

Swiss cheese security

or

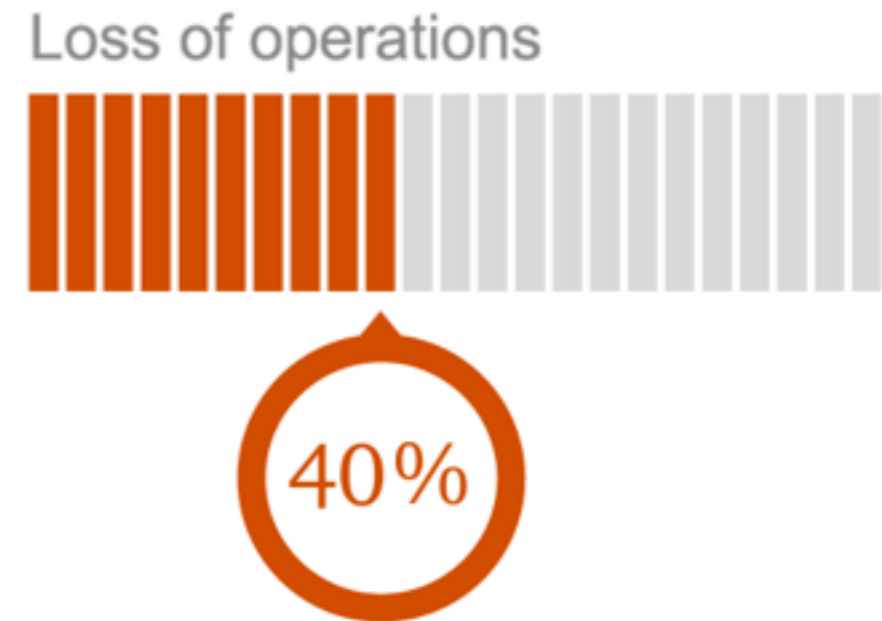
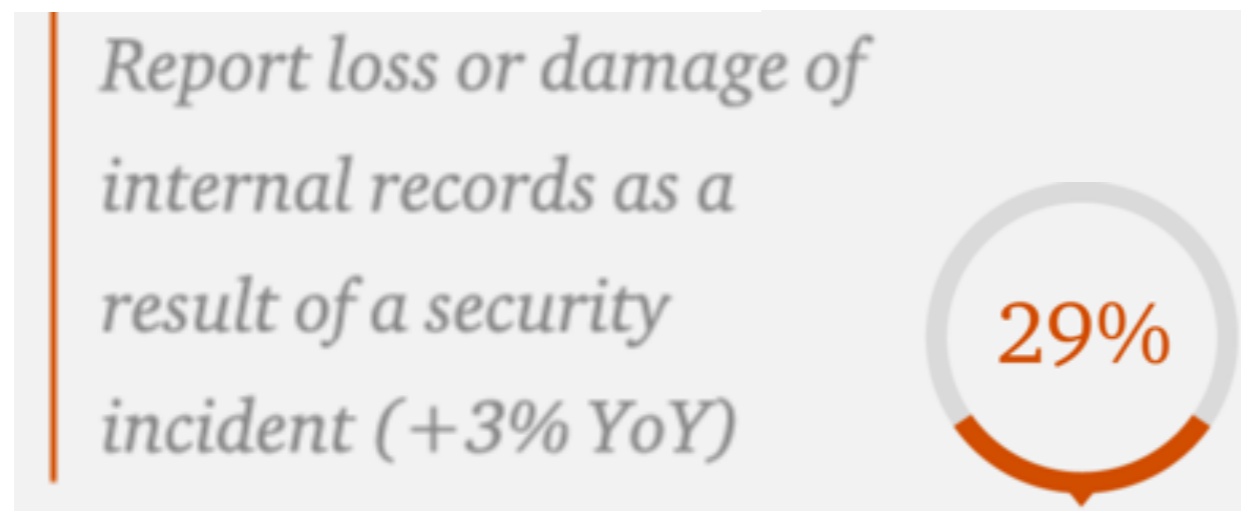
*the real challenges faced by
internet facing companies*

About me

- Almost 20 years in information security / hacking
- OWASP Project leader
- Latest research interests:
 - Large scale RSA crypto survey
 - Testing/Crashing version control systems
 - High precision-low detection network scanning tools
 - Data analysis and correlation

mainstream news and cyber security

Business leaders see new risks tied to emerging technologies



- *Cyber crime damage costs to hit \$6 trillion annually by 2021*
- *Cybersecurity spending to exceed \$1 trillion from 2017 to 2021*
- *Cyber crime will more than triple the number of unfilled cybersecurity jobs, which is predicted to reach 3.5 million by 2021*

<https://www.csoonline.com/article/3153707/security/top-5-cybersecurity-facts-figures-and-statistics-for-2017.html>

<https://www.pwc.com/us/en/cybersecurity/information-security-survey.html>



What industry says about cyber security

Raj Samani, CTO EMEA Intel Security said:

“To overcome the designs of cyber-criminals, we need to go beyond understanding the threat landscape to changing the defender-attacker dynamic. This means focusing on six key areas: We need to make it harder for hackers to obtain information and more expensive for them to launch an attack. Meanwhile on the corporate side we must improve visibility, better identify exploitation of legitimacy, improve protection for decentralised data, and detect and protect in agentless environments.”

<https://www.scmagazineuk.com/cyber-security-industry-2017-predictions-reaching-the-tipping-point/article/628904/>



How can we dispute the claims with a reality check

Companies can claim to have team of consultant testing their network and their servers, some may claim to be PCI DSS compliant and that they execute pentests regularly.

The only problem with this kind of statements is that tests are defined by a the tested company therefore we have a bias.

Companies claiming to be PCI DSS compliant can limit the test to only a portion of the services and still claim all is good.

To avoid biases, nothing beats a test executed at random from a random location using well configured tools.



Before testing

As time passes libraries gets updated and to ensure that software that depends on them is operating securely, many functions and protocols related to old and insecure technologies are removed.

This is common practice for many software libraries but it is not good for security testing, as we want to see if bad/old stuff are there.

From the OpenSSL changelog:

- SSLv2 support has been removed.
- GOST engine has been removed.
- Heartbeat for TLS has been removed.
- Remove support for all 40 and 56 bit ciphers.
- Disabled compression by default

<https://www.openssl.org/news/changelog.txt>

Custom libraries

To get real statistics data has been collected from 50 million IP at random and from 100 financial institutions.

As a client I have used a modified OpenSSL with a custom config.

- *openssl-chacha*: <https://github.com/PeterMosmans/openssl>

Compiled it from source to enable ALL protocols and ALL ciphers.


- Standard OpenSSL: ~158 ciphers
- Custom OpenSSL: ~**201** ciphers (+27 %)

Without the custom OpenSSL (*like default OpenSSL in KALI linux..*):

- no SSLv2 would have been detected
- we would have not been able to connect to ~**50.000** servers
- we would have missed ~**150.000** servers using weak ciphers.

Test results

	Rand 50M IP 2017	Bank/Fin. 2017	Bank/Fin. 2013
Supporting SSLv2	2%	2%	23%
Supporting SSLv3	13%	13%	97%
Using RC4 ciphers	18%	20%	89%
Using weak ciphers	56%	70%	43%
POODLE TLS	16%	14%	x
Padding Oracle (CVE-2016-2107)	4%	2%	x
Protocol Downgrade	26%	18%	48%
TLS Compression / CRIME	9%	2%	30%
HTTP Redirect (MITM)	28%	3%	48%
PCI DSS (all claim to be ok..)	x	21%	13%



Web encryption with SSL/TLS, defending from unknown attacks

From a random test on some financial institutions we can see that not all is good, especially regarding secure communications and client privacy.

To better understand the implications related to web encryption we can analyse some of traffic previously collected and look for issues.

As usual the problem with encrypted stuff is that people want to break encryption to see what is protected, but what can happen if we ignore that part and we test how encryption is applied?

What to look for

RSA keys can be considered insecure when:

- exponent is equal 1
- modulus length in bits is smaller than 1024
- modulus shares a prime with other moduli
- modulus is divisible by small primes
- modulus is not unique
- key (modulus, exponent) is not unique

small primes --> first 10.000 prime numbers

Exponent is equal to 1

No crypto key using RSA algorithm should use exponent 1.

Not only is possible to recover the private key, but the encrypted data is equal to the plain text data.

If exponent 1 is used, “Ciphertext == Plaintext”

Plaintext=[2374623765656] -> Ciphertext=[2374623765656]

73.560.467 keys tested —> found 28 (0.00004 %)

Modulus shares a prime factor

In theory, this should **NEVER** happen:

A key **X** has a modulus created from prime **A** and **B**.

A key **Y** has a modulus created from prime **A** and **C**.

If any two keys share any prime, the secret private key can be found and regenerated for both.

73.560.467 keys tested —> found **758.912** (1.03 %)

Modulus is divisible by small primes

A modulus should be the product of two large primes.

If it can be divided by any of the first 10.000 primes (small primes) this suggests the presence of a poorly designed or malfunctioning RSA implementation.

73.560.467 keys tested —> found **9.098** (0.012 %)

Modulus is not unique

No two distinct RSA public keys should have the same modulus, ever.

If two or more keys have the same modulus, then they have the same private key therefore both are to be considered insecure.

73.560.467 keys tested —> found **2.530.870** (3.44 %)

If any entity becomes aware of the modulus collision then it is possible for them to decrypt the traffic encrypted by all other entities using the same key.

Key is not unique

No two distinct RSA public keys should have the exact same public key, but there are exceptions.

It is allowed to "update" certificates by keeping the key (modulus, exponent) and changing metadata. This creates key duplication as keys are reused.

73.560.467 keys tested —> found **25.834.758** (35.12 %)

Sites can also use only one key for all their domains, so not all duplicated keys are implicitly bad.

Key is not unique

Attacks: CVE-2016-0800 (**DROWN**), CVE-2016-0703

A server that has SSLv2 enabled and uses a vulnerable version of OpenSSL, can be used to attack all other hostnames that appear in its certificate.

Because the security of a server cannot be assessed by just looking at its configuration, we must look for servers that use the same RSA keys and/or certificate hostname (**NOT EASY**... <https://censys.io/> can help).

Insecure protocols

POODLE attack (**protocol vulnerability**)

Problem in **CBC** encryption scheme as implemented in the **SSLv3** protocol.

An active MITM attacker can force to downgrade a connection from TLS to SSLv3, which can be exploited.

A solution to this problem exists, client and server must both implement the **TLS_FALLBACK_SCSV** indicator.

TLS_FALLBACK_SCSV (RFC 7507):

TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks

Insecure protocols

FREAK attack (**downgrade attack**)

An active attacker performing MITM can force a downgrade of the connection to 512 bit encryption, by forcing the use of **RSA_EXPORT** cipher suites (CVE-2015-0204).

The intercepted traffic, even if encrypted, can then be analysed to extract the 512 bit key, that once factored will allow the decryption of the encrypted traffic.

Software released after mid 2015 “should not” be vulnerable to this form of downgrade attacks.

TLS Interception

It is possible to “legally” intercept SSL/TLS traffic and have a software and/or an appliance act as a valid endpoint.

Communication is intercepted, certificate/keys are replaced/swapped, traffic is decrypted for inspection with:

- open source tools like “mitmproxy + SSLsplit”
- commercial HTTPS middlebox (Cisco, BlueCoat, Fortinet, Symantec, Microsoft, McAfee, etc..)

Tools and appliances can use a valid certificate intercept traffic but **in MANY cases vulnerabilities are introduced.**

TLS Interception

US-CERT, Alert **TA17-075A**, March 16, 2017
HTTPS Interception Weakens TLS Security

- *“All systems behind a hypertext transfer protocol secure (HTTPS) interception product are potentially affected.”*

Research: **The Security Impact of HTTPS Interception**

- *“..62% of traffic that traverses a network middlebox has reduced security and 58% of middlebox connections have severe vulnerabilities.”*

Interception

If a valid certificate is used by the tool/appliance intercepting the traffic how can we detect MITM attacks?

Certificates have an unique FINGERPRINT, it is created by calculating the SHA1 hash of the certificate in DER format.

GRC Fingerprints: <https://www.grc.com/fingerprints.htm>

“The remote server's REAL certificate and the SSL Appliance's FAKED certificate MUST HAVE AND WILL HAVE radically different fingerprints. They will not be remotely similar.”



Interception

Traffic MAY be intercepted (MITM) by employers, schools, governments, or whatever organisation is providing the connection.

If an appliance or a software (like an antivirus) is used then it is very likely that the connection has reduced security and that serious vulnerabilities have been introduced.

Are there blacklists of bad keys?

Very few, mainly used by IDS systems, few high-end proxy appliances and very few open source tools.

Blacklists

- Known private keys used by embedded devices
 - <https://github.com/devttys0/littleblackbox>
- Certificates related to malware
 - <https://sslbl.abuse.ch/blacklist/>
- Debian OpenSSL predictable keys
 - <https://github.com/g0tmi1k/debian-ssh>
- SSH Bad keys
 - <https://github.com/rapid7/ssh-badkeys>



The state of GPG public key cryptography, can we trust it?

To get crypto keys we downloaded all publicly available keys from PGP servers, ending up with around 9 million keys.

In theory:

- keys are created using very large prime numbers.
- keys are unique and **NEVER** share primes or modulus
- no number will be less than 100 digits long, no number will be equal to 1, and only prime numbers are used.

If any of the assumptions is false, a key is **insecure**.

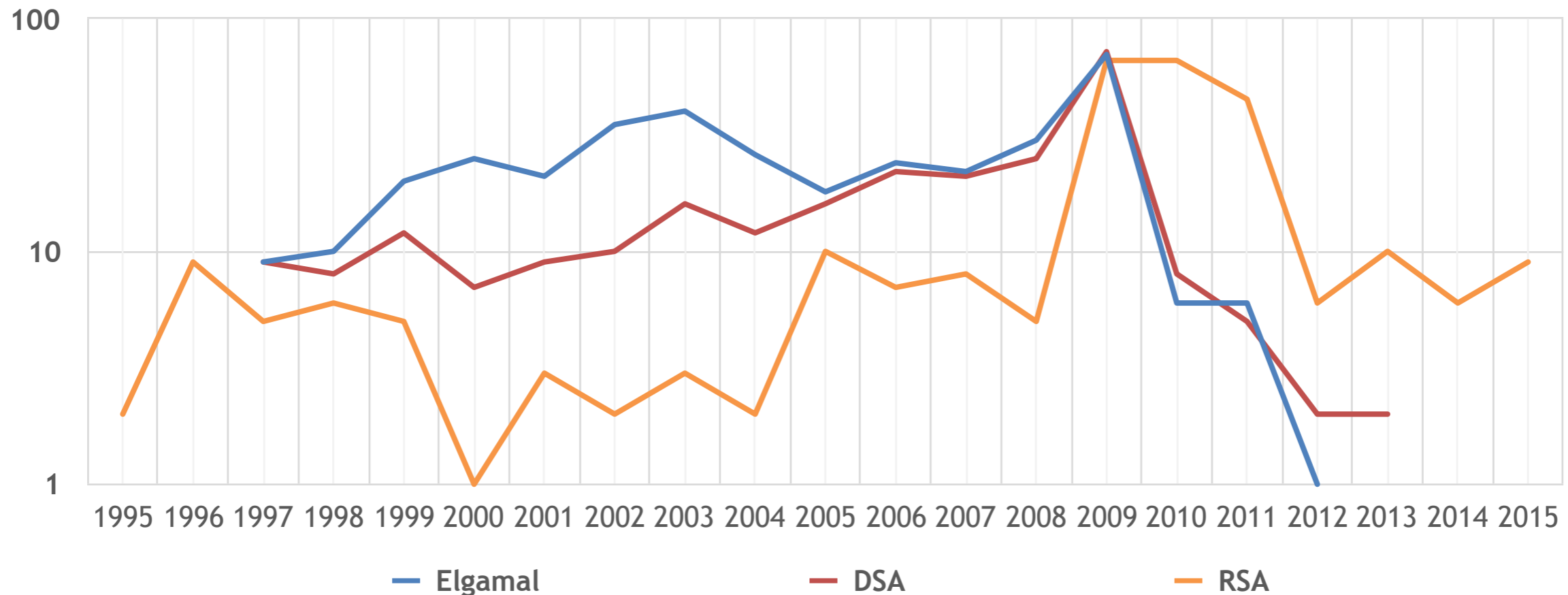
And why we should care about insecure keys?

If a key is insecure/broken, **encryption offers no protection**.

GPG / PGP

A data set (SKS dump) of 8.932.412 keys contained a total of **894 (0.01 %) insecure** keys that should be regenerated.

Vulnerable PGP Keys by alleged year of generation (log scale)



Can we trust these numbers?

Not really, as **key servers have do not guarantee key integrity.**

GPG / PGP

PGP keys with shared primes

Elgamal	357
DSA	255
RSA	263

PGP Keys with small divisors

Elgamal	363
DSA	255
RSA	266

Modulus bit length

<u>Bits</u>	<u>Count</u>
768	2
1024	179
2048	216
3072	7
4096	134
16384	1

Top 6 Divisors

<u>Divisor</u>	<u>Count</u>
641	603
6700417	603
3	320
5	170
7	127
2	109

In PGP/GPG you can also have “fingerprint collision” as demonstrated by the “Evil32” attack (<https://evil32.com/examples.html>)

```
$ gpg --keyserver pgp.mit.edu --recv-key 4BD6EC30
gpg: requesting key 4BD6EC30 from hkp server pgp.mit.edu
gpg: key 4BD6EC30: public key "Puppet Labs Release Key (Puppet Labs Release Key)"
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:          imported: 1 (RSA: 1)
```

Good Key

```
$ gpg --keyserver pgp.mit.edu --recv-key 4BD6EC30
gpg: requesting key 4BD6EC30 from hkp server pgp.mit.edu
gpg: key 4BD6EC30: public key "Puppet Labs Release Key (Puppet Labs Release Key)"
gpg: key 4BD6EC30: public key "Puppet Labs Release Key (Puppet Labs Release Key)"
gpg: no ultimately trusted keys found
gpg: Total number processed: 2
gpg:          imported: 2 (RSA: 2)
```

BAD Key

```
$ gpg --list-key --fingerprint 4bd6ec30
pub 4096R/4BD6EC30 2010-07-10 [expires: 2016-07-08]
   Key fingerprint = 47B3 20EB 4C7C 375A A9DA E1A0 1054 B7A2 4BD6 EC30
uid  Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>
```

Good Key

```
$ gpg --list-key --fingerprint 4bd6ec30
pub 4096R/4BD6EC30 2010-07-10 [expires: 2016-07-08]
   Key fingerprint = 47B3 20EB 4C7C 375A A9DA E1A0 1054 B7A2 4BD6 EC30
uid  Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>

pub 4096R/4BD6EC30 2010-07-10 [expires: 2016-07-08]
   Key fingerprint = 22A6 C997 D0F3 2A3D 984B BE13 0F65 842D 4BD6 EC30
uid  Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>
```

BAD Key

GPG / PGP

We can follow best practices and verify both keys....

```
$ gpg --verify puppet-3.6.2.tar.gz.asc puppet-3.6.2.tar.gz
gpg: Signature made Sat 12 Jul 2014 11:44:22 PM EDT using RSA key ID 4BD6EC30
gpg: Good signature from "Puppet Labs Release Key (Puppet Labs Release Key) <inf
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 22A6 C997 D0F3 2A3D 984B BE13 0F65 842D 4BD6 EC30
```

Good Key

But key verification show very similar output for both keys!

```
$ gpg --verify puppet-3.6.2.tar.gz.asc puppet-3.6.2.tar.gz
gpg: Signature made Tue 10 Jun 2014 12:44:55 PM EDT using RSA key ID 4BD6EC30
gpg: Good signature from "Puppet Labs Release Key (Puppet Labs Release Key) <inf
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 47B3 20EB 4C7C 375A A9DA E1A0 1054 B7A2 4BD6 EC30
```

BAD Key

The people behind the “Evil32” project used and GPUs to compute valid keys that have same 32 bit fingerprint (last 8 chars of full fingerprint) and same metadata, only key values (not visible..) and full fingerprint are different!

It is VERY difficult to spot GOOD and BAD keys.



GPG / PGP

In this example the network has NOT been compromised.

What has been attacked is the key in the key servers, and an attacker does not have to interact with the key owners.

Does it happen in the wild? **Yes.**

Has this kind of attack been used in target attacks? **Yes.**

Is there a way to protect key servers? **No.**

Can key owners do anything about this? **No.**

Can key owners delete the bad keys from servers? **No.**

Is there something to monitor/study this? **No.**

Are there software libraries vulnerable to this? **Yes, lots!**

Is there a solution to this? **No, use ONLY full fingerprints!**



Software companies and version controls, useful or useless?

Revision control systems (from wikipedia):

“In computer software engineering, revision control is any kind of practice that tracks and provides control over changes to source code.”

In theory any version control system should:

- keep track of added/changed/deleted items
- keep track of the time/sequence of each operation
- make sure that nothing is silently corrupted/changed

An RCS is NOT a backup system.



lets' talk about git...

Some little known facts that, if used maliciously, can be used (and have been used) to corrupt git repos.

- Commit dates

There is no guarantee (and no check) that the commit date you use is the one recorded in the main repo.

A commit with date 0 or NUL is also possible, and depending on server configuration can be stored as is, or changed to a date that the server likes..

For example a commit to a public github repo had a commit dated "May 05, 2017" but was 'recorded' as "Feb 12, 2017".



lets' talk about git...

Git can be considered a ledger, it stores a list of “stuff” that happened over time but has NO concept of files and nothing to ensure data integrity or protect against data corruption.

- Files and file names

There is no logic to identify specific file names, if two files have the same name then you can have a collision (this can be sorted out using a script to compare full file path, date and size, and find the right blob...not straight forward).

- commit size / commit hash

You can make a valid commit with size zero and hash zero.

“00 000000”

lets' talk about git...

Attack: is possible to make a commit with nothing/NUL and with date 0/NUL that can be accepted, the repository is silently corrupted, content is unreachable but still referenced.

- object/repo size

There is no command that shows how much stuff is in the repo, the one available are NOT accurate.

```
git clone https://github.com/git/git.git
```

```
git count-objects -vH | grep '-pack:' —> in-pack: 179614, size-pack: 72.70 MiB
```

```
git bundle create tmp.bundle --all
```

```
Writing objects: 100% (179614/179614), 67.90 MiB | 94.35 MiB/s, done.
```

```
Total 179614 (delta 129674), reused 179614 (delta 129674)
```

```
du -sh tmp.bundle —> 68M tmp.bundle (CORRECT)
```



lets' talk about git...

- version history

History can be changed and cannot be implicitly trusted.

The log can also be changed and is possible to “dereference” objects so nothing is pointing to them. Depending on git server config, if git finds objects that have not been linked/referenced for 30 day, it will DELETE the reference in the reflog and then the garbage collector will remove the data.

Attack:

It is possible to introduce valid commits that are pointing to themselves and commits that create ‘loops’ in the history, with areas of the reflog pointing to non referenced items. When a reference is invalid it will be removed and deleted.



lets' talk about git...

- Trailing whitespace / new blank line at EOF

If a line has a trailing white space, and a patch is committed by email, and you have pre-hooks enabled, you can have conflicting commits, empty commits, and broken code.

<https://stackoverflow.com/questions/1583406/why-does-git-care-about-trailing-whitespace-in-my-files>

<https://stackoverflow.com/questions/27059239/git-new-blank-line-at-eof>

To test how common are these kind of issues, I have downloaded and analysed **10 million repositories**.

In total I found close to **1.8 million (18%)** repositories with commits that are logged but inaccessible, with commits pointing to each other in (endless) loops, with commits that have NUL or no date, and with empty commits without date.

Data breach or "indirect" breach

What else can be found by looking at HTTP/HTTPS response traffic?

There are databases of server responses so I had a look at data from my target IP range, matched it with known signatures and found:

- **1671 Storage devices** (!!!!!)
- **435** IP phones
- **344** PLC (Programmable logic controller)
- **213** Printers
- **133** DVR/Cameras

All of it reachable, open, no password, straight access as root to the device. Some may have been Honeypots (traps) but not all...



Storage

How is possible that so many open storage devices have been found?

Sadly this seems to be a bigger issue than the one related to IoTs, as each device can hold vast amount of data ready to be downloaded by anyone.

This issue was covered briefly introduced by an Australian documentary published by “ABC Four Corners” with the title “Cyber War”.

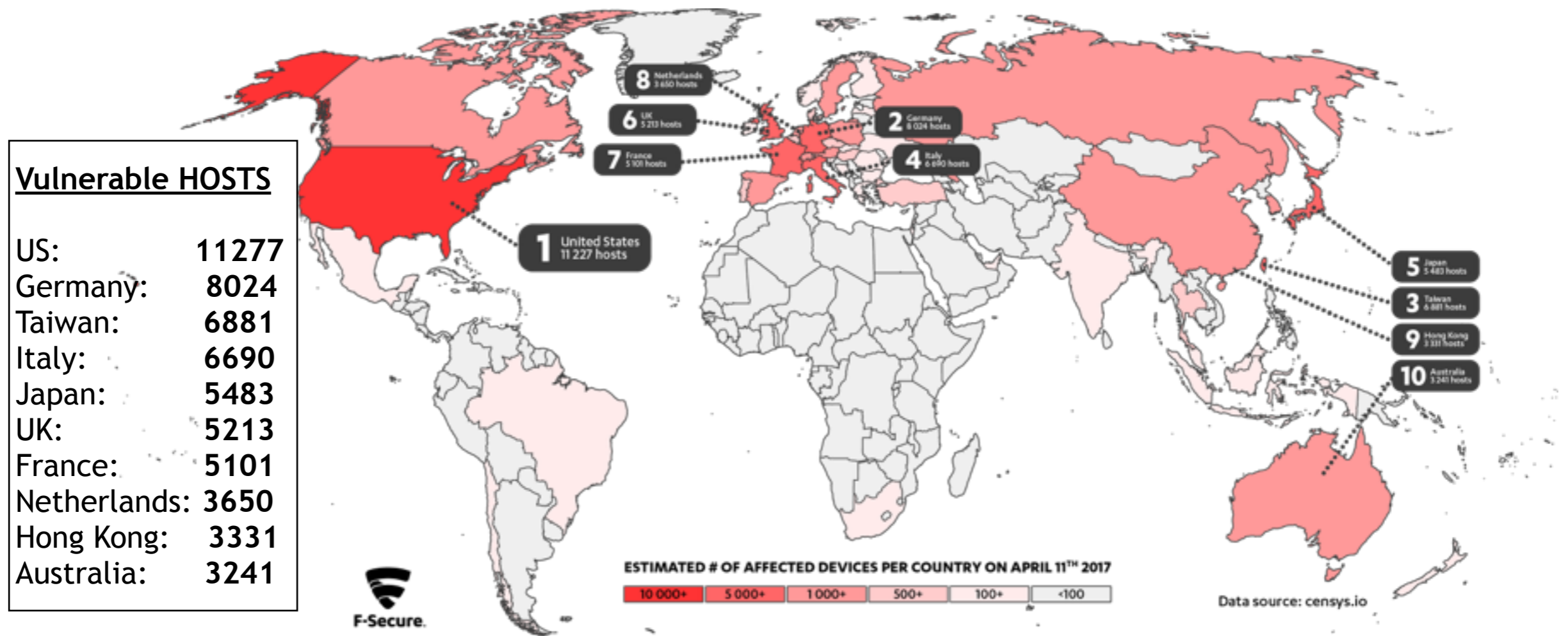
- <http://www.abc.net.au/4corners/cyber-war:-how-hackers-are-threatening-everything/7785800>

“Harry Sintonen” presented a study on QNAP NAS vulnerabilities:

- <https://www.youtube.com/watch?v=Q5Q9VQu3to4>
- <https://safeandsavvy.f-secure.com/2017/04/12/new-nas-vulnerabilities-are-pretty-much-as-bad-as-they-get/>

Storage

All relevant data can be **freely** downloaded from “censys.io” and processed to build a map of insecure NAS units.



<https://fsecureconsumer.files.wordpress.com/2017/04/affected-devices.png>

Enrico Branca - 2017



Storage

What about other storage servers?

And back from the past we have....LOTS of FTP servers !

- **4228** FTP servers (open, anonymous)

What is REALLY interesting is that many of them are NOT answering with the normal handshake so normal tools/scanners will either MISS them or mark them as CLOSED.

So what can be done to get ALL of them? We build an analyser!



Storage

In the analyser we handle all possible ways of SLOWLY and PROPERLY connecting to any form of FTP server.

Then we do the same to get all LISTING format to make sure that the server works and that is able to communicate back.

- **4228** FTP servers (open, anonymous)
 - of which **1321** FTP servers are NOT following standards

Without the logic to handle all kinds of FTP servers we would have missed **31 %** of all servers.



Storage

The amount of data in the FTP servers has not been tested because analyser was built to respect banner messages that requested no access, and no directory listing was performed to respect any country laws.

So far we have **0.01 % (5899)** of all IP that are insecure storage devices, with also many of the non standard FTP servers sitting within corporate networks.

The strange FTP servers appear to be devices that have been compromised and that are 'avoiding' standard scanners....



DNS Servers

While checking client-server interactions I noticed many servers were using DNSSEC and I decided to looking into it.

Of all domain contacted, **18 %** were using DNNSEC and some of them were using DNS firewall. How we know it?

Example: “SOA”: **NXDOMAIN**; “NS”: **NoAnswer**, “A”: **answer**

In theory this would be very good, but in practice in the firewall answers differently depending on query content and type, then it is very easy to reverse the logic and find firewall policies....



DNS Servers

Verifying DNSSEC record was also interesting, a big cloud provider is using a new DNS defence mechanism called “DNSSEC Black Lies”

“..this name does exist, just not on the one type you asked for.”

<https://blog.cloudflare.com/black-lies/>

<https://tools.ietf.org/html/draft-valsorda-dnsop-black-lies-00>

As a defence is very effective and response size is also reduced significantly, preventing NSEC and NSEC3 zone walking. This is important as many DNSSEC records did not validate properly and allowed NSEC zone walking.

Domain allowing NSEC zone walking: 24528 (**24 %**)

Domains with protected DNSSEC records: 511 (**0.5 %**)

What we found without any attack

Looking at the return traffic revealed also something else interesting.

Some companies are using WAFs (web application firewalls) that, to keep track of valid sessions, are ADDING data to the return packets as ETHERNET PADDING (a form of Ether Leak that survives routing).

A simple connection to an open port of a server “protected” by this kind of WAF appliances will immediately **reveal** ‘vendor name’, ‘appliance class’ and the WAF ‘session number’.

All this **in clear text** but visible only to packet analysers.

What we found without any attack

HTTPS - Servers supporting SSLv2	23.800 servers
HTTPS - Servers supporting SSLv3	120.000 servers
HTTPS - Protocol Downgrade	309.500 servers
Broken RSA crypto keys	758.912 keys
Broken/Malformed/Malicious GPG keys	894 keys
Web Storage servers	1671 devices
IP Phones	435 devices
PLC Controllers	344 devices
Printers	213 devices
DVR/Cameras	133 devices
Open FTP (no opensource, no pkg. dist.)	5899 servers
Vulnerable QNAP NAS (high end storage)	58.891 devices
Broken GIT repositories	1.800.000 repos
Insecure DNS servers (NSEC zone walk)	24.528 servers



Lessons learned

Many of the tested domains have been “validated” by third party companies and/or have ISO certifications, yet they fail a basic test that can even be done using free online tools.

There are plenty of tools to test the security of web servers but not all of them are using libraries customised for security testing.

Often happens that a tool performs tests for something that is not supported by the library and gets no results, then a tester assume all is fine while in reality nothing was really tested.

What happen if an individual shares a storage device that has data directly related to her/his workplace? Bad guys can access it but good guys cannot access it or test it because it is private property. What can be done?

*If there are indeed tens of thousands of storage devices out there left unguarded and accessible to anyone, there are **petabytes** of sensitive data accessed/breached every day that will never be reported.*



What to do to make things better

HTTPS:

- test your keys for small primes
- be sure that your security scanner is working properly
- use blacklists and/or use an IDS that can check RSA keys (Suricata)

GPG/PGP:

- do NOT implicitly trust keys that are in key servers
- if possible, accept a key only after comparing it with the owner

GIT:

- check for empty lines/trailing whitespace
- only one account is the admin
- one commit per change per file, and only one file at a time
- backup your code, as git is NOT a backup solution

DNS: Make sure you validate DNSSEC records, use NSEC3 and not NSEC

WAF: Check if your WAF is leaking session IDs through ETH Padding



Thank you

QUESTIONS?

P.S.: If you are interested in git security, in crypto testing, or in developing custom scanners, please get in touch!

Enrico Branca

<https://www.linkedin.com/in/ebranca>

enrico.branca@awebof.info

enrico.branca@owasp.org