One Year of SSL Internet Measurement

Olivier Levillain[*†] Arnaud Ébalard[*] Benjamin Morin[*] Hervé Debar[†]

[*]ANSSI
51 boulebard Latour Maubourg
Paris, France
first.last@ssi.gouv.fr

[†]Télécom Sud Paris
9 rue Charles Fourier
Evry, France
first.last@telecom-sudparis.eu

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 servers in the wild, we enumerated HTTPS servers on the internet in July 2010 and July 2011. We sent several stimuli to the servers to gather detailed information. We then analysed some parameters of the collected data and looked at how they evolved. We also focused on two subsets of TLS hosts within our measure: the trusted hosts (possessing a valid certificate at the time of the probing) and the EV hosts (presenting a trusted, so-called *Extended Validation* certificate). Our contributions rely on this methodology: the stimuli we sent, the criteria we studied and the subsets we focused on.

Moreover, even if EV servers present a somewhat improved certificate quality over the TLS hosts, we show they do not offer overall high quality sessions, which could and should be improved.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; D.2.8 [**Software Engineering**]: Metrics

Keywords

SSL/TLS; HTTPS; Internet measure; certificates; X.509

1. 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. Since HTTPS still represents most of the daily TLS usage, we designed our experiments to get a clear view of what browsers face on a daily basis, and whether this view is satisfying or not. We performed several campaigns in 2010 and 2011, to enumerate the HTTPS servers answering on TCP port 443. We used different stimuli to gather precise information about what was effectively supported.

Many SSL/TLS handshake parameters can be considered to assess the quality of a server's answer. Some of them are related to the protocol (version, ciphersuite, extensions) and others can be found in certificate chains (signature algorithms, root certificate, X.509 extensions). We selected various criteria and looked at them through three different subsets of the measures: all the hosts, hosts presenting trusted valid certificates, hosts presenting EV certificates.

Our contribution is therefore threefold:

- using several stimuli to enrich the data collected;
- proposing a variety of criteria to assess TLS quality;
- analysing the data through different subsets.

As such, our work is in line with the suggestions of several researchers in cybersecurity, who advocate that the field would benefit from thorough experiments (e.g., last year's keynote speaker in ACSAC 2011 [1]).

2. STATE OF THE ART

2.1 SSL/TLS: a quick tour

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 [17], rapidly followed by SSLv3 [16], which fixed

Version	Comments
SSLv2	Major structural flaws [29]. Should not be used anymore.
SSLv3	PKCS#1 flaw in early implementations [4]. Interoperability issues with newer versions.
TLSv1.0	Weakness of CBC using implicit IV [23, 12]. Workarounds exist in major software.
TLSv1.1	Minimum recommended version.
TLSv1.2	New ciphersuites (GCM mode, HMAC with SHA2 hash functions).

Table 1: Summary of SSL/TLS versions.

major conceptual flaws. Even if a compatibility mode was described, 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). TLSv1.0 [9] can be seen as a minor editorial update of SSLv3. TLSv1.1 was published in 2006 [10] and TLSv1.2 in 2008 [11]. Table 1 summarizes the different versions of SSL/TLS. Today, SSLv2 and SSLv3 should not be used anymore, and TLS versions 1.1 and 1.2 should be preferred.

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 so-called *ciphersuites* which define how to:

- authenticate the server[1];
- establish a shared secret used to derive keys;
- encrypt the application data;
- ensure the integrity of the application data.

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 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[2]. 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. `Finished` messages

[1]Mutual authentication is possible with TLS, but the algorithms used to authenticate the client are negotiated independently in the `CertificateRequest` message. This aspect of TLS is out of the scope of this article.

[2]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.

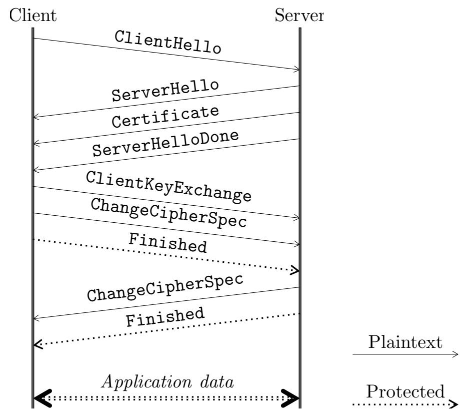

Figure 1: Example of a TLS negotiation.

are the first ones protected with the algorithms and keys negotiated.

At any moment, an `Alert` message can be sent to signal a problem, for example if no ciphersuite is acceptable, or if the client doesn't validate the certificate sent by the server.

The certificates used in TLS follow the X.509 standard [7]. The TLS Public Key Infrastructure is based on several root authorities trusted by default by clients like web browsers.

2.2 Known vulnerabilities

Early SSL/TLS versions had protocol issues. SSLv2 is flawed in numerous ways, the most problematic vulnerability being that an attacker can easily tamper with the negotiation [29]. More recently, Ray devised an attack on the renegotiation feature affecting all TLS versions [24, 25].

SSL/TLS uses many cryptographic primitives, some of which are weak, like DES or MD5 algorithms, and should not be used anymore [15, 28]. Other algorithms were wrongly implemented — Bleichenbacher described in 1998 an attack on PKCS#1 [4] — or incorrectly specified — Rogaway showed in 2002 that an adaptative chosen plaintext attack was possible on CBC, which was patched in TLSv1.1 and later proved to be exploitable in 2011 [23, 12].

It is also possible to encounter symmetric or asymmetric weak keys which lead to the loss of confidentiality in some cases (e.g. RC4 40-bit keys present in *export* ciphersuites), or allow the attacker to control the connection entirely (today, 512-bit RSA keys can easily be factored then used either for a man-in-the-middle attack or for offline decryption).

Since the authentication of the visited sites relies on certificates, processes of generation, validation and revocation thereof are critical.

- Examples of bad random number generators exist: a bug in the Debian version of OpenSSL reduced the effective entropy to only a few dozens of bits from 2006 to 2008 [8]; more recently, Lenstra et al. showed that some network devices did not produce enough entropy and reused prime numbers between two RSA generations, allowing moduli to be factored by a simple gcd algorithm [20].
- X.509 is a standard with many extensions, which have not always been correctly interpreted. For instance,

the `BasicConstraints` extension is used to distinguish server certificate from authority certificates; in 2002, Marlinspike showed that the distinction was not implemented in Webkit nor CryptoAPI [21]. He showed other vulnerabilities on the major SSL/TLS implementations [22].

- Finally, recent incidents affecting certification authorities [6, 30] have proved that the revocation system using CRLs (Certificate Revocation Lists) or OCSP (Online Certificate Status Protocol) did not really function: web browsers had to resort to black lists to limit the consequences of compromised certificates.

The TLS ecosystem is complex and it can be difficult for a client to assess a server's trustworthiness. This is particularly true for web browsers, which face a lot of servers which they have no prior knowledge of. That is why we decided to evaluate what web browsers could encounter out there. Several research teams recently led such campaigns in parallel. These studies are examined in section 8.

3. METHODOLOGY OF THE MEASURES

3.1 Enumerating HTTPS hosts

Gathering data about what a browser faces on a daily basis can be done in several ways:

- enumerating every routable address in the IPv4 space to find open HTTPS ports (TCP/443);
- contacting HTTPS hosts based on a list of DNS (Domain Name System) hostnames;
- collecting real HTTPS traffic from consenting users.

The first method is the most exhaustive, because it tests every IP in the world. However, it leads to contacting many non-HTTPS hosts. Also, it does not take into account the popularity of internet sites (i.e., discriminate sites like `www.google.com` from `randomhost.dyndns.org` or even an unnamed host).

The second option is more restrictive, but better represents user needs, and the proportion of HTTPS servers among the hosts to contact is highly optimised. Besides, this method is compliant with the TLS SNI (Server Name Indication) extension [3], 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 relevant data that would be comparable to other studies.

We chose the first method to acquire a broad vision of the HTTPS world. This method also allowed us to get consistent answers to multiple stimuli for each given host.

3.2 Description of the campaigns

In July 2010 and July 2011, we launched several campaigns to enumerate HTTPS hosts present in the IPv4 address space. We used different *stimuli* (different `ClientHello`) to grasp the behaviour of the different TLS stacks encountered.

Phase 1: finding the HTTPS hosts

The first task was to find out which hosts were accepting connections on TCP port 443. Using BGP (Border Gateway Protocol) internet routing tables, we reduced the search space from 4 billion IPv4 addresses (2^{32}) to 2 billion routable addresses. Instead of using existing tools such as `nmap` to enumerate open 443 ports, we developped homemade probes to randomize the set of routable addresses globally. For each host, the test consisted simply in a SYN-probe to determine open ports.

To prevent this first phase from being too intrusive, we bounded our upstream rate at 100 kB/s, allowing us to explore the 2 billion addresses in about two weeks.

Phase 2: TLS sessions

Once a host offering a service on port 443 was discovered, we tried to communicate with it using one (or several) TLS `ClientHello`. In this second phase, we used a full TCP handshake followed by several packets, but only with the fraction of servers listening on port 443 (about 1 percent). The second phase could thus be run in parallel with the first one.

To limit the computational impact on servers, we only recorded the first server answer (messages between `ServerHello` and `ServerHelloDone`) before ending the connection. This way, we collected the protocol and ciphersuite chosen by the server, as well as the certificate chain sent.

In the 2010 campaign, we sent only one `ClientHello` message. On the contrary, as we were interested in server behaviour, in the July 2011 campaign, we sent several `ClientHello` messages containing different protocol versions, ciphersuites and TLS extensions.

In addition to our samples, two campaigns were publicly released by the Electronic Frontier Foundation (EFF) in December 2010, allowing us to extend the data to analyse, since they employed a similar methodology to contact the servers and record the answers [13]. Table 2 describes the specificities of the `ClientHello` sent for each dataset. It contains our campaign from July 2010 (NoExt1), our seven campaigns from July 2011 (NoExt2, DHE, FF, EC, SSL2, SSL2+ and TLS12) and also includes both EFF campaigns in italics.

3.3 Issues encountered

Our July 2010 and July 2011 campaigns each took two to three weeks to complete. As explained earlier, this was necessary to avoid link saturation during the host enumeration. However, spanning our measures over several weeks has an impact on the picture of the internet we are seeing. In fact, while probing the different hosts, three factors need to be taken into account:

- the time spent acquiring the data; as the exposure time in photography, it should ideally be as short as possible, to get consistent data;
- the network load induced; sending too many packets can result in some of them getting lost at either end of the connection;
- the use of dynamic IPs in some address blocks; some ISPs change IP addresses every day or so.

Considering the network bandwidth at our disposal and the way IPs were globally randomized, we are confident we did

Id	Date	SSLv2	Max version	Ciphersuites	Extensions
NoExt1	2010/07	no	TLSv1.0	Standard Firefox suites	None
EFF-1	*2010/08*	*yes*	*TLSv1.0*	*SSLv2 + some TLSv1.0 suites*	*None*
EFF-2	*2010/12*	*yes*	*TLSv1.0*	*SSLv2 + some TLSv1.0 suites*	*None*
NoExt2	2011/07	no	TLSv1.0	Standard Firefox suites	None
DHE	2011/07	no	TLSv1.0	DHE suites only	None
FF	2011/07	no	TLSv1.0	Standard Firefox suites	EC, Reneg, Ticket
EC	2011/07	no	TLSv1.0	EC suites only	EC
SSL2	2011/07	yes	SSLv2	SSLv2 suites only	None
SSL2+	2011/07	yes	TLSv1.0	SSLv2 + some TLSv1.0 suites	Reneg
TLS12	2011/07	no	TLSv1.2	mostly TLSv1.2 suites	EC, Reneg, Ticket

Table 2: Different `ClientHello` messages sent during the campaigns. The campaigns in italics were made by the EFF. The SSLv2 column indicates that a SSLv2-compatible `ClientHello` was sent. EC means Elliptic Curves; DHE stands for Diffie-Hellman Ephemeral; finally, Reneg corresponds to the renegotiation extension [25], and Ticket to the Session Ticket one [27].

not overload links during the first phase of the campaigns. We believe that it is as close as we could get to a time-coherent snapshot. Section 6 answers questions about the impact of time on IP address stability.

Even with this in mind, randomization can be insufficient and our SYN packets may be interpreted as an attack, and our IP filtered out. A solution could have been to use several source IPs. Yet, if these addresses were not located in the same neighbourhood, we might have ended up measuring different inconsistent views of the internet, as shown in [18].

Finally, one aspect of gathering such data we did not anticipate was data storage. One easy way is to use one file per active IP, but this rapidly fills up inode/block tables, while the answers to one stimulus only take 20 GB in total. We ended up grouping answers by /8 IP ranges and developping tools to work on such files.

3.4 Global statistics on the campaigns

We first sorted the answers received for each campaign into several categories. Table 3 shows global results for the ten campaigns. It partitions the answers obtained into the following classes.

Non-TLS answers are refined in *empty* and *non-empty* answers (HTTP headers or synctactically invalid TLS messages for example). In fact, it seems common to find non-HTTPS service listening on port 443.

TLS answers can be of three types: *TLS Alerts*, *compatible TLS Handshake messages* or *incompatible TLS Handshake messages* (a `ServerHello` is considered incompatible if it contains a protocol version, a ciphersuite or extensions the client did not propose in its `ClientHello`).

If we compare the EFF-1 and SSL2+ campaigns, which correspond to similar stimuli, we obtain in both cases around 11 million valid TLS answers and 17 thousand TLS alerts, which corroborates the results, but it seems the data from the EFF-1 campaign have been post-processed to eliminate a significant part of the non-TLS results.

On the other hand, it is difficult to compare our results to the other EFF campaign, as the set of IPs probed is significantly smaller. This point is further discussed in section 6.

4. ANALYSIS METHODOLOGY

We first define subsets of the hosts contacted: the TLS hosts, the trusted hosts and the EV hosts. To assess the quality of HTTPS answers, we select several parameters regarding the TLS protocol and the certificate chain.

4.1 Subsets

TLS hosts: hosts that answered with a TLS handshake, compatible or not with the `ClientHello`.

Trusted hosts: servers which presented a server certificate for which we could build a valid chain up to a root certificate present in Firefox[3] and valid at the time of the campaign. Trusted chains not only correspond to RFC-compliant chains, but also to chains containing useless or unordered certificates and even to chains missing links[4]. Accepting the latter servers as trusted hosts conforms to most browsers' behaviour since they cache intermediate CA certificates to allow so-called *path discovery*.

`EV hosts:` EV certificates are a novelty in the internet PKI officially introduced in 2007, which aim at improving the quality of certificates. The EV guidelines [5] describe how the certificates should be issued, the audit procedures needed for the certificate authorities and the cryptographic algorithms that should be banned. An EV certificate must be issued by an EV authority (recognized by the browsers) and contain a certificate policy[5] matching the EV authority. Once a certificate is validated and recognized as EV, the browsers typically use green address bars to indicate to the user that the site is EV-trusted. The EV subset consists of hosts that sent EV chains valid at the time of the campaign. Obviously, EV hosts are a subset of the trusted hosts.

4.2 Criteria studied

Protocol version.

A `ClientHello` message includes two version fields: the external version used for the transport of this particular message (present in the so-called `Record` protocol since SSLv3) and the maximum version supported by the client, v_{max}. The standard indicates that the server should choose the

[3]This particular certificate store is easy to access and representative of many users.

[4]Of course, to build the chain up to a root certificate, the missing links had to be present in another certificate chain or in the root certificate store.

[5]Certificate policies are object identifiers (OID) contained in the `CertificatePolicies` X.509 extension.

Id	IPs with TCP/443	Non-TLS answers		TLS answers					
		Empty	Non-empty	Alerts		Incompat. HS		Compat. HS	
NoExt1	21,342,205	**40.62 %**	**13.20 %**	171,405	**0.80 %**	419	**0.00 %**	9,683,050	**45.37 %**
EFF-1	*15,579,266*	**0.00 %**	**26.79 %**	*16,509*	**0.10 %**	*88*	**0.00 %**	*11,388,447*	**73.10 %**
EFF-2	*7,777,511*	**0.00 %**	**0.91 %**	*591*	**0.00 %**	*48*	**0.00 %**	*7,705,488*	**99.07 %**
NoExt2	26,218,653	**47.28 %**	**9.35 %**	53,535	**0.20 %**	1,018	**0.00 %**	11,313,085	**43.14 %**
DHE	26,218,653	**62.79 %**	**3.19 %**	4,385,635	**16.72 %**	110,606	**0.42 %**	4,421,695	**16.86 %**
FF	26,218,653	**47.36 %**	**9.32 %**	92,239	**0.35 %**	1,007	**0.00 %**	11,262,138	**42.95 %**
EC	26,218,653	**54.26 %**	**9.36 %**	8,696,833	**33.17 %**	142,007	**0.54 %**	695,158	**2.65 %**
SSL2	26,218,653	**70.23 %**	**11.10 %**	354,760	**1.35 %**	3,826	**0.01 %**	4,533,396	**17.29 %**
SSL2+	26,218,653	**47.25 %**	**9.34 %**	17,247	**0.06 %**	127	**0.00 %**	11,361,199	**43.33 %**
TLS12	26,218,653	**54.05 %**	**9.40 %**	918,202	**3.50 %**	108,557	**0.41 %**	8,554,180	**32.62 %**

Table 3: Distribution of the answers collected for each campaign. The percentages are computed over the total number of IPs where we found a 443 open port (second column).

maximum version it supports, up to v_{max}. If no such version exists, it should terminate the handshake with an alert. We also consider that a good server should ban SSLv2.

Ciphersuite.

The range of existing ciphersuites is defined by the IANA, and currently consists of 297 suites[6]. Among those suites, the client generally offers between 10 and 30 suites. In its `ServerHello`, the server will select one.

After eliminating a lot of suites that contain seldom used algorithms (fixed Diffie-Hellman, PSK[7], SRP[8], etc.), we classify the ciphersuites in three groups: weak suites (SSLv2, DES, export suites), acceptable suites (RC4, 3DES, DSS, MD5) and strong ones (the remaining ones). We thus obtain 23 weak suites, 23 acceptable suites and 44 strong suites. Naturally, we would like the servers to select strong suites.

TLS Extensions.

As studied in a recent RFC draft [19], some servers do not support TLS extensions and among them, some implementations simply reject `ClientHello` containing extensions. This is unfortunate since this mechanism is necessary for the extensibility of the protocol:

- security fixes like secure renegotiation [25];
- support for new ciphersuites using elliptic curves [2];
- new features like session tickets [27].

From a security standpoint, when a client proposes the secure renegotiation extension, the server should support it.

Our first analysis focuses on these TLS parameters: the protocol version, the strength of the chosen ciphersuite and support for the renegotiation extension. The results are presented in section 5.

Server behaviour.

In July 2011, as we sent servers multiple `ClientHello`s, we gained unique insight into the behaviour of servers: the versions they support, their reaction to restricted ciphersuite choices, their intolerance to versions and extensions.

Section 6 gives precise statistics about these seven campaigns, by comparing the answers obtained for the different stimuli for each IP in the considered subset (TLS hosts, trusted hosts and EV hosts).

The quality of the certificate chain.

The `Certificate` message contains a list of certificates used to establish the identity of the server. This list should be strictly ordered: the first certificate is the server certificate, and each certificate is signed by the following one. The final root certificate may be omitted, since the client should already know it to trust the certificate chain.

However, some servers do not follow these rules: they forget intermediate certification authorities, they send unordered chains or useless certificates; others even send two certification paths to have a server certificate validated by two root authorities. Such behaviour is generally accepted by common TLS stacks which are laxist and provide features like *path discovery* but may break some implementations[9].

To create the trusted hosts subset, we first need to classify the certificate chains into three groups: empty or incomplete chains, trusted chains, complete but untrusted chains. The two latter classes are further refined:

- *RFC-compliant* chains do contain only the useful certificates in the correct order;
- *self-contained* chains contain useless or unordered certificates, but include all the information needed to build a complete chain;
- *transvalid*[10] chains lack *intermediate* authority to build a complete chain. To reach the root certificate, we thus need to use certificates from other chains.

In addition to this classification, section 7 contains statistics for other parameters: the cryptographic algorithms, the key sizes and the validity period of the chain built.

5. ANALYSIS OF TLS PARAMETERS

5.1 Protocol version

Table 4 shows the distribution of the versions chosen by servers for each campaign and each subset.

For standard `ClientHello` messages (i.e. NoExt1, EFF-1, EFF-2, NoExt2, FF, SSL2+), we observe the same results

[6]To be complete, we must also count the 7 SSLv2 suites and the 4 SSLv3 FIPS suites which are now obsolete or redundant.

[7]Pre-Shared Key.

[8]Secure Remote Password.

[9]One such implementation is the TLS stack present in Java

	TLS	Trusted	EV
NoExt1	TLS1 96 %	TLS1 99.3 %	TLS1 99.2 %
	SSL3 4 %	SSL3 0.7 %	SSL3 0.8 %
EFF-1	TLS1 95.4 %	TLS1 99.2 %	TLS1 98.4 %
	SSL3 4.5 %	SSL3 0.8 %	SSL3 1.6 %
	SSL2 0.1 %		
EFF-2	TLS1 96 %	TLS1 99.1 %	TLS1 98.5 %
	SSL3 4 %	SSL3 0.9 %	SSL3 1.5 %
NoExt2	TLS1 96 %	TLS1 99.4 %	TLS1 99.4 %
	SSL3 4 %	SSL3 0.6 %	SSL3 0.4 %
DHE*	TLS1 97 %	TLS1 99.8 %	TLS1 99.6 %
	SSL3 3 %	SSL3 0.2 %	SSL3 0.4 %
FF	TLS1 96 %	TLS1 99.4 %	TLS1 99.4 %
	SSL3 4 %	SSL3 0.6 %	SSL3 0.4 %
EC*	TLS1 84 %	TLS1 100 %	TLS1 100 %
	SSL3 16 %		
SSL2*	SSL2 99.9 %	SSL2 100 %	SSL2 100 %
	SSL3 <0.1 %		
	TLS1 <0.1 %		
SSL2+	TLS1 96 %	TLS1 99.2 %	TLS1 99.4 %
	SSL3 4 %	SSL3 0.8 %	SSL3 0.6 %
	SSL2 <0.1 %		
TLS12*	TLS1 98.5 %	TLS1 99.6 %	TLS1 99.5 %
	SSL3 1.4 %	SSL3 0.2 %	SSL3 0.2 %
	TLS1.1 0.1 %	TLS1.1 0.2 %	TLS1.1 0.2 %
	TLS1.2 <0.1%	TLS1.2 <0.1%	TLS1.2 <0.1%

Table 4: Distribution of the TLS versions chosen by the servers for each campaign and each subset. Campaigns with a star correspond to specific July 2011 stimuli that produced significantly less answers than NoExt2 or FF.

over time. **TLSv1.0 is the preferred version of the protocol: 95 % of the TLS hosts answer with TLSv1.0 and 5 % use SSLv3.** If we consider the same stimuli but focus on trusted or EV hosts, the proportion becomes more or less 99 % for TLSv1.0 and 1 % for SSLv3. It is worth noting that the results do not change significantly when extensions are sent, or when a SSLv2 compatibility `ClientHello` is used.

Yet, there are still many servers that do not use TLSv1.0 and choose the obsolete SSLv3. Even if the situation is better with trusted and EV hosts, such configurations should be avoided. For the SSL2+ stimulus, this represents 32,000 servers amongst the 4 million trusted servers, and about 1,000 servers amongst the 140,000 EV hosts.

The other stimuli are more difficult to interpret. Indeed, as table 3 shows, the results obtained for DHE, EC, SSL2 and TLS12 stimuli correspond to fewer TLS answers. It shows that some servers refuse to negotiate some protocol versions or some ciphersuites. Among these servers, some will emit an alert, in compliance with the standards. Others will not answer or return inconsistent TLS messages: we call this intolerance to particular versions/suites. Section 6 discusses this matter further.

5.2 Ciphersuites

Table 5 shows the distribution of ciphersuites chosen by servers for each campaign and each subset. The ciphersuites are grouped into categories: S (strong), A (acceptable), W (weak)[11] and N represents suites that were *not* proposed by the client.

	TLS	Trusted	EV
NoExt1	S 65 %	S 59 %	S 59 %
	A 35 %	A 41 %	A 41 %
EFF-1	S 64 %	S 62 %	S 54 %
	A 36 %	A 38 %	A 46 %
EFF-2	S 64 %	S 61 %	S 56 %
	A 36 %	A 39 %	A 44 %
NoExt2	S 73 %	S 68 %	S 80 %
	A 27 %	A 32 %	A 20 %
DHE*	S 95 %	S 98.6 %	S 98.9 %
	A 2.5 %	A 1.4 %	A 1.1 %
	N 2.5 %		
FF	S 73 %	S 68 %	S 80 %
	A 27 %	A 32 %	A 20 %
EC*	S 83 %	S 99.2 %	S 98.9 %
	N 17 %	N 0.8 %	N 1.1 %
SSL2*	W 99.9 %	W 100 %	W 100 %
	N 0.1 %		
SSL2+	S 71 %	S 67 %	S 80 %
	A 29 %	A 33 %	A 20 %
	W 0.1 %		
TLS12*	S 98.8 %	S 100 %	S 100 %
	N 1.2 %		

Table 5: Distribution of the ciphersuites chosen by the servers for each campaign and each subset. Percentages below 0.1 % are ignored.

For the first three campaigns (NoExt1, EFF-1 and EFF-2), around 65 % of the answers contain strong suites. For trusted or EV subsets, this proportion is smaller. This discrepancy is explained by the fact that the A category contains the popular `TLS_RSA_WITH_RC4_128_MD5` suite. While we consider this suite only acceptable[12], it is commonly considered the most efficient one. For the NoExt2, FF and SSL2+ campaigns from July 2011, proportions are roughly the same: 72 % for TLS hosts, 68 % for trusted hosts and 80 % for EV hosts, which indicates a minor improvement in the suite choices, especially for EV hosts.

We did not expect servers to answer with ciphersuites *not* proposed in `ClientHello`, as it is not compliant with the specifications. This phenomenon is significant in the DHE and EC campaigns, when the servers faced a limited choice. We also witness this behaviour with TLS12 stimulus, essentially because we chose not to propose the popular `RC4_MD5` ciphersuite[13]. This is a manifestation of server intolerance to DHE/EC/TLSv1.2.

Another way to look at the ciphersuites chosen by servers is to compute the proportion of them that provide Perfect Forward Secrecy (PFS). Such suites do not allow the decryption of past sessions if the server private key is compromised[14]. Results are presented in table 6. We did not

ClassPath.

[10]The term *transvalid* was first used by the EFF to describe such chains.

[11]Such suites were only proposed in the SSL2 and SSL2+ `ClientHello`.

[12]On one hand, statistical vulnerabilities on RC4 limit the quantity of data that should be protected without refreshing the keys. On the other hand, it is strongly discouraged to use MD5 nowadays.

[13]However, other classical non-DHE non-EC suites were proposed in TLS12 `ClientHello`.

[14]With TLS ciphersuites, this property is obtained with

		TLS	Trusted	EV
NoExt1	2010-07	41 %	44 %	20 %
EFF-1	2010-08	43 %	46 %	19 %
EFF-2	2010-12	41 %	42 %	19 %
NoExt2/FF SSL2+	2011-07	37 %	39 %	11 %
TLS12	2011-07	0 %	0 %	0 %

Table 6: Proportion of the ciphersuites chosen by the servers that provide Perfect Forward Secrecy. The percentage is computed over the total number of compatible answers.

include the DHE and EC campaigns (since all the proposed suites offers PFS) nor the SL2 stimulus (since none of the suites proposed offers PFS).

If we look at the first four lines, corresponding to four different dates, around 40 % of TLS hosts chose PFS suites. The corresponding trusted hosts figures are a little higher, whereas the proportions drops with EV hosts, especially in July 2011 where only 11 % of the servers chose PFS.

The result for the TLS12 stimulus can be explained by the fact that the PFS suites proposed in this case were all compatible with TLSv1.2 only, leaving no choice to non-TLSv1.2 implementations but not to offer PFS.

For a standard stimulus, about two thirds of TLS hosts choose strong ciphersuites. Recently, the proportion seems to have grown to 80 % for EV hosts. However, only 40 % of TLS hosts, and less than 20 % of EV hosts choose a suite offering the Perfect Forward Secrecy.

5.3 Secure renegotiation

As discussed in section 4.2, we expect servers to implement RFC 5746 for secure TLS renegotiations. However, only three out of the ten stimuli studied proposed this extension (FF, SSL2+[15] and TLS12), all sent in July 2011. As a result, we can not assess the evolution in time of this parameter.

If we now consider only those three stimuli proposing the extension, we have the same results: 53 % of the TLS hosts accept RFC 5746 extension, which becomes 65 % when we focus on trusted hosts. For EV hosts, the proportion is even better: 80 %. However, 20 % of EV hosts (around 28,000 servers) still do not support secure renegotiation as of July 2011.

It is important to notice that servers are only vulnerable if they do not support the extension *and* if they accept to renegotiate. As we did not pursue the connection, nor tried to renegotiate, we are not able to reliably tell how many servers are indeed vulnerable. Yet, from the client's perspective, the only way to be sure that the server is not vulnerable is if it supports the secure renegotiation extension.

As of July 2011, one third of the trusted hosts and 20 % of EV hosts do not support secure renegotiation.

ephemeral Diffie-Hellman key exchanges, as opposed to the RSA encrypted key exchange.

[15]The SSL2+ `ClientHello` did not propose the extension *per se*, but rather used a dedicated ciphersuite to signal the support of secure renegotiation.

6. ANALYSIS OF SERVER BEHAVIOUR

Using the collected data, we tried to compare the answer types for a given IP in different campaigns. However, we found out that such a comparison was irrelevant when we compared data collected at different times.

6.1 When comparison really makes sense

We compared the list of IPs from the TLS subset between our measure of July 2010 and the EFF measure of August 2010: 7.5 million IPs are present in both sets but 2 million were only seen by our trace, and nearly 4 million were only present in the EFF trace. So a vast proportion of TLS hosts do not have a stable IP over a month or a year.

This fact is even more visible when comparing our measures in 2010 and 2011 (using the NoExt2 measure): 5.5 million IPs correspond to TLS hosts in both cases, but about 4 million (resp. 6 million) IPs are present only in the 2010 (resp. 2011) trace. It is thus clear we can only do per-IP comparisons between measures that were made at the same time.

This basic comparison sheds light on the surprising figures obtained in EFF-2 campaign: between the EFF experiments conducted in August and December, 7.5 million IPs represent TLS hosts both times, about 4 million have disappeared in EFF-2 whereas only 60,000 IPs are new in EFF-2. This can be explained by the fact that the EFF did not launch the first phase (enumerating IP with TCP/443 open) again in December and reused the list of IPs from August 2010.

6.2 Error margin for July 2011 campaigns

We now focus on the seven measures of July 2011 that were conducted simultaneously[16] on the same address pool to understand server answers against different stimuli. We need to check that the servers we contacted were the same during all the communications. Let's first compare the IPs corresponding to a TLS answer between two similar campaigns: NoExt2 and FF. Over 99.6 % of IPs corresponding to a TLS host in one measure also do in the other. The correlation is even better if we focus on trusted or EV hosts.

To confirm that, we also compare the server certificates returned by the servers. Considering the set of IPs that answered at least once with a server certificate, we count for each IP the number of different server certificates received over the seven communications. More than 99.6 % of them presented the same server certificate each time they answered with a valid `ServerHello`. If we compute the same statistics using the list of IPs presenting at least once a trusted certificate (resp. an EV certificate), the error margin is even better, since 99.9 % served only one certificate. The hosts that do not consistently send the same server certificate (0.4 %) define our error margin.

6.3 Understanding hosts with multiple stimuli

We can now refine the statistics about DHE/EC/TLSv1.2 intolerance. Let us focus on the servers that correctly answered with a compatible `ServerHello` to the NoExt2, FF and SSL2+ stimuli. We thus obtain a list of apparently valid IPs for each subset: 11.2 million TLS hosts, 4.0 million trusted hosts and 141,972 EV hosts. Tables 7, 8, 9 present the answers of this sample groups to the DHE, EC

[16]For a given IP, all `ClientHello` messages were sent within a 10 minute-timeframe.

and TLS12 stimuli. The answers are partitioned into the categories defined in section 3.4.

	TLS	Trusted	EV
Compatible Handshake	39 %	42 %	13 %
Alert	38 %	28 %	71 %
Intolerant servers	**23 %**	**30 %**	**16 %**
Non-TLS answer	22 %	30 %	16 %
Incompatible Handshake	1 %	0 %	0 %

Table 7: Answers to the DHE stimulus.

	TLS	Trusted	EV
Compatible Handshake	6 %	10 %	5 %
Alert	76 %	68 %	82 %
Intolerant servers	**18 %**	**22 %**	**13 %**
Non-TLS answer	17 %	22 %	13 %
Incompatible Handshake	1 %	0 %	0 %

Table 8: Answers to the EC stimulus.

	TLS	Trusted	EV
Compatible Handshake	76 %	74 %	86 %
Alert	7 %	5 %	2 %
Intolerant servers	**17 %**	**21 %**	**12 %**
Non-TLS answer	16 %	21 %	12 %
Incompatible Handshake	1 %	0 %	0 %

Table 9: Answers to the TLS12 stimulus.

The Alert line shows the proportion of servers that refuse to negotiate and assert this choice. This should happen if none of the proposed ciphersuites is acceptable. Theoretically, this should not happen with a protocol version, since the servers of the sample groups accepted TLSv1.0 and could have answered with this version. Yet, our TLS12 stimulus did not contain all the suites of the NoExt2/FF/SSL2+ `ClientHello`s so it is legitimate for a server to accept the latter stimuli but send an Alert to the TLS12 message.

The last two lines represent servers that do not respond correctly to the stimulus. As they correctly answered three other stimuli, we would have expected an Alert message to signal the negotiation failure. We call such servers DHE-, EC- or TLSv1.2-intolerant, and their behaviour does not conform to the standards.

For each case (DHE, EC, TLSv1.2), the proportion of intolerant servers is very important: about 20 % globally, more than 12 % for EV servers.

Another disappointing fact is the very low proportions of servers supporting DHE and EC suites, especially for EV hosts (13% for DHE, 5 % for EC).

Finally, let's consider the proportion of the sample groups answering correctly to a SSLv2 `ServerHello` when the stimulus is a pure SSLv2 `ClientHello` (SSL2 stimulus). Table 10 shows the answers received. We did not expect TLS servers to behave correctly, since SSLv2 uses different messages and is now deprecated. In fact, we would have expected fewer servers to accept negotiating a SSLv2 session.

Many TLS servers are still fully compatible with SSLv2, whereas they should not negotiate the obsolete version of the protocol.

	TLS	Trusted	EV
Compatible Handshake	**40 %**	**27 %**	**8 %**
Alert	2 %	2 %	1 %
Non-TLS answer	58 %	71 %	91 %
Incompatible Handshake	0 %	0 %	0 %

Table 10: Answers to the SSL2 stimulus.

7. ANALYSIS OF CHAIN QUALITY

In this section, we only consider four campaigns (NoExt1, EFF-1, EFF-2 and NoExt2) which correspond to different dates (July 2010, August 2010, December 2010 and July 2011). The results of this section were similar for the three standard July 2011 stimuli (NoExt2, FF, SSL2+). For the four campaigns, trusted hosts represent around 35 % of the TLS hosts (about 4 million) and EV servers represent 1 % of the TLS hosts (100 to 140,000). 10.2 million unique certificates were analysed, that were gathered from 10.9 million unique certificate chains.

The certificate chains we study are the chains built by our verification program, i.e. the best certificate chain we could build from the certificates sent and all the certificates gathered. We prefer trusted chains over non-trusted chains, and chose RFC-compliant chains (R) over complete but unordered chains (C) over transvalid chains (T). The partition of certificate chains built along these latter categories are given in table 11. It is interesting to notice that EV hosts often present unordered or even transvalid chains[17], which leads to incompatibilities with some TLS stacks[18]. Generally, servers send 2 or 3 certificates and the certificate chains we build also contain the same number of certificates. However, some servers send more certificates. The maximum we saw was 150 in EFF-2 and corresponded to a trusted server.

More than 40 % TLS hosts do not send RFC-compliant chains. The values for EV hosts are even worse (around 85 %). These observations are stable from 2010 to 2011.

	2010-07	2010-08	2010-12	2011-07
TLS	R : 60 % C : 9 % T : 4 % I : 27 %	R : 61 % C : 8 % T : 3 % I : 28 %	R : 59 % C : 10 % T : 6 % I : 25 %	R : 54 % C : 10 % T : 6 % I : 30 %
Trusted	R : 69 % C : 21 % T : 10 %	R : 71 % C : 19 % T : 10 %	R : 67 % C : 21 % T : 12 %	R : 62 % C : 24 % T : 14 %
EV	R : 11 % C : 78 % T : 11 %	R : 13 % C : 76 % T : 11 %	R : 16 % C : 74 % T : 10 %	R : 12 % C : 83 % T : 5 %

Table 11: Partition of the certificate chains built in (R)FC-compliant, (C)omplete although not RFC-compliant and (T)ransvalid chains. (I)ncomplete chains are chains we could not build completely.

RSA is the main algorithm used in the certificates sent: the proportion of certificate chains containing only RSA keys

[17]As mentioned earlier, transvalid chains are chain missing *intermediate* certificate authorities, the root certificate being optional in the RFC.

[18]For example, the Java TLS implementation can not validate such chains.

is higher than 99 % for TLS hosts, and reaches 100 % for trusted and EV hosts. We thus would like to assess the cryptographic robustness of such RSA certificate chains. The criterium to measure RSA key robustness is the minimum RSA key length found in the certificate chain. The statistics for this parameter are given in table 12. It appears that mean RSA key lengths are increasing with time. The 84 % 1024-bit and 13 % 2048-bit chains measured in 2010 turn into 52 % and 48 % respectively. The shift is even better for Trusted hosts, since we have 86 % of 1024 bit chains and 13 % of 2048 bit chains in July 2010 that have become 52 % and 48 % in July 2011. In 2011, all the EV servers present 2048 bit robust chains, which can be explained by the EV guidelines [5] that specify this as a minimum key size for EV certificates from December 2010.

	2010-07	2010-08	2010-12	2011-07
TLS	1147	1135	1197	1303
Trusted	1149	1133	1220	1514
EV	1950	1897	2042	2048

Table 12: Mean RSA robustness of the chains.

This table does not include extreme values: few servers present huge RSA keys (up to 16384 bit long) or very short keys (512 or 768 bits). Such weak RSA keys represent 3 % of TLS hosts in the first two campaigns and less than 2 % for the last two. What is more serious is that 512 and 768 bit certificates are present in 2 % of trusted hosts in July 2010. Fortunately, this number has dropped to less than 0.1 % (less than 2,500 servers) in July 2011.

Finally, the last parameter we study is the validity period of the chain (i.e. the intersection of the validity periods of the certificates in the chain). The mean values are represented in table 13. As expected, trusted and EV validity periods are reasonable (mostly one or two years). However TLS hosts do contain anomalies (chains that are never valid, or valid until the year 9999), that are hopefully skimmed by the trusted filter. Another trend we observe is that the typical validity of EV certs has moved from 365 days to 730 days between July 2010 and December 2011.

	2010-07	2010-08	2010-12	2011-07
TLS	2561	5020	2328	2659
Trusted	701	728	728	744
EV	551	555	612	652

Table 13: Mean validity period (in days).

The chain validity period and the key robustness are well understood parameters that have improved over time. They have reached acceptable values in the EV subset: at least 2048 bit RSA keys, 1- or 2-year validity. This is fortunate, as EV was designed specifically to take these parameters into account.

8. RELATED WORK

Several projects aiming at understanding the SSL landscape have performed similar measures on the internet. We discuss three of them: two which were presented at security conferences (BlackHat, DefCon and CCC) and one covered in an academic paper.

As explained in section 3, the EFF performed two similar campaigns. They presented their results about the certificates gathered in 2010 [13, 14]. Even if the global methodology resembles ours (enumerating TCP/443 open ports on the IPv4 space), they used standard tools to find TLS hosts (`nmap`) whereas we developped specific tools to fully randomize the IPv4 space. Besides, the set of IPs initally probed was not exactly the same: they used a restricted list of /8 prefixes following the IANA information. Additionally, in their second measure (EFF-2), they did not enumerate the TCP/443 hosts again and used the August 2010 list instead. Another main difference in our approaches is the stimulus: the EFF sent a SSLv2 `ClientHello` whereas we tried to obtain more information by sending different stimuli. Contrary to this study, our work does not focus exclusively on the certificate chains sent by trusted hosts; but we broaden the criteria to assess the quality of servers' answers and to show trends by focusing on different subsets of hosts.

In 2010, Ivan Ristic from Qualys SSL Labs presented another SSL survey focusing this time on a DNS enumeration of HTTPS hosts [26]. He established several connections with each of the servers tested. His goal was to assess the quality of TLS answers from servers reachable via a DNS hostname. His results concerning protocol support match ours: SSLv2 is still widely supported, at least with a compatible `ClientHello`, while TLSv1.1 and TLSv1.2 are virtually inexistant. Our results also concur with his findings about the quality of the `Certificate` message sent by servers that are not strictly RFC-compliant. Like SSL Labs, we work on more criteria than just the certificates, but the difference in host enumeration makes our work complement theirs. In addition, focusing on the EV hosts allows us to present additionnal results. Since April 2012, SSL Labs have launched SSL Pulse, a dashboard providing daily measures for 200,000 HTTPS sites, which gives a partial but interesting insight on trends in SSL deployements.

Finally, in 2011, Ralph Holtz et al. from the University of München gathered and studied different data sets: active probing of popular sites, passive monitoring on a 10Gb link and the EFF campaigns [18]. They provide a thorough analysis of the certificates received for each of the campaigns studied. One of their results is to compare the certificates received by clients from different source addresses and to spot suspicious certificates from these sets. Using real world traffic is a very interesting source of information and once again, it is complementary to full IPv4 host enumerations.

9. CONCLUSION

From July 2010 to July 2011, we gathered data from full IPv4 HTTPS host enumerations, evaluated the quality of TLS answers and described trends in time.

We found that some well studied parameters, like RSA key sizes, are improving, but most of the criteria we analysed are not well taken into account, even if some parameters have improved in a year. For example, a lot of servers are still intolerant to some ciphersuites or to recent TLS versions. The quality of the certificate chains sent by servers is also not acceptable, since many HTTPS hosts send `Certificate` messages that do not comply to the standard, which makes some TLS stacks fail.

There is a pressing need for a quality label representing the overall quality of TLS sessions (server configuration, implementation and cryptographic parameters). The only widespread existing label is Extended Validation, which is visually recognisable in web browsers. However, EV only deals with the format of the certificate issued to servers, and does not take into account the other parameters. In fact, global TLS statistics were even better than EV statistics for some parameters. One way of improving the SSL landscape would thus be to create a new label or to extend EV constraints to cover all the criteria relevant to security: support for recent TLS versions and for most secure ciphersuites, preference for PFS suites, strict RFC-compliance, support of known security extensions.

Recent initiatives like the EFF Decentralized SSL Observatory (passive monitoring through a browser plugin) or SSL Labs' SSL Pulse should help monitoring parts of the SSL landscape, as the EFF calls it. It may also be useful to browse the full IPv4 space again to compare global views of the SSLiverse over time. Indeed, using only DNS scans or passive monitoring does not allow for really comparable statistics.

Further work could also include new stimuli to refine the data obtained. We could study other parameters like the certificate revocation methods. To improve our notion of trust, it would be useful to take other certificate trust stores into account (e.g. Internet Explorer, Opera). Finally, what we noticed was that many servers did not behave like common known TLS stacks, so it would be interesting to investigate their answers to try and fingerprint the stacks encountered.

Acknowledgment

The work in this paper has been partially sponsored by the EC 7th Framework Programme as part of the ICT VisSense project (grant no. 257497). The authors would like to thank the Applied and Fundamental Research Division of the French Network and Information Security Agency (ANSSI) for their comments and suggestions.

10. REFERENCES

[1] T. Benzel. The science of cyber security experimentation: the DETER project. In Robert H'obbes' Zakon, John P. McDermott, and Michael E. Locasto, editors, *ACSAC*, pages 137–148. ACM, 2011.

[2] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). RFC 4492 (Informational), May 2006.

[3] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 3546 (Proposed Standard), June 2003.

[4] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In *CRYPTO*, 1998.

[5] CA/Browser Forum. EV SSL Certificate Guidelines version 1.3, 2010.

[6] Comodo. Report of Incident - Comodo detected and thwarted an intrusion on 26-MAR-2011, 2011.

[7] 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), May 2008.

[8] Debian. DSA-1571-1 openssl – predictable random number generator, 2008.

[9] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999.

[10] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006.

[11] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008.

[12] T. Duong and J. Rizzo. BEAST: Surprising crypto attack against HTTPS, 2011.

[13] P. Eckersley and J. Burns. An Observatory for the SSLiverse, Talk at Defcon 18, 2010.

[14] P. Eckersley and J. Burns. Is the SSLiverse a safe place?, Talk at 27C3, 2010.

[15] Electronic Frontier Foundation. *Cracking DES. Secrets of Encryption Research, Wiretap Politics & Chip Design.* O'Reilly, 1998.

[16] A. Freier, P. Karlton, and P. Kocher. The SSL Protocol Version 3.0, 1996.

[17] K. Hickman. The SSL Protocol, 1994-1995.

[18] 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 *IMC'11*, 2011.

[19] A. Langley. Unfortunate current practices for HTTP over TLS. Internet Draft, 2011.

[20] A. Lenstra, J. Hughes, M. Augier, J. Bos, T. Kleinjung, and C. Wachter. Ron was wrong, Whit is right. Cryptology ePrint Archive, Report 2012/064, 2012.

[21] M. Marlinspike. Internet Explorer SSL Vulnerability, 2002.

[22] M. Marlinspike. More Tricks For Defeating SSL In Practice, 2009.

[23] B. Moeller. Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures, 2002-2004.

[24] M. Ray. Authentication gap in TLS renegotiation, 2009.

[25] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov. Transport Layer Security (TLS) Renegotiation Indication Extension. RFC 5746 (Proposed Standard), February 2010.

[26] I. Ristic. Internet SSL Survey, Talk at BlackHat 2010, 2010.

[27] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 4507 (Proposed Standard), May 2006.

[28] M. Stevens, A. Sotirov, A. Lenstra J. Appelbaum, D. Molnar, D.A. Osvik, and B.D. Weger. Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In *Crypto 2009, LNCS 5677*, pages 55–69, 2009.

[29] S. Turner and T. Polk. Prohibiting Secure Sockets Layer (SSL) Version 2.0. RFC 6176 (Proposed Standard), March 2011.

[30] Vasco. DigiNotar reports security incident, 2011.