Clusters of Re-used Keys

Stephen Farrell,
Trinity College Dublin
stephen.farrell@cs.tcd.ie
Article version 0.8, Compiled on 2018/06/09 at 22:24
work-in-progress

***Abstract*—We survey the long-term cryptographic public keys, (for SSH, e-mail and HTTP protocols), on hosts that run the SMTP protocol in ten countries. 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. When 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 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 re-use can create undesirable security and privacy dependencies between cluster members. A range of causes for key sharing have been confirmed, including multi-homed hosts, mirroring, large-scale use of wildcard public key certificates, cloning virtual machines that already contain host keys and vendors shipping products with hard-coded or default key pairs. Discussions with local (Irish) asset-owners to better understand the reasons for key re-use and to possibly assist with improving network posture are ongoing, and we 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 scans of populations of hosts on the Internet, two each in Ireland (IE) and Estonia (EE) and one each in Finland (FI) Luxembourg (LU), Uraguay (UY), New Zealand (NZ), Namibia (NA), Singapore (SG), Slovenia (SI) and Portugal (PT). The hosts in question offer some mail service, that is, these 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 those hosts on port 25 and other commonly used ports. We see unexpectedly large-scale re-use of keys across clusters of hosts and Autonomous Systems (ASes). Figure 1 shows 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 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]. 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 generally used for host authentication, though historically some may also have been used directly for RSA key transport. Re-using the same keys for many different instances of services can create vulnerabilities or increase the probability or impact of some attacks. Re-use can however 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 services for which that private key is used.

A. Safe Key Re-Use

Not all re-use of keys is bad. There are situations where it is reasonable to re-use a key pair for different services, 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 is reasonable to use one key pair to protect services on

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

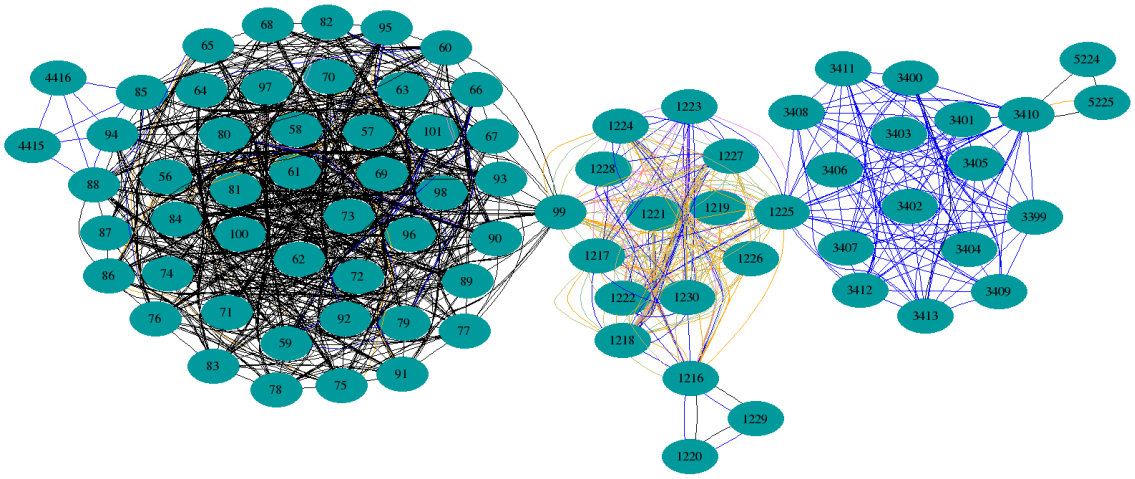


Fig. 1: Cluster 835 from run FI-20180326 is an 80 host cluster with many key re-uses and interesting structure. Nodes represent hosts. Node colour represents AS – this cluster has hosts in only one AS. Edges represent pairs of ports where the same key is used by both hosts. Edge colour reflects the combination of ports as per Figure 6. Black edges are SSH, blue are mail protocols and orange is the web. Section V has more detail of graphs.

say port 587 (email submission) and port 25 (email submission and transfer). Where services on the same host are less related, for example, web and mail service, it seems a little less reasonable to re-use the same key pair, however, it is certainly arguable that an attacker who 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 network interfaces, each with it's own IPv4 address, then it will appear as multiple hosts and so that host's addresses will show up as a cluster in our scan. The number of hosts of this kind in our scans is unknown but discussion with Irish hosters has confirmed this as non-negligible.

There are also cases where re-using a key across multiple hosts can be defended. For example, if the set of hosts for which a key is re-used are virtual machines (VMs) that are, and always will be, running on physical hardware 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 may be cases where a Hardware Security Module (HSM) 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, for example, when a TLS Server Name Indication [9] for the intended name/application is provided. Given our scans are 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.

From Internet vantage points however, observers cannot distinguish between any of the above and less desirable cases.

B. Dangers of Key Re-Use

Simply using different keys for each service instance does not mean that service is more secure, better managed, nor that it is independent of other hosts. However, there are 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 contain hosts belonging to very different entities. Some specific additional risks include:

Leaks: Where private keys are stored as files on disk, leaks are quite possible and a leak from one host can affect all cluster members. If we notionally say the cost of a leak of any of the 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 cluster member is $n * p$. If those considerations do apply then the risk of the cluster scenario would 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 an attacker to masquerade as any other host from the cluster in question.

Misdirected Mail: If an attacker can manipulate mail routing (via MX resource records or BGP), and has the 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 masquerade as any of the hosts in a cluster enables an active attacher to capture user credentials, for example passwords sent in IMAP or SMTP transactions. Even worse, if SSH password logins are 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 [11] then the ability to masquerade as any host in the cluster would allow theft of HTTP cookies, breaking the same origin policy on which the web depends. Given new web technologies such as alt-svc, [12] the ORIGIN frame [13] and secondary certificates [14] in HTTP version 2, breach of one web server in a cluster may allow masquerading as any other, all from the one 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 [15]), 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 the corresponding public key certificate ought be revoked if the public component is part of some Public Key Infrastructure (PKI).[16] 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 private key compromise, then some hosts may remain at risk even after revocation of some certificates.

Theory: It is now feasible to do security proofs for realistic protocols, and indeed TLS1.3 has been the subject of such formal studies. [17], [18] 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 IP addresses "move" between clusters (see Section VI-A). If data recovery tools are available in a VM, keys belonging to others could be recovered. While this risk isn't directly caused by key re-use, the impact and success probability may be increased.

[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

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 unwise 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. One might think of this as wondering if small-to-middling sized data might better enable researchers 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.) We 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 useful 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 Transfer Protocol (SMTP) [19] port 25 (and hence offer some mail service) from the censys infrastructure.[3] [20] Subsequently, in March/April 2018, we ran scans locally using ZMap/ZGrab [21] for Ireland, Estonia, Finland and Portugal, but still limited to hosts who listen on port 25 (according to ZMap). Later, we added scans for the additional countries shown in Table IV. For each port-25 listener seen we record SSH and/or TLS details for 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.

Our expectation was that hosts running SMTP could be a useful population to examine as they would hopefully have a

[3] https://censys.io

higher 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.

All source code required to replicate this study or do a similar-scale scan has been published[4] under the MIT license. Our scans typically require hours to a few days to run due to the limits of our systems and built-in delays between scan stages to consume 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 in 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 co-ordinating script ("skey-all.sh") either copies addresses from a previous scan or (via "IPsFromMM.py") uses a geo-location database 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, less than 1% of the addresses probed will have a port 25 listener and will therefore be retained for further scanning. From our scanning host, (see Table II) 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 to the host and port in question and to record details seen, including public key fingerprints and other parameters. This is done via the "FreshGrab.py" script and takes a number of hours or even days. 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 an outdated geo-location database, changes in routing, or for some other reason, but doesn't affect our main findings.)

Analyse: We analyse records to detect key re-uses by iterating through the stored key fingerprints. The script that does this is "SameKeys.py". We also check the names found in SMTP banners, X.509 certificate subject and SubjectAltName (SAN) fields against the forward and reverse DNS to assist with asset-owner identification.[5]

Report: We produce graphs for each cluster using graphviz "dot" format [22] (optionally rendered as SVG files) and create a JSON file per cluster containing relevant details. This is done by the "ReportReuse.py" script. We also have tooling to compare runs to check for either cross-border linkages "cross-border.sh" and to compare runs of the same population taken at different times ("dot-r1r2.sh").

Most of the steps above are run by the "skey-all.sh" shell script. Consult the "README.md" file in the code repository for more detail. There are also some additional scripts for installing required dependencies ("install-deps.sh") and for validating clusters ("check-keys.sh") described further below, or in the code repository.

B. Fingerprints and Clustering

We base clusters on the SHA256 [23] fingerprint of public keys, as reported by ZGrab. We also record whether TLS server certificates are "browser-trusted" (those can still be expired certificates) 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. 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 (TLS) ports. We do see many cases where SSH host keys are re-used across sometimes many 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 the next host in the scan shares keys with both.

C. Scoping and Geo-location

Both censys and our local 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). 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.

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

[5]We only query up to 100 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 - 100 SANs should be enough to identify any asset-owner, which is why we're interested in SANs in this study.

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

D. Other Tooling

There are some additional tools in the code repository in the "clustertools" and "cross-border" directories. 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 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. The "cross-border" directory contains the "cross-border.sh" script which is a stateful script for checking linkages between new and existing scans. The "superclusters.sh" script uses that output to generate a large latex/PDF document with details of cross-border linkages. Additional tools are added as analysis warrants.

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 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.

Some hosts are sometimes not accessible during validation, and appear as discrepancies that may disappear in a later run. Presumably some of this is due to intrusion prevention systems reacting to our validation traffic. 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 for the cluster shown in Figure 1 produced the output shown in Figure 2.

F. (Lack of) Infrastructure

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 a small virtual server or normal laptop. Specifically, we currently run scans from a Virtual Private Server (VPS) with the parameters listed in Table II. Only the "Grab" phase of the process is run on this host - the rest is better done on a typical laptop.

```
$ 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" shows no discrepancies for SSH but 41 mismatches out of 585 pairs of ports for TLS. Most mismatches are due to a lack of response but there were four real key changes.

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

G. Ethical Considerations

As we're doing active scans it is appropriate to consider whether there are ethical implications of this work. As our scans are of hosts that listen on port 25 (i.e., email servers) those don't raise the ethical isues that would arise with hosts operated by or for individual people. We are also much less likely to record personal information in banners from such hosts.

Our scanning rate is low enough that we do not believe we negatively 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 of our scanner. 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 will not publish scan data, as that could assist attackers. In communications with asset-holders, we anonymise IP address and name information involving any other asset-holders also involved in the set of clusters concerned. We do however include all AS numbers involved in clusters as asset-holders may have relevant contacts that can help with mitigation.

IV. Related Work

Early work surveying the use of cryptographic keys on the Internet included Heninger et al's seminal work [24] 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. [2], [25], [26], [3]. The properties of email security deployments have also been studied, for example, by Durumeric et al. [27] and Holz et al. [28]. Albrecht et al, carried out

Internet-wide scans of SSH usage [5] finding "about 2^{24}" servers in their scans. Holz et al's "TLS in the Wild" [28] perhaps comes closest to this work, as it notes the kind of re-use described here. Gasser et al. [29] similarly describe SSH host key re-uses found in their 2013 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

Tables III and IV provide an overview of the runs in this article. Figures 3 and 4 show the cluster size distributions.[7]

In order to validate that our initial results don't reflect some oddity of the four countries scanned (e.g. some European middle-sized Atlantic country bias), we ran scans of six additional countries. NA, NZ, SG and UY provide better geographic coverage, and LU and SI are added to include central Europe. The HARK percentages for these countries are consistent with our first four cases, except for NA, which is much lower. It is not clear if this is due to some real-world cause (e.g. fewer data centres), or if that is an artefact of having so few mail servers.

Clearly countries further away from the scanning host require more time to scan. In the case of NZ, scanning averaged 12s per IP address, and DNS checks were also slower, thus extending the overall scan time.

For a physically small country, SG is much bigger than our others in terms of the number of port-25 listeners found, and hence in terms of our scan size. We see 73k port-25 listeners, 51k of which do crypto, and 3k clusters. That leads to a more space and memory intensive scan - the grab phase alone produces 2.2G of data, and the analysis phase required using a machine with 8GB of RAM and 16GM of swap - at peak, the analysis code was using about 95% of the RAM and 60% of the swap space. The full SG scan, including images, requires 17GB of storage.

The NA scan produces so little data that our normal illustration of clustersizes, with a log-scale x-axis, isn't really useful. Figure 5 presents that information at a more appropriate scale.

For each run we report the following:

Country: IE, EE, FI, PT, LU, UY, NZ, NA, SG, or SI in this article.

Scan start/Analysis dates: Scans can take multiple days. Final analyses reported here were sometimes months later.

IPs from ZMap: the number of port-25 listeners from censys for 2017 scans, or, for local scans, the number ZMap finds based on the MaxMind prefixes.

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 - we don't scan 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 SSH or TLS can be seen. There is no quality judgement as to whether keys are good or bad, certified or not, certificate expired or not, etc.

Some crypto%: is *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 keys 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 contactable host/port combination this would be 100%.

Hosts with only local keys: are 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 is *100*hosts-in-clusters/some-crypto*.

Max, Median and Average cluster size: are 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 7, 8, 9 and 10 are sample cluster graphs from scans, to give a sense of the range of clusters we see.[8]

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 17 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 (sufficiently)

[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. Versions 0.7 and earlier of this article missed a small number of cross-border links due to a script bug fixed before version 0.8 was posted.

[8] https://down.dsg.cs.tcd.ie/runs/ shows all graphs.

TABLE III: Overview of runs (a)

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
Analysis	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

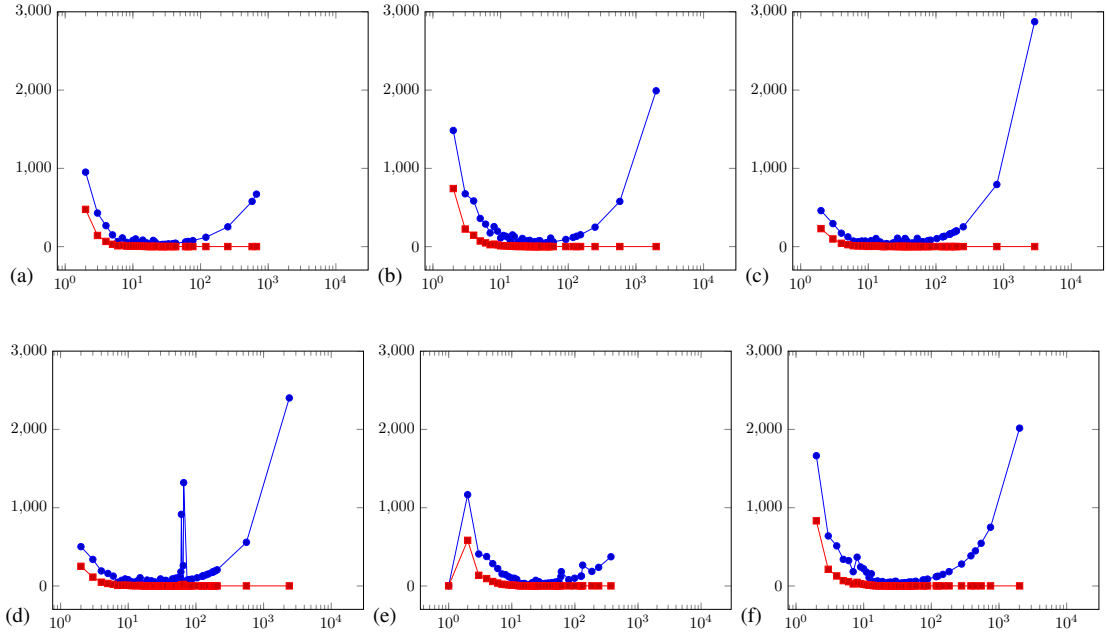


Fig. 3: Cluster size distributions for runs (a) IE-20171130, (b) IE-20180316, (c) EE-20171130, (d) EE-20180324, (e) FI-20180326, (f) PT-20180403. Blue circles show the number of hosts in clusters of given size, red squares reflect the number of clusters of given size. The x-axis is logarithmic.

TABLE IV: Overview of runs (b)

Country (year)	LU	UY	NZ	NA	SG	SI
Scan start	2018-04-23	2018-04-24	2018-04-25	2018-04-30	2018-04-30	2018-05-14
Scan end	2018-04-24	2018-04-25	2018-04-29	2018-04-30	2018-05-22	2018-05-20
IPs from ZMap	6800	1878	23333	551	73608	9759
"out of county"	4	0	3	0	14	0
"In country" IPs	6796	1878	23330	551	73594	9759
No crypto seen	686	336	11872	172	21871	1882
Some Crypto	6110	1542	11458	379	51723	7877
Some crypto%	89%	82%	49%	68%	70%	80%
Total crypto host/ports	21284	4806	33424	820	129346	26330
Total unique keys	5622	1355	10657	504	35364	7421
Percent keys vs. max	26%	28%	31%	61%	27%	28%
Hosts with only local keys	1966	720	5814	299	22644	3529
Hosts in clusters	4144	822	5644	80	29079	4348
HARK	**67%**	**53%**	**49%**	**21%**	**56%**	**55%**
Number of clusters	446	180	811	30	3050	612
Max cluster size	1012	245	281	8	2800	948
Median cluster size	19	7.5	25.5	4	52.5	19.5
Average cluster size	73	29.67	47.12	4.4	179.47	66.88

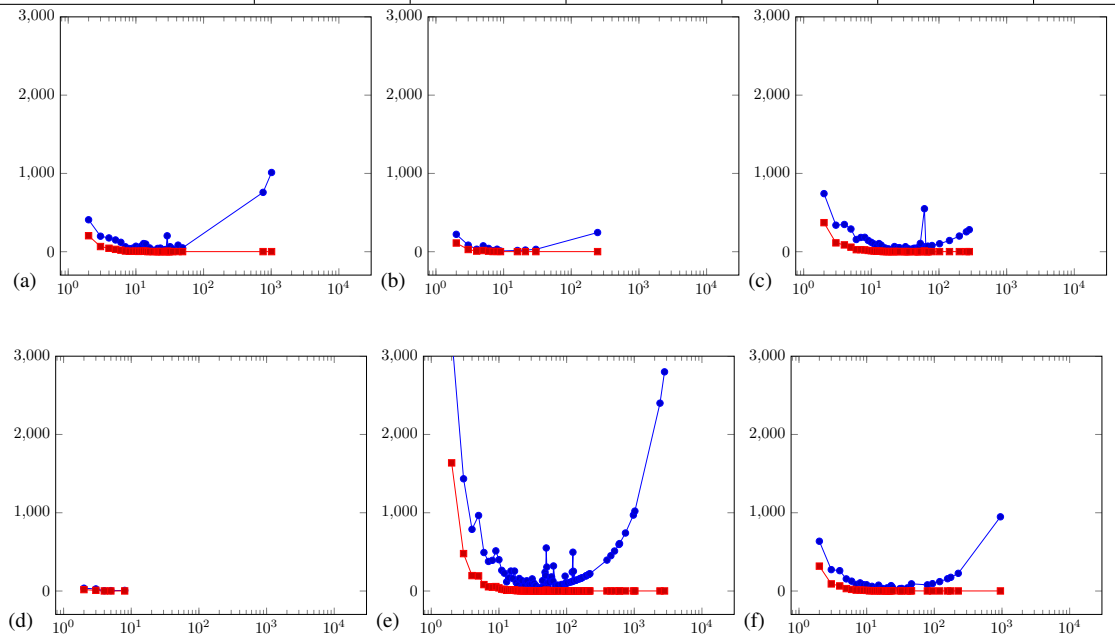
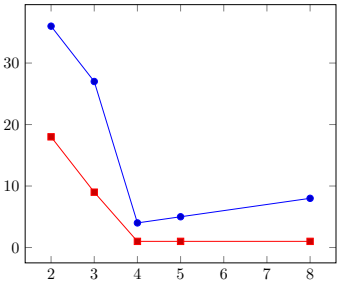


Fig. 4: Cluster size distributions for runs (a) LU-20180423, (b) UY-20180424, (c) NZ-20180425, (d) NA-20180430, (e) SG-20180430, (f) SI-20180514. Blue circles show the number of hosts in clusters of given size, red squares reflect the number of clusters of given size. The x-axis is logarithmic. NA figures are so small this rendering isn't useful so Figure 5 presents NA at a more appropriate scale.

Fig. 5: NA cluster sizes at a more appropriate scale. Blue circles show the number of hosts in clusters of given size, red squares reflect the number of clusters of given size. The biggest cluster has 8 hosts, and there are 18 clusters of size 2.



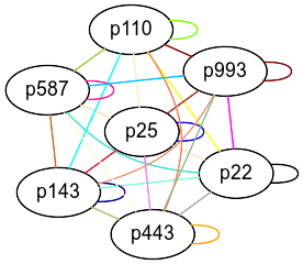


Fig. 6: Edge colours in graphs. Nodes are port numbers, edge colours are as used between nodes in cluster graphs when the pair of ports re-use the same key.

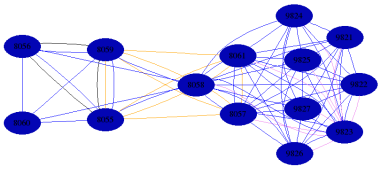


Fig. 7: Cluster 112 from run EE-20180324 is a 14 host cluster with mail and web key re-uses.

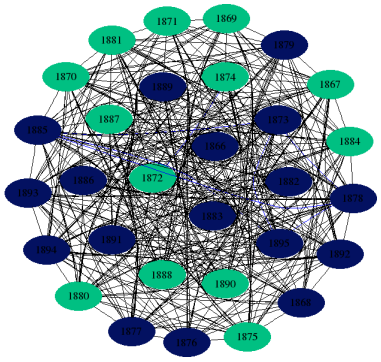


Fig. 8: Cluster 60 from run FI-20180326 is a 30 host cluster involving mostly SSH host-key re-uses, but with some re-use of key on mail ports. 13 of the hosts are in one AS, and 17 in another.

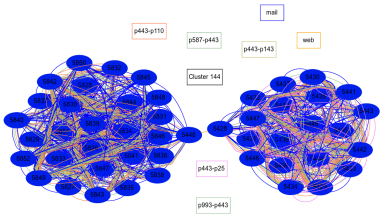


Fig. 9: Cluster 144 from run PT-20180403 is a 48 host cluster with two clear “lobes”

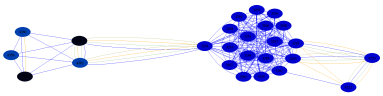


Fig. 10: Cluster 240 from run FI-20180326 is a 25 host, 2 AS, cluster with mail and web key re-uses.

unpredictably for each run. The “ReportReuse.py” script that produces the graphviz dot files used to generate graphs takes an argument specifying whether to anonymise the IP addresses like this. Node colour is based on the ASN of the host. Edge colours are specific to the combination of ports - the colours used are as shown in Figure 6.

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 25. The set of hosts in run IE-20171130/Cluster-10 turns into run IE-20171130/Cluster-333 shown in Figure 24. One additional host is added to the cluster and some

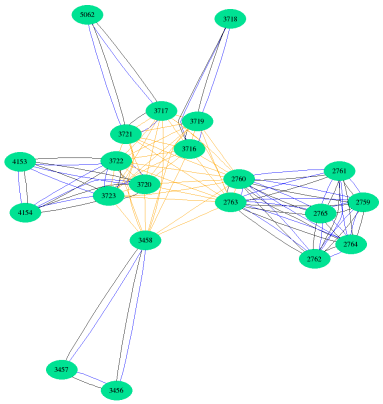


Fig. 11: Cluster 355 from run FI-20180326 is a 21 host cluster with various mail/web key re-uses and some apparent structure.

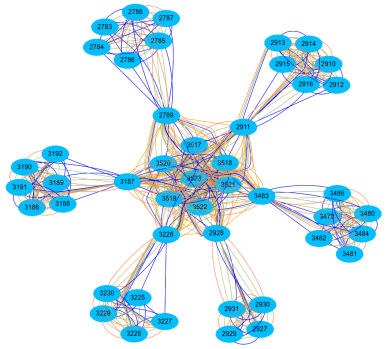


Fig. 12: Cluster 152 from run SI-20180514 has 46 hosts sharing variously on mail and web ports in one AS.

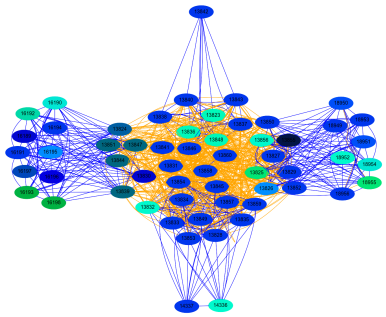


Fig. 13: Cluster 128 from run SG-20180430 has 58 hosts sharing variously on mail and web ports in 15 different ASes.

of the linkages have undergone changes. Cluster 355 from the FI-20180326 run, shown in Figure 11, also shows some interesting structure as does cluster 835 (Figure 1) from the same run. Figures 12 and 13 are additional examples.

B. 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.[9] 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.

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

TABLE V: Most re-used SSH Host Keys

Run	Cluster	Most Re-uses	Cluster	"Pure" Re-uses
IE-20171130	363	32	89	10
IE-20180316	199	48	103	25
EE-20171130	401	209	18	26
EE-20180324	26	542	28	173
FI-20180326	115	60	368	26
PT-20180403	130	136	340	10
LU-20180423	54	32	90	11
NA-20180430	24	3	none	none
NZ-20180425	15	246	109	8
SG-20180430	242	729	13	137
SI-20180514	95	19	91	14
UY-20180424	58	7	110	3

C. Communicating with Asset-Holders

We have developed tooling, "ah-tb.sh", "'ah-ranges.sh" and "ClusterAnonOthers.py", to assist with communication with asset-holders. The former two create a directory containing the set of cluster files corresponding to an input regular expression or set of prefixes 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 results into whatever process suits the asset-holder. In most cases so far we have started by contacting vendors or local ASes. (Exceptions being an educational institution where we had contacts and a case where we were the holder of a re-used key pair and so could open a ticket with the relevant hoster.) It's likely we may move on to contacting the owners of hosts in clusters directly but time will tell.

D. Most used SSH Keys

For each run, we checked for the clusters that had the "most re-used" SSH host keys, and for the largest cluster that was a "pure" SSH cluster, with no TLS keys at all. Table V shows the results for each run. As can be seen, for run EE-20180324 there is one cluster (26) with 558 hosts where 542 of those use the same SSH host key. For the same run, cluster 28 has 173 hosts all of which use the same SSH host key, and do no other cryptography. The largest "pure" SSH cluster, seen so far, has 173 hosts sharing a single SSH key, in one AS and is shown in Figure 14. The 2nd-largest has 127 hosts sharing one SSH host key across 5 different ASes in Singapore and is shown in Figure 15. (That key, perhaps oddly, is not seen in any other country so far.) These numbers are generated using the "biggest22.sh" script.

VI. Irish Results

In this section we report on the Irish runs, with additional detail for selected clusters. Of the 1,437 clusters seen in the IE-20180316 run, 129 (9%) involve more than one AS. The use of less desirable ciphersuites is as shown in Table VII. Recall that RSA key transport combined with many copies of a private key is a bad combination. (RC4 is independently

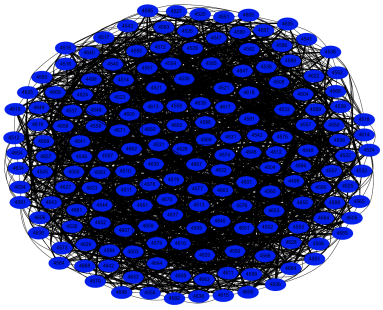


Fig. 14: Cluster 28 from run EE-20180324 has 173 hosts sharing a single SSH host in one AS.

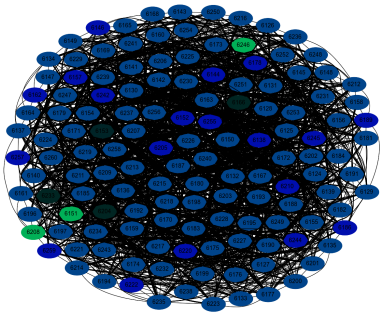


Fig. 15: Cluster 13 from run SG-20180430 has 137 hosts sharing a single SSH host key over 5 ASes.

bad.) Information on the certificates seen on TLS ports is shown in Table VI. In terms of the threat against web origins mentioned in Section II, there are 3,918 hosts that present browser-trusted certificates on port 443 and that are in clusters in this run. The max cluster size for port 443 is 1991, next is 171, then 152. Cluster 32 (see Figure 18) seems like a case where different web origins are being used "behind" a single wildcard certificate. We see 25 cases where a key used on port 443 is sometimes presented via a browser-trusted certificate but elsewhere 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.

A. Time Evolution

Clusters evolve over time. The primary attributes of clusters are key fingerprints and IP addresses that can change independently, under the control of whomever administers a host. Keys

TABLE VI: TLS certificate types for run IE-20180316. The rightmost column is the number of listeners on that port that 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

TABLE VII: Less Desirable Cipherstuites in IE-20180316

Code	Ciphersuite Name	Count
x000A	TLS_RSA_WITH_3DES_EDE_CBC_SHA	2
x0035	TLS_RSA_WITH_AES_256_CBC_SHA	86
xC011	TLS_ECDHE_RSA_WITH_RC4_128_SHA	87
x002F	TLS_RSA_WITH_AES_128_CBC_SHA	1818
x0005	TLS_RSA_WITH_RC4_128_SHA	2806

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 two runs, IE-20171130 and IE-20180316, we examined the changes in the "forward" (from 2017 to 2018) and "reverse" directions with the overall results shown in Table VIII. 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 key 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 is the"IP and FP linked" category of clusters 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 changed between runs.

The "complex" category covers relationships that are more complicated - Figure 16 shows the clusters in this category. Those changes show cases of two clusters merging and of clusters splitting into two or three clusters based on either IP address or keys. In one case two clusters evolve into three clusters via both IP address and keying changes. It seems that most of the possible kinds of change happen, even if in small numbers, and even if such changes initially seem improbable.

B. Selected Cluster Details

Table IX lists details for a selection of clusters. Further notes for some of those are below. These figures are produced by the "ClusterStats.py" script.

Cluster 3: This cluster is the largest in the run. All 1991 hosts are in the same AS, one of the world's hyper-scalers who hosts machines in Ireland. This cluster doesn't seem to be strongly associated with Ireland. There are a total of 1995 host/port combinations, 1991 using the same key with browser-trusted certificates for port 443. A Certificate Transparency (CT) search via crt.sh for that key fingerprint indicates that key has been in use since at least late 2014. The other key in this cluster is used for 4 hosts on port 25 without a browser-trusted certificate.

These hosts may be used for some marketing related service. There are many different SMTP banners. For each, the name in the banner is used as the hostname for a DNS

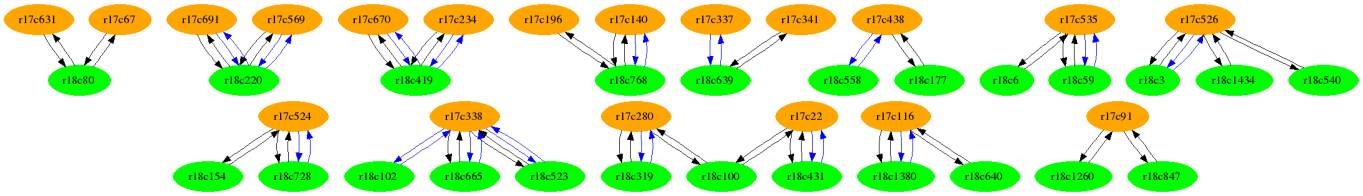


Fig. 16: The clusters with "complex" time evolution between runs IE-20171130 (orange) and IE-20180316 (green). Blue arrows show key fingerprint linkage, black arrows show IP linkage.

TABLE VIII: Cluster evolution - Categories in the evolution from IE-20171130 to IE-20180316. Numbers are the number of clusters in each category.

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

name that matches the wildcard certificate. So if the SMTP banner is "foo" and the certificate is for "*.xyz.example.com" then we see a DNS entry for "foo.xyz.example.com". These names are commonly of the form "[company]-mkt-dev1-1" or "[company]-mkt-prod2" and there are 293 such 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. The same keys, banners and other names were also see in the IE-20171130 run. At that time there were 671 members of the cluster.

Cluster 665: This is the second largest in the run with hosts in five ASes - the numbers in each being 433, 111, 17, 15 and 2. The AS with 433 hosts is a well-known local hoster/registrar. The AS with the 111 hosts is a local ISP/hoster. The ASes with 17 and 15 hosts are international with a local presence. The AS with 2 hosts is a local Internet/computing consultancy. There are 475 unique reverse DNS names associated with this cluster and 155 unique names extracted from TLS certificates, the most common of which occurs 990 times. The names from banners and certificates may imply that a hosting control panel tool is involved in these re-uses. Forum posts related to that tool appear to indicate that using TLS settings is considered challenging. We see 7 cases where the same TLS key is used sometimes with browser-trusted certificates, and sometimes with self-signed certificates.

Cluster 9: This cluster is the third largest in the run. All 249 hosts in this cluster are in the same AS, which is a local hosting company. The cluster shows a mixture of re-uses of SSH (46 keys for 248 hosts) and TLS keys (137 keys over 1148 ports). All but one of the 248 reverse DNS names seen are below a local hosting company's ".ie" 2LD. Most names in certificates also match the same 2LD. In addition, there are another 8 seemingly unrelated names in the ".ie" and ".com" namespaces seen in browser-trusted certificates.

Cluster 52: This cluster is the seventh largest in the run, see Figure 17. All 118 hosts in this cluster are in the same AS, a hyper-scaler with a local presence, and all hosts only do crypto on port 443. All hosts use the same browser-trusted and non-wildcard certificate, issued in mid-2016. The associated name may indicate an association with mail delivery. The certificate for these hosts includes a name like "example.net" with a SAN for "www.example.net" but there is no A record for the relevant "www.example.net" in the DNS at the time of writing - there are MX, SPF, A and perhaps other DNS records for the "example.net" value in question..

Cluster 32: This cluster has 92 hosts, see Figure 18, all 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, issued in mid-2017. "check-keys.sh" shows some discrepancies. 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 SMTP banner names, mostly below the same .co.uk second-level domain, but with the leftmost 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.

Cluster 199: Cluster 199 has 48 hosts, see Figure 19, all part of a small local telco AS, with which the author was previously unfamiliar. All hosts use the same SSH host key, and 7 TLS keys are used over 237 ports. A run of "check-key.sh" showed no discrepancies from the the cluster was detected. Almost all of names seen are the same and are of the form "[tool].example.com" where "tool" is a well-known hosting tool. Four names are for what seem to be unrelated 2LDs under the ".ie" ccTLD, three of which produce an NXDOMAIN response when queried for in the public DNS, and one of which returns an A record. Three names are for apparently unrelated names under the ".com" gTLD, all of which return A records currently.

Cluster 144: This cluster has 26 hosts and is one of two matching the median cluster size, see Figure 20. All 26 hosts in this cluster are in the same AS, which belongs to a local ISP. There are 17 2LD names, most (10) below "ie" with others below ".com", ".co.uk" and one below ".eu". Most certificates however reflect the name of a popular hosting tool. "check-keys.sh" detected no mismatches - all 125 host/ports remained the same.

Cluster 177: This cluster has 26 hosts and is one of two matching the median cluster size, see Figure 21. All 26 hosts are in the same AS, which belongs to a local ISP. There are

TABLE IX: Selected Clusters in IE-20180316 run. SSH and TLS key counts are per-key, re-uses are not counted here. Browser-trusted and wildcard certificate counts and the "most re-used" count are per-port, i.e., multiple ports on the same host add to the count.

Cluster	3	665	9	52	32	199	144	177	103	194	338
IPs	1991	578	249	118	92	48	26	26	25	22	21
ASes	1	5	1	1	1	1	1	1	1	1	4
Crypto Ports	1995	3405	1396	118	165	285	125	128	25	121	138
SSH ports	0	488	148	0	0	48	0	24	25	21	16
SSH keys	0	310	46	0	0	1	0	1	1	1	6
TLS ports	1995	2917	1148	118	165	237	125	104	0	180	122
TLS keys	2	169	137	1	8	7	17	2	0	13	18
Browser-trusted	1	517	1052	118	74	232	125	0	0	96	5
Wildcards	1991	226	501	0	72	0	0	0	0	0	1
Most Re-Used	1991	1298	911	118	70	231	109	78	25	88	85
Cluster	333	227	462	1227	111	8	76	648	804	639	1389
IPs	21	15	14	5	5	4	3	2	2	2	2
ASes	1	10	6	1	2	1	2	1	1	1	1
Crypto Ports	135	20	16	13	31	9	3	2	2	10	2
SSH ports	19	0	0	5	1	4	3	2	2	0	2
SSH keys	19	0	0	1	1	4	1	1	1	0	1
TLS ports	116	20	16	8	30	5	0	0	0	10	0
TLS keys	17	2	2	7	1	2	0	0	0	2	0
Browser-trusted	7	1	0	3	27	4	0	0	0	1	0
Wildcards	1	0	0	1	17	4	0	0	0	0	0
Most Re-Used	47	19	15	5	39	4	3	2	2	9	2

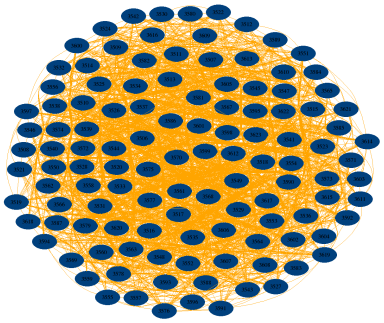


Fig. 17: Cluster 52 has one hundred and eighteen hosts sharing Web server keys and is the largest cluster that renders with graphviz for this run. There are six larger clusters.

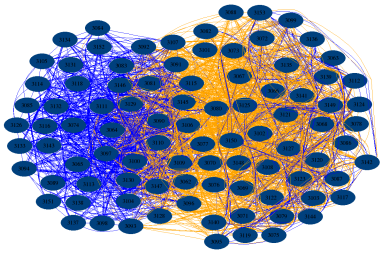


Fig. 18: Cluster 32, has 92 hosts sharing web and mail host keys.

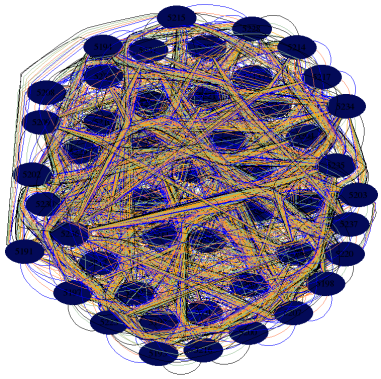


Fig. 19: Cluster 199 has 48 hosts

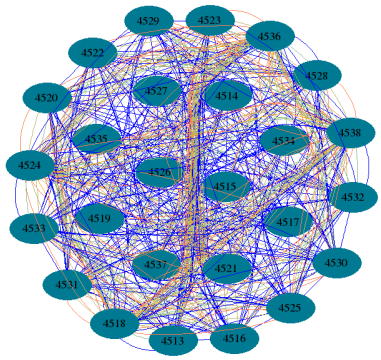


Fig. 20: Cluster 144, has twenty-six hosts.

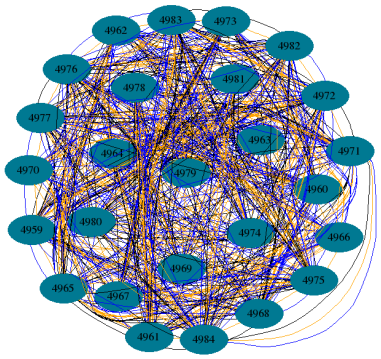


Fig. 21: Cluster 177, has twenty-six hosts.

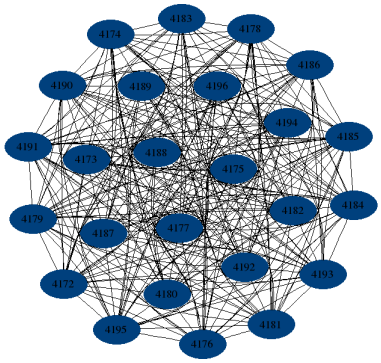


Fig. 22: Cluster 103, has twenty-five hosts sharing SSH host keys and is the largest "pure" SSH cluster in this run.

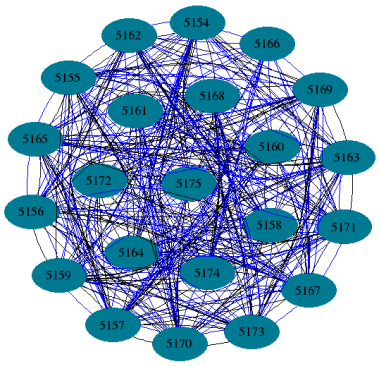


Fig. 23: Cluster 194 has twenty-two hosts that re-use lots of keys.

Fig. 24: Cluster 333 - Fig. 25 grown a little a few months later

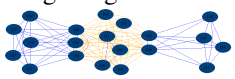


Fig. 25: Cluster 10 from IE-20171130.

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 24 times. "check-keys.sh" shows some change on port 443 since the test run. Most of the names used are for one 2LD below ".info" - at the time of writing the relevant name servers for the 2LD appear unavailable. The most common names used in this cluster is the literal string "imap.example.com" presumably indicating some lack of care in provisioning.

Cluster 103: This cluster has 25 hosts and is the largest "pure" SSH cluster in the run - see Figure 22. All 25 hosts are in the same AS, which is a hyper-scaler with a local presence. The only crypto used seems to be SSH with all hosts using the same SSH host key, and with no TLS services in use. 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.

Cluster 194: This cluster has 22 hosts. See Figure 23, all 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.

Cluster 333: Cluster 333 has 21 hosts, see Figures 24 and 25, all 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, 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.

We saw this cluster as cluster 10 in our IE-20171130 scan.

Cluster 338: Cluster 338 has 21 hosts in four different ASes - see Figure 26. There are 128 host/port combinations, 85 of

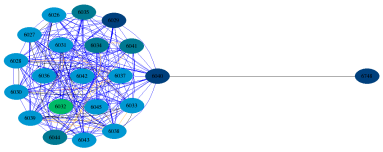


Fig. 26: Cluster 338 has 21 hosts sharing variously over four different ASes.

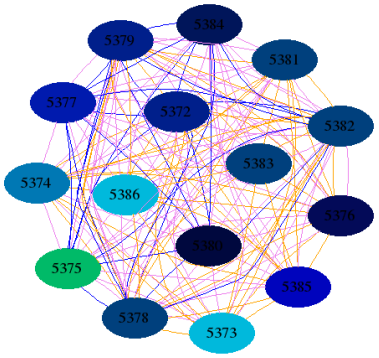


Fig. 27: Cluster 227 has 15 hosts sharing variously over ten different ASes.

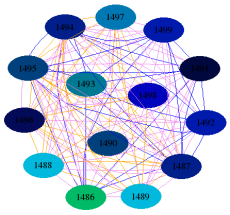


Fig. 28: Cluster 150 from run IE-20171130 overlaps cluster 227.

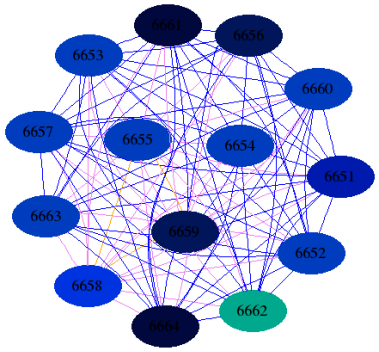


Fig. 29: Cluster 462 has 14 hosts sharing variously on ports 25 and 443 web over 6 different ASes.

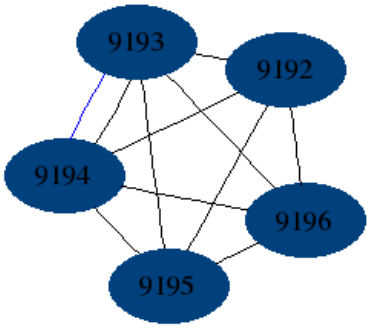


Fig. 30: Cluster 1227 has five hosts that re-use the same SSH host-keys, and two of whom share a key for port 25.

which use the same key and all of which are mail ports. The 9 unique SMTP banners have names that vary widely. The most commonly used key in this cluster (used 85 times for mail protocols, but not for the web) is also seen 69 times within a cluster in Estonia, but is not seen in any clusters in either Portugal nor Finland.

Cluster 227: Cluster 227 has 15 hosts - see Figures 27 and 28. Ten 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 VII for details. The name of a well-known anti-spam vendor features for most of the hosts in this cluster. That and the fact that this is seen in different countries would seem to suggest that a hard-coded key, or some similar problem, may be a cause.

Cluster 462: Cluster 462 has 14 hosts, see Figure 29, in 6 different ASes, 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." At least one of these hosts is offering a live service. This was confirmed by using the "gc.sh" script to extract the certificate from a connection to the MX host for one of the domains named in this cluster. This cluster overlaps with clusters from runs in all other locales. We have confirmed that a product that ships with a default key pair is the cause. Following discussion with the vendor concerned they stated that they will fix the issue and contact the affected customers.

Cluster 1227: Cluster 1227 has 5 hosts, see Figure 30, all within a single AS operated by a hyper-scaler. There are 13 host/port combinations, 5 are shared SSH host-keys. Two are shared port 25 keys. 6 other unique keys are used for port 25 and port 443. The three port 443 certificates are browser-trusted, no other certificates are browser-trusted. "check-key.sh" reported 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.

Cluster 111: Cluster 111 has 5 hosts who make lots of use of one key, see Figure 31. 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. Each host has an SMTP banner of the form "[foo].example.com" where foo is either "server1" (one occurrence) or "hosting2" (four occurrences),

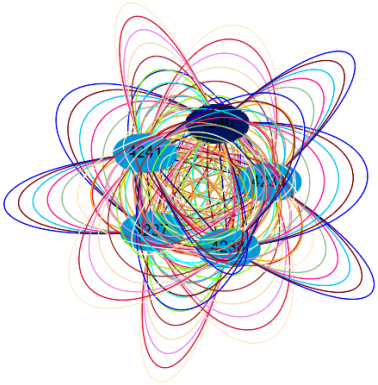


Fig. 31: Cluster 111 has five hosts that re-use the same keys for almost everything. As can be seen the individual edges become less useful at this point.

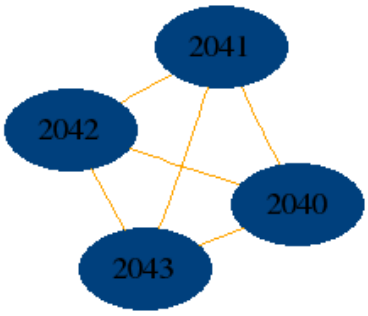


Fig. 32: Cluster 8 - a representative four host cluster where the same TLS key is used on port 443.

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.

Cluster 8: Cluster 8 has 4 hosts who share one key for port 443, see Figure 32, all 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.

Cluster 76: Cluster 76 has 3 hosts who share SSH host-keys, see Figure 33. Two of the hosts are within a 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.

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

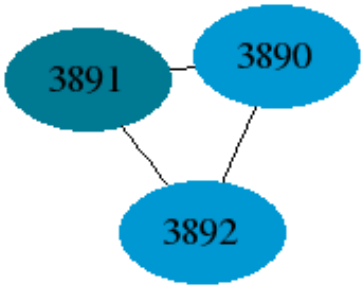


Fig. 33: Cluster 76 consists of three hosts, in two different ASes and shows SSH host-key re-use.

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.

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.

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.

Cluster 1389: This cluster has 2 hosts who share an SSH host-key and is the smallest (JSON file) in the run. Both hosts are in an AS 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.

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).

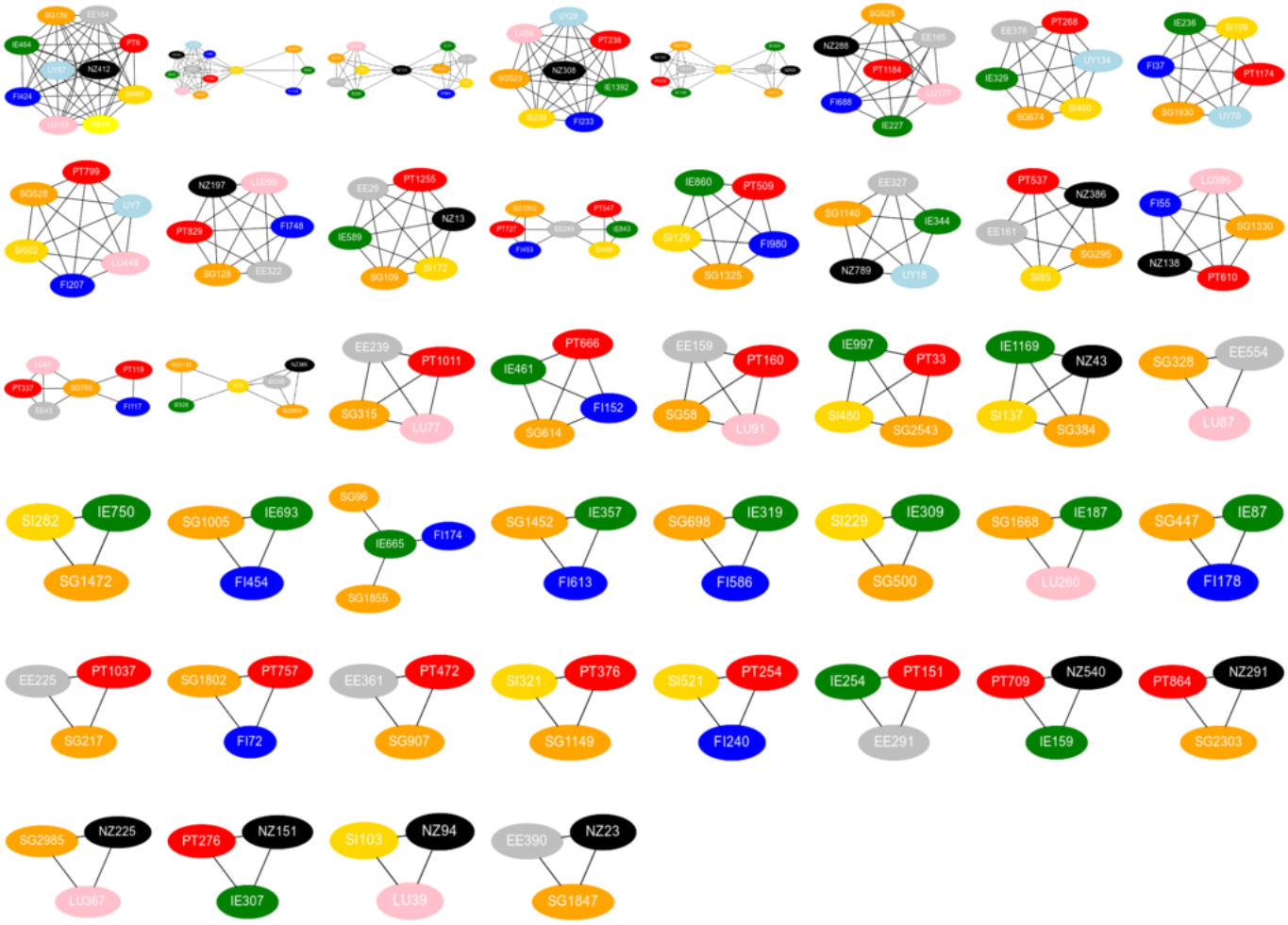


Fig. 34: Cross-border overlaps for clusters with more than one link. Nodes represent clusters. There are an additional 57 cases where two clusters in different countries are linked only to one another. (NA=yellow;NZ=black;UY=lightblue;PT=red;IE=green;EE=grey;SI=gold;SG=orange;LU=pink;FI=blue;)

TABLE X: Counts of cross-border links.

-	NA	NZ	UY	PT	IE	EE	SI	SG	LU	FI
NA	x	1	1	1	1	1	1	1	1	1
NZ	1	x	4	13	14	13	12	28	9	7
UY	1	4	x	6	6	4	6	7	4	6
PT	1	13	6	x	15	17	16	26	10	15
IE	1	14	6	15	x	12	18	45	6	25
EE	1	13	4	17	12	x	11	25	10	6
SI	1	12	6	16	18	11	x	21	7	9
SG	1	28	7	26	45	25	21	x	15	22
LU	1	9	4	10	6	10	7	15	x	7
FI	1	7	6	15	25	6	9	22	7	x

VII. Cross-border results

When there is a common key overlap in two clusters from different runs, then we call that a cross-border link, and the set of linked clusters a super-cluster. Figure 34 shows the accumulated set of cross-border super-clusters and table X summarises the numbers of cross-border links. It is notable that there is no pair of countries without cross-border links. Almost all of the "super-cluster" graphs are fully connected. All but one of those that are not seem (so far) to be two such, connected via one cluster.

Discussions with asset-holders seem to indicate they are more interested (or perhaps puzzled/scared) by cross-border links, and hence presumably more motivated to take action as a result, perhaps justifying expending some effort on analysis of super-clusters. That analysis is ongoing.

The "cross-border.sh" script produces figure 34 and table X. That script is stateful, allowing the user to add newly scanned countries to the set for which cross-border links have been analysed.[10] In total there are 7,469 hosts involved in some "super-cluster".

The "superclusters.sh" script produces a LaTeX/PDF output that provides more detail for each cross-border cluster. That script also retrieves certificates and SSH host keys as a verification step and as certificates sometimes contain useful information not previously recorded as meta-data. For exam-

[10]There was a bug in that script that missed a few cross-border links. That was fixed before version 0.8 of this article. Earlier versions showed a few less cross-border links in Figure 34. Those missing links actually made the structures look more interesting, but version 0.8 fixes that;-)

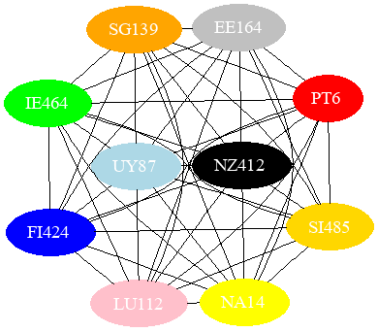


Fig. 35: Cross-border supercluster SCNA14

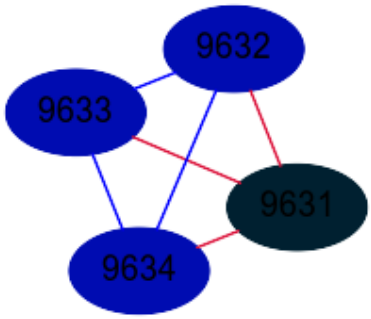


Fig. 36: Cluster EE164, part of SCNA14

ple, in one case, we two self-signed certifiates for the same key, with different extensions in one (related to IMAP it seems) that hadn't previously been picked up.

The outputs from that script however are identifying and the document is very long (approx. 380 pages for the 10 countries), so we only reproduce a small subset of those outputs for selected "super-clusters" in this section.

SCNA14 This super-cluster seems to be built around a single key that is correlated with use of the a specific email server securiy product in banners and DNs. Figure 35 shows the graph for supercluster SCNA14. and table XI summaries the linked clusters. Figures 36-45 are graphs for the country-specific clusters that make up SCNA14.

An email was sent to the product vendor on 20180531. An answer has yet to be received.

SCSG867 is a very small super-cluster but (for now) a puzzle. We see one key on four different IP addresses, one in LU and three in SG, with those three in two different

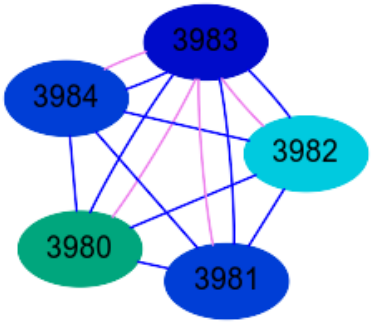


Fig. 37: Cluster FI424, part of SCNA14

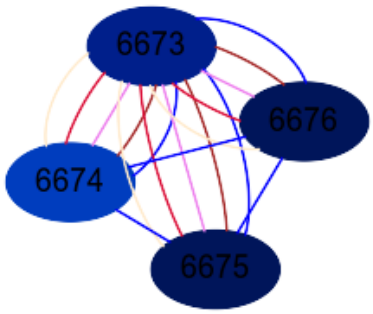


Fig. 38: Cluster IE464, part of SCNA14

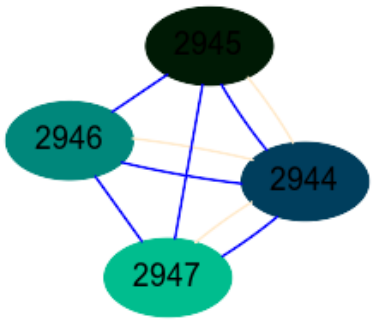


Fig. 39: Cluster LU112, part of SCNA14

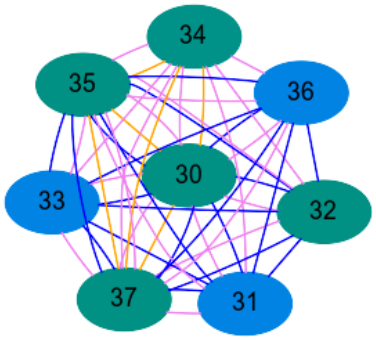


Fig. 40: Cluster NA14, part of SCNA14

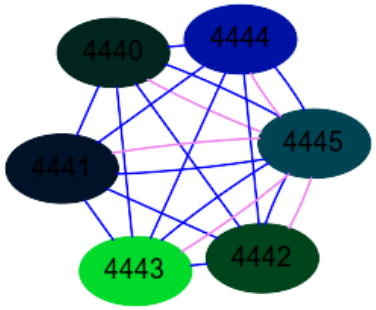


Fig. 41: Cluster NZ412, part of SCNA14

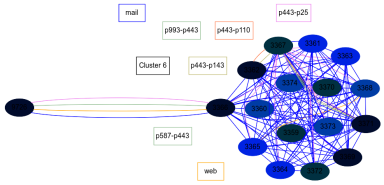


Fig. 42: Cluster PT6, part of SCNA14

TABLE XI: Summary of SCNA14 clusters.

Name	EE164	NA14	IE464	FI424	SG139	SI485	UY87	LU112	PT6	NZ412	Total
IP addrs	4	8	4	5	22	3	7	4	17	6	80
ASes	2	3	3	4	8	1	3	4	4	6	38
Port count	4	16	9	6	49	7	14	5	42	10	162
SSH ports	0	0	1	0	3	0	0	0	1	0	5
SSH keys	0	0	1	0	3	0	0	0	1	0	5
TLS ports	4	16	8	6	46	7	14	5	41	10	157
TLS Keys	1	4	1	1	2	4	5	1	10	4	33
B-T certs	0	0	0	0	0	0	2	0	7	2	11
W/C certs	0	0	0	0	0	0	1	0	2	0	3
Most key re-uses	4	11	8	6	45	3	8	5	24	7	

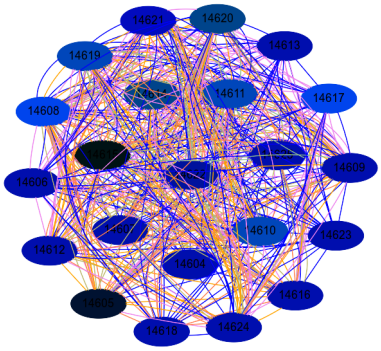


Fig. 43: Cluster SG139, part of SCNA14

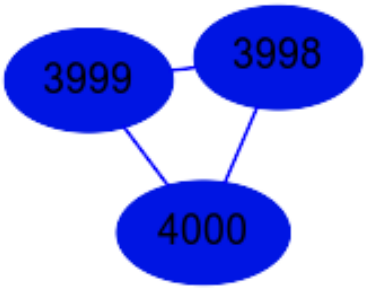


Fig. 44: Cluster SI485, part of SCNA14

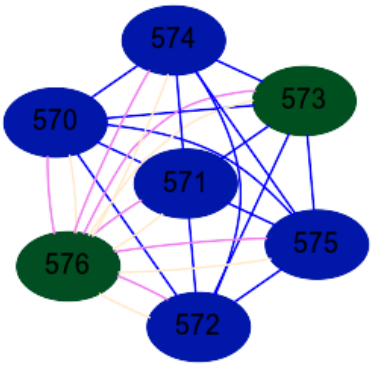


Fig. 45: Cluster UY87, part of SCNA14

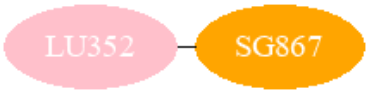


Fig. 46: Cross-border supercluster SCSG867

TABLE XII: Summary of SCSG867 clusters.

Name	LU352	SG867	Total
IP addrs	2	3	5
ASes	1	2	3
Port count	9	9	18
SSH ports	2	2	4
SSH keys	1	2	3
TLS ports	7	7	14
TLS Keys	4	2	6
B-T certs	6	0	6
W/C certs	0	0	0
Most key re-uses	4	4	

ASes. The key is used on port 25 on all hosts, and on port 587 on one of the SG hosts. Figure 46 shows the graph for this supercluster. Table XII summaries the clusters in SCSG867. Figure 47 shows the graph for cluster 352 in run LU, a part of SCSG867. Figure 48 shows the graph for cluster 867 in run SG, a part of SCSG867. All TLS ports in the clusters involved use cipher-suite 133210, which is TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 and hence relatively modern. At least two of the four hosts with the linking key present the same self-signed certificate on port 25.

Exploration of DNS and additional examination of banners determined that the host in LU is associated with a company who develop a hosting tool. As it is possible that that is the source of key re-use a mail was sent to that organisation on 20180602. An answer has yet to be received.

SCPT151 is a super-cluster involving 21 hosts in 3 countries, 17 of which share an SSH host key. Figure 49 shows the graph for this supercluster. Table XIII summaries the clusters in SCPT151. Figure 50 shows the graph for cluster 291 in run

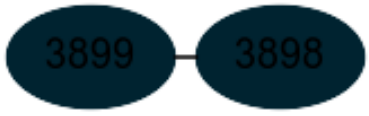


Fig. 47: Cluster LU352, part of SCSG867

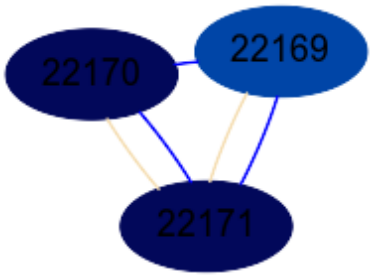


Fig. 48: Cluster SG867, part of SCSG867

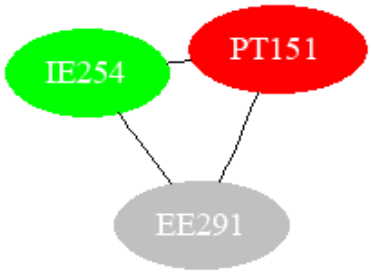


Fig. 49: Cross-border supercluster SCPT151

EE, a part of SCPT151. Figure 51 shows the graph for cluster 254 in run IE, a part of SCPT151. Figure 52 shows the graph for cluster 151 in run PT, a part of SCPT151.

Coincidentally, the author is in control of one of the hosts in IE254, a VPS rented from a local hosting company. It appears that the cause is an ECDSA key that was included in the image when the image was created. While RSA keys were created when the VPS was purchased, the ECDSA key files are dated one year before that date. This appears to be a tooling issue. The author currently has an open ticket with the hosting company concerned.

VIII. Discussion

Some clusters are deliberate, some not. The former could be as a result of an attack but more likely result from tooling that makes it easier to end up re-using keys. Asset owners

TABLE XIII: Summary of SCPT151 clusters.

Name	PT151	EE291	IE254	Total
IP addrs	8	2	11	21
ASes	2	2	2	6
Port count	49	4	42	95
SSH ports	4	2	11	17
SSH keys	1	1	1	3
TLS ports	45	2	31	78
TLS Keys	4	1	14	19
B-T certs	22	0	11	33
W/C certs	14	0	0	14
Most key re-uses	42	2	11	

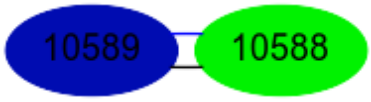


Fig. 50: Cluster EE291, part of SCPT151

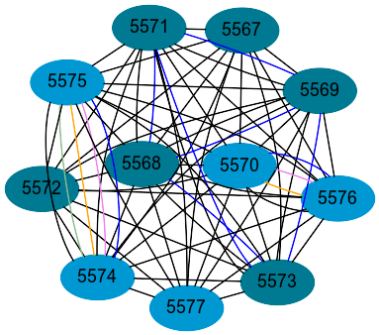


Fig. 51: Cluster IE254, part of SCPT151

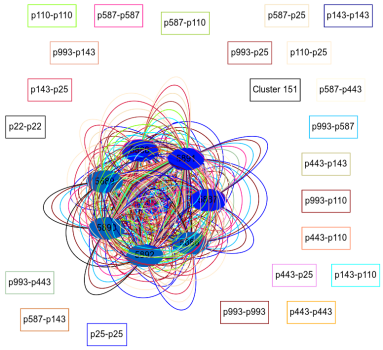


Fig. 52: Cluster PT151, part of SCPT151

may or may not know or care about the existence of clusters. In this section we describe confirmed and likely causes of re-use. In some cases causes are clear from inspection, in others, we accept asset-holders' assertions - attempting to validate for specific clusters would likely be too intrusive.

A. Confirmed Reasons for Key Re-Use

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 host cause re-use. This is not obvious to SSH clients, as the host-key hash presented is 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 shouldn't grow further and may disappear.

One system administrator using Puppet[11] 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 a new 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.

Speaking to asset holders, this seems to be an issue that has

[11] https://puppet.com/

come up in the past,[12] has been fixed, but that tends to recur, perhaps indicating 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.

Software/Hardware with Default Keys: Some clusters were due to a vendor product that ships with a demonstration key pair in order to allow customers to establish TLS connections. While the vendor's documentation warns that this should be replaced, and describes how to replace this in the usual ways, it seems that at least some customers didn't follow or see those instructions and we see some tens of hosts using the same keys for this reason in our scans. It is unclear why the vendor doesn't generate a new key pair for each deployment. Following discussion with the vendor, they stated that they are going to contact customers experiencing this issue and will update their product to gnerate new key pairs at install time. There are other clusters that appear to be similarly caused by other vendor's products.

Hosts with multiple addresses: One hoster has an offering of a "dedicated" server that can be configured with up to 20 IPv4 addresses. Typically, the tenant for such cases is also a hoster, so one sees different web-sites configured somewhat similarly. Cluster 128 in IE-20180316 is an example of this. Another offer from the same hoster leads to size 2 clusters for similar reasons.

Large scale use of wildcard certificates: Some clusters use a private key where the public key is in an X.509 wildcard certificate that is being presented by 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. It is not yet clear if the reason for these uses of wildcard certificates are mainly economic or e.g. to make system administration simpler.

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, and as with wildcard certificates may be related to economic or other causes.

B. Possible Reasons for Key Re-Use

Ongoing discussion with asset-holders may confirm or refute the ideas below. However, at the time of writing, these causes cannot yet be considered confirmed.

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 administrators forgot to turn off or reconfigure the old machine after setting up the new, perhaps only making changes to forward DNS records.

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.

Anycast: If a service uses anycast addressing, multiple instances of the service may have been deployed with the same keys. In such cases, our scans may see multiple instances via different unicast addresses and treat this as key re-use. 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 seem to 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.

Honeynets: Some honeynets may be deployed with re-used keys, either because that is considered to better attract attack traffic, or because those deploying don't really need to care about the transport security properties of attack traffic. Comparing our scans to honeynet IPs supplied by other researchers we found no matches.

Cross-Border Clusters: More investigation of the clusters shown in Figure 34 is needed. We have confirmed that a product shipping with a default key pair is the cause for some of these. Others remain to be investigated.

IX. 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.

Regular Key Rotation: The most effective way to avoid being a part of one of these clusters is likely to simply rotate cryptographic keys at a 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 have access to keys in contravention of local policy, so the pain is likely worthwhile.

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 administration 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.

Use a PKI that encourages key rotation: The combination of certbot[13] and the LetsEncrypt Certification Authority (CA) results in keys being changed every few months, so setting up cron jobs on each host to renew certificates in that manner will remove re-uses fairly quickly. Note that this can be done as easily for certificates used for mail as for the web and could go a long way to de-clustering generally. There may be a perceived downside to doing this as certificates will end up

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

[13] https://certbot.eff.org/

in CT 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 strong argument.

PKIs could impose limits: CAs are in a good position to see some of the re-uses described here and could refuse to issue certificates for some kinds of key re-use. Getting agreement from relevant players for such policy changes could be challenging.

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.

X. 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 declustering 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 faster or on even more modest machines. At present the scanning process is fairly memory intensive and slow in parts - use of a database would likely be a significant improvement.

Infrastructure: Building a relatively modest server, with a few terabytes of storage and good bandwidth would also speed up scans. The author is investigating building such infrastructure in Ireland and would be happy to help anyone else who wanted to do that instead, or as well, in Ireland or some other locale.

Other Populations: The populations scanned to date are geographically bounded and all run SMTP listeners. It could also be interesting to scan other sets of related hosts, e.g. belonging to the same industry 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 CT: We have not, but could, check for additional information based on searching CT logs [30], for example the crt.sh[14] 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.

[14]https://crt.sh/

Reduced "noise": If one considers multi-homed hosts to be "noise" in the output of these scans, then methods to identify such hosts could reduce that noise and help identify less desirable cases of key re-use.

Slowly doing more runs: Another obvious thing to do is to run this for other countries. That has been started. We could extend out tooling to support 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: In discussion with asset-holders, we are investigating 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 for key re-uses.

Longitudinal Studies Re-running these scans over time will likely produce interesting results in terms of how clusters form, live and (hopefully) dissipate. This work is ongoing.

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.

Protocol Design: Typically, application and protocol designers making use of SSH or TLS would assume that keys are unique per endpoint, or nearly so. Given this article shows that diverges from the current reality, protocol designers may need to take widespread key re-use into account, for example in threat models for new protocols.

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. If CAs required key rotation, or even notified key holders that re-use has been seen, that could be a significant help in breaking up clusters.

XI. 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, likely due to a mixture of technology limitations and operators not yet being very familiar with managing keys at scale. Better understanding the causes of key re-use and of ways to avoid or dissipate clusters of key re-use could be a fruitful avenue for further research.

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[15], 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; Richard Clayton and Alexander Vetterl for the honeynets point and data. Thanks also to the asset-holders who co-operated in finding reasons for these results, and the vendor who has committed to improving their product, but who 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] A. Barth, "The Web Origin Concept," RFC 6454 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–20, Dec. 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6454.txt

[12] 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

[13] 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

[14] 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

[15] 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.

[16] 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

[17] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A comprehensive symbolic analysis of tls 1.3," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1773–1788.

[18] K. Bhargavan, C. Brzuska, C. Fournet, M. Green, M. Kohlweiss, and S. Zanella-Béguelin, "Downgrade resilience in key-exchange protocols," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 506–525.

[19] 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

[20] 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.

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

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

[23] 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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] O. Gasser, R. Holz, and G. Carle, "A deeper understanding of ssh: results from internet-wide scans," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–9.

[30] 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

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.

[15] https://responsible.ie/