamous Debian bug [5]. The authors found that 1.5% of scanned hosts showed weak keys and that the rate of fixing was slow. In 2011, Holz *et al.* published an analysis of X.509 certificates [6] obtained from large-scale scans over 1.5 years. Complementing their data with passive monitoring, they found that, at the time, only 18% of certificates would be accepted by a Mozilla browser without a warning. Results in another publication by Vratonjic *et al.* [7] supported this. In 2007, Schönwälder *et al.* [8] performed large-scale SNMP measurements. Similarly to our research, their goal was to better understand the deployment and usage of a specific protocol.

In 2012, Heninger *et al.* presented results of an Internet-wide sweep of TLS/SSL and SSH servers [1]. Their publication is an excellent example of combining measurement methods and cryptographic insights. They downloaded both X.509 certificates and SSH host keys and recovered 6.2 million unique SSH host keys. The authors' focus was on detecting weak keys. They showed that poor entropy in pseudo-random number generators may cause RSA and DSS keys to be recoverable, and demonstrated this for 0.03% of all RSA keys and more than 1% of DSS keys. DSS keys are described to be more vulnerable due to the importance of good random number generators for every message. Lenstra *et al.* had conducted a similar study a few months earlier [2] with different conclusions. They studied available X.509 data and found more vulnerable RSA than DSS keys, and thus concluded that RSA is weaker than DSS. Heninger *et al.* showed by active measurements that many of these SSH keys could be found on embedded devices, with poor entropy, or were deployed as manufacturer-default keys. However, the authors note that *'the lack of uniquely identifying host information [prevented them] from distinguishing default keys from keys generated with insufficient entropy'*. In many respects, we treat the work of Heninger *et al.* as our point of departure: we focus on the deployment properties of SSH as a result of network management practices.

Concerning large-scale scans in general, Leonard and Loguinov reported on service discovery scans on Internet scale [9]. We use their IP generation principle, too. In 2013, Durumeric *et al.* presented another fast scanner that computes target IP addresses using modular group arithmetic [10].

IV. Scanning process and data sets

We describe our scanning method and the data sets we obtained from our scans.

a) Scanner: Our scanner is built on a producer-consumer model. An *IP generator* produces addresses as input to *port scanner* threads to determine if TCP port 22 is open. The output of these threads is then fed to slightly modified OpenSSH instances. As in [9], our IP generator uses a *Linear Congruential Generator* to create a series of pseudo-random integers which we interpret as IP addresses. This ensures that generated IP addresses are distributed uniformly over the IPv4 space and no address is repeated. We scanned from a single IP address from our own Autonomous System, AS56357. We used BGP information from our border routers to filter out IP addresses that were not in any announced prefix. Each of our OpenSSH instances carries out a full SSH handshake. In this regard, our scanner is different from the one Heninger *et al.* used [1] as our OpenSSH implementation supports multiple types of key exchanges compared to just Diffie-Hellman Group 1 and stores additional data like authentication mechanisms offered by servers.

b) Scanning periods and data sets: We conducted three scans in January, April and July 2013. Each scan took about 5–12 days. The resulting data set is presented in Table I, with the number of hosts with which we could carry out SSH handshakes and the keys we downloaded. Naturally, the difficulty in giving a precise account of hosts lies in the fact that the equation 'distinct IP address = distinct host' does not hold. Hosts may have several interfaces, and interfaces may also be assigned more than one IP address. For simplicity, we

	Jan 2013	Apr 2013	Jul 2013
Scanning period	January 5–12	April 18–29	July 23–27
IPs with SSH	12.0M	10.2M	8.8M
Keys	21.9 M (9.8 M)	18.5M (8.7M)	16.0M (7.4 M)
SSH2...			
RSA	10.9M (4.6M)	9.2M (4.0M)	8.0M (3.6M)
DSS	9.3M (4.3M)	7.8M (3.7M)	6.7M (3.3M)
NIST-P256	734k (450k)	758k (493k)	780k (519k)
NIST-P384	54 (35)	48 (34)	45 (31)
NIST-P521	696 (556)	655 (517)	734 (529)
SSH1-RSA	984k (538k)	721k (417k)	528k (310k)

TABLE I: Data sets gained in our scans. Numbers in brackets indicate distinct keys.

still use the term 'host', but it should be kept in mind that when we give numbers for hosts, these are *upper* bounds.

We found SSH servers in 32,000 ASes in each scan. This is interesting as the current estimate of the Internet's size is around 40,000 ASes, meaning that 1/4 of ASes either block public SSH access or do not use SSH. The number of IPs with SSH servers decreased from scan to scan. One confirmed reason, although likely the less dominating factor, is that our blacklist grew over time. One large hosting provider asked for the blacklisting of 830,000 addresses. The other factor, although plausible, is harder to confirm: systems blacklisted our IP address despite the measures we describe in Section VI.

The number of recorded IP addresses common to both the January and April scans was 6 million; and those common to both April and July was 4 million. About 2. 8 million IP addresses appeared in all 3 scans, i.e., between 24–32% of the IP addresses in the respective scans. AS reassignments were not responsible for this: only 92,000 IP addresses changed their AS during the scanning period. However, we know from [11] and [12] that a large portion of the IPv4 space is dynamically assigned and IP occupancy often shorter than 24 hours. This suggests that many SSH installations are actually found on dynamic IP addresses. This is in line with our findings concerning broadband access devices (see Section V).

c) Enriching data sets: We used a fast DNS query tool [13] to obtain PTR and SSHFP records for IP addresses we investigated in more detail. For these, we also looked up the responsible AS and WHOIS contacts. Finally, we mapped them to country and city using the Maxmind geo-IP databases [14]. For some ASes, we investigated the device types in a network. Our methods of choice were `nmap` service scans on ports 20, 80, 443 and 8080. We elaborate on this in Section V.

V. Results

We present the results we obtained from our scans. Additionally, we classify these results and give recommendations.

A. Protocol versions

SSH 1 is flawed and should not be used anymore. In our scans, we could group hosts into servers that only support the old SSH 1 variants (SSH 1.3–1.51), those that only support SSH 2 versions, and those that offer both (pseudo-version 1.99). In all three scans, more than 90% of all SSH servers supported only SSH 2. The share of hosts that support both SSH 1 and SSH 2 is decreasing: between January and July 2013, it fell from 7.9% to 6.2%, while SSH 2-only hosts increased by 2%. The use of SSH 1 (0.5% in July) could be almost negligible – but this percentage corresponds to an absolute value of 42,000 hosts. We believe these are probably older systems: major Linux distributions, for example, disable SSH 1 by default. Overall, however, this finding can be viewed as very good. If a recommendation can be given at all, then it should be to disable SSH 1 for good on clients and servers.

B. Server versions

The server string sent to the client reveals which SSH implementations and vendors are used on the server side. To some degree, this allows to identify older, non-updated SSH servers. Updates are an important process to remove vulnerabilities. There is an important caveat to take into account here: since our scanner executes the SSH protocol faithfully, we can only report the server string as sent to us. However, SSH servers may have been (manually) patched without this being reflected in their server strings.

Figure 1 shows the most frequently used SSH servers and development over time. For better visualisation, we grouped OpenSSH versions with the suffix *p* together with the original version. The *p* identifies ports to non-OpenBSD systems. We find that OpenSSH is by far the most popular SSH server, running on about 65% of SSH hosts. Dropbear is used on 20% of the scanned SSH hosts. Other servers occurred much less frequently. Worth mentioning are Cisco implementations (4%) and ROSSSH, a MikroTik implementation (about 2%). All other servers occurred on less than 1% of SSH hosts. Some SSH servers use custom vendor identifiers. In our July scan, we found more than 32,000 vendor strings that occurred at most 5 times. Almost all of these are either random-looking Base64-like strings or hex-encoded characters. These strings were chosen intentionally or randomly set during the installation.

Concerning the age of servers, we found that OpenSSH 4.3 dominates the SSH landscape, even though it is from early 2006. There are several known vulnerabilities for this version. Fortunately, we can also see that the use of OpenSSH 4.3 is declining (but more than 1.5 million servers remain). Only 6.6% of OpenSSH servers in our July 2013 scan have a version number of 6.0 or above. OpenSSH 6.0 was released in April 2012. The latest OpenSSH version, 6.2, was published in March 2013. It appears in only 1% of version strings in our scan of July. Similarly, only 6.5% of Dropbear servers are more recent than February 2012 (version 2012.55). Whatever one's view on the update rate of SSH servers may be, we find it intriguing that the adoption of new server versions seems to be extremely slow.

C. Weak keys

We investigated how many SSH servers presented RSA keys that must be considered cryptographically weak due to irregu-

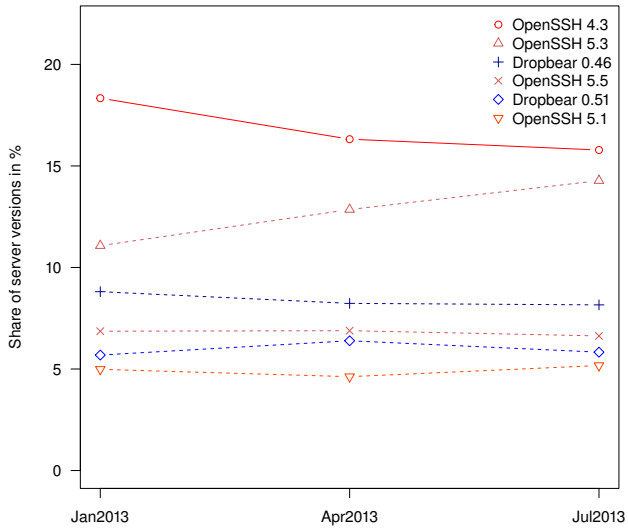

Fig. 1: Most frequently encountered SSH server versions over all scans.

Heninger *et al.*	**Jan 2013**	**Apr 2013**	**Jul 2013**
0.03%	0.016%	0.013%	0.014%

TABLE II: Coprime weak RSA keys across all hosts.

larities during key generation. In doing so, we reproduced the results by Heninger *et al.* and determined to which degree the situation has improved since the authors' disclosure.

1) Coprime weakness: Heninger *et al.* showed in [1] that a number of RSA public keys exist whose moduli share one prime number with the modulus of a different public key. In such cases, the private keys can be calculated very fast. We used the original authors' implementation [15] to determine how many coprime vulnerable keys still exist. Table II summarises our results. While the numbers seem to have fallen in the year that passed, they also do not seem to change any more – the changes in the last two scans are likely due to measurement inaccuracy and rounding effects. This can be seen as an indicator that Heninger *et al.*'s responsible disclosure is working, albeit slowly.

We were surprised by one effect: when we filtered for SSH 1 keys, we found that about 2.4% of them had a coprime weakness, and the number is not decreasing. This hints that entropy problems were present from the beginning of deployment. Furthermore, it may be an indication that the very old systems do not receive attention anymore. As SSH 1 should not be used anyway, it does not constitute a major weakness, however.

2) Debian-weak keys: A second well-known vulnerability is the use of SSH keys created on Debian-based Linux systems with a version of OpenSSH that contained a serious bug in key generation [5]. These flawed versions were distributed from 2006 to 2008. Their public-private key pairs can be precomputed with reasonable effort. H. D. Moore published a list [16] of known-weak keys which we used, too. The good news is that, five years after the bug was found, Debian-weak

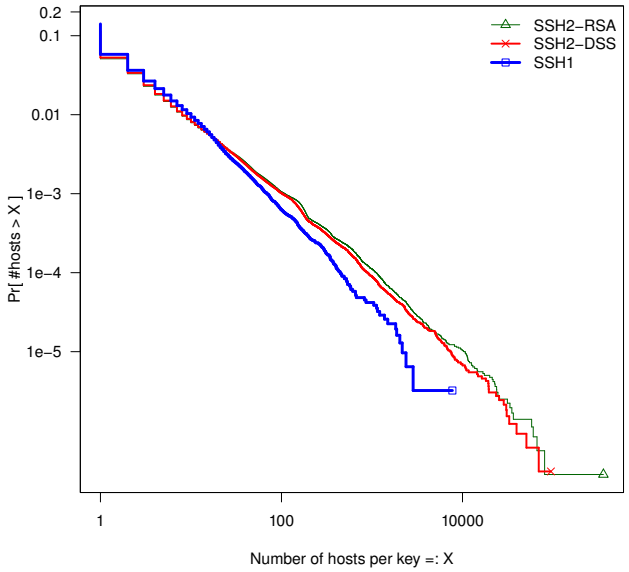

Fig. 2: CCDF: frequency of case 'key occurs on more than X hosts' (for July 2013).

	Jan 2013	**Apr 2013**	**Jul 2013**
SSH-1.x	530 k (54.16%)	364 k (50.70%)	261 k (49.51%)
SSH-2.0	12.7 M (61.03%)	10.5 M (59.02%)	8.9 M (57.46%)

TABLE III: Duplicate host keys fetched during the scans.

keys are rare and becoming even less common, thus continuing the trends documented in [1], [6]. In January, 0.017% of SSH 1 keys were vulnerable, and 0.077% of SSH 2 keys. In July, the numbers had decreased to 0.011% and 0.063%, respectively. The fact that more SSH 2 keys seem to be affected may hint that the majority of SSH 1 keys were generated before 2006 and these servers do not receive updates.

D. Duplicate non-weak keys

More than half of all fetched host keys were used on multiple IP addresses. This phenomenon was also reported by Heninger *et al.*, who distinguished weak keys on devices with poor entropy (see above), devices with default keys, and simple reuse of the same key on different hosts. We were interested in the latter two phenomena as they relate to network management practices. We investigated which of the two phenomena occurs in which networks and to which degree it constitutes a security weakness. Our analysis was mostly carried out on the data set from July 2013, with the exception of one AS.

The first thing to note is that duplicate *yet non-weak* keys occur much more frequently than cryptographically weak keys. Ranking their occurrences together, the first weak key appeared on rank 1337 in our data set. Figure 2 visualises how commonly the case that 'a key occurs on more than X hosts' occurs over the entire IPv4 space. Table III shows the number of non-unique keys for all three scans.

In our study, we investigated the 10 most frequent duplicated keys, which together account for about 6% of all keys.

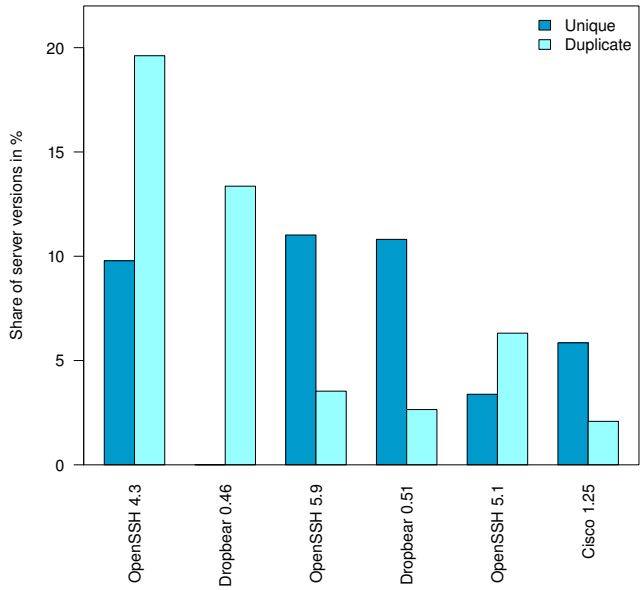

Fig. 3: Market share of servers with unique and duplicate keys.

To this end, we enriched our data set with the DNS PTR resource record for all IPs in our data set. This yielded 6.2 million names, of which 5.8 million were distinct. Second, we determined the operators of Autonomous Systems and their registered location on the globe. Third, we used the Maxmind geo-IP database to retrieve country and city of an IP address. Fourth, for the cases of mixed hosting/broadband access we also carried out `nmap` service scans. One fact to take into account here was that most IP addresses we investigated were dynamic, and some changed on a daily basis. As service scans on several ports take longer, it was not possible to unambiguously match IP addresses to SSH keys *and* running services. We thus used the service scans primarily to gain a picture how the AS was used.

a) General findings: Figure 3 relates duplicate keys to market share of server versions. Some servers, such as OpenSSH 4.3, are responsible for a large portion of duplicate keys. Dropbear 0.46 distributes almost exclusively duplicate keys. It is known that Dropbear is often used in embedded systems, so this points at devices with default keys. However, it is interesting that Dropbear 0.51 has a larger share when offering unique keys, which could mean it is primarily used for other purposes. For the following analysis, we group by probable cause of key duplication.

b) Mixed hosting/broadband ASes: Among the top 10 keys, a pattern for key duplicates was found in ASes with IPs for both (Web) hosting and broadband access devices. For example, the most common key in our data set was an RSA key found primarily in just 10 AS (more than 99% of 360,000 occurrences in July and 560,000 in January). The only feature that all these hosts seemed to share was that they employed Dropbear 0.46. This particular key had already been found by Heninger *et al.*, although only 240,000 times. The key was of particular interest because 80% of occurrences were found in the two AS of the same Taiwanese ISP (which we will call *TW1* and *TW2*), particularly in Taipei, and another 16% in that of a mainland Chinese ISP (*CN*). The rest were distributed over Central America, the Caribbean and the USA. This geographic pattern was the same across all scans. Almost all (99.5%) of DNS names in Taiwan and China contained the string 'dynamic', which already pointed at use for broadband access devices. Thanks to the database created by the authors of [1], we learned that they had identified the key as a default key in a DSL device, too. It seems plausible that it is a device sold primarily in the above mentioned markets, at least with this key. The AS in question did not only present this key, however – we found a large number of other keys. Figure 4 shows the distribution for *TW1*. The security finding here is that the use of a shared key among so many different systems is a serious weakness, as it cannot be safely used for purposes like remote administration. This is even more so as we found two other extremely common keys in *TW1* (ranks 5 and 6, 66,000 and 60,000 times, respectively). We informed the ISPs, but did not receive a reply.

c) (Mostly) safe broadband access devices: We found a similar device as above for the key at rank 9 in our database. The corresponding IP addresses were almost exclusively in Singapore and belonged to a broadband provider (*SG*). The PTR records were unique strings in that provider's domain. The SSH server was overwhelmingly Dropbear 0.52 (99%). Again, this seemed to be a case of default key in a device for broadband access. The situation was better here than above, however: all bar 38 devices were within this AS. However, we did find the other 38 occurrences as far away as New Zealand. We can only speculate how the 'escaped keys' got there – possibly by simple labour relocation. Figure 4 shows the distribution for AS *SG* – it once again seems to be used for multiple purposes as there is a large number of other and even unique keys.

d) Home entertainment devices: We encountered an entirely different case of key reuse for an RSA and a DSS key that occurred 58,000 times and 50,000 times in our scans (ranks 7 and 8), respectively – but distributed across well over 1,500 ASes in more than 50 countries. More than 80% of the IP addresses were in the USA or a closely related market like the UK, Canada or Australia, however. The SSH server revealed the solution as it contained a vendor-specific string that belongs to a home entertainment solution by a well-known and globally acting corporation. One of their products is a fully configured 'off-the-shelf' Linux-based entertainment server solution. We verified this by scanning port 80, which revealed itself to use a Red Hat-based Apache version, which this particular vendor also advertised. Connecting to several of the IP addresses with a browser, we confirmed the brand. Additional credibility was lent to this as 38% of the AS with this key carried the term 'cable' in their WHOIS entry – this type of broadband access is fast. Needless to say, this kind of setup is extremely dangerous in terms of security.

e) SSH gateways: We encountered an entirely different setup for the second most common key in our database, found

in the AS of a large German ISP (*DE*, 95,000 occurrences). The key was strictly limited to this one AS. It occurs on 75% of hosts that we found in that AS, although many different keys are used there (see Figure 4). The next most frequent key in that AS occurred only 41 times. Resolving the IPs to host names (PTR record), we found that a third of them pointed to just one domain; the rest resolved mostly to different domains. The SSH server was always a modified Dropbear version and the service scans showed that only the Apache Web server was used in this AS. This seemed to point at a hosting situation with a very different setup. Two plausible reasons were that we were either facing an SSH gateway or that Virtual Machine images with the same key had been deployed. The former would be a good solution, the latter a very poor one. Unfortunately, these setups are hard to distinguish without very intrusive scans like OS scans. We thus contacted the ISP. They confirmed we had identified 'shared hosting' products, and that SSH traffic is directed via a gateway. This allows simpler management and enables them to switch SSH keys on the real machines without customers encountering SSH host key warnings. The private key is never exposed to a customer. When done properly, as seemed to be the case here, this SSH setup seems quite reasonable.

We found what seemed to be similar cases of 'key clusters' (ranks 3 and 4) in the AS of a Japanese corporation's branch in the USA (*US*/*JP*), where two keys represented 80% of fingerprints. We also found a cluster in an AS in the USA (*US*, rank 10). The variance in these networks concerning SSH servers and Web servers was higher, however. We contacted the ISPs, but did not receive an answer. Their AS setup may be a mix of the ones in Germany and the ones in Taiwan and China.

f) Per-customer keys: An AS of interest was that of another large German company, where we found about 60,000 keys that occurred more than once in their AS, but no single key was used more than 200 times. These keys did not occur in our top 10, but their existence pointed at an entirely different management practice. Our suspicion was that Virtual Machine images might be distributed to customers without regenerating the host keys. We contacted the ISP. They confirmed that Virtual Machines were used, but that SSH keys are generated at deployment time. They investigated and offered a plausible explanation: their customers are allowed to bind several IP addresses to an interface. In such cases, the host key would be the same for all interfaces, and we would have mistaken distinct IP addresses for distinct machines (note our caveat from Section IV). However, they could also confirm that there were some cases where customers did indeed reuse keys across *different machines*. Note that the type of hosting is entirely different here, with the responsibility of key management on the side of the customer. This latter case of reuse is a potential vulnerability, especially if customers run public-facing services with attack surface.

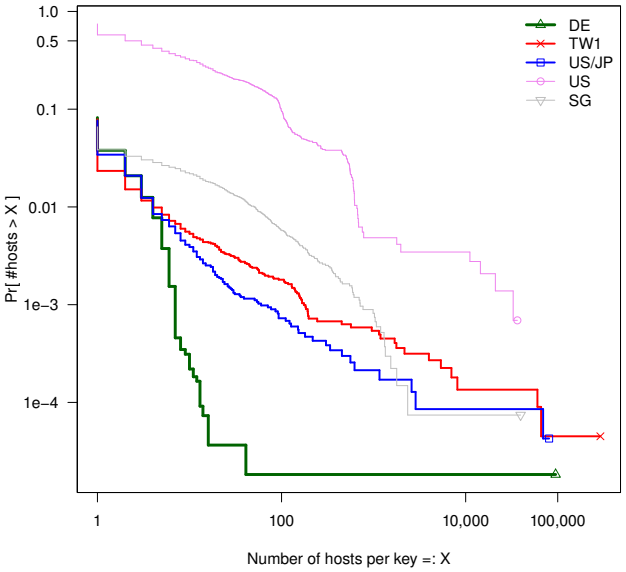

Fig. 4: CCDF: 'key occurs on more than X hosts', for selected AS.

E. Use of SSHFP

A way to make SSH connections more secure, especially initial connections, is to use the DNS to store an SSH fingerprint in a dedicated DNS resource record, SSHFP. This is defined in RFCs 4255 and 6594. DNSSEC can be used to sign these records. We determined whether such records are in use and accurate. We looked up the PTR records of IPs with active SSH servers, and then forward-resolved the thus obtained domain name. We restricted our analysis only to such IP/domain combinations where the two matched. Our results thus constitute a lower bound. We found 2,070 distinct domains employing a total of 4,374 SSHFP records (a domain may have multiple records, one for every key type it uses). We considered a domain secured with SSHFP if at least one SSHFP resource record matched a key. We found 94% of the 2,070 domains are correctly secured with SSHFP records. However, only 660 of these also had correctly secured their SSHFP records with DNSSEC. The rest transmitted the SSHFP record via insecure DNS.

A relatively low number of domains deploy the SSHFP record. It is plausible to assume that those operated by technology aficionados (e.g., open source projects) are more likely to do so. On the one hand, one could argue that 94% correctly deployed records are a good value. On the other hand, 6% wrong records can be taken as an early warning that deploying security-relevant resource records must not be taken lightly. Sites considering such a move should definitely have their DNS and SSH administration tightly integrated.

F. Cryptographic algorithms

We investigated which cryptographic algorithms were advertised by servers. Concerning symmetric ciphers, we found 111 different ciphers sent by servers. This is more than three times the number of officially registered algorithms. For July

2013, the most frequent supported cipher was 3DES, offered by 99.5% of hosts. AES offers in CTR mode, with a key length of 128 bit or more, summed up to 69.9%; in CBC mode the number was 87.7%. RC4 was also offered frequently (67.1%). Due to recent progress in security analysis on CBC mode and RC4 [17], [18], their usage is discouraged. Recent versions of OpenSSH prefer AES-CTR and AES-GCM instead of the aforementioned ciphers. Very few servers also offered weak ciphers such as DES or RC2 (0.03%). The numbers for April were almost the same.

All servers advertised support for HMAC for integrity protection. The less CPU-intensive UMAC increased its market share from 40% to 45% during our investigation period.

Regarding key exchange mechanisms, the RFC requires both `diffie-hellman-group1-sha1` and `diffie-hellman-group14-sha1`. The former is supported by almost all hosts. However, we found that, among all scans, a non-negligible fraction of servers did not implement the latter. The number decreased from 32% in January 2013 to less than 29% by the end of July 2013. Some Dropbear implementations and very old OpenSSH servers do not provide both methods. The reduction can thus be explained by the changes in market share. This finding shows again that old SSH software is still widespread.

G. Key lengths

We investigated the lengths of host keys for SSH 1 and SSH 2. Figure 5 shows the distribution for the different key types in July 2013. Concerning SSH 1, we find that about 9% of all keys are shorter than 1,024 bit. These key lengths are considered too short by today's standards – RSA at 768 bit was factored in 2009, and the NIST recommends at least 2,048 bit keys. However, our finding must be viewed against the background of the relatively rare occurrence of SSH 1 and its other flaws. Furthermore, as the plot shows, most remaining keys are 1,024 bit or longer. Concerning the more important SSH 2, we see an entirely different distribution. Keys with a length of less than 1,024 bit make up for only about 5%. Keys with a length of 1,024 bit make up for more than 50%, and those with a length between 1,024 bit and 2,048 bit add another 44%. Longer keys are much rarer, although there are a few examples of 16,384 bit keys, which can be regarded as extremely conservative from a security perspective. We did not find any major changes in the key length distribution over the course of our scans. We also compared our numbers to the findings of Heninger *et al.* The authors had reported an occurrence of 0.08% of 512 bit SSH 2 keys. In April 2013, we found 0.3% of keys to have this length. This means there are about 50,000 hosts with 512 bit keys.

H. Security recommendations

SSH servers are generally deployed in a secure way. We recommend to disable SSH 1 for good and to make sure that the server software is updated regularly. Conservative security policies include avoiding RC4 and AES-CBC due to recent discoveries [17], [18], in addition to proven weak ciphers

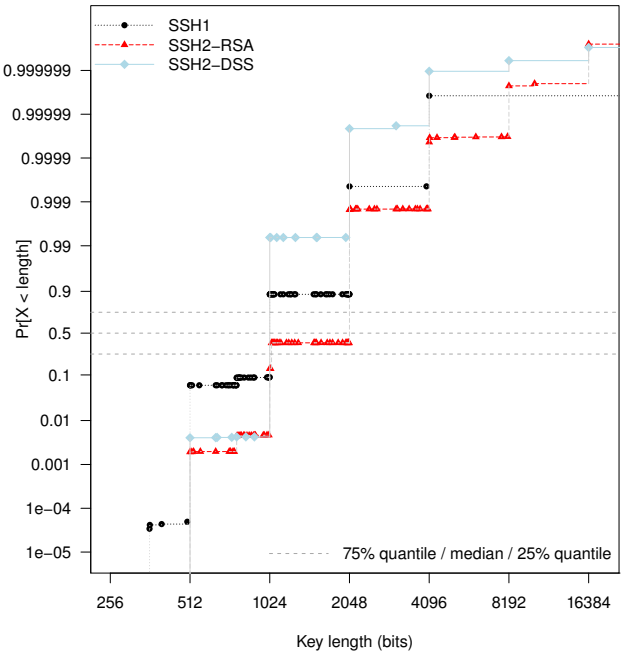

Fig. 5: Cumulative distribution of key lengths in July 2013. (Note the unusual atanh scaling (double-ended pseudo-log) of the y axis.)

such as RC2 or DES. The HMAC and UMAC authentication schemes can currently be regarded as safe. Key exchange mechanisms should be based on Diffie-Hellman to guarantee Perfect Forward Secrecy.

There are four things which need to be kept in mind when creating SSH keys: (1) RSA as well as DSS keys should be 2,048 bits or longer. In the case of elliptic curve cryptography, all three specified ECDSA key lengths provide enough security. (2) When using 'off-the-shelf' devices, SSH keys should be regenerated where possible. (3) In general, devices with poor sources of entropy should either be avoided or their SSH keys generated on other, stronger machines with more entropy. (4) Updates of SSH server software should be carried out regularly. This removes weaknesses and ensures support for newer ciphers and longer keys is guaranteed.

VI. Ethical considerations

SSH is a highly sensitive security protocol that we scan on such a large scale that an Autonomous System is likely to be hit several times during a day. This means our scans trigger certain intrusion detection systems, which in turn may cause unnecessary work for administrators.

To minimise negative effects, we implemented several cautionary measures during our scans. As a first precaution, we notified about 20 CERTs, watchlist services like the Internet Storm Center [19], and several blocklist operators before our scans. As we had to carry out full handshakes, we sought for a way to label our scans as less threatening. First, we supplied contact information in the SSH username[1]. Second,

[1] `university-security-research-see-https-pki-net-in-tum-de`

we changed the authentication method to a non-existent one[2] and added additional information with a URL in the comment section of the SSH server string.

As a consequence of our scans we received a number of complaints by email. We replied to every such email in person, stating our intentions, offering to blacklist the affected system and providing links to further information. Over the course of all three scans, we received a total of 191 abuse emails. About 80% of these were automated complaints sent by intrusion detection systems, firewalls or filtering software such as `fail2ban`. On 10 occasions, friendly conversations with administrators followed our reply – mostly, these were requests for further information. We received only one email mentioning possible legal consequences for us. We replied to this with a second, longer explanatory email and blacklisted the affected system. This seemed to resolve the issue as we did not receive any further replies. Our blacklist now contains entries from 17 parties. We need to add that our group has been active in other scanning projects before (e.g., SSL) and has also carried out (much shorter) SSH scans for testing purposes before the full scans started. Some blacklist entries stem from the earlier scans. Only one hosting provider requested blacklisting for their range. The total number of IP addresses requested to be blacklisted is roughly one million – about 830,000 of these requested by the mentioned provider.

With the above precautions implemented, we ultimately decided to carry on with our scans on Internet scale. The position we take is that research can contribute to a safer Internet as a whole. Furthermore, carrying out scans over a period of time has given us valuable pointers. For example, one conclusion that we draw is that such scans should not be carried out too frequently, but *that there is also no need to – the rate of change is slow.* We recommend to always notify watchlist and blocklist operators, CERTs, and previously affected parties as far as they are known.

We believe it is important for the scientific community to be able to reproduce our results. We encourage further analysis of our data sets, which we are going to release on our site [20]. However, to prevent abuse, certain precautions are necessary. In part, we derive the measures we take from the practices described by Allman and Paxson in [21]. Our goal is to avoid that an attacker can use sensitive information like weak keys or server version to stage attacks without having to carry out the same kind of large-scale scan that we did. Thus, we must avoid that vulnerable servers can be identified without interacting with them. Approaches like *k-anonymity* and *l-diversity* have limitations when the attacker is in possession of other data sets that he can link to ours. A better way to protect against abuse is thus to remove indications of vulnerability from the data set, and to suppress IP addresses entirely, replacing them with random numbers. This still allows researchers to reproduce most of our statistical results, but prohibits potential attackers from learning more from our data set than they could learn by scanning themselves. We will also be happy to share our data sets in their entirety with other research groups if they are willing to enter an agreement with us that they must protect the data to the same standards and will not engage in misuse.

[2] `security-research@bozen.net.in.tum.de`

VII. Summary and conclusion

We have carried out the largest and longest SSH scans to date to determine the state of SSH deployment on the Internet. We generally found that relatively old software is still common on the Internet, and the rate of change is slow. The SSH 2 protocol seems to mostly have displaced the flawed SSH 1, however. We reproduced the results by Heninger *et al.* and found that while the number of cryptographically weak keys is low and has somewhat decreased since the disclosure, improvements are now on the border of measurement inaccuracies. Among our key contributions is also an analysis in which networks strong, yet duplicate keys occur, and what reasons this might have. We found setups that offered varying degrees of security: we found well-kept and centralised SSH gateways and geographically restricted devices with default keys, but also devices that are geographically wide-spread or even sold globally. We analysed the use of the SSHFP records and found that these resource records are mostly accurate, although it is hard to draw conclusions with respect to other records holding cryptographic information. Unfortunately, most of them are not secured with DNSSEC. Concerning the actual ciphers and key lengths, our results show that key lengths are sufficient in the case of SSH 2 and ciphers give practically no reason for concern.

Our data sets will be provided on our homepage [20], in a pseudonymised format to counter potential use in attack preparation. We invite the scientific community to download these data sets and carry out further investigations.

Acknowledgements We thank Nadia Heninger and Zakir Durumeric for looking up some keys in their database and H.D. Moore for providing us with the blacklist of Debian-weak keys. We thank Ondrej Mikle for his DNS scanner.

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under support code 01BY1203C, project *Peeroskop*, and 16BP12304, EUREKA project *SASER*, and by the European Commission under the FP7 project *EINS*, grant number 288021.

References

[1] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your Ps and Qs: Detection of widespread weak keys in network devices," in *Proc. 21st USENIX Security Symposium*, Bellevue, WA, USA, Aug 2012.

[2] A. Lenstra, J. Hughes, M. Augier, J. Bos, T. Kleinjung, and C. Wachter, "Ron was wrong, Whit is right," Cryptology ePrint Archive, Report 2012/064. http://eprint.iacr.org/2012/064.

[3] N. Provos and P. Honeyman, "ScanSSH – scanning the Internet for SSH servers," in *15th USENIX Systems Administration Conference*, Baltimore, MD, USA, Dec 2001.

[4] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage, "When private keys are public – results from the 2008 Debian OpenSSL vulnerability," in *Proc. 9th ACM SIGCOMM Internet Measurement Conference (IMC)*, Chicago, IL, USA, Nov 2009.

[5] Debian Project, "Debian security advisory: DSA-1571-1 openssl – predictable random number generator," http://www.debian.org/security/2008/dsa-1571, 2008.
[6] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL Landscape: a thorough analysis of the X.509 PKI using active and passive measurements," in *Proc. 11th ACM SIGCOMM Internet Measurement Conference (IMC)*, Berlin, Germany, Nov 2011.
[7] N. Vratonjic, J. Freudiger, V. Bindschaedler, and J.-P. Hubaux, "The inconvenient truth about Web certificates," in *10th Workshop on Economics of Information Security (WEIS 2011)*, Fairfax, Virginia, USA, Jun 2011.
[8] J. Schonwalder, A. Pras, M. Harvan, J. Schippers, and R. van de Meent, "SNMP traffic analysis: approaches, tools, and first results," in *Proc. 10th IFIP/IEEE Int. Symposium on Integrated Network Management (IM)*, Munich, Germany, 2007.
[9] D. Leonard and D. Loguinov, "Demystifying service discovery: implementing an Internet-wide scanner," in *Proc. 10th ACM SIGCOMM Internet Measurement Conference (IMC)*, Melbourne, Australia, Nov 2010.
[10] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast internet-wide scanning and its security applications," in *Proc. 22nd USENIX Security Symposium*, Washington, D.C., USA, Aug 2013.
[11] J. Heidemann, Y. Pradkin, R. Govindan, and C. Papadopoulos, "Census and survey of the visible Internet," in *Proc. 8th ACM SIGCOMM Internet Measurement Conference (IMC)*, Vouliagmeni, Greece, October 2008.
[12] Y. Xie, F. Yu, and K. Achan, "How dynamic are IP addresses?" in *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug 2007.
[13] O. Mikle, "DNS scraper," https://github.com/hiviah/dns-scraper, 2013.
[14] Maxmind, "GeoLite free downloadable databases," http://dev.maxmind.com/geoip/legacy/geolite/, 2013.
[15] N. Heninger and J. A. Halderman, "Fast pairwise GCD computation," Web site with source code: https://factorable.net/resources.html, 2012.
[16] H. D. Moore, "Debian OpenSSL predictable PRNG toys," Web site with weak keys: http://digitaloffense.net/tools/debian-openssl/, 2008.
[17] M. R. Albrecht, K. G. Paterson, and G. J. Watson, "Plaintext recovery attacks against SSH," in *Proc. 30th IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2009.
[18] D. J. Bernstein, "Failures of secret-key cryptography," Talk at FSE 2013. http://cr.yp.to/talks/2013.03.12/slides.pdf, 2013.
[19] Internet Storm Center, http://isc.sans.edu, 2013.
[20] SSH@MEASR.NET, http://ssh.measr.net, 2014.
[21] M. Allman and V. Paxson, "Issues and etiquette concerning use of shared measurement data," in *Proc. 7th ACM SIGCOMM Internet Measurement Conference (IMC)*, San Diego, CA, USA, Oct 2007.