Clusters of Re-used Keys

Stephen Farrell,
Trinity College Dublin
stephen.farrell@cs.tcd.ie
Article version 0.1, Compiled on 2018/03/29 at 19:13
work-in-progress

***Abstract*—This article describes a survey of long-term cryptographic public keys observed in real deployments of secure-shell, e-mail and web protocols in two similarly-sized countries – Ireland and Estonia. We find that keys are very widely re-used across multiple IP addresses, and even autonomous systems. From one run scanning 18,268 hosts in Ireland that run at least one TLS or SSH service, approximately 53% of the hosts involved are using keys that are also seen on some other IP address. For each key, if two IP addresses share that key, then those two IP addresses are considered members of the same cluster. In the same scan we find a maximum cluster size of 1,991 hosts and a total of 1,437 clusters, mostly with relatively few hosts per cluster (median cluster size was 26.5, most common cluster size is two). In that scan, of the 54,447 host/port combinations running cryptographic protocols, we only see 20,053 unique keys (36%), indicating significant key re-use across hosts and ports. We describe the methodology followed and the published source code and public data sources that enable researchers to replicate, validate and extend these results. Clearly, such key sharing can create undesirable security and privacy dependencies between cluster members. The author is currently starting the process of contacting local (Irish) asset-owners to try establish the reasons for this key sharing and to possibly assist with improving network posture.**

***Index Terms*—Internet measurement, security, privacy, cryptographic key management**

I. Introduction

This article describes scans of two populations of hosts on the Internet, in Ireland (IE) and Estonia (EE). The hosts in question offer some mail service, that is, the hosts in question are contactable at IPv4 addresses that listen on TCP port 25. Our scans record information relating to the long-term cryptographic keys used by a number of services on those hosts. We see unexpectedly large-scale re-use of cryptographic keys across clusters of hosts in these scans.

At the time of writing the author is attempting to contact asset owners for some of the hosts in the Irish (IE) population scanned in order to try establish why and how key re-use has occurred, whether or not that was deliberate, and whether the asset-owners can or would prefer to move away from re-using long-term cryptographic keys on multiple hosts. This version is therefore a work-in-progress to assist in those discussions and will be updated as those proceed.

Manuscript received MMM dd 2018; Stephen Farrell is with Trinity College, Dublin 2, Ireland (email: stephen.farrell@cs.tcd.ie, https://www.cs.tcd.ie/Stephen.Farrell/)

II. Background

Over the last five years, the proportion of Internet traffic that is encrypted, particularly using the Transport Layer Security (TLS [1]) protocol has been increasing consistently. [2], [3] [1] The Secure-Shell (SSH [4]) protocol has also been nearly ubiquituously used for systems administration for many years. [5]

These increases in the use of encryption are consistent with the consideration that "Pervasive Monitoring is an attack" [6] [7] which reflects some of the general drivers behind recent increased deployment of cryptographic protocols. However, in addition to making use of these protocols, we also want endpoints to move beyond opportunistic security [8] and properly manage keys, especially long-term asymmetric keys, as otherwise we won't gain all. the intended benefits of increased deployment of security protocols.

For TLS and SSH, the long term key pairs in which we are interested are essentially used for host authentication, though historically they may also have been used directly for RSA key transport. Re-using the same long-term asymmetric key pair values for many different instances of services can create vulnerabilities or increase the probability or impact of some attacks. This can be relatively easily avoided if different key pairs are used for each instance, and if keys are regularly rotated.

In our scans we do see many keys re-used in clusters on different hosts. A **cluster** is defined here to be the largest set of hosts/IP addresses in our population such that, each host in the cluster shares at least one public key (for some service) with another host in the cluster. In other words, two hosts are in the same cluster, if they share a private key, regardless of the port with which that private key is used.

A. Safe Key Re-Use

Not all re-use of keys is bad. There are certainly situations where it is reasonable to re-use a key pair for different services on a single host, for example if a public key is certified for multiple related DNS names (perhaps smtp.example.com and mail.example.com) that resolve in the public DNS to the same IP address, and where the same server instance listens on both ports. In such cases it seems fairly clearly reasonable to use one key pair to protect services on say port 587 (email submission) and port 25 (email submission and mail transfer).

Where services on the same host are less related, for example, web and mail service, it seems a little less reasonable

[1] See also https://letsencrypt.org/stats/#percent-pageloads for recent graphs.

to re-use the same key pair, however, it is certainly arguable that if an attacker manages to gain access to a host with software-only key storage, a successful attack that gains access to one key pair on the host also results in access to all the key pairs on that host.

There are also cases where re-using a key across multiple hosts can be defended. For example, if the set of hosts for which the key is re-used are all virtual machines that are, and always will be, running on the same physical hardware, and all under the control of one entity. Another reasonable case might be if the set of hosts for which the key is re-used are all "behind" a middlebox or hypervisor that is not externally visible at the IP layer and where that middlebox offers a cryptographic front-end with the full knowledge of the hosts visible on the public Internet.

There may also be cases where our scans show the same key being used, but where the "real" application will use a unique key, if for example, a TLS Server Name Indication [9] for the intended name/application is used. Given our scans start off based on IP addresses, we would miss such cases. Nonetheless, even having the same default key pair for port 443 on multiple hosts can be problematic.

Of course, simply using different keys for a contactable IP address does not in itself imply that a service is more secure, better managed, nor that it is independent of other hosts, given current virtualisation trends. Note though that from Internet vantage points, observers cannot easily distinguish between these and less desirable cases.

B. Dangers of Key Re-Use

While the previous sub-section describes some situations where key re-use can be defended, there are also clear dangers potentially associated with key re-use.

1) Risk of Leaks: Where many copies of private keys are stored as files on disk, which is common, a leak from one host affects all. If we notionally say the cost of a leak of any of the host keys is the same and is C and there are n hosts, then the cost of one cluster key leaking is $n * C$. If the probability of a key leaking from any host is p, then, we can say that the probability of some key leaking from some host is $n * p$. If those considerations do apply then the risk of the cluster scenario could be $n^2 * p * c$ instead of $n * p * c$ if each host has a different key. In other words, the cluster scenario is riskier, and the bigger the cluster grows, the faster the risk get worse. In the argument here we're making reasonable assumptions about independent events and equal costs and probabilities. While those assumptions will not always apply, they often will, and the external observer cannot know when they do not apply, and so would be wise to assume they do, when considering what services to use.

2) Masquerade: If used for authentication, then re-use of key pairs across hosts means that a breach of any host in a cluster enables a successful attacker to also masquerade as any host from the cluster in question.

3) Credentials: Being able to masequerade as any of the hosts in a cluster is likely to allow an active attacher to capture user credentials, for example passwords sent in IMAP or SMTP transactions. Even worse, if SSH password logins are still enabled, such an attacker can make use of those privileged user credentials. Given user credentials (especially passwords) are frequently re-used in many places, this kind of attack may extend beyond the cluster to other hosts at which those credentials can be used.

4) Web Origins: If the hosts in a cluster represent different web origins then the ability to masquerade as any host in the cluster would allow theft of web cookies, breaking the same origin policy on which the web depends. Given new web technologies such as alt-svc [10] and the ORIGIN frame [11] in HTTP version 2, breach of one web server in a cluster should allow masquerading as any other, all from the initial host.

5) Key Transport: If a long term RSA key pair is or has ever been used for key transport, and if an attacker has a record of those sessions, then breach of any host using that key pair allows the attacker to directly decrypt all recorded sessions.

6) Million Messages: Any unpatched vulnerability related to use of a private key that requires multiple messages to be sent by the attacker (e.g. the ROBOT [2] attack [12]), can likely be exploited more efficiently and more stealthily, if there are more service endpoints to which the attacker can send messages.

7) Cross Protocol: Key re-use increases the potential for cross-protocol attacks, and the likelihood that some service software supports older versions of SSH or TLS/SSL, that may be vulnerable to such attacks.

8) Revocation: If a key is known to be compromised, then it ought be revoked if the public component is part of some Public Key Infrastructure (PKI[13]). With key re-use it may be that the same private key continues to be used on some hosts whilst being replaced on others. If the reason for revocation related to potential private key compromise, then those less-well managed hosts will remain at risk even after revocation of some certificates and replacement of some key pairs.

9) Theory: It is now feasible to do security proofs for realistic protocols, and indeed TLS1.3 has been the subject of such formal studies. It is not clear that widespread key re-use was considered in such studies. The impact, if any, on such proofs is uncertain.

10) Laziness: Re-use of keys can clearly be a result of careless management. Advertising attributes that can reasonably be taken to indicate carelessness to the public Internet seems like a bad plan for a service operator.

C. Research Questions

The overarching research question behind this work is to investigate whether or not local measurement of Internet security posture is more useful (compared to Internet-scale measurement) in helping asset holders to improve security posture. This study tackles one aspect of that work.

As part of this work, we define a metric related to key re-use: the percentage of hosts in a population doing some cryptography using keys that are known to be used for multiple IP addresses; versus hosts where we only see keys being used

[2] https://robotattack.org/

for one IP address. (Of course, the scope of our measurement means that hosts could be mis-identified as being in the latter set, if in fact some key is also used elsewhere.) Call this the "hosts are re-using keys" (HARK) percentage.

One hypothesis of this work is that reducing HARK could correlate with improvements in security, and be a reasonable indicator of whether a population are managing security more carefully than in the past. We do not necessarily aim to reduce HARK to zero, nor do we currently claim to know what value might be optimal or a good target. But of course leaving room for future work is also a fine feature for a proposed new metric.

III. Methodology

A. Overview

During October and November 2017, we extracted scans of Irish (IE) and Estonian (EE) hosts that listen on port 25 from the censys infrastructure. [14] [3] Subsequently, starting in March 2018, we began to scan locally using ZMap/ZGrab [15] ourselves, but still limited to hosts who listen on port 25 (according to ZMap).

Our expectation was that hosts that run the SMTP protocol (mail servers that listen on port 25) could be a useful population to examine as they hopefully represent hosts with a better than average probability of being well managed. Initial examination of that data showed that keys were being re-used more frequently than expected and that there were clusters of hosts re-using private keys in various ways.

For each host we record SSH or TLS details for each of the ports listed in Table I. (Censys scans did not include port 587 in November 2017.) We developed analysis code to identify and analyse the clusters of hosts re-using keys as described below.

All source code required to replicate this study or do a similar-scale scan (of perhaps 20-30k hosts) has been published [4] under the MIT license. Each scan of that scale requires a few days to run on our very modest infrastructure due to built-in delays between scan stages to consume less bandwidth. About 3GB of storage is required for results. As-is, our tools take the contry-code (e.g. IE or EE) as an input and should work for other countries with similar numbers of port 25 listeners.

A speedier or larger-scale scan could likely be done with minor code changes given better bandwidth, processing power and storage. Note though that one of the research questions we want to explore is whether such "local" scans can be more effective in helping asset owners mitigate risks, so we're sanguine about the speed and scale limitations, and happy to be able to demonstrate that quite limited resources are sufficient.

Data sets from the scans are not being published as doing so could assist an attacker with "lateral" movement to other hosts in a cluster. Such an attacker would also be assisted in attacking users attempting to authenticate to hosts in a breached cluster. However, it is important to note that, as shown by this article, it is relatively simple to detect these clusters.

[3] https://censys.io

[4] https://github.com/sftcd/surveys/

TABLE I: Ports Scanned

Port	Protocol	Key Fingerprint Used
22	secure-shell	SSH host key
25	SMTP	TLS Server Public
110	Pop3	TLS Server Public
143	IMAP	TLS Server Public
443	HTTPS	TLS Server Public
587	SMTP Submission	TLS Server Public
993	IMAP	TLS Server Public

B. Steps in the Process

The following are the steps in the process run by our code, (we don't go into significant detail, as consulting the code is the better option for such):

Select addresses. The first step is to establish the set of IP addresses to scan. That can be based on a previous scan or a set of prefixes from a geo-location database. Our coordinating script ("skey-all.sh") either copies addresses from a previous scan or (via "IPsFromMM.py") uses a geo-location database and ZMap to decide which ranges to scan. Running ZMap can take from a few hours to a day depending on the number of addresses in the scan, and the available bandwidth. Typically, about 1% or so of the hosts scanned will have a port 25 listener and will therefore be retained for further scanning. From our scanning hosts, having ZMap send about 150 probes per second seems to result in few or no lost answers.

Grab. The next step is to use ZGrab to attempt to connect with the host and port in question and to record details seen, including keys and other cryptographic parameters. This is done via the "FreshGrab.py" script and takes a number of hours. Before calling ZGrab we check if the IP address being considered is correctly geo-located, as in some cases addresses may be in the "wrong" country. (This could be due to changes in routing, or due to ambiguities in how MaxMind and ZMap work.)

Analyse. Having accumulated data, we analyse the records and detect the key re-use. The "SameKeys.py" script does this, and additionally compares some of the names found (e.g. in SMTP banners or X.509 certificates) against the forward and reverse DNS.

Report We then produce graphs for each cluster using graphviz "dot" format [16] (optionally rendered as SVG files) and create a single JSON file per cluster with relevant details. This is done by the "ReportReuse.py" script.

All the steps above are run by the "skey-all.sh" shell script. Consult that script (or the "README.md" file in the code repository) for more detail. There are also some additional scripts for installing required components ("install-deps.sh") and for producing the LaTex source ("make-tex.sh") used in the results section below and for validating clusters ("check-keys.sh") as described further below.

C. Fingerprints and Clustering

We base clusters on the SHA256 [17] fingerprint of the keys, as reported by ZGrab. We also record whether e.g. TLS

server certificates are "browser-trusted" or not and some other meta-data. For TLS services, the hash input is the encoded SubjectPublicKeyInfo field of the X.509 certificate presented by the server. For SSH, we use the SSH key hash, as produced by "ssh-keygen."

We include checks for cross-protocol key re-use and do see quite a lot of that both on individual hosts and between hosts. It is not uncommon to see the same key apparently being used for port 25 on one IP address and port 443 on another. The only cross-protocol re-use that we have not yet seen is between port 22 (SSH) and other ports, though we do see many cases where SSH host keys are being re-used across multiple hosts.

Once two hosts have the same key fingerprint for any pair of ports, then we assign those hosts to the same cluster. This involves iterating over the set of records more than once, e.g., when we "join" two existing clusters having found that a host shares keys with both.

D. Scoping and Geo-location

Both censys and our local re-scans make use of MaxMind [5] for geo-location. We use the free GeoLite2 databases (for ASNs, City and Country). The "mm_update.sh" script downloads the databases needed for our scans. When using censys, we select the set of IPv4 addresses that have listeners on port 25 and that have the relevant Country Code (IE or EE) for the sample concerned. When re-scanning locally, we start with the list of country-specific prefixes from MaxMind and later discard any specific IP addresses that no longer appear to have the correct country code. Censys' geo-location appears to be more accurate than our local scans, which is unsurprising. However, some inaccuracy in geo-location doesn't affect our main conclusions with respect to key re-use.

E. Validation

In order to increase our confidence that these clusters are real, we have a validation script ("check-keys.sh") that reads individual cluster files and uses different tooling to check again if the cluster is as before. This is to guard against e.g., bugs in ZGrab or elsewhere in our code. For SSH, we use the "ssh-keyscan" binary to connect to the hosts in question and re-check the key hashes. For TLS services (all the others), we use the "openssl" binary to re-check the fingerprints.

During validation, we often see additional SSH host keys, as it appears that ZGrab (at least as we use it) finds fewer keys than ssh-keyscan. If you run the validation script from a network that e.g. doesn't allow outbound port 25 connections (which is not uncommon), then you'll get some false negatives, as the validation script won't be able to connect to port 25 on the hosts in the cluster. Similarly, if a host is not accessible at all during validation, that will show up as discrepancies that may disappear in a later run.

We do not currently re-validate all clusters as part of runs, but do that selectively when looking at individual clusters of interest.

[5] https://www.MaxMind.com/

TABLE II: A very modest scanner

Parameter	Value
Processor	AMD Opteron 62xx class CPU
CPU	25% of 1 Core
RAM	0.75 GB
Disk	7.5 GB
Bandwidth	Unlimited @10 Mbps

F. (Lack of) Infrastructure

As stated previously, being able to do local scanning using very modest "infrastructure" seems like a benefit. In this case, we did make initial use of the presumably well-engineered censys infrastructure, but, thanks to open-source technology and open databases, we are able to run our scans from an extremely modest virtual server or normal laptop.

Specifically, we currently run scans from a modest Virtual Private Server (VPS) with the parameters listed in Table II. Only the "Grab" phase of the process needs to be run on this host - at the scale of scan we're doing, the rest can be done just fine on a typical laptop.

G. Ethical Considerations

As we're doing active scans it is appropriate to consider whether there are ethical implications of this work. Given that our current scans are of hosts that listen on port 25 (i.e., email servers) we feel that these scans have fewer ethical implications than those that might involve hosts that are operated by individual users.

Given our preference for modest scanning infrastructure, our scanning rate is low enough that we are not likely to affect any running services. We use the default ZMap block list and have published a web page and DNS TXT record that can be found from the source address from which we scan. So far, nobody has asked us to not scan them, if someone did, we would add them to the ZMap block list. As stated previously, we do not intend to publish scan data, as that could assist attackers in some cases.

IV. Results

A. Reported values

For each specific run below we report the following:

- Country - IE or EE, in this article.
- Scan start - the date on which the scan was started, scans can run over multiple days.
- Scan end - the date on which the last analysis code was run - for the 2017 scans reported here this is not close to the scan start due to the time taken to develop the analysis code.
- IPs from ZMap - is the number we got from censys for 2017 scans, or the (overestimated prefixes) we got from MaxMind for 2018 scans.
- Judged "out of country" - is a count of specific IP addresses that MaxMind doesn't consider to be in the right country. Note there is the usual ambiguity here with respect to Ireland/Northern Ireland/UK here. In our scans, we don't further consider a host if MaxMind considers it

to be in the ”UK” though we could presumably extend to the island of Ireland via lat/long co-ords if those were available. For Estonia, there were quite a few addresses considered to be in Sweden, but the author is not sufficiently locally knowledgeable to know if there is any rationale behind that other than some inaccuracy in the prefixes in the geo-location database.

- No crypto seen - these are the in-country hosts that, when contacted, ran no SSH or TLS services that our scanner could detect.
- Some crypto - these hosts have at least one port (not necessarily port 25) where a server key can be detected. There is no quality judgement as to whether the specific keys are good or bad, certified or not, certificate expired or not, etc. (We do store that but it’s not reflected in this number.)
- Percent with some crypto = 100 * some-crypto / in-country
- Total crypto hosts/ports - is a count of all of the ports seen on all of the hosts that run TLS or SSH
- Total unique keys - is a count of all of the key fingerprints seen, across all hosts/ports - keys are only counted once, regardless of the number of hosts on which a key is seen.
- Percent keys vs. max - if a different key were seen on every possible host/port combination this would be 100%
- Keys only seen on one host - this is the number of hosts for whom none of their keys are seen on any other host in the run
- Hosts in clusters - this is the total number of hosts that are in some cluster
- HARK - the “Hosts Are Re-using Keys” percentage - which is 100 * hosts-in-clusters / some-crypto
- Biggest cluster size - the number of hosts in the biggest cluster
- Median and average cluster size - as you’d expect

We also present a graph showing the number of clusters of each size and the number of hosts in clusters of each size.

The specific runs documented in this article are:

1) Results of run IE-20180316-181141: Table III provides the oveview of this run. Cluster sizes are distributed as shown in Figure 1.

2) Results of run IE-20171130-000000: Table IV provides the oveview of this run. Cluster sizes are distributed as shown in Figure 2.

3) Results of run EE-20180324-214756: Table V provides the oveview of this run. Cluster sizes are distributed as shown in Figure 3.

4) Results of run EE-20171130-000000: Table VI provides the oveview of this run. Cluster sizes are distributed as shown in Figure 4.

B. Graphing

Graphs provide a sometimes useful way to visualise clusters. These were very useful during debugging when e.g., impossible asymmetries in graphs showed up problems that needed to be addressed. Simply flicking through the graph images was a useful way to spot such anomalies.

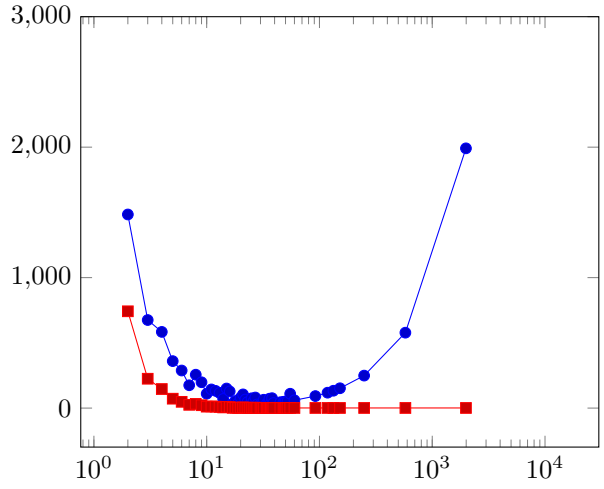

Fig. 1: Clustersize distribution for run IE-20180316-181141
circle = number of hosts in clusters of given size;square = number of clusters of given size;x = log clustersize

TABLE III: Overview of run IE-20180316-181141

Country	IE
Scan start	2018-03-16 18:11:41
Scan finish	2018-03-25 16:04:17
Number of IPs from ZMap	24774
Judged “out of county”	1233
“In country” IPs	23541
No crypto seen	5273
Some Crypto	18268
Percent with some crypto	77%
Total crypto host/ports	54447
Total unique keys	20053
Percent keys vs. max	36%
Keys only seen on one host	8570
Hosts in clusters	9698
HARK	53%
Number of clusters	1437
Biggest cluster size	1991
Median cluster size	26.5
Average cluster size	87.78

For the medium to large clusters in these runs, the more complex graphs aren’t that useful, but do at least give an impression of scale. Some few of these graphs do show some structure that may prove useful when investigating causes.

For the largest graphs, the graphviz package fails to produce any graph. The “try-render-problematic.sh” script attempts to use graphviz in various ways and does succeed in generating images for all but the largest graphs in our runs to date. Graphviz has a number of tools for rendering graphs that perform differently, some slower, making nicer graphs (e.g. “sfdp”), but failing for more graph instances, others (e.g. “neato”’) quicker and more robust, but producing less readable output in general for our graphs. All of the graphs below were produced using the “sfdp” tool, except in Figure 11 which was

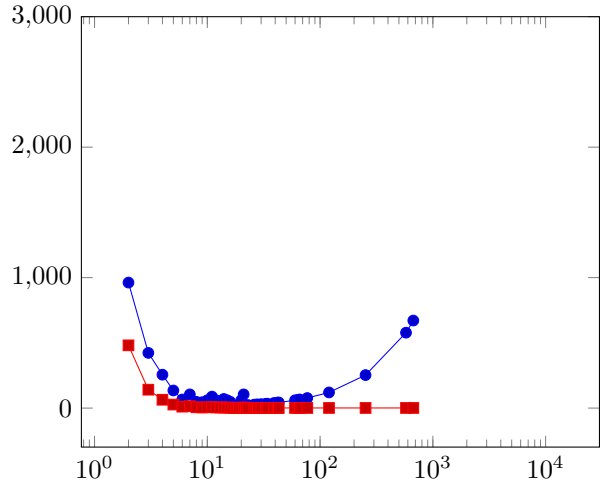

Fig. 2: Clustersize distribution for run IE-20171130-000000
circle = number of hosts in clusters of given size;square = number of clusters of given size;x = log clustersize

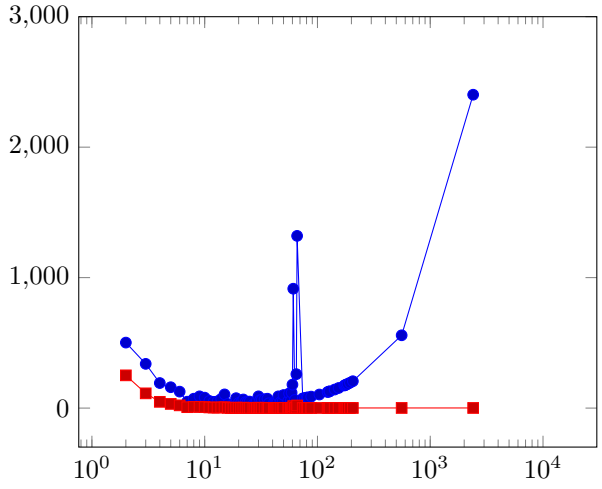

Fig. 3: Clustersize distribution for run EE-20180324-214756
circle = number of hosts in clusters of given size;square = number of clusters of given size;x = log clustersize

TABLE IV: Overview of run IE-20171130-000000

Country	IE
Scan start	2017-11-30 00:00:00
Scan finish	2018-03-27 19:15:57
Number of IPs from ZMap	23616
Judged "out of county"	0
"In country" IPs	23616
No crypto seen	13277
Some Crypto	10339
Percent with some crypto	43%
Total crypto host/ports	31832
Total unique keys	9234
Percent keys vs. max	29%
Keys only seen on one host	5406
Hosts in clusters	4933
HARK	47%
Number of clusters	815
Biggest cluster size	671
Median cluster size	21
Average cluster size	63.2308

TABLE V: Overview of run EE-20180324-214756

Country	EE
Scan start	2018-03-24 21:47:56
Scan finish	2018-03-29 01:36:07
Number of IPs from ZMap	17827
Judged "out of county"	1334
"In country" IPs	16493
No crypto seen	1519
Some Crypto	14974
Percent with some crypto	90%
Total crypto host/ports	80019
Total unique keys	20014
Percent keys vs. max	25%
Keys only seen on one host	3303
Hosts in clusters	11671
HARK	77%
Number of clusters	639
Biggest cluster size	2402
Median cluster size	42
Average cluster size	98.0435

produced by "neato" as sfdp times out in our build with that input as we impose a two-minute timeout for building each specific graph.

Figures 5 through 15 are sample cluster graphs, to give a sense of the range of clusters we see. Unless otherwise stated, all are from the IE-20180316-181141 run detailed below. **TODO: re-run check-keys.sh for any clusters shown**

The node numbers in the graphs are local indexes of the IP addresses in our data set. These are essentially determined by ZMap which hashes the set of input ranges we give it, resulting in node and cluster numbers changing (sufficiently) unpredictably for each run. The "ReportReuse.py" script that produces the graphviz dot files used to generate these graphs has a command line argument specifying whether to anonymise the IP addresses like this. Node colour is set based on the ASN of the host. Edge colours are specific to the combination of ports on which the same key is seen.

Few of these graphs display interesting structure – one that does is cluster 9 from the IE-20171130-000000 run, shown in Figure 12. The set of hosts in run IE-20171130-000000/Cluster-9 turns into run IE-20171130-000000/Cluster-333 shown in Figure. 13. One additional host is added to the cluster and some of the linkages have undergone changes.

V. Related Work

Early work surveying the use of cryptographic keys on the Internet included Heninger et al's seminal work [18] identifying re-used keys and keys with common factors. Since

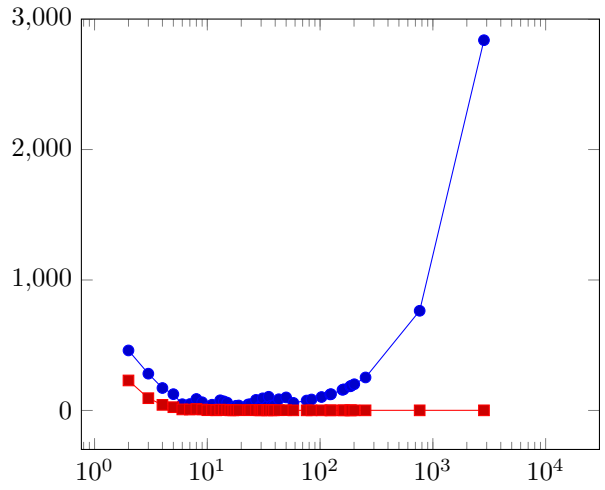

Fig. 4: Clustersize distribution for run EE-20171130-000000
circle = number of hosts in clusters of given size;square = number of clusters of given size;x = log clustersize

TABLE VI: Overview of run EE-20171130-000000

Country	EE
Scan start	2017-11-30 00:00:00
Scan finish	2018-03-28 15:04:20
Number of IPs from ZMap	12775
Judged “out of county”	0
“In country” IPs	12775
No crypto seen	997
Some Crypto	11778
Percent with some crypto	92%
Total crypto host/ports	47555
Total unique keys	11867
Percent keys vs. max	24%
Keys only seen on one host	4053
Hosts in clusters	7725
HARK	65%
Number of clusters	310
Biggest cluster size	2838
Median cluster size	29.5
Average cluster size	134.318

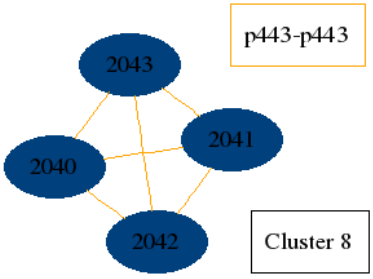

Fig. 5: Cluster 8 - a representative four host cluster where the same TLS key is used on port 443.

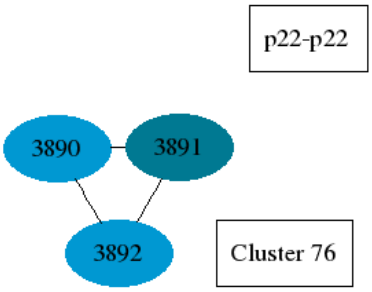

Fig. 6: Cluster 76 consists of three hosts, in two different ASes and shows SSH host-key re-use.

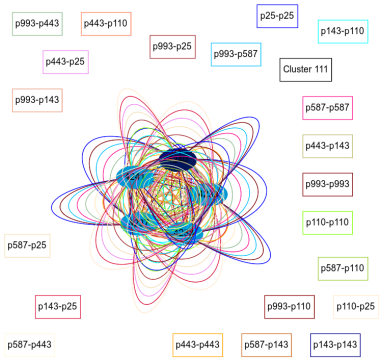

Fig. 7: Cluster 111 has five hosts that re-use the same keys for almost everything. As can be seen the individual edges become less useful at this point.

ZMap/ZGrab and censys have become available many people have studied the properties of populations of cryptographic keys for example [2], [19], [20], [3].

The properties of email security deployments have also been studied, for example, by Durumeric et al. [21] and Holz et al. [22]. Albrecht et al, carried out Internet-wide scans of SSH usage in 2015 [5] finding “about 2^{24}” servers in their scans.

To our knowledge, those and the many other studies of the TLS and SSH ecosystems have focused more on the protocol or cryptographic properties seen, and did not consider the

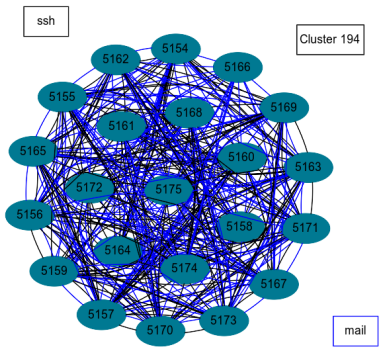

Fig. 8: Cluster 194 has twenty-two hosts that re-use lots of keys. Once there are more than 10 hosts in a cluster, we no longer distinguish re-use for the same protocol, but simply add one edge when the same mail related ports use the same keys on a pair of hosts.

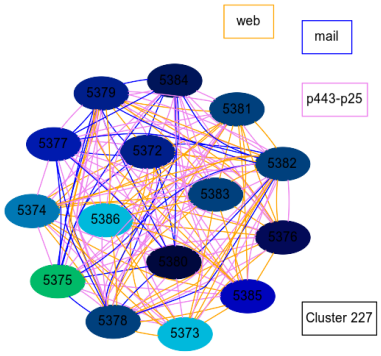

Fig. 9: Cluster 227 has fourteen hosts sharing variously over ten different ASes.

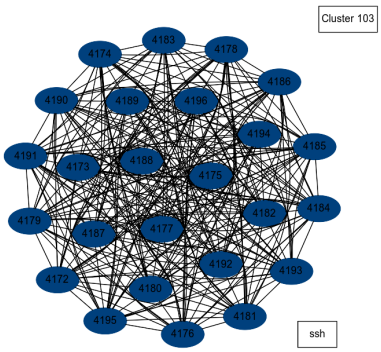

Fig. 10: Cluster 103, has twenty-four hosts sharing SSH host keys and is the largest "pure" SSH cluster in this run.

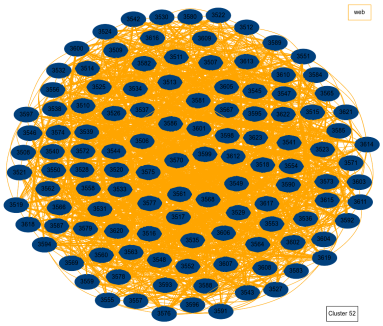

Fig. 11: Cluster 52 has one hundred and eighteen hosts sharing Web server keys and is the largest cluster that renders with graphviz for this run. There are six larger clusters.

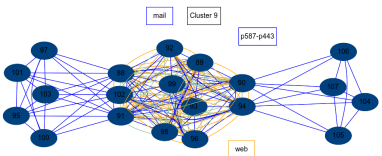

Fig. 12: Cluster 9 from IE-20171130-000000 - a rare one with some apparent structure.

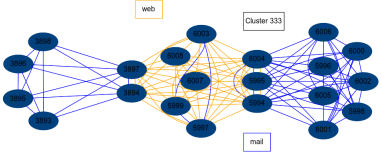

Fig. 13: Cluster 333 - Fig. 12 grown a little a few months later

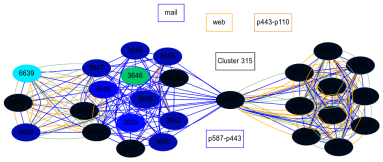

Fig. 14: The set of clusters and graphs for Estonian runs are generally graphically similar to Irish equivalents. As one example, EE-20171130-000000/Cluster315 shown here is a twenty-seven host cluster involving six ASes and various mail and web key re-uses.

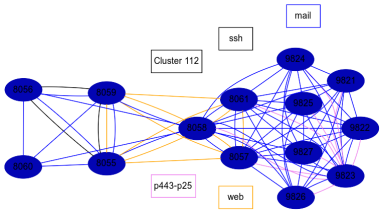

Fig. 15: Cluster 112 from run EE-20180324-214756 is a fourteen host cluster involving and various mail and web key re-uses.

clustering aspect studied here.

VI. Discussion

TODO: most content here after chat with asset holders

A. Possible Reasons for Key Re-Use

Clusters could have been created deliberately or accidentally.

The former case could be as a result of an attack or, more likely, to make someone's life easier or due to limitations in tooling that don't make key re-use apparent to administrators. Asset owners may or may not know or care about the existence of the clusters.

As of now, this section is speculative. We hope that changes as we engage with asset owners, so this section is expected to change significantly.

1) Copied configs/VMs: We expect that some of the key re-use detected here is due to config files being copied between hosts, either as part of some orchestration process, or within virtual machine images.

2) SSH Host Key Reuse scenario: One more specific way in which SSH host-key clusters could have come about is if one starts the "sshd" daemon on a host prior to creating a virtual machine image as a clone of that host, and subsequently loads that image onto a number of hosts. "sshd" will have created it's host keys before the image is created and all hosts will then have the same host keys. This won't be obvious to clients, as the host-key hash presented is perhaps unlikely to be recorded.

B. De-Clustering

If one considers these clusters undesirable, then the question arises as to how one might migrate away from large clusters. In this section we suggest ways in which administrators might move away from key re-use.

1) Use a PKI that encourages key rotation: As previously pointed out, the combination of certbot and LetsEncrypt results in keys being changed every few months, so setting up cron jobs on each host to renew certificates in that manner with LetsEncrypt or some other Certification Authority (CA) will remove re-uses fairly quickly. Note that this can as easily be done for certificates used for mail as for the web and this could go a long way to de-clustering generally. There may be a perceived downside to doing this as the certificates will likely end up in Certificate Transparency logs, however, the fact that scans such as ours can in any case see those keys and detect key re-use seems to argue that that is not a very convincing argument.

2) SSH client notification: Whether or not having SSH clients warn about the re-use of host keys would be an effective improvement is something that could be tested with systems administrators. In principle, SSH clients could warn a user that the same host key has been seen for multiple entries in the "known_hosts" file.

VII. Future work

1) Talk to asset owners: The author plans to start discussing these results with local asset holders in the immediate future and will be updating this as events warrant. But there are clearly plenty of clusters to go around, and also different locales so replication of this work would be interesting, both to validate (or falsify!) these results, but also (in the former case) to investigate whether some common local approaches to de-clustering emerge in different locales.

2) Efficiency: Whilst it isn't a problem that these scans take days, as the underlying data is unlikely to change quickly, it would nonetheless be better to improve the efficiency of the tools so they could be run on even more modest machines. At present they are fairly memory intensive and slow in parts - use of a database would likely be a significant improvement.

3) Infrastructure: Building a relatively modest server, with modest storage (a few terabytes) and excellent bandwidth would also speed up the scans. The author plans to investigate such infrastructure locally in Ireland and would be happy to help anyone else who wanted to do that instead, or as well, in some other locale.

4) Other Populations: The populations we have scanned to date are geographically bounded. It would also be interesting to scan sets of related hosts, e.g. belonging to the same sector or making use of the same technologies. The scanning tools used here could also be used within enterprise networks to check internal and externally visible hosts.

5) Check the rest of the Internet: While the local clusters are interesting, it seems like an obvious extension to check if keys being re-used locally are also used elsewhere. Starting from locally detected clusters may be a useful way to approach that at Internet-scale.

6) Check in CT: We have not, but could, check for additional information based on searching Certificate Transparency (CT) logs [23], for example the crt.sh[6] web interface offered by Commodo does allow searching based on the TLS fingerprints we find. We have not (yet) done the work to develop an application to use a CT API for the populations we have scanned, but have verified that at least some of our fingerprints do occur in CT logs.

7) Slowly doing more runs: Another obvious thing to do is to run this for other locales, and to extend to other geographic scopes, e.g. city or regional scale.

8) IPv6: Investigating the IPv6 addresses associated with the names detected here could also be of interest, if it extends any of the clusters. (Generic IPv6 scanning is of course of interest but not as a specific extension of this work.)

9) Mitigations and Incentives: Whilst discussing with asset holders, it will be interesting to investigate better mitigations and the incentives that might motivate administrators to do better in this space. For the former, one could speculate that administrative tools may be making accidental key re-use too likely and run-time/monitoring tools are presumably not looking out for key re-uses.

10) Key Rotation: Even if a host starts out as part of a cluster, it ought be a normal part of applications using crytpography to periodically rotate to new key pairs. That should result in clusters being broken up relatively quickly. TLS server certificates using the LetsEncrypt (tm) CA and certbot do seem to default to changing keys during certificate renewal which is also relatively frequent. If CAs had policies that called for key rotation, or notified key holders that key re-use has been detected, that could be a significant help in breaking up clusters.

11) Longitudinal studies: Running cron jobs with these scans over time may produce interesting results in terms of how clusters form, live and (hopefully) dissipate.

12) Better Metrics: While the HARK metric is simple and easy to understand, it does not capture the changes in risk possibly associated with cluster sizes and density. It could be interesting to investigate whether metrics used for other clusters of risk (e.g. health related metrics) could provide more accurate metrics. And of course, determining whether or not any metric related to these clusters is useful in practice will also be useful.

VIII. Conclusion

The HARK numbers were a surprise to the author. One conclusion is that doing measurement is a good, perhaps especially when researchers first dip a toe in these waters with a background that others haven't previously brought to the space.

More on topic, though, the clusters seen here do seem to indicate some failings in key management, possibly due to a mixture of technology limitations and operators still being less familiar with managing keys at scale.

In the end though - rotate the keys!

Acknowledgements

Initial data on which this survey was built was made available at no cost to the author as a researcher by Censys.io. Thanks for that.

[6] https://crt.sh/

Thanks to all those who attended the September 2017, “responsible” workshop [7], the discussion at which provided the inspiration for this work.

REFERENCES

[1] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–104, Aug. 2008, updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5246.txt

[2] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, “Measuring https adoption on the web,” in *26th USENIX Security Symposium*, 2017, pp. 1323–1338.

[3] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz, “Mission accomplished?: Https security after diginotar,” in *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017, pp. 325–340.

[4] T. Ylonen and C. Lonvick (Ed.), “The Secure Shell (SSH) Transport Layer Protocol,” RFC 4253 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–32, Jan. 2006, updated by RFCs 6668, 8268, 8308, 8332. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4253.txt

[5] M. R. Albrecht, J. P. Degabriele, T. B. Hansen, and K. G. Paterson, “A surfeit of ssh cipher suites,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1480–1491.

[6] S. Farrell and H. Tschofenig, “Pervasive Monitoring Is an Attack,” RFC 7258 (Best Current Practice), RFC Editor, Fremont, CA, USA, pp. 1–6, May 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7258.txt

[7] S. Farrell, “Why pervasive monitoring is bad,” *IEEE Internet Computing*, vol. 18, no. 4, pp. 4–7, 2014.

[8] V. Dukhovni, “Opportunistic Security: Some Protection Most of the Time,” RFC 7435 (Informational), RFC Editor, Fremont, CA, USA, pp. 1–11, Dec. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7435.txt

[9] D. Eastlake 3rd, “Transport Layer Security (TLS) Extensions: Extension Definitions,” RFC 6066 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–25, Jan. 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6066.txt

[10] M. Nottingham, P. McManus, and J. Reschke, “HTTP Alternative Services,” RFC 7838 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–20, Apr. 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7838.txt

[11] M. Nottingham and E. Nygren, “The ORIGIN HTTP/2 Frame,” RFC 8336 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–11, Mar. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8336.txt

[12] H. Bck, J. Somorovsky, and C. Young, “Return of bleichenbacher’s oracle threat (robot),” Cryptology ePrint Archive, Report 2017/1189, 2017, https://eprint.iacr.org/2017/1189.

[13] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–151, May 2008, updated by RFC 6818. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5280.txt

[14] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, “A search engine backed by internet-wide scanning,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 542–553.

[15] Z. Durumeric, E. Wustrow, and J. A. Halderman, “Zmap: Fast internet-wide scanning and its security applications.” in *Usenix Security*, 2013, pp. 605–620.

[16] E. Gansner, E. Koutsofios, and S. North, “Drawing graphs with dot,” 2006.

[17] D. Eastlake 3rd and T. Hansen, “US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF),” RFC 6234 (Informational), RFC Editor, Fremont, CA, USA, pp. 1–127, May 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6234.txt

[18] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, “Mining your ps and qs: Detection of widespread weak keys in network devices.” in *USENIX Security Symposium*, vol. 8, 2012, p. 1.

[19] L. Valenta, D. Adrian, A. Sanso, S. Cohney, J. Fried, M. Hastings, J. A. Halderman, and N. Heninger, “Measuring small subgroup attacks against diffie-hellman.” *IACR Cryptology ePrint Archive*, vol. 2016, p. 995, 2016.

[20] M. Hastings, J. Fried, and N. Heninger, “Weak keys remain widespread in network devices,” in *Proceedings of the 2016 Internet Measurement Conference*. ACM, 2016, pp. 49–63.

[21] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman, “Neither snow nor rain nor mitm...: An empirical analysis of email delivery security,” in *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 27–39.

[22] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar, “Tls in the wild: An internet-wide analysis of tls-based protocols for electronic communication,” *arXiv preprint arXiv:1511.00341*, 2015.

[23] B. Laurie, A. Langley, and E. Kasper, “Certificate Transparency,” RFC 6962 (Experimental), RFC Editor, Fremont, CA, USA, pp. 1–27, Jun. 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6962.txt

[7] https://responsible.ie/

Stephen Farrell is a research fellow at Trinity College Dublin, from which he received his PhD in 2008. His research interests include security and privacy and communication in unusual and stressed environments. Between 2011 and 2017, Stephen was Internet Engineering Task Force (IETF) security area director. Stephen is a senior technical advisor for M3AAWG and a co-founder of Tolerant Networks Limited.