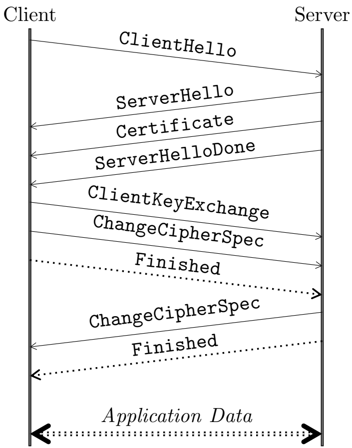
`concerto`: A Methodology Towards Reproducible Analyses of TLS Datasets

Olivier Levillain, Maxence Tury and Nicolas Vivet

ANSSI

Real World Crypto
January 6th 2017

SSL/TLS in a nutshell

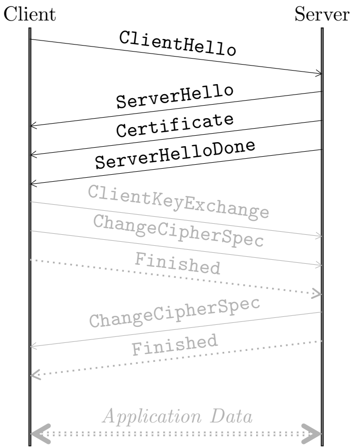


SSL/TLS: a security protocol providing

- server (and client) authentication
- data confidentiality and integrity

SSL/TLS is a fundamental basic block of Internet security

SSL/TLS data collection



Interesting criteria to study the ecosystem

- protocol features and cryptographic capabilities
- certificates and trust aspects
- server behaviour

Different methodologies

- Full IPv4 scans
- Domain Names scans
- Passive Observation

Stimulus choice (version, suites, extensions)

`concerto`: motivation

The tools used to produce the data for [ACSAC'12]

- `parsifal`, a home-made parser generator, to parse the answers
- (mostly undocumented or even not versionned) various scripts

In 2015, we tried to run similar analyses on new campaigns

- problem: several criteria had to evolve (trust stores, weak suites)
- how to compare the situation now and then?
- how to include new, external, datasets?

The `concerto` way, towards reproducible analyses

- keep the raw data and the associated metadata
- automate the analysis process
- run it from scratch when needed

concerto, step by step

Context preparation

- NSS certificate store extraction from source code
- metadata injection (stimuli, certificate store)

Answer injection

- answer type analysis
- raw certificate extraction

Certificate analysis

- certificate parsing
- building of all* possible chains

Statistics production

- TLS parameters, certificate chain quality, server behavior

Interlude: challenges with the data quality

What can a TLS server answer to a client proposing the following ciphersuites: `AES128-SHA` and `ECDH-ECDSA-AES128-SHA`?

Interlude: challenges with the data quality

What can a TLS server answer to a client proposing the following ciphersuites: `AES128-SHA` and `ECDH-ECDSA-AES128-SHA`?

A `AES128-SHA`

B `ECDH-ECDSA-AES128-SHA`

C an alert

Interlude: challenges with the data quality

What can a TLS server answer to a client proposing the following ciphersuites: `AES128-SHA` and `ECDH-ECDSA-AES128-SHA`?

- A `AES128-SHA`
- B `ECDH-ECDSA-AES128-SHA`
- C an alert
- D something else (`RC4_MD5`)

Interlude: challenges with the data quality

What can a TLS server answer to a client proposing the following ciphersuites: `AES128-SHA` and `ECDH-ECDSA-AES128-SHA`?

A `AES128-SHA`

B `ECDH-ECDSA-AES128-SHA`

C an alert

D something else (`RC4_MD5`)

- sadly, this can be explained
- worth mentionning: some servers select the NULL ciphersuite

Interlude: challenges with the data quality

What can a TLS server answer to a client proposing the following ciphersuites: `AES128-SHA` and `ECDH-ECDSA-AES128-SHA`?

A `AES128-SHA`

B `ECDH-ECDSA-AES128-SHA`

C an alert

D something else (`RC4_MD5`)

- sadly, this can be explained
- worth mentionning: some servers select the NULL ciphersuite

E a `ServerHello` *missing* two bytes

Interlude: challenges with the data quality

What can a TLS server answer to a client proposing the following ciphersuites: `AES128-SHA` and `ECDH-ECDSA-AES128-SHA`?

A `AES128-SHA`

B `ECDH-ECDSA-AES128-SHA`

C an alert

D something else (`RC4_MD5`)

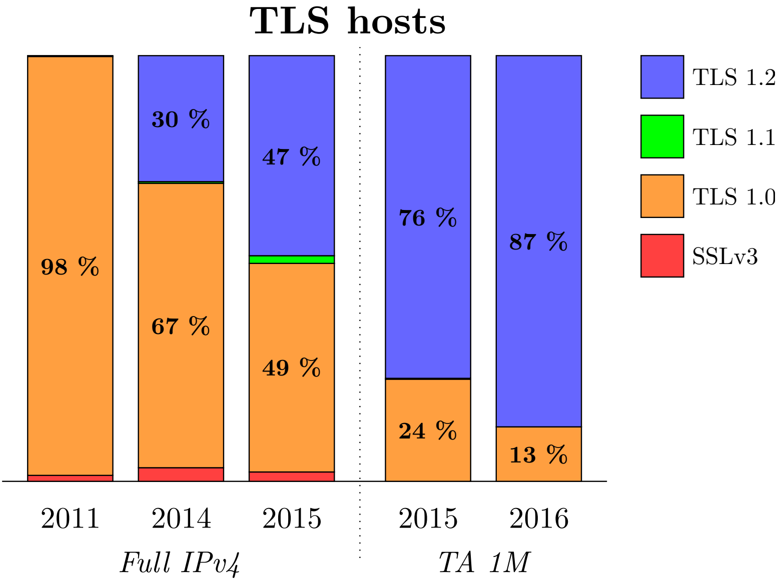
- sadly, this can be explained
- worth mentionning: some servers select the NULL ciphersuite

E a `ServerHello` *missing* two bytes

Our answers:

- `parsifal`, an open-source framework, to develop robust binary parsers
- use metadata (the used stimulus), to spot inconsistencies

Evolution of TLS versions



Certificate chains: theory and practice

The `Certificate` message is specified as follows:

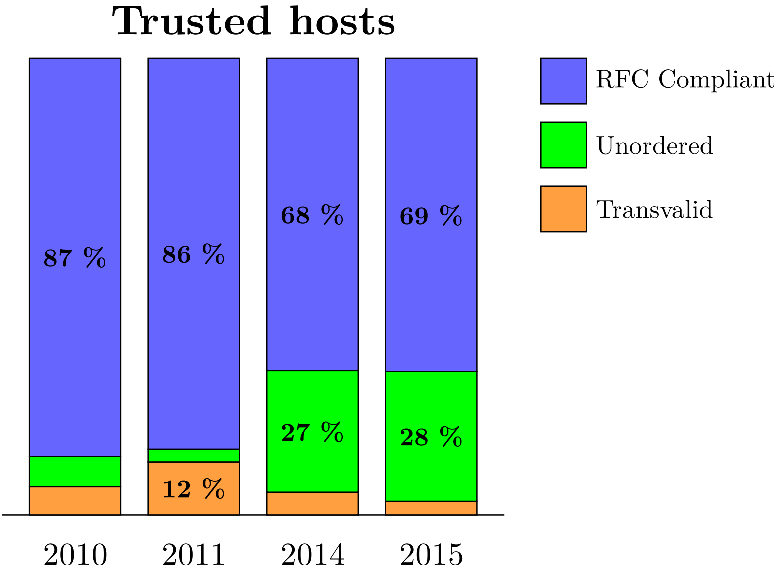
- the server certificate first
- each following CA cert must sign the preceding one
- the root CA may be ommited

The reality is otherwise:

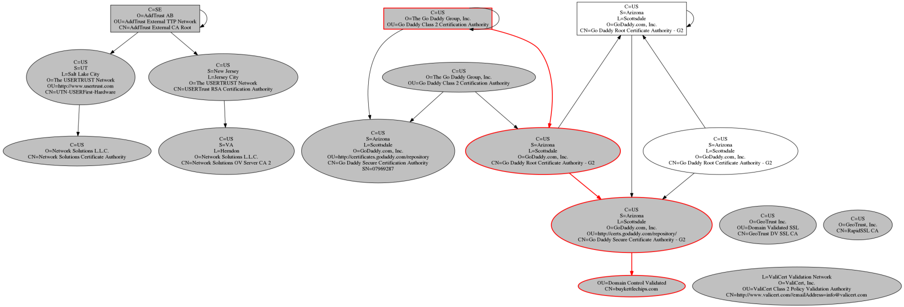
- unordered messages
- certificate repetition
- presence of useless certificates
- missing certificates (EFF calls such chains transvalid)

TLS 1.3 relaxes the strict order constraint

Evolution of certificate chain quality



Exemple of a certificate chain



Challenges in the certificate chain building phase

Actually, `concerto` does *not* build all possible chains, for two reasons

- X.509v1 certificates generated by appliances
  - X.509v1 have no extension, so they used to be considered as CA
  - however, we encounter too many of them in some campaigns
    - 140,000 similar self-signed *distinct* certificates
    - 20 billion signatures to check, for isolated self-signed certificates
  - only X.509v1 trust roots are considered as CAs

Challenges in the certificate chain building phase

Actually, `concerto` does *not* build all possible chains, for two reasons

- X.509v1 certificates generated by appliances
  - X.509v1 have no extension, so they used to be considered as CA
  - however, we encounter too many of them in some campaigns
    - 140,000 similar self-signed *distinct* certificates
    - 20 billion signatures to check, for isolated self-signed certificates
  - only X.509v1 trust roots are considered as CAs
- Crazy cross-certification
  - there exist mutually cross-signed CAs...
  - where each CA has emitted several distinct certificates with the same public key
  - one way to go is to create an equivalence class of CAs
  - the other is to limit the number of transvalid certificates

Interlude: some figures about certificates

RSA Key Sizes (full IPv4 scan in 2015)

- (TLS hosts) 384 - 16384
- (Trusted hosts) 1024 - 4096

Maximum observed size of a `Certificate` messages (EFF data in 2010)

- 150 certificates
- including (only) one duplicate
- including 113 trusted roots

Misc (from 2017 HTTPS TopAlexa 1M `scans.io` data)

- 9% RSA-SHA1 signatures (and 976 RSA-MD5)
- 5% X.509v1 certificates (and 3 X.509v4)

Server behaviour

You can take advantage of multiple stimuli to grasp server behaviour

Feature intolerance

- Using our IPv4 multi-stimuli campaigns (2011 and 2014)
- EC- and TLS 1.2-intolerance has regressed between 2011 and 2014

SSLv2 support

- 40% of HTTPS servers were still accepting SSLv2 in 2014
- all vulnerable to DROWN attack
- the situation was worse in practice (SMTPS servers in particular)

Implementation choices, limitations and future work

Current `concerto` design rationale

- store enriched data in CSV tables
- split data processing into simple tools
- avoid tools requiring a global view when possible

Future work

- more sophisticated backends
- more polished statistics and report tools
- inclusion of other relevant data sources (e.g. revocation info, CT)

Conclusion

To analyse the SSL/TLS ecosystem, we need

- up-to-date high quality data
  - with clean collection methodologies
  - with associated metadata
  - possibly using multiple stimuli
- methodologies and tools to allow for reproducible analyses
  - to compare results regarding different datasets
  - to understand trends on relatively long periods

`concerto` is a first step to accomplish the second part

- `parsifal` and `concerto` v0.3 are available online
- there is some documentation on the GitHub repository
- don't hesitate to drop a mail if you are interested in the tool

Questions ?

Thank you for your attention

olivier.levillain@ssi.gouv.fr

https://github.com/ANSSI-FR/parsifal
https://github.com/ANSSI-FR/concerto

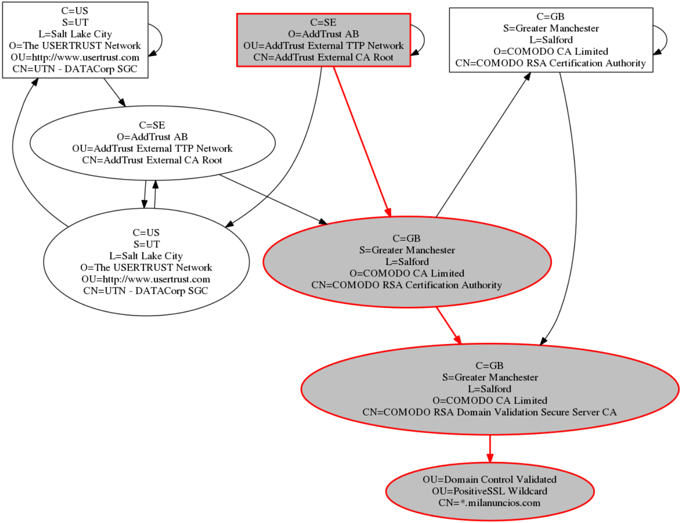
More information and results in my PhD thesis
https://www.ssi.gouv.fr/publication/une-etude-de-lecosysteme-tls/
(manuscript in English, beyond the page in French)

Backup slides

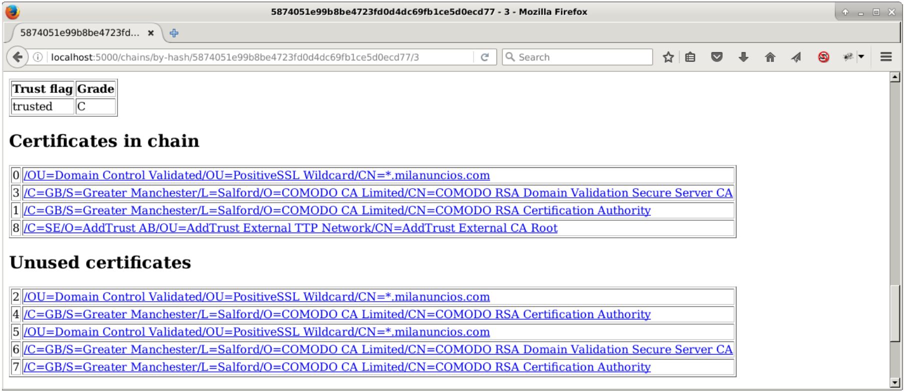
Typical figures for a full IPv4 HTTPS campaign

Table	N rows
Server answers	40 M
(including TLS answers)	30 M
Distinct `Certificate` messages	20 M
Parsed certificates	10 M
Unparsed certificates	100
Verified links	14 M

More certificate examples



More certificate examples



Trust flag	Grade
trusted	C

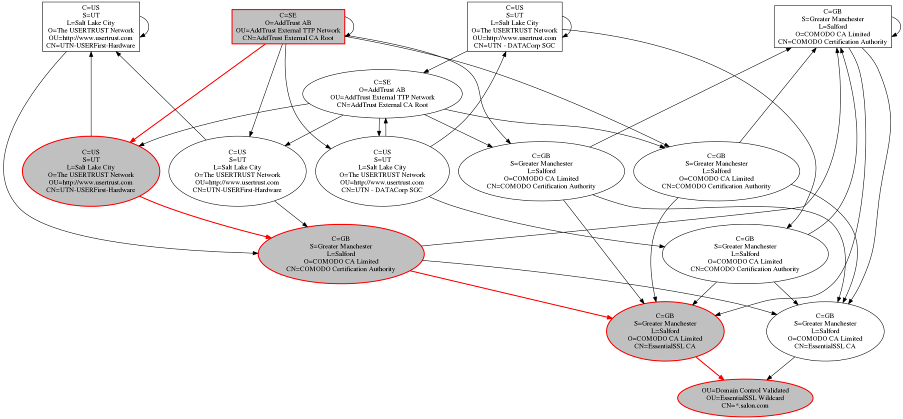
Certificates in chain

0	/OU=Domain Control Validated/OU=PositiveSSL Wildcard/CN=*.milanuncios.com
3	/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Domain Validation Secure Server CA
1	/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Certification Authority
8	/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root

Unused certificates

2	/OU=Domain Control Validated/OU=PositiveSSL Wildcard/CN=*.milanuncios.com
4	/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Certification Authority
5	/OU=Domain Control Validated/OU=PositiveSSL Wildcard/CN=*.milanuncios.com
6	/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Domain Validation Secure Server CA
7	/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Certification Authority

More certificate examples



More certificate examples

e4cf0f8285f0a89b8e6dc075721373fc67c1b75d - 29

/OU=Domain Control Validated/OU=EssentialSSL Wildcard/CN=*.salon.com

Complete	True
N Transvalid	0
Ordered	True
Chain validity period	2011-05-23 00:00:00 --- 2013-05-22 23:59:59

Grades

Trust flag	Grade
trusted	A

Certificates in chain

0	/OU=Domain Control Validated/OU=EssentialSSL Wildcard/CN=*.salon.com
1	/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=EssentialSSL CA
2	/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO Certification Authority
3	/C=US/S=UT/L=Salt Lake City/O=The USERTRUST Network/OU=http://www.usertrust.com/CN=UTN-USERFirst-Hardware
4	/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root

Chain seen 1 times

Campaign	Location		Timestamp	Valid?	Link to the chain
1	salon.com	54.85.20.41	2015-08-14 06:01:23	False	e4cf0f8285f0a89b8e6dc075721373fc67c1b75d

Analysing the certificate chains

To analyse these chains properly, `concerto` uses the following tools:

- `inject`
- `injectAnswers`
- `parseCerts`
- `prepareLinks`
- `checkLinks`
- `buildChains`

Analysing the certificate chains

To analyse these chains properly, `concerto` uses the following tools:

- `inject` to record trust CAs from your reference store
- `injectAnswers` to parse server messages and extract certificates
- `parseCerts` to parse the certificates
- `prepareLinks` to identify the possible links between certificates
- `checkLinks` to check the cryptographic signature
- `buildChains` to try and built all* the possible chains