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

Olivier Levillain
ANSSI
olivier.levillain@ssi.gouv.fr

Maxence Tury
ANSSI
maxence.tury@ssi.gouv.fr

Nicolas Vivet
ANSSI
nicolas.vivet@ssi.gouv.fr

***Abstract*—Over the years, SSL/TLS has become an essential part of Internet security. As such, it should offer robust and state-of-the-art security, in particular for HTTPS, its first application. Theoretically, the protocol allows for a trade-off between secure algorithms and decent performance. Yet in practice, servers do not always support the latest version of the protocol, nor do they all enforce strong cryptographic algorithms. To assess the quality of HTTPS and other TLS deployment at large, several studies have been led to grasp the state of the ecosystem, and to characterize the quality of certificate chains in particular.**

In this paper, we propose to analyse some of the existing data concerning TLS measures on the Internet. We studied several datasets, from the first public ones in 2010 to more recent scans. Even if the collection methodology and the used tools vary between campaigns, we propose a unified and reproducible way to analyse the TLS ecosystem through different datasets. Our approach is based on a set of open-source tools, `concerto`.

Our contribution is therefore threefold: an analysis of existing datasets to propose a unified methodology, the implementation of our approach with `concerto`, and the presentation of some results to validate our toolsets.

I. Introduction

SSL (Secure Sockets Layer) is a cryptographic protocol designed by Netscape in 1995 to protect the confidentiality and integrity of HTTP connections. Since 2001, the protocol has been maintained by the IETF (Internet Engineering Task Force) and has been renamed TLS (Transport Layer Security).

SSL/TLS primary objective was to secure online-shopping and banking web sites. With the so-called Web 2.0, its usage has broadened drastically: services provided by Google, Yahoo!, Facebook or Twitter now offer a secure access using TLS. Furthermore, other services like SMTP or IMAP benefit from the security layer; there also exists several VPN (Virtual Private Network) implementations relying on SSL; finally, some Wifi access points use TLS as an authentication protocol (EAP-TLS).

Several flaws have been discovered in TLS, leading to revisions of the standard. Moreover, TLS is subject to various configuration and implementation errors. As TLS usage is so ubiquitous on the Internet, it is legitimate to assess its security. A lot of studies have been led since 2010 to test the behaviour of HTTPS servers world wide. Recently more tools and scans have been made available, allowing for more data to be collected.

Yet, the conditions used to collect data are not always clearly stated, and some of the proposed analyses were not easy to reproduce. This is why we studied several datasets to apply the same reproducible methodology on them.

Section II briefly describes SSL/TLS to present the basic notions needed to understand how data are usually collected. Then, section III presents the different methodologies of measures used in practice for SSL/TLS. Section IV lists several datasets and analyses that have been published since 2010. Section V describes `concerto`, the toolset we developed to analyse the different datasets in a reproducible and homogeneous way, while section VI presents several challenges we had to face with our tools. Finally, section VII contains results obtained on the different datasets.

II. SSL/TLS in a nutshell

SSL (Secure Sockets Layer) is a protocol originally developed by Netscape in 1995 to secure HTTP connections using a new scheme, `https://`. The first published version was SSLv2, rapidly followed by SSLv3, which fixed major conceptual flaws. SSLv2 and SSLv3 use different message formats.

In 2001, the evolution and the maintenance of the protocol were handed to the IETF (Internet Engineering Task Force) which renamed it TLS (Transport Layer Security). TLS 1.0 can be seen as an editorial update of SSLv3 but the cryptographic janitorial work fixed a weakness in the CBC padding [18]. TLS 1.1 was published in 2006 (fixing another CBC flaw [5]) and TLS 1.2 [4], the current version, in 2008, providing new cryptographic algorithms (GCM mode, SHA2 support). Today, SSLv2 and SSLv3 should not be used anymore, and TLS 1.1 and 1.2 should be preferred. The IETF TLS working group has been working on TLS 1.3 since 2014.

To establish a secure session between a client and a server, SSL/TLS uses handshake messages to negotiate its parameters: the version of the protocol, the cryptographic algorithms and the associated keys. The algorithms are described by *ciphersuites* which define how to authenticate the server, establish a shared secret used to derive keys, and how to encrypt and ensure the integrity of the application data.

A. A typical TLS connection

Figure 1 presents a handshake between a client and a server. First, the client contacts the server over TCP and proposes several versions and ciphersuites; this initial message, `ClientHello`, also contains a nonce. If the server finds an acceptable ciphersuite, it responds with several messages: `ServerHello`, containing the selected version

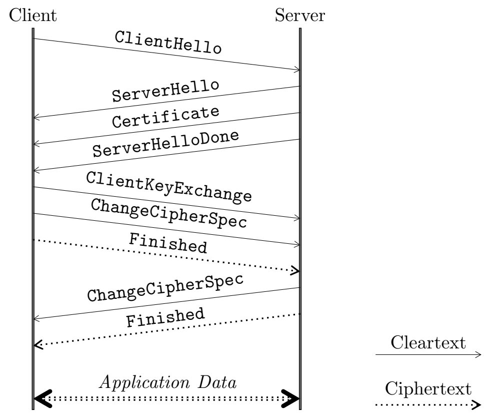

Fig. 1. Example of a TLS negotiation.

and ciphersuite, the `Certificate` message, containing the chain of certificates for the site contacted, and an empty `ServerHelloDone` message ending the server answer. Then, the client checks the certificates received and sends a `ClientKeyExchange` message, carrying a random value encrypted with the public key of the server[1]. At this point, the client and the server share this secret value, since the server can decrypt the `ClientKeyExchange` message. Finally, the `ChangeCipherSpec` messages activate the negotiated suite and keys, and the `Finished` messages ensure the integrity of the handshake *a posteriori*, as they contain a hash of all the handshake messages previously exchanged; they are the first messages protected with the negotiated algorithms and keys.

At any moment, an `Alert` message can be sent to signal a problem, e.g. if no ciphersuite is acceptable, or if the client does not trust the certificate chain.

Existing studies mainly focus on the first flight of server messages (from `ServerHello` to `ServerHelloDone`) since it is composed of cleartext messages containing the parameters chosen by the server (protocol version, cryptographic algorithms, supported extensions) and its certificate chains.

B. X.509 certificates in TLS

The certificates used in TLS follow the X.509 standard. The TLS Public Key Infrastructure is based on several root authorities trusted by default by web browsers. The server certificate is sent in the `Certificate` message, along with its certificate *chain*, that is the ordered set of certificates needed to build, link by link, a certificate path to a trusted root. Figure 2 presents a typical chain, where `S` is the server certificate, `I` is the certificate of an intermediate authority, and `R` is the certificate of a trusted root; it also describes three different `Certificate` messages representing this chain. A conforming TLS implementation must send `S`, `I` and `R`, *in that order* (the first `Certificate` message of figure 2). A server can omit the root certificate since the client needs to know (and trust) `R` to validate the chain[2]. In theory, all other `Certificate` messages should be discarded.

In practice, a lot of TLS servers present an unordered certificate chain (e.g. the second message of figure 2). It is relatively easy to accomodate such messages; moreover, unordered chains are sometimes justified in case there exists two valid certificate chains, leading to different root certificates: what is the correct `Certificate` message to produce for the chain to be accepted by clients trusting at least one of the root certificates? This matter was discussed within the TLS working group, and the order constraints should be relaxed in TLS 1.3.

Another problem with chains is that a smaller proportion of servers even omit intermediate certificates (as in the third message of the figure). To be able to build a chain in this case, the client has to grab missing certificates using out-of-band mechanisms (by relying on an X.509 extensions when present[3], or maintaining a cache of previously seen certificates).

Some implementations, like the Java TLS stack, do not try and repair broken `Certificate` messages, which leads to errors when a client uses such a stack. On the contrary, repairing broken chains may be complex to implement and can lead to security flaws, as was recently shown in OpenSSL (CVE-2015-1793).

For these reasons, we would like to assess the quality of certificate chains, w.r.t several criteria: cryptographic algorithms, key sizes, validity periods, but also the intrinsic quality of the sent chain (order and completeness).

III. Different methodologies of measures

Gathering data about the HTTPS server ecosystem can be done in several ways:

1) enumerating every routable address in the IPv4 space to find open HTTPS ports (443/tcp) and establish TLS sessions with them;
2) contacting hosts based on a list of hostnames (e.g. Top Alexa 1 Million[4]);
3) collecting real HTTPS traffic from consenting users.

The first method seems to be the most exhaustive, because it tests every IP in the world. It is thus possible to speak with many different TLS implementations to broaden our knowledge of the ecosystem. However, it leads to contacting many non-HTTPS hosts. Also, it does not take into account the popularity of Internet sites: it does not discriminate sites like `google.com` from `random.dyndns.org` or even from an unnamed host (e.g. an ADSL modem).

The second option is more restrictive, but better represents user needs, and the proportion of HTTPS servers among the hosts to contact is higher. Besides, this method is compliant with the SNI (Server Name Indication) extension [1], which allows a client to contact different virtual hosts at the same address.

Finally, the last one is completely passive and is really centered on users' habits. In this case it is important to have access to the traffic of many different consenting users to get

[1]For the sake of simplicity, the negotiation presented here uses RSA encryption as key exchange algorithm, but other mechanisms exist, like DHE-RSA where an ephemeral Diffie-Hellman is signed by the server with its private key.

[2]Actually, this assumption, which is true with the classic Public Key Infrastructure, does not necessary hold with new trust models like DANE [11] where the client may only know *the hash* of the expected trusted root.

[3]However, by construction, those extensions have not yet been checked.

[4]http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

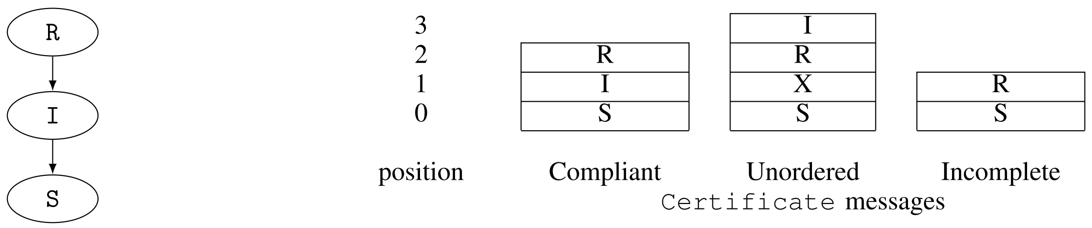

Fig. 2. A typical certificate chain, and different `Certificate` messages representing this chain. The position column indicates the position in the message.

relevant data that would be comparable to other studies. On the one hand, such measures are truly reprentative of real traffic, but on the other hand, the stimuli used to test the servers are not controlled by the experimenter, contrary to the other methods. Table I summarizes the differences between these methods.

IV. Available datasets

This section presents some TLS datasets we studied. It only focuses on HTTPS datasets, and it is not an exhaustive list of all the existing studies, but this inventory helped us gather information about the expected properties of datasets to allow for relevant analyses.

A. 2010: the first public campaigns

The EFF (Electronic Frontier Foundation) performed two campaigns to explore the *SSLiverse* in August and December 2010. They presented their results about the gathered certificates [8], [9]. They published some of their tools, as well as the data, both processed (the certificate database) and unprocessed (the raw answers). This allowed us to understand their collection methodology and to analyse their data ourselves. These campaigns follow the first methodology described earlier (Full IPv4 space).

B. 2011: analysis of the impact of different points of view

In 2011, Holz et al. from the University of München gathered and studied different data sets: active probing of popular sites, passive monitoring on a 10 GB link and the EFF campaigns [12]. They provided a thorough analysis of the certificates received for each of the campaigns studied. In particular they compared the certificates received by clients from different source addresses and obtained differences in these sets. Using real world traffic is a very interesting source of information and it is complementary to full IPv4 host enumerations.

Since the researchers' datasets were either already available (EFF) or only partial (the collected server certificates for popular sites from different locations), we did not investigate this dataset further.

C. 2010-2011-2014: multiple stimuli to grasp the server behavior

In 2012, we presented another study of the HTTPS ecosystem [17], based on several full IPv4 scans: the EFF dataset and other campaigns we led in 2010 and 2011. Our contribution was to use several `ClientHello` messages to probe the servers, allowing to get information about the servers' behavior against uncommon stimuli. Several statistical results were also presented using different prisms: restricting the datasets to trusted hosts or to EV hosts. We also gathered similar data in 2014.

We used custom Python scripts to scan open 443 ports and to retrieve server answers to different `ClientHello` messages. The data was then analysed with open-source tools based on Parsifal [16].

D. Since 2012: SSL Pulse

Since 2009, Qualys has proposed an online tool called SSL Server Test[5]. In 2010, Ristic presented a SSL survey, based on these tools, focusing on a DNS enumeration of HTTPS hosts [19]. His goal was to assess the quality of TLS answers from servers reachable via a DNS hostname: for example, at the time, SSLv2 was still widely supported, while TLS 1.1 and TLS 1.2 were virtually inexistant. Since April 2012, the study has been extended into SSL Pulse [15], a dashboard aiming at measuring the effective security of SSL on a set of popular sites. SSL Pulse provides statistics on a list of 200,000 servers. Even if these tools give an insight into the HTTPS ecosystem, they do not represent a usable dataset as is.

E. 2012: the Internet Census

In March 2013, an independent researcher published results on an Internet-wide scan on multiple ports in 2012 [3]. To complete this so-called Internet Census, the researcher took control of open embedded devices accross the Internet[6] and used them as a botnet to probe the Internet. The collected data (600 GB of compressed data, 9 TB uncompressed) were made public via BitTorrent.

Concerning TLS data, we observed that the answers were truncated well before the end of the first server flight, making it impossible to get relevant data in an homogeneous way. Moreover, on top of the obvious unethical aspect of the approach, it was shown that the data, though apparently authentic, "suffered from qualitative problems such as methodological flaws or lack of metadata" [14].

F. 2013-2015: ZMap and ZGrab

Since 2013, several tools have been published to lead Internet-scale scans, like ZMap [7], an asynchronous scanner relying on SYN-cookie to track answers to given probes. Such tools allow for scanning the Internet very quickly (in a matter of minutes or hours), which may lead to data loss; the usual

[5]https://www.ssllabs.com/ssltest/

[6]Many online devices expose administration interfaces with default passwords, and the attacker/researcher could easily take control of them.

	Full IPv4	**Hostname-based**	**Passive analysis**
HTTPS servers coverage	Nearly 100 %	Subset of named sites	Subset of visited sites
TLS features coverage (typical number of stimuli)	Limited (10)	Large (100)	Uncontrolled
User representativeness	No	Mostly	Fully
Network invasiveness	Important	Moderate	None
SNI-awareness	No	Yes	Partial
Examples of campaign	[8], [17], [7]	[15], [13]	[12]

TABLE I. COMPARISON OF THE DIFFERENT METHODS TO COLLECT HTTPS DATA.

countermeasures are to randomize the adress space and to reemit packets.

In particular, Durumeric et al. used ZMap and other tools to analyse the HTTPS certificate ecosystem in 2013 [6]. They estimate the packet loss in the scanning phase at 2 %, which they compensate by keeping the IP addresses accross campaigns (they launched 110 campaigns over 14 months).

For our study, we considered two of the recurring datasets provided by Durumeric et al.: the Top Alexa 1M scans and the full IPv4 campaigns, both on the 443/tcp port and with TLS 1.2 `ClientHello`.

G. 2016: multiple stimuli with SNI

In August 2016, we realised several scans using domain names coming from the Top Alexa 1M list with stimuli used by the Firefox browser and other less common stimuli crafted using the scapy TLS implementation [2]. In September 2016, we reproduced the experiment on the `.fr` zone, as provided by the Afnic registry.

For these campaigns, we first resolve the domain names. In case of a resolution failure, a second attempt is made with the `www.` prefix. We thus obtain 99 % of unique IPv4 addresses virtually hosting less than 100 domains each. However, some IP addresses are shared by a large quantity of domains (sometimes more than 100,000). In this case, we take a random sample of only 10 domains.

Then, for each stimulus, we perform an HTTPS scan on every retained domains using a tool called `probe_server` [16]. The program specifies the requested virtual host via the SNI extension by dynamically adapting the stimulus. Finally, the answers are written as received, even in case of non TLS compliant responses, in a dedicated binary format for latter analysis with `concerto`.

These campaigns were initiated from an Autonomous System and an IPv4 prefix dedicated and documented in the Internet Routing Registries for Internet measurements. The bandwidth was closely monitored to ensure no data loss.

H. Summary

Table II summarizes the studied campaigns and their properties. To be able to do reproducible analyses and to confront the state of the TLS ecosystem between different points in time, the data collection must follow certain rules. First, as shown in [12], the connections must be issued from a unique source using a well-defined and stable stimulus (it was not the case in the Internet Census 2012).

For entire IPv4 measures, it is important to find an acceptable trade-off between the duration of the campaign and the packet loss. As the time spent acquiring the data (the exposure time) increases, more IP addresses can appear or disappear during the campaign: we would prefer this time to be as short as possible. On the contrary, sending too many packets can overload some network links or trigger alarms, leading to dropped packets. Early campaigns usually spanned their measures accross two to three weeks, but more recent datasets tend to choose shorter exposure time (around 1 day). To accomodate packet loss, it is necessary to randomize the IPs; it is also possible to send multiple SYN packets or to remember past results in case of recurrent measures.

With such guarantees on the collection phase, we can work on the data (the messages sent by the servers). Usually, we will also need the following metadata:

- the `ClientHello` message used as stimulus for the campaign, to check the conformance[7] and the quality of the server answer;
- the timestamp of each answer, to validate the certificate at retrieval time;
- one (or several) certificate store(s) to check the validity of the certificate chain against a given configuration.

V. `CONCERTO`: REPRODUCIBLE ANALYSES OF TLS DATASETS

Armed with different datasets, we propose a modular methodology to allow for reproducible analyses. Our goal is to feed a database with the data and the relevant metadata to produce statistics and to allow for finer-grain requests.

The different phases of our methodology are the following (appendix A pictorially describes the cinematics of our tools): context preparation, answer injection, certificate analysis and statistics computation.

A. Context preparation

Two simple tools are used to make the database aware of the campaign context: `injectStimulus` and `injectCerts`.

The first one documents the versions, ciphersuites and extensions proposed by the client during the campaign. It will allow us to spot anomalies as those described in [17], i.e. when a server chooses a ciphersuite that was not proposed.

[7]Strange as it may seem, some servers, presented with an impossible choice of ciphersuites, will choose a ciphersuite that was not proposed by the client.

	Campaign type	Date	Available	Retained
EFF [8]	1	2010	yes	yes
Holz et al. [12]	1 + 2 + 3	2011	partially	no
our IPv4 campaigns [17]	1	2010, 2011 and 2014	privately	yes
SSLPulse [15]	2	recurring since 2012	no	no
Internet Census [3]	?	2012	yes	no
Durumeric et al. [7], [6]	1 + 2	recurring since 2013	yes	yes
our SNI campaigns	2	2016	yes	yes

TABLE II. CAMPAIGN DESCRIPTION. TYPES ARE 1 FOR FULL IPV4 CAMPAIGNS, 2 FOR HOSTNAME-BASED SCANS AND 3 FOR PASSIVE ANALYSES; THE LAST COLUMN INDICATES THE STUDIED DATASETS.

The second tool is the first step allowing to flag certificates, chains and ultimately hosts with a certificate store. Before checking that a given host is trusted according to a certificate store, we need to make sure the corresponding certificates are loaded.

In some cases, it may be useful to compare campaigns using the same certificate store at different times. For example, Mozilla products use the certificate store from the NSS library. We wrote some scripts to extract the trusted certificates from NSS source.

B. Answer injection

Here, the goal is to process the bulk data to extract the relevant data for each host (which can be identified by a domain name or an IP). First, we need to characterize the answer type (non-TLS contents, SSL/TLS alerts or SSL/TLS Handshake messages). Depending on this type, we may gather more information: protocol version, alert type, chosen cipher-suite, certificate chains.

Of course, this step will be different for each dataset. For EFF datasets, we had to parse the messages sent by each server with the `injectAnswerDump` tool; actually, we could reuse the same tools for the datasets used in [17] that the authors shared with us. On the contrary, for ZGrab results obtained from `scans.io`, we had to write a different tool, `injectZGrabResults` to process the compressed custom JSON format (where the messages had already been parsed).

Finally, this step produces an `answers` table containing one line per host, a `chains` table with unique certificate chains and the set of all unique certificates collected for the given campaign in raw format. If the context information about the `ClientHello` used for the considered campaign is present, `injectAnswerDump` can also use it to enrich `answers` and flag inconsistencies.

C. Certificate analysis

In this step, we first parse all certificates with `parseCerts` to extract relevant information such as public keys, distinguished names and some extensions.

Then, we build and check all the existing links, i.e. the list of (s, i) couples where s is a certificate signed by i. This operation, led by `prepareLinks` and `checkLinks`, is the most expensive operation, since it considers all the available certificates, with no regard to their origin. The reason we use two different tools is that the second step, checking cryptographic signatures, can be parallelized, whereas the first program only quickly enumerates all possible links based on distinguished names and extensions.

Next, we build all possible chains from the certificate chains presented by the servers in their TLS `Certificate` message. The idea is to start from the server certificate, and to try every possible certificate path using the previously computed links. As a matter of fact, with real-world data, the number of possible chains can be huge because of cross-certification between authorities. That is why we introduced a parameter, `max-transvalid` in our algorithm, to limit the number of out-of-sent-chain certificates that can be used while building a chain[8].

Following the validated links, `flagTrust` starts from trusted certificates to recursively flag trusted certificates and trusted chains. It is possible to call `flagTrust` several times with different trust stores.

Finally, `rateChains` takes into account the previous information (e.g. the order of the chain, the presence of unused certificates and the trust flag) to grade the built chains.

D. Statistics computation

With all the computed tables, we are finally able to produce statistics on whole campaigns, such as the preferred cipher-suites for a given `ClientHello`, the proportion of servers supporting TLS 1.2, or the quality of certificate chains sent by servers. The advantage of our toolset is that these statistics can easily be computed in a reproducible way on diverse datasets, which is obviously interesting with recurring campaigns such as those published on the `scans.io` website. Moreover, the flexibility provided by our trust flags allows for finer-grained statistics on restricted subsets, e.g. hosts presenting an trusted chain or hosts presenting an RFC-compliant certificate chain.

E. Typical figures

Our toolset is made of programs written in OCaml, Python or Shell, that work on a shared directory containing the *database*. In practice, the processed data is stored into simple (but sometimes huge) CSV files. It would be interesting to experiment with other backends, yet this does not seem necessary at this point. Indeed, we only have two ways of processing data in our tools: either we process them one row at a time (e.g. to check possible links) or we need to load the whole table to build a graph (e.g. to enumerate all possible links). In both cases, CSV files suit our needs. Figure 3 presents typical figures for a single-stimulus full IPv4 campaign.

[8]This number does *not* include the root certificate that may legitimately be omitted.

Table	N rows	Size
`answers.csv`	40 M	4 GB
`chains.csv`	20 M	2 GB
`parsed_certs.csv`	10 M	6 GB
`links.csv`	14 M	1 GB
`built_chains.csv`	120 M	12 GB
`trusted_certs.csv`	6 M	300 MB
`trusted_chains.csv`	9 M	450 MB

Binary contents	N	Size
raw certificates	10 M	10 GB

Fig. 3. Some figures regarding a typical full IPv4 campaign.

VI. Challenges

A. Handling X.509v1 certificates

The initial version of the X.509 standard did not include extensions. Since extensions are now used to indicate the fact that a given certificate is a certificate authority (CA), X.509v1 certificate used to be implicitly trusted as CAs. Nowadays, trusted roots do not emit such certificates anymore, but trusted roots themselves sometimes still use the old format.

In an early version of `concerto`, we considered all X.509v1 certificates as CAs. Yet, this led to a huge combinatorial explosion with several classes of certificates: some appliances or virtual machine management engines produce unique X.509v1 certificates for each equipment or instance, with the same subject and the same issuer. Though, with X.509v1 certificates, as we can not rely on extensions such as *SubjectKeyIdentifier* and *AuthorityKeyIdentifier* to remove irrelevant possible links to check, the mere presence of n of such appliances would trigger n^2 certificate pairs to check.

For example, a full IPv4 campaign contains around 1.2 million X.509v1 distinct certificates. Among those, we observe certificate clusters, where the subject and issuer fields are the same for all certificates. The biggest one contain more than 140,000 distinct certificates.

To avoid having to check such an amount of bogus links, we decided to only consider as CAs the X.509v1 certificates included in the trusted roots, that is a dozen certificates. This reproduces the behaviour of modern TLS stacks.

B. Tuning the `max-transvalid` parameter

During our study, we observed that the `max-transvalid` parameter (the number of out-of-sent-chain certificates to use when building possible certificate chains) could lead to large tables. On the contrary, using a small value makes us miss several built chains. In practice, the only programs relying on those built chains are `rateChains` and `computeChainsStats`, the tools computing statistics on chain quality.

To assess the impact of the parameter on chain quality statistics, we experimented with `max-transvalid` values from 1 to 6 on a /8 prefix: we confronted the chain quality analysis to assess the loss of precision. For 1 to 3 values, the number of incomplete chains decreases while the number of transvalid chains increases; after 3, the results does not evolve anymore up to the last value we computed. To avoid producing too much data while avoiding missing too much built chains, we chose to use 3 as our `max-transvalid` parameter for all further analyses concerning chain quality.

C. Complete view of certificate ecosystem

In order to not overload the servers hosting many domains, we have decided to probe only a sample of 10 domains on these servers. This is enough for analysing the answers but not to get a complete view of the available certificates. Among the servers hosting more than 100 domains in the `.fr` zone file and returning a valid TLS answer, 71 % of them seem to always provide the same certificate. The rest (29 % of these highly mutualised servers) take the SNI extension into account and serve different certificates for different domain names. Most of the certificates hosted on these SNI-aware servers are issued by Let's encrypt. Thus one way to solve this issue would be to retrieve additional certificates from another source such as Certificate Transparency repositories.

A dedicated tool, `injectCT`, was thus recently added to the `concerto` tool suite in order to load certificates recorded in Certificate Transparency, and might provide interesting future results.

VII. Some results

`concerto` has enabled us to produce several figures concerning the available datasets. The main goal of this section is to show some of the offered possibilities.

A. Big picture

To test our approach, we considered several datasets, marked as retained in table II. For each of them, we used the NSS trust store that was in the source code during the considered scan.

Table III describes the number of hosts with an open 443/tcp port, and the proportion of TLS answer types in each campaign[9]. There seems to be significant differences between the EFF campaigns and the other datasets. This can be explained by the fact that the SYN scan was first completed before mounting TLS sessions, several weeks later; in the meantime, dynamic IP addresses would have moved. Moreover, it seems the SYN scan was not repeated in December. Finally, the results show that some post-processing was made before publishing the results, to eliminate most of non-TLS hosts. If we look at the other datasets, we observe that the number of TLS-speaking hosts on port 443 is growing. The recent trend to favor HTTPS where possible may be in response to revelations concerning pervasive network monitoring [10].

As stated in section IV-G, the using of SNI extensions for our 2016 `.fr` scans, along with the observed concentration of many hosts over a few IP addresses, prompted us to query only 10 arbitrary domains per IP fronting 100 domains or more. As such, comparisons between this campaign and the previous, exhaustive ones do not seem appropriate. Still, it remains of some interest to note that, with the same stimulus, only 0.6% of the IP addresses exhibited strictly more than one answer type over the different domains they hosted.

[9]For multi-stimuli campaigns, we selected a standard stimuli for this table.

Campaign Date	Campaign Source	N hosts with open port	TLS answers	
Full IPv4 campaigns (1 host = 1 IP)				
2010/07	[17]	21,342,205	9,854,653	**(46 %)**
2010/08	EFF	15,665,414	11,476,107	**(73 %)**
2010/12	EFF	7,777,511	7,706,123	**(99 %)**
2011/07	[17]	26,218,630	11,355,060	**(43 %)**
2014/03	[17]	40,126,218	29,055,642	**(72 %)**
2015/08	scans.io	46,231,300	34,087,781	**(74 %)**
Top Alexa 1 Million campaigns (1 host = 1 domain name)				
2015/08	scans.io	684,365	357,282	**(52 %)**
2016/08	Our scans	676,024	604,529	**(89 %)**

TABLE III. Proportion of TLS answers for each campaign. TA stands for Top Alexa 1M, and FR for the .fr DNS zone.

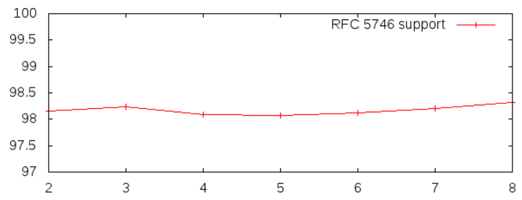

Fig. 5. Proportion of TLS/trusted hosts supporting secure renegotiation.

B. Trends concerning TLS parameters

TLS 1.2 support: In our datasets, we use TLS 1.2 stimuli in 2011, 2014, 2015 and 2016. Figure 4 present the protocol version chosen by TLS and Trusted hosts for each of the relevant IPv4 campaigns. The figures show that TLS 1.2 support went from almost non-existent in 2011 to a majority of servers among trusted hosts in 2016.

RFC 5746 support: Another simple trend to identify using `concerto` relates to the proportion of servers supporting secure renegotiation among Top Alexa 1M servers. For example, we downloaded monthly ZGrab results for February to August 2016 and ran our toolsets to extract information about RFC 5746 support (the secure renegotiation extension). Figure 5 contains the results.

C. Some figures about certificates

Looking at 2010, 2011, 2014 and 2015 full IPv4 campaigns, we could produce some statistics on certificates, presented in table IV. We first count unique certificates for each selected campaign. Then for 2010/2011 and 2014/2015, we count the number of common certificates within a one-year period.

	2010	2011	2014	2015
N unique certificates	4,772,324	5,332,427	9,939,328	10,785,323
Common certificates	1,995,691		3,299,539	
Proportion of common certs	42 %	37 %	33 %	30 %

TABLE IV. Number of unique certificates across campaigns.

Chain robustness: For all the full IPv4 campaigns we considered, certificate chains overwhelmingly contained only RSA keys. We thus focus on the RSA-only certificate chains and we consider the following criterion: the minimum length of all the moduli in the certificate chain. We call this the chain robustness. Figure 6 represent the repartition of said robustness for the studied campaigns.

The median robustness went from 1024 bit to 2048 between 2010 and 2015. For trusted hosts, it is also true for the first quartile. This means that 2048 bit has finally become the new standard. The change was a little quicker for trusted hosts, probably because the CA/Browser Forum required Extended Validation certificates to be at least 2048 bit long since December 2010.

For all the considered campaigns, the maximum key robustness was 16384 bit for TLS hosts and 4096 bit for trusted hosts.

It is however worrying that the minimum key robustness was 512 until 2014 for trusted hosts (it was 384 until 2014 for TLS hosts). Even if the situation improved in 2015, where the minimum was 1024 bit, such weak RSA keys should never be used today.

Statistics on incomplete/unordered chains: In section II-B, we saw how chains are supposed to be sent in `Certificate` messages. Yet, for various reasons, certificates are not always ordered, and some links may even be absent.

Table 7 presents proportion of RFC-compliant, unordered and incomplete chains encountered in different full IPv4 campaigns. We also refined our analysis by restricting the statistics to trusted hosts. For both subsets, it seems unordered chains are becoming more common, which can be explained by server certificates linking to multiple trust anchors.

Further results with Neo4j: While we used several custom scripts to produce the previous results, storing the processed data into CSV files also allows for easy importing into third-party applications. We were quickly able to feed selected metadata from the answers of a TLS campaign into a Neo4j graph database. Our goal was to produce additional results through Cypher queries. As a graph-oriented query language, it seemed like a natural candidate for certificate chains probing.

Following simple `cut` operations which isolated the relevant columns from the CSV answers, chains and certificates produced from one of our Top Alexa 1M scans, we build the nodes and relationships of a new Neo4j database. Using the provided `neo4j-import` tool with standard processing capacities, the whole operation takes no more than 20 seconds. We proceed in creating an index on the certificate hashes.

We present two results among several one-liner queries, which demonstrate the usefulness of Neo4j. Table V displays the reuse rate of server certificates over the subset of 602,875 hosts which presented a TLS `Certificate` message: the most reused certificate is thus presented on almost 9,000 domain names. From the same campaign, table VI shows the prevalent CAs for the issuance of the 298,985 distinct non-self-signed leaf certificates.

D. Study of some intolerances

When sending TLS `ClientHello`s, one would expect the sum of hosts answering with alert and handshake messages to stay unchanged across stimuli. Yet, we observed that under circumstances such as TLS 1.2 `ClientHello`s, several servers seem to panic and either abort the connection abruptly, or answer with an irregular `ServerHello` (e.g. with a cipher suite that was not advertised in the first place). This phenomenon is designated as *server intolerance*.

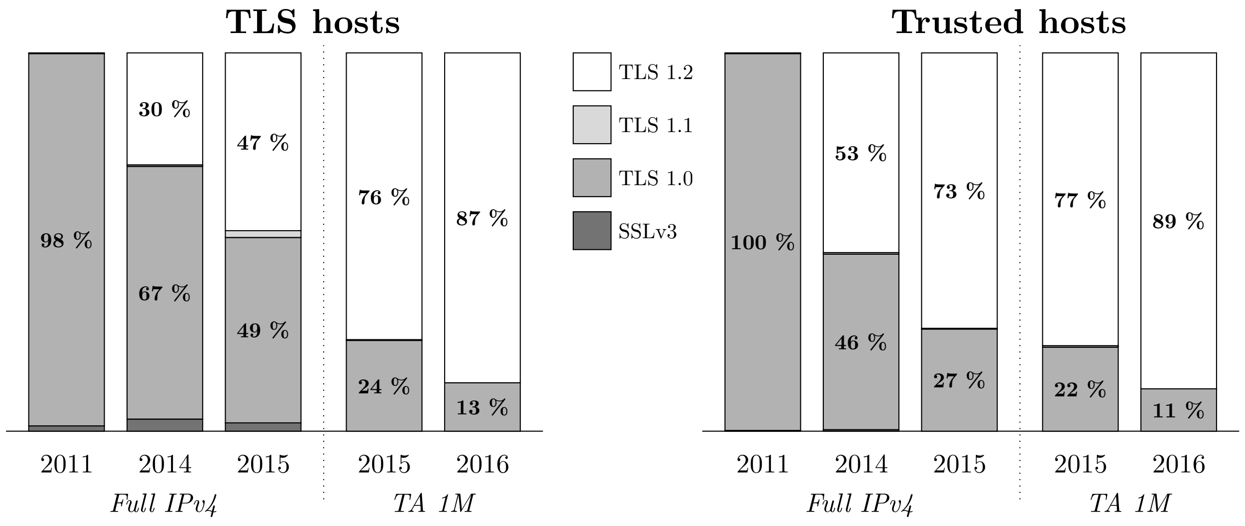

Fig. 4. Evolution of the protocol versions chosen by the servers (for TLS and trusted hosts).

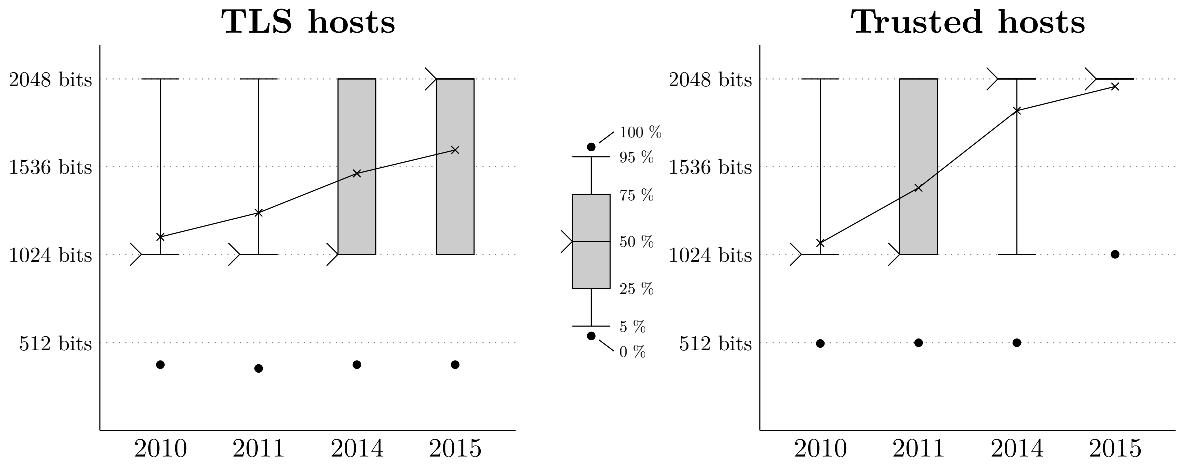

Fig. 6. Evolution of the RSA key robustness (in bits) in certificate chains between 2010 and 2015. The box plot shows different percentiles for the value repartition, and the line represents the mean value for each campaign. In some cases, the maximum value is off-chart and is not represented.

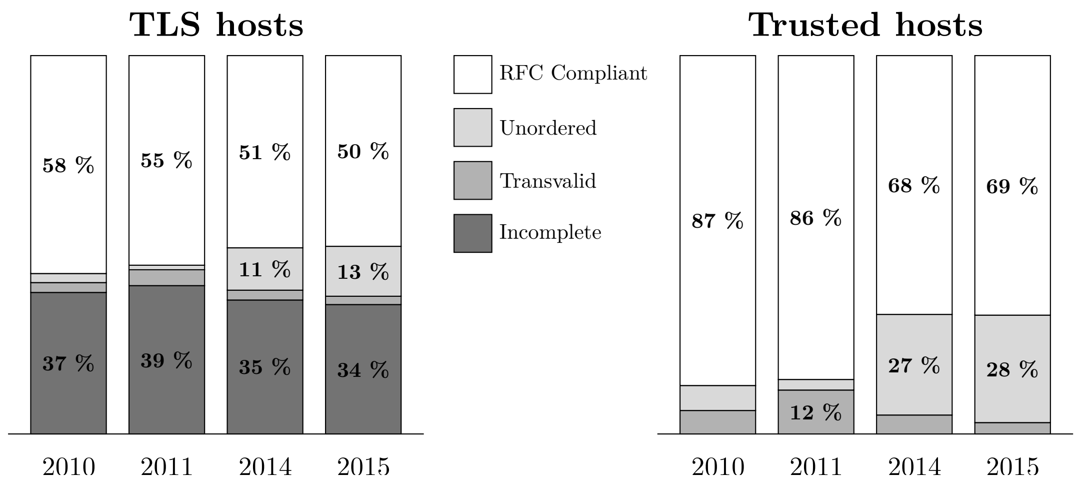

Fig. 7. Quality of chains over time (for TLS and trusted hosts).

Certificate subject	Representation
/OU=Domain Control Validated/OU=Hosted by HostGator.com, LLC./OU=PositiveSSL Wildcard/CN=*.hostgator.com	1.44 %
/OU=Domain Control Validated/OU=Hosted by BlueHost.Com, INC/OU=PositiveSSL Wildcard/CN=*.bluehost.com	0.78 %
/OU=Domain Control Validated/OU=CoreSSL DV Wildcard/CN=*.xserver.jp	0.77 %
/C=US/S=California/L=Mountain View/O=Google Inc/CN=misc-sni.blogspot.com	0.76 %
/C=US/S=New York/L=New York/O=Tumblr Inc./CN=*.tumblr.com	0.73 %
/OU=Domain Control Validated/OU=PositiveSSL Wildcard/CN=*.webhostbox.net	0.72 %
/OU=GT90704249/OU=See www.rapidssl.com/resources/cps (c)14/OU=Domain Control Validated - RapidSSL(R)/CN=*.sakura.ne.j p	0.61 %
/CN=*.wpengine.com	0.57 %
/C=US/S=California/L=Mountain View/O=Google Inc/CN=*.googleusercontent.com	0.50 %

TABLE V. UNIQUE SERVER CERTIFICATE REUSE OVER A TOP ALEXA 1M CAMPAIGN ON 2016/08/01.

Certificate subject	Representation
/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO ECC Domain Validation Secure Server CA 2	14.9 %
/C=GB/S=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO RSA Domain Validation Secure Server CA	14.0 %
/C=US/S=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./OU=http://certs.godaddy.com/repository//CN=Go Daddy Secure Certificate Authority - G2	9.6 %
/C=US/O=Let's Encrypt/CN=Let's Encrypt Authority X3	6.7 %
/C=US/O=GeoTrust Inc./CN=RapidSSL SHA256 CA - G3	6.0 %
/C=US/S=TX/L=Houston/O=cPanel, Inc./CN=cPanel, Inc. Certification Authority	4.0 %
/C=BE/O=GlobalSign nv-sa/CN=AlphaSSL CA - SHA256 - G2	2.9 %
/C=US/O=GeoTrust Inc./CN=RapidSSL SHA256 CA	2.9 %
/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure Server CA - G4	2.8 %

TABLE VI. LEAF-SIGNING CERTIFICATES MARKET SHARE OVER A TOP ALEXA 1M CAMPAIGN ON 2016/08/01.

Table VII shows intolerance to the TLS 1.2 versions in the 2014 full IPv4, whereas table VIII describes server intolerance to a PFS-only stimulus, that was added for the 2016 Top Alexa 1M campaign.

	TLS	Trusted
Compatible Handshake	99.2 %	99.4 %
Alert	0.1 %	0.1 %
Intolerant servers	**0.7 %**	**0.5 %**

TABLE VII. SERVER INTOLERANCE TO A TLS 1.2 STIMULUS IN THE 2014 FULL IPV4 CAMPAIGN.

	TLS	Trusted
Compatible Handshake	93.3 %	92.8 %
Alert	3.6 %	3.8 %
Intolerant servers	**3.1 %**	**3.4 %**

TABLE VIII. SERVER INTOLERANCE TO A PFS-ONLY STIMULUS IN THE 2016 TOP ALEXA 1M CAMPAIGN.

E. DROWN

Using multiple stimuli is also useful to spot hosts vulnerable to the DROWN attacks. First, using an SSLv2 stimulus, we can flag all the server certificates that are presented by a server still accepting the obsolete protocol version. Then, we can count the number of servers presenting a flagged certificate. These are vulnerable to DROWN, since there exists an SSLv2 server using the same certificate that an attacker can mount the attack against.

For multi-stimuli, full IPv4 campaigns, we thus get 55 % of vulnerable servers (57 % among trusted servers) in 2011 and 46 % of vulnerable servers (40 % among trusted servers) in 2014. With our most recent multi-stimuli campaigns run on Top Alexa 1 Million servers in 2016, the situation is better: 6.6 % of overall vulnerable servers (and 5.2 % for trusted hosts).

It is however important to notice that these figures, however alarming, are still an understatement for two reasons: first, we did not look at shared RSA *public keys* across different certificates (which should have a minor impact); last and not least, we did not take into account other secure protocols such as SMTPS, where RSA certificates are often reused, and where SSLv2 is still very common.

VIII. CONCLUSION

Since 2011, SSL/TLS has received a lot of attention, which has led the community to discover several flaws. In parallel, several efforts have led researchers to launch campaigns to better know the actual state of TLS deployment. Today, such datasets are ubiquitous and easily available, but it is not that easy to study them in a reproducible and uniform way.

With `concerto`, our goal was to be able to reproduce previous analyses, and to provide a sound methodology to assess the quality of TLS deployment in the wild. In this article, we presented several datasets we could study, and tried our approach on them. We could analyse these campaigns and easily reproduce several indicators accross different campaigns.

One of the major advantages of using many small tools is the flexibility regarding trust stores: using `flagTrust` once the data has been inserted and processed allows us to quickly check for the dependency of several IPs on a given set of trusted certificates. We believe `concerto` could also be used to produce efficient differential analysis regarding some indicators, as was shown with the RFC 5746 support example.

`concerto` has been publicly released as open source software[10]. Future work concerning `concerto` will be to improve backend efficiency and better handle new kinds of datasets such as Certificate Transparency logs. Another useful improvement would be to add revocation information to the database.

[10]https://github.com/ANSSI-FR/concerto

References

[1] D. E. 3rd, “Transport Layer Security (TLS) Extensions: Extension Definitions,” RFC 6066 (Proposed Standard), Internet Engineering Task Force, Jan. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6066.txt

[2] P. Biondi and the Scapy community, “Scapy,” http://www.secdev.org/projects/scapy/, 2003-2016. [Online]. Available: http://www.secdev.org/projects/scapy/

[3] C. Botnet, “Internet Census 2012: Port scanning /0 using insecure embedded devices,” http://internetcensus2012.bitbucket.org/paper.html, 2012. [Online]. Available: http://internetcensus2012.bitbucket.org/paper.html

[4] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[5] T. Duong and J. Rizzo, “Here come the XOR ninjas,” *Ekoparty Security Conference*, Sep. 2011.

[6] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, “Analysis of the HTTPS certificate ecosystem,” in *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain*, Oct. 2013, pp. 291–304.

[7] Z. Durumeric, E. Wustrow, and J. A. Halderman, “ZMap: Fast Internet-wide Scanning and Its Security Applications,” in *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA*, Aug. 2013, pp. 605–620.

[8] P. Eckerseley and J. Burns, “An Observatory for the SSLiverse,” *Defcon 18*, Aug. 2010.

[9] ——, “Is the SSLiverse a safe place?” *27. Chaos Communication Congress*, Dec. 2010.

[10] S. Farrell and H. Tschofenig, “Pervasive Monitoring Is an Attack,” RFC 7258 (Best Current Practice), Internet Engineering Task Force, May 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7258.txt

[11] P. Hoffman and J. Schlyter, “The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA,” RFC 6698 (Proposed Standard), Internet Engineering Task Force, Aug. 2012, updated by RFCs 7218, 7671. [Online]. Available: http://www.ietf.org/rfc/rfc6698.txt

[12] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, “The SSL landscape: a thorough analysis of the x.509 PKI using active and passive measurements,” in *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC ’11, Berlin, Germany*, Nov. 2011, pp. 427–444.

[13] H. Kario, “Security Pitfalls: Monthly Scan Results,” https://securitypitfalls.wordpress.com/, 2014-2015. [Online]. Available: https://securitypitfalls.wordpress.com/

[14] T. Krenc, O. Hohlfeld, and A. Feldmann, “An internet census taken by an illegal botnet: a qualitative assessment of published measurements,” *Computer Communication Review*, vol. 44, no. 3, pp. 103–111, 2014.

[15] S. Labs, “SSL Pulse: Survey of the SSL Implementation of the Most Popular Web Sites,” https://www.trustworthyinternet.org/ssl-pulse/, 2012-2015. [Online]. Available: https://www.trustworthyinternet.org/ssl-pulse/

[16] O. Levillain, “Parsifal: A Pragmatic Solution to the Binary Parsing Problems,” in *35. IEEE Security and Privacy Workshops, SPW 2014, San Jose, CA, USA*, May 2014, pp. 191–197.

[17] O. Levillain, A. Ébalard, B. Morin, and H. Debar, “One Year of SSL Internet Measurement,” in *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA*, Dec. 2012, pp. 11–20.

[18] B. Möller, T. Duong, and K. Kotowicz, “Google Security Advisory: This POODLE Bites - Exploiting The SSL 3.0 Fallback,” http://www.openssl.org/~bodo/ssl-poodle.pdf, Sep. 2014. [Online]. Available: http://www.openssl.org/~bodo/ssl-poodle.pdf

[19] I. Ristic, “Internet SSL Survey,” *Black Hat USA*, Aug. 2010.

Appendix

A. *Tool cinematics*

In the following figures, the database tables are represented by grey parallelograms, programs by rectangles and binary files by circles (grey ones correspond to binary data inside the database).

Figure 8 presents the overall process to inject data, analyse them and produce some statistics, assuming that we use the NSS trust store at validation time and that we know the `ClientHello` message sent. This was essentially the standard process we applied to all the campaigns studied.

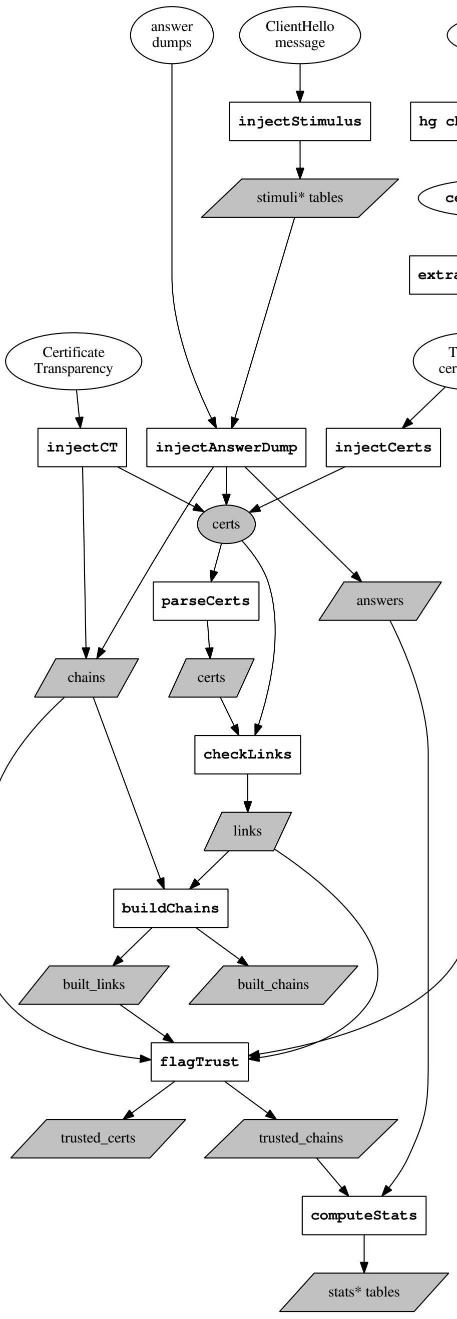

Fig. 8. Simplified version of the overall process for a typical campaign.