Clusters of Re-used Keys

Stephen Farrell,
Trinity College Dublin
stephen.farrell@cs.tcd.ie

Article version 0.4, Compiled on 2018/04/20 at 10:00
work-in-progress

***Abstract*—This article describes a survey of long-term cryptographic public keys observed in deployments of secure-shell, e-mail and web protocols in four similarly-sized countries – Ireland, Estonia, Finland and Portugal. 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. If two IP addresses share a 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. Scans in the other countries demonstrate the same issue. 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 in discussons with some local (Irish) asset-owners to try establish the reasons for key sharing and to possibly assist with improving network posture, and will continue to incorporate resulting findings in revisions of this article.**

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

I. Introduction

This article describes six scans of populations of hosts on the Internet, two each in Ireland (IE) and Estonia (EE) and one each in Finland (FI) and Portugal (PT). The hosts in question offer some mail service, that is, those hosts 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 and Autonomous Systems (ASes) in these scans, Figure 1 is one of the more structured clusters seen in these scans.

At the time of writing the author is discussing these findings with relevant 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.[1][2], [3] The Secure-Shell (SSH) [4] protocol has also been nearly ubiquitously 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 achieve all the 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 see many keys re-used in clusters of hosts. A **cluster** is the largest set of IP addresses in a scan population such that each host in the cluster shares at least one public key 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 service for 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 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]https://letsencrypt.org/stats/#percent-pageloads has recent graphs.

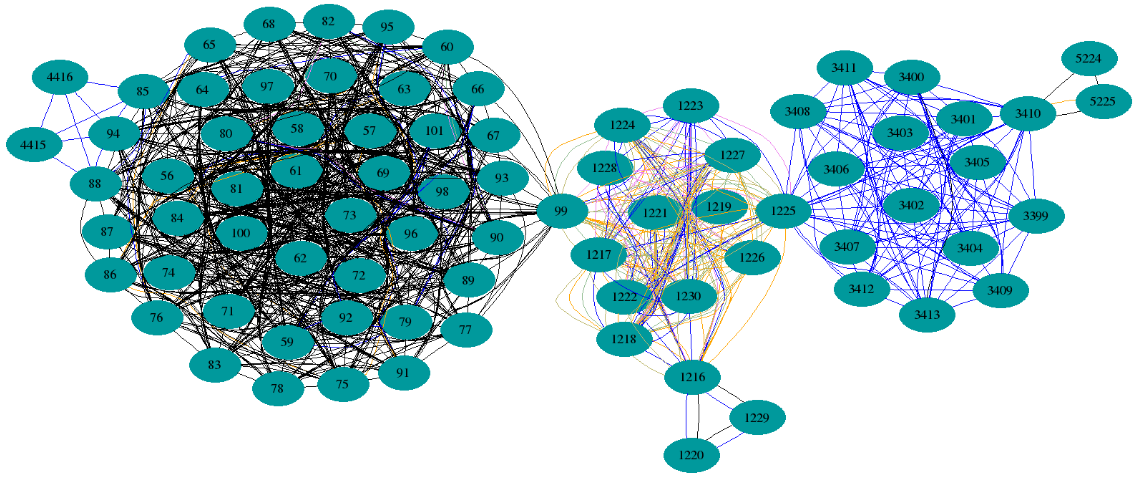

Fig. 1: Cluster 835 from run FI-20180326 is an 80 host cluster with many key re-uses and interesting structure. Nodes represent hosts. Node colour represents AS – this cluster has hosts in only one AS. Edges represent pairs of ports where the same key is used by both hosts. Edge colour reflects the combination of ports as per Figure 4. Black edges are SSH, blue are mail protocols and orange is the web. Section V has more detail of graphs.

to re-use the same key pair, however, it is certainly arguable that an attacker woh manages to gain access to a host with software-only key storage should be considered to have access to all the key pairs on that host.

If a single (physical, "bare metal") host has multiple interfaces, each with it's own IPv4 address, then it will show up here as multiple hosts. The number of hosts of this kind in our scanning populations is unknown, but it is certainly reasonable to "re-use" keys in such cases.

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 physical hardware that is 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.

Sometimes people might mirror two hosts for redundancy, including the keys used. If the private keys on both instances are considered to only be at risk from the same entities (e.g. both are hosted by one entity), then this scenario also seems defensible. If two different hosters were used for better redundancy, then having the same private key at risk to two entities seems less defensible.

There could be cases where a Hardware Security Module (HSM) that holds keys in secure storage is used by different hosts (say in one rack) and where the HSM for some reason cannot operate with different keys for every host. In such cases, re-using keys that never leave the HSM might be defensible. In other cases, a rack might be in a less physically trusted environment, creating a need for remote access to a HSM, which in turn might affect the HSM:host ratio making key re-use more likely.

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, having the same default key pair for port 443 on multiple hosts can be problematic.

Of course, simply using different keys for each service instance does not in itself imply that 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*

There are clear potential dangers associated with key re-use. Clearly, any legitimate holder of a private key has the capability to attack other cluster members sharing that key, and our results do show cases where clusters appear to contain hoosts belonging to very different entities. Some speciifc additional risks include:

Leaks: Where private keys are stored as files on disk, a leak from one host may affect all cluster members. 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. While the details of this argument won't always apply, it seems reasonable that risk should increase with cluster-size faster than linearly under most threat-models.

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.

Misdirected Mail: If an attacker can manipulate mail routing (via MX resource records) or routing information (via BGP), and the attacker has a copy of relevant private keys, then the attacker can masquerade as the mail recipient's domain and intercept email, even if the recipient's domain enforces strong authentication of mail transport, e.g., via MTA-STS. [10]

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 passwords are frequently re-used in many places, this attack may extend beyond the cluster to other hosts at which those credentials can be used.

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 [11], the ORIGIN frame [12] and secondary certificates [13], in HTTP version 2, breach of one web server in a cluster will allow masquerading as any other, all from the breached host.

Key Transport: If an RSA key has ever been used for key transport, and an attacker has a record of such sessions, then breach of any host using that key allows the attacker to directly decrypt all recorded sessions.

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 [14]), can likely be exploited more efficiently and more stealthily, if there are more service endpoints to which the attacker can send messages.

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

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[15]). 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.

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.

Data recovery@ It appears that keys and addresses "move" between clusters (see Section VI-A. If data recovery tools are accessible to e.g., a virtual machine, it could be that keys belonging to others could be recovered. While this risk isn't directly caused by key re-use, it's impact could be increased.

Laziness: Re-use of keys can 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.

[2] https://robotattack.org/

TABLE I: Ports Scanned. (2017 scans did not include port 587).

Port	Protocol	Key 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

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. One might think of this as wondering if small to middling sized data might better enable researcherss to gain insight, compared to big data.

We define a metric related to key re-use: the percentage of hosts in a population doing cryptography using keys that are known to be used for multiple hosts; versus hosts where we only see keys being used for one IP address. (Of course, hosts could be mis-identified as not re-using keys, even if some key is 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

In November 2017, we extracted scans of Irish (IE) and Estonian (EE) hosts that listen on the standard Simple Mail Trasfer Protocol (SMTP [16]) port 25 (and hence offer some mail service) from the censys infrastructure. [17][3] Subsequently, in March/April 2018, we ran scans locally using ZMap/ZGrab [18] for Ireland, Estonia, Finland and Portugal, but still limited to hosts who listen on port 25 (according to ZMap).

Our expectation was that hosts running SMTP could be a useful population to examine as they would hopefully have a better than average probability of being well-managed. Initial examination of the 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 and/or TLS details for each of the ports listed in Table I. (Censys scans did not include port 587 in November 2017.) We wrote analysis code to identify and analyse the clusters of hosts re-using keys as described below.

[3]https://censys.io

All source code required to replicate this study or do a similar-scale scan has been published[4] under the MIT license. Our scans typicaly require a few days to run on our very modest infrastructure, due to limits of that infrastructure and built-in (default 100ms) delays between scan stages to require less bandwidth. About 3-6GB of storage is required per run. Our tools take as input a country-code (e.g. IE) and should work for any country with a similar number of port 25 listeners.

A speedier or much 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 smal-scale local scans can be (more) effective in helping asset owners mitigate risks, so we're sanguine about speed and scale limitations, and happy to demonstrate that quite limited resources are sufficient.

Data from scans are not being published as doing so could assist an lateral movement, or attract attackers to clusters. However, note that, as shown by this article, it is relatively simple to detect these clusters.

A. Scanning Process

The following are the steps in our scans, (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 probed 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. We analyse the records to detect key re-uses. The script that does this "SameKeys.py" also compares the names found in SMTP banners or X.50 certificate subject and SubjectAltName (SAN) fields against the forward and reverse DNS.[5]

Report We produce graphs for each cluster using graphviz "dot" format [19] (optionally rendered as SVG files) and create a JSON file per cluster containing 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 validating clusters ("heck-keys.sh") as described further below.

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

[5]We only query up to 20 SANs per certificate - in runs we have seen some gigantic certificates with more than 1500 SANs - querying each would be too time consuming for the benefit gained - 20 SANs should be enough to identify any asset-owner, which is why we're interested in SANs in this study.

B. Fingerprints and Clustering

We base clusters on the SHA256 [20] fingerprint of public keys, as reported by ZGrab. We also record whether e.g. TLS server certificates are "browser-trusted" (which can be expired) 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 see quite a lot of that both on individual hosts and between hosts. It is not uncommon to see the same key 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., to "join" two existing clusters having found that a host shares keys with both.

C. Scoping and Geo-location

Both censys and our local re-scans make use of MaxMind[6] for geo-location. We use the 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 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.

D. Other Tooling

There are some additional tools in the code repo in the "clustertools" directory – the "ipoverlaps.sh" script compares two runs and says which clusters overlap with which, see Section VI-A for discussion of the output of this tool. The "fpsfromcluster.sh" script shows host many occurrences of each fingerprint are seen in a cluster. The "check-no-ssh-cross-protocol.sh" script checks for any cases where an SSH host-key is used for a TLS port - no such case has been seen in

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

```
$ check-keys.sh -i cluster835.json
Running check-keys.sh at 20180402-141156
Starting at 20180402-141156, log in
   validation-results-20180402-141156.out
Doing cluster835.json
infile,ipcount,22count,matches,mismatches
835,80,66,1042,0
infile,ipcount,tlscount,matches,mismatches
835,80,140,544,41
```

Fig. 2: Validation of cluster 835 (Figure 1)using "check-keys.sh". This show no discrepancies for SSH host-keys but 41 mismatches out of 585 pairs of ports for TLS. Most mismatches are due to a lack of response but there were also four real key changes.

the five runs reported here. The "ClusterGetCerts.py" script makes a fresh connection to the TLS ports from a cluster file and extracts the X.509 certificates seen in text form. There is also a script for producing the LaTex source ("make-tex.sh") used in the results section below. Additional tools may be added as the analysis proceeds.

E. Validation

In order to increase our confidence that these clusters are real, we have a validation script ("check-keys.sh") that reads a list of cluster files and uses different tooling to check if the cluster is as before. This is to guard against e.g., bugs in ZGrab or in our clustering 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 (in "s_client" mode) to re-check 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 also see some real discrepancies for some clusters, but that is to be expected, e.g., due to key rotation for browser-trusted certificates that have expired.

We do not re-validate all clusters as part of runs, but do that selectively when looking at individual clusters of interest. A validation run of Cluster 835 (shown in Figure 1), produced the output shown in Figure 2.

F. (Lack of) Infrastructure

As stated previously, being able to do local scanning using very modest "infrastructure" seems like a benefit. 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.

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

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 or for 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. In communications to date with autonomous system asset-holders, we anonymise IP address and name information that involves other asset-holders, e.g. when a cluster has members from multiple ASes. We do however include the AS numbers of other hosts in clusters shared with the asset-holder in question, as they may already have relevant eontacts.

IV. Related Work

Early work surveying the use of cryptographic keys on the Internet included Heninger et al's seminal work [21] identifying re-used keys and keys with common factors. Since ZMap/ZGrab and censys have become available many people have studied the properties of populations of cryptographic keys for example [2], [22], [23], [3].

The properties of email security deployments have also been studied, for example, by Durumeric et al. [24] and Holz et al. [25]. 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 clustering aspect studied here.

V. Results

Table III provides the oveviews of each of the six runs done for this article. Figure 3 shows the cluster size distributions for

TABLE III: Overview of runs

Country (year)	IE(2017)	IE(2018)	EE(2017)	EE(2018)	FI(2018)	PT(2018)
Scan start	2017-11-30	2018-03-16	2017-11-30	2018-03-24	2018-03-26	2018-04-03
Scan end	2018-04-15	2018-03-25	2018-04-14	2018-03-29	2018-04-01	2018-04-05
IPs from ZMap	23616	24774	12775	17827	37012	19782
"out of county"	0	1233	0	1334	506	63
"In country" IPs	23616	23541	12775	16493	36506	19719
No crypto seen	12959	5273	796	1519	26106	4169
Some Crypto	10657	18268	11979	14974	10400	15550
Some crypto%	45%	77%	93%	90%	28%	78%
Total crypto host/ports	25935	54447	45067	80019	34263	63907
Total unique keys	12889	20053	15502	20014	11686	12202
Percent keys vs. max	49%	36%	34%	25%	34%	19%
Hosts with only local keys	5651	8570	3176	3303	4675	4143
Hosts in clusters	5006	9698	8803	11671	5725	11407
HARK	**46%**	**53%**	**73%**	**77%**	**55%**	**73%**
Number of clusters	823	1437	521	639	1029	1512
Max cluster size	671	1991	2874	2402	373	2016
Median cluster size	21	26.5	36	42	24	30
Average cluster size	63.23	87.78	121.18	98.04	50.65	117.51

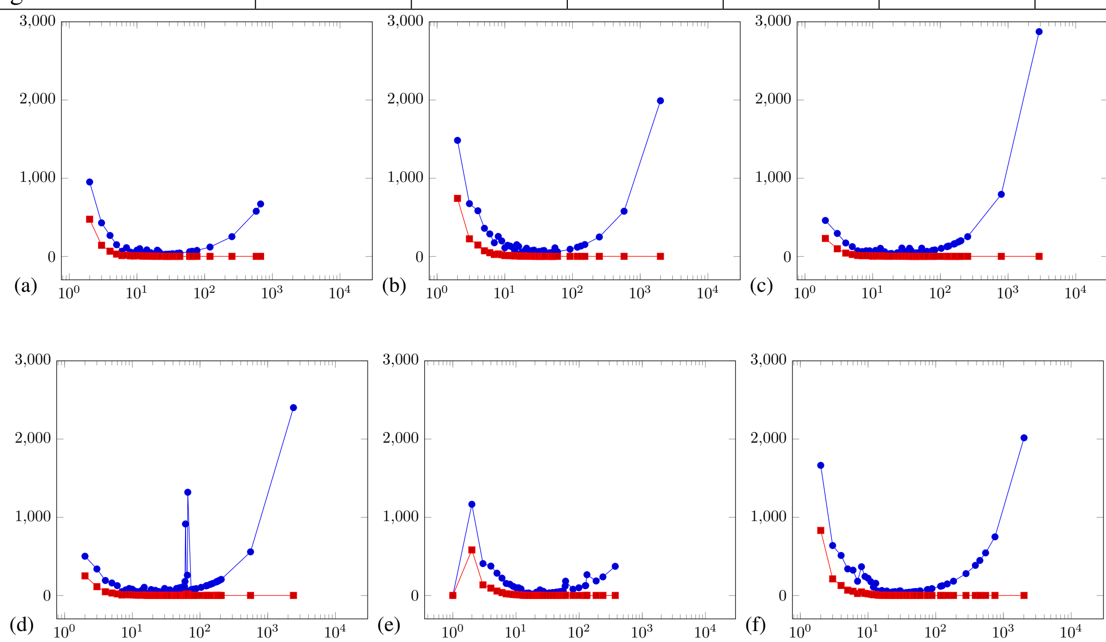

Fig. 3: Clustersize distributions for runs (a) IE-20171130, (b) IE-20180316, (c) EE-20171130, (d) EE-20180324, (e) FI-20180326, (f) PT-20180403. Blue circles show the number of hosts in clusters of given size, red squares reflect the number of clusters of given size. The x-axis is logarithmic.

these runs.[7]

For each specific run below we report the following:

Country: IE, EE, FI, or PT in this article.

Scan start/end dates: Scans take multiple days. For the 2017 scans here the latency is due to the time taken to develop the analysis code.

IPs from ZMap: the number we got from censys for 2017 scans, or the addresses from MaxMind for local scans.

Judged "out of country": the number of addresses that MaxMind doesn't consider to be in the right country. There is the usual ambiguity here with respect to Ireland/Northern Ireland/UK. but we don't consider hosts MaxMind says are in the "UK." 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 inaccuracy in the prefixes from the geo-location database.

No crypto seen: hosts that ran no SSH or TLS services our scanner could detect.

Some crypto: hosts that have at least one port (not necessarily port 25) where a server key can be detected. There is no quality judgement as to whether keys are good or bad, certified or not, certificate expired or not, etc.

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.

% keys vs. max: if a different key were seen on every possible host/port combination this would be 100%

Hosts with only local keys: the number of hosts such that none of their keys are seen on any other host in the run

Hosts in clusters: the number of hosts that are in some cluster

HARK: the "Hosts Are Re-using Keys" percentage = $100 * hosts - in - clusters/some - crypto$

Max, Median and Average cluster size: as you'd expect

A. Graphing Clusters

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. Flicking through the graph images was a useful way to spot such anomalies. Figures 5, 6, 7 and 8 are sample cluster graphs from scans, to give a sense of the range of clusters we see.

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 could prove useful when investigating causes, e.g. Figure 1.

For the largest graphs, the graphviz package fails to render the graph as an image. 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 12 which was 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.

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 (suffi-

[7]Fixes for some bugs in analysis code caused small changes to the 2017 run figures between versions 0.3 and 0.4 of this article.

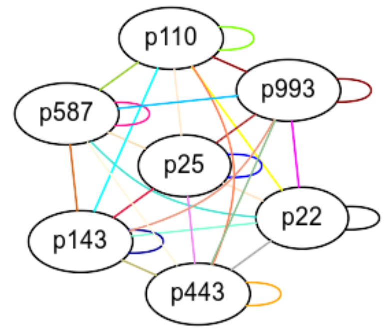

Fig. 4: Legend for edge colours in graphs. Nodes are port numbers, edge colours are as used between host-nodes in other graphs when the pair of ports re-use the same key. The loopback colours (e.g. black from p22 to itself) are used when a key is re-used on the same port on two hosts.

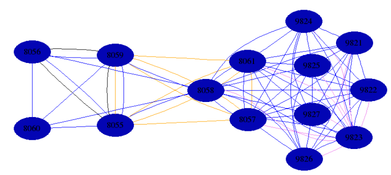

Fig. 5: Cluster 112 from run EE-20180324 is a 14 host cluster with mail and web key re-uses.

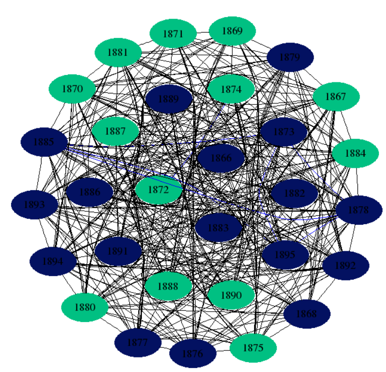

Fig. 6: Cluster 60 from run FI-20180326 is a 30 host cluster involving mostly SSH host-key re-uses, but with some re-use of key on mail ports. 13 of the hosts are in one AS, and 17 in another.

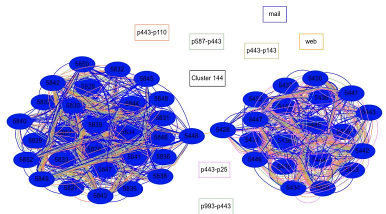

Fig. 7: Cluster 144 from run PT-20180403 is a 48 host cluster with two clear “lobes”

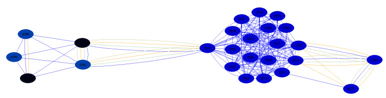

Fig. 8: Cluster 240 from run FI-20180326 is a 25 host, 2 AS, cluster with mail and /web key re-uses.

ciently) 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 - the colours used are as shown in Figure 4.

Once there are more than 10 hosts in a cluster, we no longer distinguish re-use for the same protocol via different edge colours, but simply add one edge when the same mail related ports use the same keys on a pair of hosts. So if two hosts in a larger cluster use the same key for both ports 25 and 143, then only one edge will be created on the graph for that cluster and that edge will be coloured for “mail.” Even in such larger graphs, if two hosts share a key on different ports, e.g. port25 on one host has the same key as port 143 on the second host, then, as these situations are less common, we continue to add edges for those, and the graph will have a ”p25-p143” entry in the legend.

Relatively few of these graphs display interesting structure – one that does is cluster 10 from the IE-20171130 run, shown in Figure 20. The set of hosts in run IE-20171130/Cluster-10 turns into run IE-20171130/Cluster-333 shown in Figure 19.

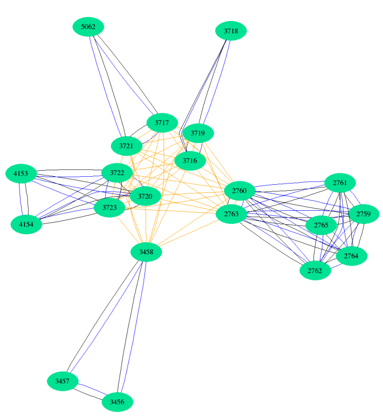

Fig. 9: Cluster 355 from run FI-20180326 is a 21 host cluster with various mail/web key re-uses and some apparent structure.

One additional host is added to the cluster and some of the linkages have undergone changes. Cluster 355 from the FI-20180326 run, shown in Figure 9, also shows some interesting structure as does cluster 835 (Figure 1) from the same run.

B. *Cross-border Overlaps*

We check for fingerprint overlaps between the clusters found in different runs using the “fpoverlaps.sh” script. We find overlaps as shown in Figure 10. Note we only checks hosts for which key re-use has already been detected within a run – additional re-uses may exist that are not detected by this script.

Overall we see 89 cases of cross-border cluster overlap. One of those is between IE-20180316/227 (a 15-host, 10 AS cluster, Figure 21), EE-20180324/165 (a 2-host, 2 AS, cluster) and FI-20180326/688 (a 2-host cluster) and PT-20180403/1184 (a 5-host, 4 AS cluster). This seems to be related to common software or hardware as one vendor’s name is mentioned in banners for all four clusters.

C. *Word Clouds for Clusters*

While cluster graphs help to understand the scale and possible structure of a cluster, if we aim to understand more we need to delve into the details as to which hosts, belonging to which asset-holders, are sharing keys. The cluster details stored include names from AS names, banners, reverse DNS, and X.509 certificates, but it can be tedious to extract that data from whatever bulk format (in our case JSON files) in which it is stored. We have experimented with generating a word-list for each cluster, based on the names we find, repeated as often as they are in the underlying cluster data. From those word-lists, we build a word cloud image.[8] Doing that for all clusters and flicking through the images has been useful for identifying some asset-holders for local (Irish) clusters, but hasn’t been useful for clusters from other locales. Local knowledge therefore does seem to help in identifying some asset-holders – whether that results in much change is still to be determined. The word-list and word-cloud images are built using the “wordle.sh” script. If we get permission from some asset-holder to include an example, we will, but for now, we don’t as that would be identifying information.

D. *Communicating with Asset-Holders*

We have developed tooling, “ah-tb.sh” and “ClusterAnonOthers.py”, to assist with communication with asset-holders. The former creates a directory containing the set of cluster files corresponding to an input regular expression and the latter anonymises a set of cluster files, except those from one specified AS. Anonymous records have an IP address of ”XXX.XXX.XXX.XXX” and no names from banners, certificates or DNS. We do not anonymise the AS of the anonymised IP addresses.

To date, our practice has been to communicate with asset-holders via an existing contact, and to then try funnel our

[8]https://en.wikipedia.org/wiki/Tag_cloud

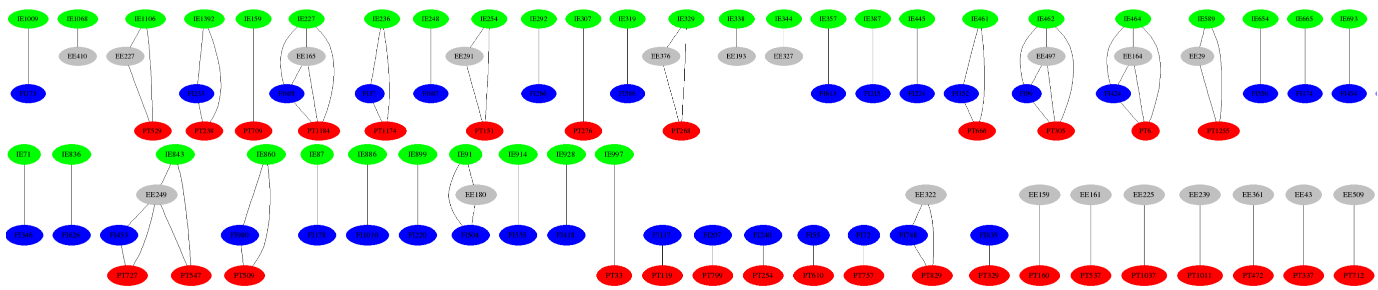

Fig. 10: Cross-border Cluster Links. Nodes represent clusters, edges are present when there is at least one fingerprint in common between the clusters. (green=IE-20180316;blue=FI-20180326;gray=EE-20180324,red=PT-20180403)

TABLE IV: TLS certificate details for keys from clusters in run IE-20180316. The rightmost column is the number of listeners on that port who are members of clusters. Recall there are 9,698 hosts in clusters in this run.

Port	Browser Trusted	Wildcard Cert	Listeners
25	294	173	4277
110	311	49	3998
143	328	59	4109
443	396	241	7331
587	307	0	3321
993	345	0	4024

results into whatever process suits the asset-holder. In all but one case so far we have started by contacting local ASes. (The exception being an educational institution where we had contacts already.) It's likely we may move on to contacting the owners of hosts in clusters directly but time will tell.

VI. Irish Results

In this section we report in more detail on the Irish runs, with additional detail of selected clusters.

Of the 1,437 clusters seen in the IE-20180316 run, 129 (9%) invlolved more than one AS. The use of less desirable ciphersuites is as shown in Table V. Recall that RSA key transport combined with many copies of a private key is a bed combination. (RC4 is independently bad.)

Information on the certificates seen on TLS ports is shown in Table IV. In terms of the threat against web origins mentioned in Section II, there are 5205 hosts with browser-trusted certificates that are in clusters in this run. The max cluster size (only considering port 443) is 1991, next is 171, then 152. Cluster 32 (see Figure 13 seems like a case in point where apparently different origins are being used "behind" a single wild-card certificate. We see 25 cases where a key used on port 443 on more than one host sometimees is presented via a browser-trusted certificate but also sometimes presented without such a certificate. We see one case where a key has different sets of names associated with the same key in different browser-trusted certificates.

Of the 684 keys that are shared on more that one host in this run, the average number of hosts per key is 7.81 (median is 2, max is 1991, then 183 etc.).

TABLE V: Less Desirable Cipherstuites used in IE-20180316

Keys	Code	Ciphersuite Name
2	x0A	TLS_RSA_WITH_3DES_EDE_CBC_SHA
86	x35	TLS_RSA_WITH_AES_256_CBC_SHA
87	xC011	TLS_ECDHE_RSA_WITH_RC4_128_SHA
1818	x2F	TLS_RSA_WITH_AES_128_CBC_SHA
2806	x05	TLS_RSA_WITH_RC4_128_SHA

A. *Time Evolution*

Clusters evolve over time. The primary attributes of clusters are key fingerprints and IP addresses that can change independently, under the contol of whomever administers a host. Keys can be rotated (as we'd suggest), or new re-uses can be seen. IP addresses can be re-purposed, or remain stable. And of course, our clusters are sets, so membership can change based on key and/or IP address changes. (We could, but have so far not, extend our concept of cluster evolution to encompass naming, based on all the usual forms of name.)

The "dot-r1r2.sh" script analyses two runs to produce this analysis. For the two Irish runs, (20171130 and 20180316), we examined the changes in the "forward" (from 2017 to 2018) and "reverse" directions with the overall results as per Table VI.

The "disappeared" category covers clusters that we no longer see at 20180316. The "appeared" category are those created (or first seen) in the 20180316 run. The "IP-linked" category are clusters that share some IP address(es) with exactly one cluster in the other run, but have no keys fingerprint in common. The "FP-linked" are clusters that share key fingerprints with one cluster in the other run, but have no IP addresses in common. The most common linkage are the"IP and FP" category that have at least one IP address and at least one key fingerprint in common (not necessarily, but commonly, on the same IP address). This is what one would expect if nothing changes between runs. The "complex" category covers relationships that are more complicated - Figure 11 shows the clusters in this category.

B. *Specific Clusters*

We now describe a selected subset of the clusters seen in more detail. We do not yet attempt to present a fully consistent level of description of each cluster – we may evolve towards

TABLE VI: Cluster evolution - from IE-20171130 to IE-20180316.

Category		Category	
Disappeared	168	Appeared	777
IP-linked	36	FP-linked	16
IP and IP-linked	584		
Complex-20171130	19	Complex-20180316	24

that, or it may be that the aspects of interest for each cluster can only be established via inspection, in which case different descriptions for each are a reasonable outcome. We hope to learn more as we discuss these clusters with asset-owners.

1) IE-20180316/Cluster-3:

- This cluster has 1991 hosts and is the largest in the run. All the IP addresses are in the same ASN, which is one of the world's hyper-scalers who happen to host machines in Ireland, so this cluster doesn't seem to be particularly associated with Ireland.
- There are a total of 1995 host/port combinations. 1991 of those use the same key with browser-trusted certificates for port 443. A CT search via crt.sh for that SPKI hash seems to indicate that key has been used since late 2014. The other 4 use one key that is not browser-trusted for port 25.
- A run of "check-keys.sh" shows that the p25 keys appear to have changed.
- Hosts in the cluster do appear to be listening on port 25 - 3 randomly selected hosts from the cluster all accepted connections from telnet on port 25.
- 1972 of the hosts have the same public key certificate, which is a wild-card certificate. It may be be that these hosts are used for marketing related services.
- There are 1980 different SMTP banners. For each one, it appears that the hostname in the banner is used as the hostname for a DNS name that matches the wildcard certificate. So if the SMTP banner is "foo" and the certificate is for "*.example.com" then we see a DNS entry for "foo.example.com" as one might expect.
- Many of the banner hostnames are of the form "[company]-mkt-dev1-1" or "[company]-mkt-prod2" There are 293 different company names in the set, some of which are internationally known brand names, and only a few of which are related to Ireland. An HTTP GET to port 443 on one of those hosts returned a 404. We didn't further investigate the web site content.
- All TLS host/port combinations report use of the TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 ciphersuite,[9] so there may not be a current concern with RSA key transport.
- The same keys, banners and other names were also see in the IE-20171130 run. At that time there were 871 members of this cluster.

2) IE-20180316/Cluster-665:

[9]See https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4

- This cluster has 578 hosts and is the second largest in the run. The hosts in this cluster are in five ASes. The number of hosts in each the these ASes is 433, 111, 17, 15 and 2. The ASN with 433 hosts is a well-known local hoster/registrar. The ASN with the 111 hosts is a local ISP/hoster. The ASNs with 17 and 15 hosts are international with a local presence. The ASN with 2 hosts is a local Internet/computing consultancy.
- 517 of the TLS server certificates seen are browser-trusted whereas 2400 are not. There are a total of 3405 host/port combinations with crypto. 1298 of those use the same key. The next most commonly used key is used for 570 ports, then 375, 219, 86, and so on down to 2 - there are 65 keys used twice. There are 480 keys used only on one port.
- There are 457 reverse DNS names associated with this cluster and 20 unique names from TLS certificates - two for tool names, three as wildcards, i.e. of the form "*.example.com" and 15 for what appear to be regular hostnames one might find in DNS.
- The names from banners and certificates may imply that default keys for a particulat hosting control panel tool are being re-used. Forum posts for that tool do appear to indicate that controlling these settings is considered challenging.

3) IE-20180316/Cluster-9:

- This cluster has 249 hosts and is the third largest in the run. All 249 hosts in this cluster are in the same AS, which is a local hosting company.
- 96 of the TLS server certificates seen are browser-trusted whereas 1052 are not. There are a total of 1396 host/port combinations with crypto. 911 of those use the same key. The next most commonly used key is used for 90 ports, then 51, 18, 18, and so on down to 2 - there are 9 keys used twice. There are 145 keys used only on one port.

4) IE-20180316/Cluster-52:

- This cluster has 118 hosts and is the seventh largest in the run. See Figure 12.
- All 118 hosts in this cluster are in the same AS, which is a hyper-scaler with a local presence.
- All 118 of the TLS server certificates seen are browser-trusted and use the same certificate, issued in mid-2016.
- There are a total of 118 host/port combinations with crypto.
- The name associated with these keys sounds like it's related to mail delivery. But who knows?
- The certificate for these hosts includes a name like "example.com" with a SAN for "www.example.com" but there is not A record for the relevant www.example.com in the DNS today

5) IE-20180316/Cluster-32:

- This cluster has 92 hosts. See Figure 13. All 92 hosts in this cluster are in the same AS, which is a hyper-scaler with a local presence.
- There are a total of 165 host/port combinations with crypto. 70 of the 74 port 443 keys are the same, and map to one wildcard certificate. "check-keys.sh" shows

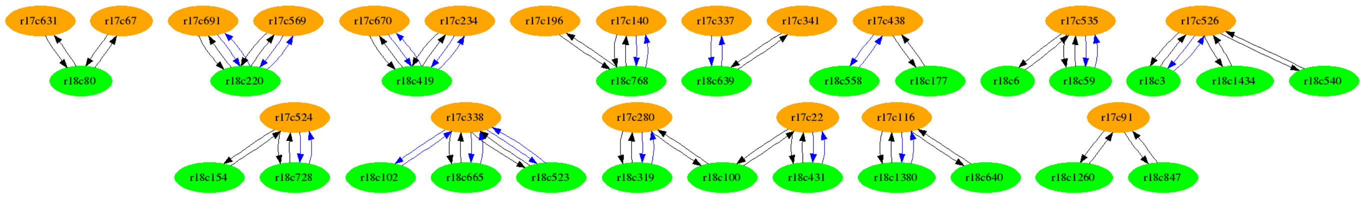

Fig. 11: The clusters with "complex" time evolution between runs IE-20171130 (orange) and IE-20180316 (green). Blue arrows show key fingerprint linkage, black arrows show IP linkage.

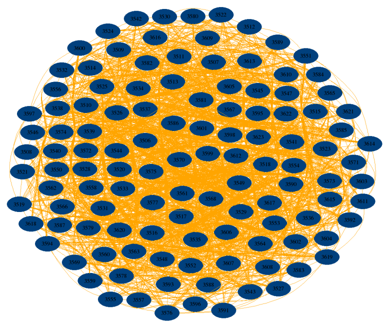

Fig. 12: Cluster 52 has one hundred and eighteen hosts sharing Web server keys and is the largest cluster that renders with graphviz for this run. There are six larger clusters.

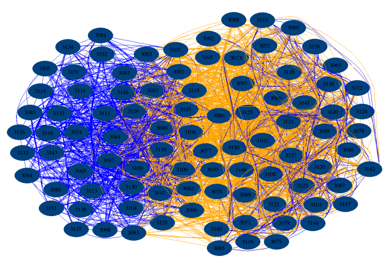

Fig. 13: Cluster 32, has 92 hosts sharing web and mail host keys.

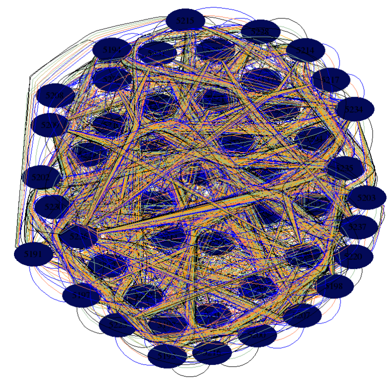

Fig. 14: Cluster 199 has 48 hosts

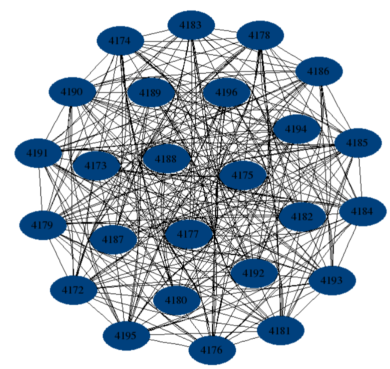

Fig. 15: Cluster 103, has twenty-five hosts sharing SSH host keys and is the largest "pure" SSH cluster in this run.

some discrepancies. This cluster has no SSH keys. The TLS discrepancies do include some key changes, with 4 new key fingerprints being seen on a number of the cluster's hosts.

- This cluster features about 50 different DNS names, mostly below the same .co.uk 2LD, but with the leftnmost label reflecting different organisations. These names may relate to a UK based mobile web application development company, so the use of one wildcard certificate could be undesirable.

6) IE-20180316/Cluster-199:

- Cluster 199 has 48 hosts who make lots of use of one key. See Figure 14.
- All 48 hosts are part of a small local telco ASN. (With which the author was unfamiliar.)
- There are 285 host/port combinations, 231 of which use the same key. 48 of those use another key, and then 5 ports use unique keys. 232 ports use browser-trusted certificates, whereas 5 do not.
- check-key.sh produced no discrepancies.
- All but one of the SMTP banners use the same hostname. The one exception appears appears to be a certificate with a SAN for what could be a .ie second-level domain (2LD) except that that value is not registered. The rest have names of the form "cpanel.[foo].com".

7) IE-20180316/Cluster-103:

- This cluster has 25 hosts and is the largest "pure-" SSH cluster n the run. See Figure 15. All 25 hosts in this cluster are in the same AS, which is a hyper-scaler with a local presence.
- There are a total of 25 host/port combinations with crypto.
- "check-keys.sh" shows some discrepancies. The discrepancies all seem to relate to hosts not responding, except for one host that does appear to now be using different host-keys.

8) IE-20180316/Cluster-144:

- This cluster has 26 hosts and is one of two matching the

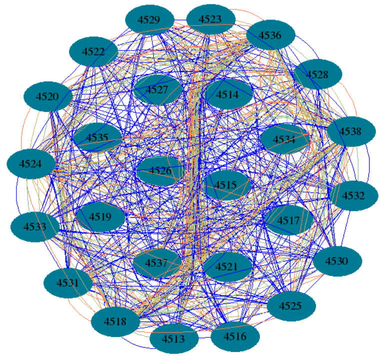

Fig. 16: Cluster 144, has twenty-six hosts.

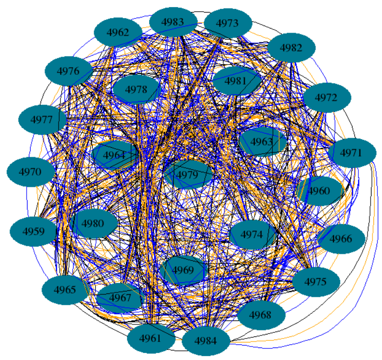

Fig. 17: Cluster 177, has twenty-six hosts.

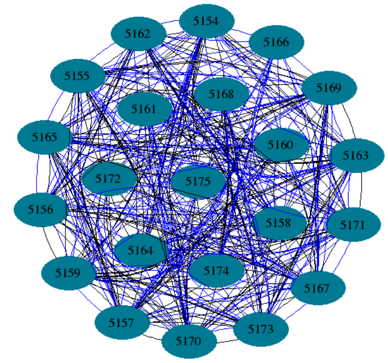

Fig. 18: Cluster 194 has twenty-two hosts that re-use lots of keys.

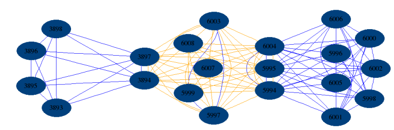

Fig. 19: Cluster 333 - Fig. 20 grown a little a few months later

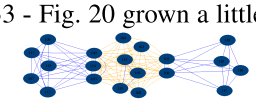

Fig. 20: Cluster 10 from IE-20171130.

median cluster size See Figure 16.

- All 26 hosts in this cluster are in the same AS, which belongs to a local ISP.
- There are a total of 125 host/port combinations with crypto. 109 of those use the same key. Each of the others has a unique key. The most-used key is used mail ports (110,143,587,993) on all 26 hosts. That same key is used for port 443 on 5 hosts. 16 other hosts use a unique key for port 443.
- "check-keys.sh" detected no mismatches - all 125 host/ports remain the same

9) IE-20180316/Cluster-177:

- This cluster has 26 hosts and is one of two matching the median cluster size See Figure 17.
- All 26 hosts in this cluster are in the same AS, which belongs to a local ISP.
- There are a total of 128 host/port combinations with crypto. Three keys are used for all of these, one 78 times, one 26 times and one 24v times.
- "check-keys.sh" shows some change on port 443 since the test run.

10) IE-20180316/Cluster-194:

- This cluster has 22 hosts. See Figure 18.
- All 22 hosts in this cluster are in the same AS, which belongs to a local ISP.
- There are a total of 121 host/port combinations with crypto. One key is used for 88 of those, one key for 21, and 12 keys are used for one port each. For 96 ports, a browser-trusted certificate is used, 4 ports use a certificate that is not browser-trusted.
- "check-keys.sh" shows a number discrepancies for SSH, but none for mail or web.
- An interesting variety of names are used for SMTP banners. Some relate to bitcoin, others to finance, and more to gaming.
- A certificate associated with the most-used key was issued in late 2017.

11) IE-20180316/Cluster-333:

- Cluster 333 has 21 hosts. See Figures 19 and **??**.
- All hosts are within a hyper-scaler's local AS.
- There are 135 host/port combinations, with one key used 47, 26, 15, 10, 5 and 2 times each, and with 30 other keys used for one port each. 7 ports use certificates that are browser-trusted, 109 ports do not. One of the browser-trusted certificates uses a key that has been in use since 2014, across multiple certificates/CAs, another uses a key certified only once so far in early 2018, a third was issued in March 2018, and for a Dutch-sounding .eu DNS name not reflected in the SMTP banners.
- check-key.sh shows up some mail and web port discrepancies, with cases of hosts not answering and of changed fingerprints. All of the SSH ports were the same, about 20% of the mail and web ports showed discrepancies.
- SMTP banners are mixed. There are 21 different values, 19 with a common 2LD, which seems to be that of a Dutch Internet consultancy, but with two more outside that namespace. Of those, one has a .nl 2LD that matches the hostname used in some of the 19 banners, the last one seems to have no relation to the others.
- A search at crt.sh shows certificates for this public key dating back to late 2014.

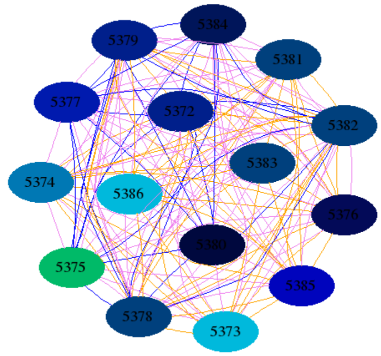

Fig. 21: Cluster 227 has 15 hosts sharing variously over ten different ASes.

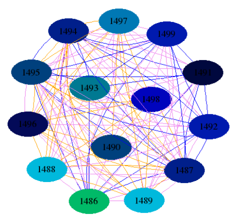

Fig. 22: Cluster 150 from run IE-20171130 overlaps cluster 227.

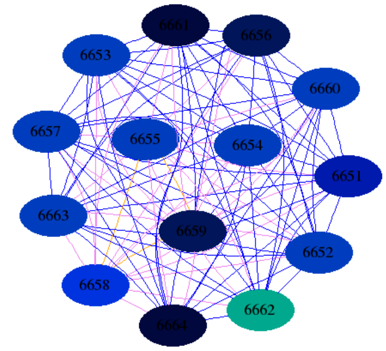

Fig. 23: Cluster 462 has 14 hosts sharing variously on ports 25 and 443 web over 6 different ASes.

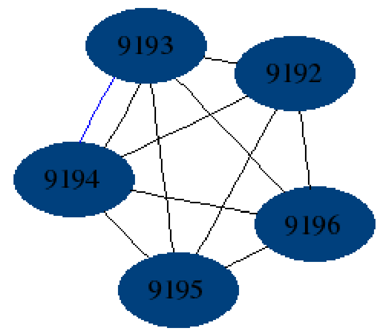

Fig. 24: Cluster 1227 has five hosts that re-use the same SSH host-keys, and two of whom share a key for port 25.

- We saw this cluster as cluster 9 in our IE-20171130-000000 scan.

12) IE-20180316/Cluster-227:

- Cluster 227 has 15 hosts. See Figures 21 and 22. 10 different ASes are involved in this cluster, of many kinds, from hyper-scalers, through local ISPs to small local hosters and even the local NREN.
- There are 20 host/port combinations, 19 of which use the same key. The odd one out is a port 443 listener. The others are for ports 25 or 443. One of those certificates is browser-trusted and is for a medium-sized food processing company in Ireland.
- SMTP banners have names that vary widely.
- check-keys.sh reports about 50% mismatches, mostly it seems due to inability to connect to some hosts.
- Since run IE-20171130, this cluster has grown in size by one host, but two others have changed, i.e., since then, two hosts have dropped out, and three have joined. (Assuming hosts just weren't unavailable at the wrong moment.) This cluster also overlaps with clusters in other runs - see Section V-B for details.

13) IE-20180316/Cluster-462:

- Cluster 462 has 14 hosts. See Figure 23. 6 different ASes are involved in this cluster, mainly local ISPs.
- There are 16 host/port combinations, 15 of which use the same key. The most used key is a 1024 bit RSA key.
- The SMTP banners vary but generally name Irish organisations.
- check-keys.sh reports mismatches on port 443, but none on port 25.
- The certificate used on port 25 in most cases has a SAN of the form "[vendor-appliance] Demo Certificate." It is not known if these hosts are offering a live service or are just used for testing. In at least one case the IP address of the HOST matches the MX address for the domain as per the SMTP banner seen, which implies these keys are in live use.

14) IE-20180316/Cluster-1227:

- Cluster 1227 has 5 hosts. See Figure 24.
- All hosts are within a single AS operated by a hyper-scaler
- There are 13 host/port combinations, 5 are the shared SSH host-keys. Two are the shared port 25 keys. 6 other unique keys are used for port 25 and port 443 on the hosts that don't share a port 25 key. The three port 443 certificates are browser-trusted, no other certificates are browser-trusted.
- check-key.sh reports no discrepancies.
- There are four different SMTP banners, that appear unrelated. The two identical banners are on the hosts that share a key for port 25.

15) IE-20180316/Cluster-111:

- Cluster 111 has 5 hosts who make lots of use of one key. See Figure 25.
- Four hosts are within a local hoster's AS. One is within a local telco/ISP's AS.
- There are 31 host/port combinations, all but one use the same key which is used on ports 110, 143, 443, 587 and 993. The odd one out is the only SSH host-key in use.
- check-key.sh says all keys remain the same.

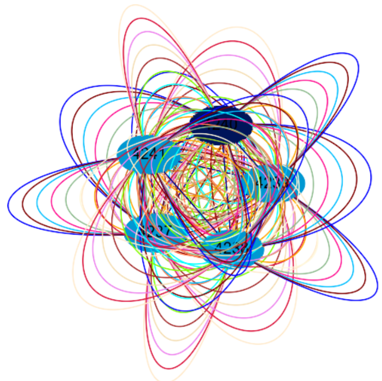

Fig. 25: Cluster 111 has five hosts that re-use the same keys for almost everything. As can be seen the individual edges become less useful at this point.

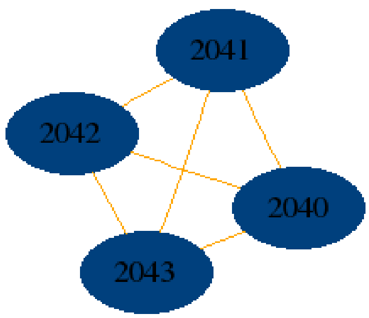

Fig. 26: Cluster 8 - a representative four host cluster where the same TLS key is used on port 443.

- Each host has an SMTP banner of the form "[foo].example.com" where foo is either "server1" (one occurrence) or "hosting2" (four occurrences), whilst the (same) public key certificate for each host is for "*.example.com".
- A search at crt.sh shows certificates for this public key dating back to late 2014.

16) IE-20180316/Cluster-8:

- Cluster 8 has 4 hosts who share one key for port 443. See Figure 26
- All hosts are within one AS operated by a hyper-scaler.
- There are 9 host/port combinations, 4 use the same key, all for port 443. Each host has a unique key for port 22. One of the hosts has a unique key for port 25. The port 443 keys are browser-trusted, the port 25 key is not.
- check-key.sh says there are 3 discrepancies - one of the hosts seems to be unresponsive.
- Each host has an SMTP banner of the form "foo.example.com" whilst the (same) public key certificate for each host is for "*.example.com". A search at crt.sh shows only one matching certificate issued in mid 2017.

17) IE-20180316/Cluster-76:

- Cluster 76 has 3 hosts who share SSH host-keys. See Figure 27.
- Two of the hosts are within one local AS operated by a hosting company. Another is within an AS operated by a local ISP/hoster.
- Each host only does crypto on port 22.
- check-key.sh says all 3 keys remain the same.
- There are two SMTP banners with names that don't appear to obviously match. One is a DNS bogon.

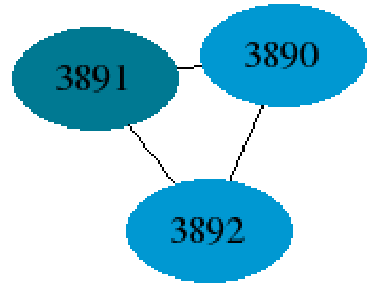

Fig. 27: Cluster 76 consists of three hosts, in two different ASes and shows SSH host-key re-use.

18) IE-20180316/Cluster-648:

- This cluster has 2 hosts who share an SSH host-key and is the 3rd smallest (file) in the run.
- The SSH host-keys are 1024-bit RSA, so presumably quite old.
- Both hosts are in an ASN operated by the local NREN. Reverse DNS indicates they belong to a 3rd level educational institute.
- Neither host has any other cryptographic ports.
- The SMTP banner indicates a name that matches one of the hosts. Reverse DNS for that host also matches the banner. The other host has
- "check-keys.sh" reports both keys are still active.

19) IE-20180316/Cluster-804:

- This cluster has 2 hosts who share an SSH host-key and is the 2nd smallest (file) in the run.
- Both hosts are in an ASN operated by the local NREN, in a range that appears to be for non-universities.
- Neither host has any other cryptographic ports.
- "check-keys.sh" reports only one of those keys are still the same at the time of writing. That could be some Firewall/Intrusion Detection System (IDS) behaviour, ssh-keyscan results for one of the hosts varies depending on whether one uses names or IP addresses, and IPv4 or IPv6. Manual inspection shows that the host keys remain the same.
- Neither IP is in our IE-20171130-000000 scan.

20) IE-20180316/Cluster-639: This is a two-host cluster where a single key is used for mail and web services. The same two IPs are also cluster IE-20171130/341 but with entirely different keys and even with some port differences (SSH is absent in IE-20180316/639 but present in IE-20171130/341). Yet cluster IE-20171140/337 shares keys with IE-20180316/639, though again for a different set of ports. The IPs from IE-20171130/337 do not show up at all in the IE-20180316 run. The AS for all four IPs is a local hoster.

21) IE-20180316/Cluster-1389:

- This cluster has 2 hosts who share an SSH host-key and is the smallest (file) in the run. Both hosts are in an ASN operated by a major multinational, and hoster. Reverse DNS suggests these hosts are in the ranges used for the multinational's hosted customers.
- Neither host has any other cryptographic ports.
- "check-keys.sh" reports those keys are still the same at the time of writing.
- Neither IP is in the IE-20171130 run.

22) Summary of Clusters of Size 2: There are 742 clusters of size 2. In total, there are 170 different combinations of ports seen in this set of clusters, of those:

- 150 only involve common port 25 keys.
- 143 only involve common port 443 keys.
- 41 only involve common keys for both ports 25 and 443 (one unique key per host).
- 27 only involve common SSH host-keys.
- 24 only involve independently common keys for ports 25 and 443 (two unique keys per host).

VII. Discussion

Clusters could have been created deliberately or accidentally. The former case could be as a result of an attack or, more likely, is the result of limitations in tooling that don't make key re-use apparent to administrators, or that make it easy to end up re-using keys. Asset owners may or may not know or care about the existence of clusters of key re-use.

A. Confirmed Reasons for Key Re-Use

In this section we describe causes for key re-use that have been confirmed by asset-holders. Note that we simply accept asset-holders' assertions to this effect - attempting to validate for specific clusters would likely be too intrusive.

SSH Host Key Reuse: Some SSH host-key clusters were caused by starting the "sshd" daemon (thus generating the SSH host key pair) on a host prior to creating a virtual machine image as a clone of that host, so that subsequent loads of that image onto another of host caused the re-use. This is not obvious to clients, as the host-key hash presented is perhaps unlikely to be recorded. This was confirmed for cluster IE-20180316/Cluster 35. The asset-holder in that case now has new scripting so that cluster sholdn't grow further and may disappear.

One system administrator using Puppet[10] had installed the SSH server on a base system used to generate images. Even though the Puppet SSH module was activated on the imaged hosts, the original SSH host key from the image was already in place, and so there was no requirement for the Puppet agent on the image to generate one. The fix was to delete the existing SSH host key from all imaged hosts, and either generate a new image-specific host key manually (or just let the Puppet agent do it in due course). The base image was then modified appropriately to prevent the situation described arising again.

Copied configs/VMs: Some of the key re-use detected here is due to VM cloning with config files or templates being buggy or re-used. Speaking to asset holders, this seems to be an issue that has come up in the past,[11] been fixed, but that tends to recur, perhaps indicating some tooling changes could be useful.

Mirrored Hosts: One asset-holder contacted the author describing a mirroring arrangement as outlined in Section II-A. Presumably this kind of mirroring is the cause for a number of the clusters of size two and perhaps even three.

B. Possible Reasons for Key Re-Use

As of now, this section is speculative. As discuession with asset-holders continues, we hope to move points here to the section above.

Old/New Hosts: It could be that some of the size-two clusters represent hosts where one is a "new" version of the other, but where the administrators forgot to turn off or re-configure the old machine after setting up the new.

Anycast: If a service uses anycast addressing, it is possible that multiple instances of the service will have been deployed with the same keys. In such cases, scans such as ours are likely to see those instances via unicast addresses so see this as a key re-use across hosts. It is unclear why it might be desirable for anycast services to re-use SSH or TLS server keys, unless clients for those services are pinned to the specific host keys, which would be a fairly brittle design."

Bad Random Number Generators: It is possible that some of the key re-uses detected here are the result of bad random number generators causing the same keys to be generated at different times and places.

Software/Hardware with Hard-coded Keys: If some software package or piece of equipment ships with hard-coded or default keys, then that would clearly show up in our results.

Test Equipment left Running: If test equipment is installed with default or test keys, and never updated to use real keys then key re-use is highly likely. Some of the SMTP banners we see in results seem to imply that this may have occurred.

Large scale use of wildcard certificates: Some of the clusters we've seen are re-using a private key where the public key is in an X.509 wildcard certificate that is being used on many hosts. There seem to be at least two different cases here: a) where the hostnames covered by the wildcard all seem to relate to one organisation and b) where the hostnames seem to map to many different organisations, e.g. hostnames of the form of "[customer].example.com" – in the latter case, key re-use may be more risky, if the different customers are not expected to be mutually trusting.

Mega-SANs: Cluster 46 (with 12 hosts) in the PT-20180403 run has a certificate with 1,577 SANs for port 443. While that particular key does not appear to be re-used on other hosts, other (mail related) keys on that host are. That seems like quite a concentration of risk.

Cross-Border Clusters:

More investigation of the clusters shown in Figure 10 is needed. It seems fairly sure (but not yet fully-confirmed) that software-related issues are more likely to give rise to such

[10] https://puppet.com/

[11] https://technodrone.blogspot.ie/2013/01/the-ssh-key-problem-with-cloned-linux.html

overlaps, whereas many configuration related issues are more likely to give rise to clusters that don't cross borders or ASes so much. For example, clusters 464, 462 and 227 each give a very strong impression of being caused by products that ship with default keys - it is hard to envisage how key re-use across so many ASes and locales could happen otherwise, and each of those clusters prominentely mentions a well-known vendor name.

C. 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.

Measurement: If the adage "you can't manage what you don't measure" is considered to have some validity, then presumably a first step in de-clustering could be to monitor for the existence of key re-use. Systems administratation and monitoring tools could relatively easily integrate the kind of key re-use detection described here, and thereby enable administrators to decide to take action (or not) when they see unexpected re-use.

Regular Key Rotation: The most basic way to avoid being a part of one of these clusters is likely to be to simply rotate all cryptographic keys at some frequency that is acceptable for the application context. This can of course cause problems if e.g. hashes of keys have been stored in applications. However, not rotating keys brings with it the significant risk that e.g. former employees may continue to have access to systems in contravention of local policy, so the pain is likely worthwhile.

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 imply that that is not a very convincing argument.

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.

VIII. Future work

Talk to Asset Owners: The author has begun discussing these results with local (Irish) asset-holders 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.

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.

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.

Other Populations: The populations scanned to date are geographically bounded and all run SMTP listeners. 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.

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.

Check in CT: We have not, but could, check for additional information based on searching Certificate Transparency (CT) logs [26], for example the crt.sh[12] 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.

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.

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.)

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.

Key Rotation: Even if a host starts out as part of a cluster, it ought be a normal part of applications using cryptography 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

[12] https://crt.sh/

clusters.

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

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) might be more meaningful. And of course, determining whether or not any metric related to these clusters is useful in reducing risk would be a fine thing.

IX. 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. 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 by Censys.io. Thanks for that. Thanks to all those who attended the September 2017, "responsible" workshop[13], the discussion at which provided the inspiration for this work. Thanks to: David Malone for help with understanding some clusters; Mike Bishop for the point about HSMs remote from the rack. Thanks also to the asset-holders who co-operated in finding reasons for these results, but who would prefer to remain anonymous.

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] D. Margolis, M. Risher, B. Ramakrishnan, A. Brotman, and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)," Internet Engineering Task Force, Internet-Draft draft-ietf-uta-mta-sts-15, Apr. 2018, work in progress. [Online]. Available: http://tools.ietf.org/html/draft-ietf-uta-mta-sts-15

[11] 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

[12] 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

[13] M. Bishop, N. Sullivan, and M. Thomson, "Secondary Certificate Authentication in HTTP/2," Internet Engineering Task Force, Internet-Draft draft-ietf-httpbis-http2-secondary-certs-00, Dec. 2017, work in progress. [Online]. Available: http://tools.ietf.org/html/draft-ietf-httpbis-http2-secondary-certs-00

[14] 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.

[15] 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

[16] J. Klensin, "Simple Mail Transfer Protocol," RFC 5321 (Draft Standard), RFC Editor, Fremont, CA, USA, pp. 1–95, Oct. 2008, updated by RFC 7504. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5321.txt

[17] 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.

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

[19] E. Gansner, E. Koutsofios, and S. North, "Drawing graphs with dot," 2006.

[20] 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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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

[13] 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.