Coming of Age: A Longitudinal Study of TLS Deployment

Platon Kotzias
IMDEA Software Institute
Universidad Politécnica de Madrid

Abbas Razaghpanah
Stony Brook University

Johanna Amann
ICSI/Corelight/LBNL

Kenneth G. Paterson
Royal Holloway, University of London

Narseo Vallina-Rodriguez
IMDEA Networks Institute
ICSI

Juan Caballero
IMDEA Software Institue

ABSTRACT

The Transport Layer Security (TLS) protocol is the de-facto standard for encrypted communication on the Internet. However, it has been plagued by a number of different attacks and security issues over the last years. Addressing these attacks requires changes to the protocol, to server- or client-software, or to all of them. In this paper we conduct the first large-scale longitudinal study examining the evolution of the TLS ecosystem over the last six years. We place a special focus on the ecosystem's evolution in response to high-profile attacks.

For our analysis, we use a passive measurement dataset with more than 319.3B connections since February 2012, and an active dataset that contains TLS and SSL scans of the entire IPv4 address space since August 2015. To identify the evolution of specific clients we also create the—to our knowledge—largest TLS client fingerprint database to date, consisting of 1,684 fingerprints.

We observe that the ecosystem has shifted significantly since 2012, with major changes in which cipher suites and TLS extensions are offered by clients and accepted by servers having taken place. Where possible, we correlate these with the timing of specific attacks on TLS. At the same time, our results show that while clients, especially browsers, are quick to adopt new algorithms, they are also slow to drop support for older ones. We also encounter significant amounts of client software that probably unwittingly offer unsafe ciphers. We discuss these findings in the context of long tail effects in the TLS ecosystem.

ACM Reference Format:
Platon Kotzias, Abbas Razaghpanah, Johanna Amann, Kenneth G. Paterson, Narseo Vallina-Rodriguez, and Juan Caballero. 2018. Coming of Age: A Longitudinal Study of TLS Deployment. In *2018 Internet Measurement Conference (IMC '18), October 31-November 2, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3278532.3278568

1 INTRODUCTION

The Transport Layer Security (TLS) protocol is the most widely used encrypted communication protocol on the Internet. However, in order to stay secure, it has had to constantly evolve in the face of each new attack and vulnerability that is discovered. Over the last few years various TLS vulnerabilities such as BEAST, Lucky 13, POODLE, Heartbleed, FREAK, Logjam, and multiple attacks against RC4 have been discovered. The Snowden revelations have also highlighted weaknesses in TLS, specifically the reliance on RSA key transport for establishing keying material, a method that can be passively broken by an entity in possession of the server's RSA private key. Addressing these attacks requires changes to the protocol, to server-, or to client-software, or to all of them simultaneously.

Prior work highlights different parts of the TLS ecosystem like specific attacks [6, 9, 10, 17, 41, 44, 44, 63, 74, 82], problems of the PKI [7, 46, 54, 60] or problems of TLS usage in specific areas like on mobile devices [47, 71, 83]. However, to the best of our knowledge, no prior work has examined the specific impact of security issues on protocol deployment.

In this paper, we conduct a large-scale longitudinal study examining the evolution of the TLS ecosystem since 2012 both on the client and on the server side. We analyze trends and evolution of the ecosystem, putting a special focus on changes occurring in response to specific high-profile attacks. For this, we use a combination of passive and active measurement data. Our passive measurements have been running continuously since February 2012 and currently contain protocol information about more than 319.3B TLS connections. The active measurement data provided to us by Censys [42] contains SSL and TLS scans of the entire IPv4 address space starting from August 2015.

To identify the patching behavior and evolution of specific clients we also create the—to our knowledge—largest TLS client fingerprint database to date, consisting of 1,684 fingerprints. These fingerprints allow us to attribute 69.26% of TLS connections to specific TLS-using programs or libraries. Using these fingerprints we discover that while clients, especially browsers, are quick to adopt new algorithms, they are also slow to drop support for older ones. We also encounter significant amounts of client software that probably unwittingly offer unsafe ciphers. We discuss these findings in the context of long tail effects arising from the desire to maintain backwards compatibility, software abandonment, and the difficulties faced by users of TLS (in a broad sense) in keeping up-to-date with proper TLS usage.

Our analysis, shows radical changes in the TLS ecosystem over the last 6 years. In 2012, 90% of TLS connections used TLS 1.0, while today 90% use TLS 1.2, with TLS 1.3 traffic increasing rapidly. In 2012, the use of RC4 and CBC-mode for encryption was prevalent; today RC4 has almost completely disappeared in response to attacks, while CBC-mode accounts for about 10% of traffic. RC4 and

IMC '18, October 31-November 2, 2018, Boston, MA, USA

ACM ISBN 978-1-4503-5619-0/18/10…$15.00
https://doi.org/10.1145/3278532.3278568

CBC-mode have been largely replaced by modern AEAD schemes, especially AES-GCM. We observe significant declines in the advertisement and use of export cipher suites, anonymous cipher suites, and vintage algorithms like 3DES. We also see a rise in the use of forward-secret cipher suites, now accounting for more than 90% of connections. Ephemeral elliptic-curve-based Diffie-Hellman key exchange (ECDHE) now dominates RSA key transport in TLS handshakes, with Curve25519 gaining in popularity and accounting for more than 20% of connections today.

We correlate major changes in the use of cipher suites and TLS extensions with the timing of specific TLS attacks and assess the impact of security research. We observe a few cases, like HeartBleed, where the ecosystem changed very quickly. We also observe a strong correlation in time between Snowden revelations and the change to forward-secret ciphers. On the other hand, it took several years for RC4 usage to reduce significantly after attacks against RC4 were discovered. Moreover, we do not observe a change in our traffic after CBC attacks, possibly due to the lack of better available options at the time or due to the existence of patches. At last, server support for SSL3 is still embarassingly high, despite the severity of POODLE and RC4 attacks.

Our main contributions are as follows: *(i)* We conduct a large-scale longitudinal study on the evolution of the TLS ecosystem since 2012. We place a special focus on high-profile attacks and detail the impact that these attacks have had on the ecosystem. *(ii)* We present the largest, and longest-running, passive TLS monitoring dataset comprising 319.3B TLS connections collected since February 2012. We use our passive dataset, together with a large active scan dataset, to detail the changes to the TLS ecosystem since 2012. Where possible, we correlate the changes that we observe in these datasets with the timing of specific attacks on TLS. *(iii)* We create the largest database of TLS client fingerprints to-date. We use these fingerprints to identify the evolution of client software on the Internet. We will release our TLS client fingerprints to the community after publication of this paper[1].

The remainder of this paper is structured as follows: §2 covers the background. §3 introduces the datasets used in our paper and §4 details our TLS client fingerprinting. §5 examines the impact TLS vulnerabilities had on the TLS ecosystem and §6 investigates TLS characteristics not directly related to attacks. §7 discusses the implications of our findings. Finally, §8 presents the related work and §9 concludes.

2 BACKGROUND

This section gives a brief overview of relevant parts of the TLS protocol, as well as the TLS vulnerabilities we analyze. For a detailed TLS description, we refer the reader to RFC 5246 [40]. Note that while we predominantly use the term TLS, our measurements also cover the earlier Secure Sockets Layer (SSL) protocol.

2.1 TLS Connection Establishment

To establish a TLS connection, the client and server first negotiate the parameters of the connection using Client Hello and Server Hello messages. These two messages are not encrypted, allowing passive observation. The client first sends a Client Hello message to the server listing its capabilities such as the maximum protocol version, the cipher suites, and elliptic curves it supports. The server then chooses its preferred options, among those offered by the client, and informs the client of its choices in the Server Hello message. Thus, the server chooses the final protocol version, cipher suite, and other parameters that will be used in the secure channel.

Table 1 lists the release dates of all SSL/TLS versions. A key feature added in TLS 1.0 over the earlier SSL protocol versions are TLS extensions used for adding TLS functionality and TLS protocol message formats [40]. The client includes in the Client Hello all the extensions that it supports and the server simply ignores extensions that it does not understand. As of March 2018, 28 TLS extensions have been standardized [57]. It is also possible for anyone to define their own extensions.

2.2 TLS Attacks

Many attacks against SSL/TLS have been discovered in recent years. As a result, both the protocol specification and individual implementations have been significantly revised. In some cases, just changing client and/or server configurations was sufficient mitigation – for example to avoid using weak cipher suites. In other cases, where good alternatives were not widely supported or where support for legacy was needed, more invasive changes to implementations were required. Finally, as we will see, clients and servers have changed the versions of the protocol they negotiate in response to attacks. Below, we provide a brief description and a timeline of the most notable attacks, ordered by disclosure date.

BEAST (09/06/2011) BEAST allows a MITM attacker to decrypt data passing between a webserver and an end-user browser for CBC cipher suites in TLS versions 1.0 and earlier [26, 41]. The attack exploits these versions' reliance on predictable IVs; these are not present in TLS 1.1 and 1.2. BEAST requires the attacker to have very fine control over the placement of chosen plaintext blocks in the client's messages. Client-side mitigation was possible and widely implemented. As a response to BEAST, server operators were encouraged to enforce the use of RC4 suites whenever TLS 1.0 and lower is offered by the client [74].

Lucky 13 (06/12/2012) Lucky 13 is a cryptographic timing attack against TLS implementations using CBC mode [10, 29]. All TLS cipher suites using CBC-mode encryption are potentially vulnerable to this attack. The best counter-measure is to switch to using authenticated encryption with associated data (AEAD) cipher suites in TLS 1.2, but at the time of the attack's disclosure, most implementations did not support TLS 1.2. Consequently, and in view of the security issues in RC4 disclosed soon after Lucky 13, most implementations took steps to try to limit the timing leaks. Follow-up work has shown that this was not always successful [8, 14, 79].

RC4 attacks (12/3/2013 and later) This sequence of attacks [9, 20, 27, 33, 48, 85] exploited biases in the output of the RC4 stream cipher to recover plain-texts that are sent repeatedly under the protection of TLS, e.g., HTTP cookies or passwords. The attacks grew stronger over time, making the use of RC4 increasingly indefensible as an encryption option, and spurring the adoption of TLS 1.2 with its stronger, modern AEAD cipher suites.

[1] https://github.com/platonK/tls_fingerprints

Heartbleed (03/12/2013) Heartbleed is an OpenSSL bug allowing remote attackers to obtain sensitive information from process memory via packets that trigger a buffer over-read [44, 87]. It was introduced into the software in 2012 and publicly disclosed in April 2014 [30]. Due to the severity of the attack and the ease of exploiting it, Heartbleed attracted a lot of media attention.

POODLE (14/05/2014) POODLE is a cryptographic exploit which takes advantage of TLS clients' willingness to fall back to SSL 3 and the specifics of CBC-mode padding in SSL 3 [31, 63]. Since RC4, the only other encryption option in SSL 3, was already considered to be weak, the only real countermeasures to this attack are to disable SSL 3 and support TLS downgrade protection.

FREAK (03/03/2015) Export-grade ciphersuites have limited key lengths to satisfy US cryptography export laws of the 1990s. They offer approximately 40 bits of security and can today be broken quite easily [82]. Since the early 2000s, with the relaxation of export controls, cryptography no longer has to be hobbled in this way. FREAK allows an MITM attacker to downgrade TLS connections to use export-grade cryptography (512-bit RSA for key transport) [17, 32, 51]. The attack is possible when a client connects to a web server that supports RSA_EXPORT cipher suites and the client requests an RSA cipher suite. The attack also relies on a client-side bug where the client accepts handshake messages of the special format encountered in export handshakes, when no such handshake was requested by the client. To mitigate this vulnerability, both clients and servers should disable support for export cipher suites.

Logjam (20/05/2015) Logjam, reminiscent of FREAK, allows an MITM attacker to attack connections if the server supports DHE_EXPORT cipher suites and the client requests a DHE cipher suite [6, 34]. The effect is to downgrade the security of TLS connections to the level of export-grade cryptography (512-bit DH key exchange). Logjam takes advantage of the protocol-level flaw in TLS that the server signature in a DHE handshake does not cover the entire handshake but only the DH parameters. To mitigate this vulnerability, both clients and servers should disable export cipher suites.

Sweet32 (31/08/2016) The DES and Triple-DES (3DES) ciphers with their 64-bit block size are vulnerable to a birthday-bound attack on CBC-mode which makes it possible for an MITM attacker to recover plaintext from long-duration connections [18, 37]. To counter this attack it is necessary to stop using 64-bit block ciphers or to re-key the session frequently.

Other Attacks. Besides the aforementioned attacks, other TLS vulnerabilities such as CRIME [28], SLOTH [35] and DROWN [36] were also publicly disclosed. They are not included in our study because we are not able to study them in detail using our dataset .

3 DATASETS

This section introduces the datasets used in our study.

3.1 ICSI SSL Notary

The main data source for this study is the ICSI SSL Notary [12]. The Notary passively collects metadata about outgoing SSL/TLS connections on all ports from several universities and research networks mainly located in North America. The dataset consists of 319.3B connections collected over 6 years, from February 2012 to March 2018. During this period it collected 31.5M unique certificates. This number excludes short-lived certificates used by GRID Computing and Tor [68].

Version	Release Date
SSL 2	Feb. 1995
SSL 3	Nov. 1996
TLS 1.0	Jan. 1999
TLS 1.1	Apr. 2006
TLS 1.2	Aug. 2008
TLS 1.3	Aug. 2018

Table 1: Release dates of all SSL/TLS versions.

The Notary uses the Bro Network Security Monitor [1] to collect its data. We significantly extended the TLS-related features in Bro since we started our data collection back in 2012.

The Notary dataset is fundamentally different from active scan datasets because it focuses on *connections* instead of *servers*. It captures how TLS is actually used on the Internet and includes the interplay between clients and servers. It also emphasizes connections to services that users commonly use and de-emphasizes services rarely accessed by users.We consider this a feature since it shows the current makeup of the traffic on the Internet.

We note that our dataset exhibits artifacts of the collection process that are beyond our control. As we leverage operational setups that run our analysis on top of their normal duties, we must accept occasional outages, packets drops (e.g., due to CPU overload) and misconfigurations. As such our data collection effort is designed as a "best effort" process: we take what we get but generally cannot quantify what we miss. However, given the large total volume across the sites, we consider the aggregate as representative of many properties of real-world SSL/TLS activity.

3.2 Censys

We complement the passive Notary dataset with active scans from Censys [42]. Censys performs periodic Internet-wide TLS scans and provides a search engine to explore the collected scan data. Censys uses a combination of ZMap [46] and ZGrab [4] for data collection.

We use multiple Censys datasets: First, we use TLS scans of the IPv4 address space on TCP port 443 covering 46M hosts and 535M unique certificates. Second, we use HTTPS scans of the top 1M Alexa most popular websites. Both scans offer the same set of cipher suites as a 2015 version of Chrome including a number of strong ciphers such as AES-GCM cipher suites with forward secrecy, as well as weaker CBC, RC4, and 3DES cipher suites. Finally, we use weekly scans of the IPv4 address space on TCP port 443 that offer SSL 3 as the sole supported protocol version and other scans that look for Export-grade cipher suite support. Censys scans are available starting from August 22nd 2015; in our paper we use the data till May 13 2018.

These datasets provide a temporal view of publicly-reachable TLS servers active in the IPv4 address space over 32 months, allowing us to study how the choices of SSL/TLS versions and cipher suites by

servers change over time, and whether these changes correspond to attack disclosures and vulnerability reports.

3.3 Ethical Considerations

The passive data collection performed by the ICSI SSL Notary was cleared by the responsible parties at each contributing institution. Note that the ICSI SSL Notary specifically excludes or anonymizes sensitive information, such as client IP addresses. Censys performs a number of steps to make sure that data is collected ethically; these are outlined in [42].

4 TLS CLIENT FINGERPRINTING

As of May 2018, IANA has registered almost 200 cipher suites, 28 TLS extension, and 35 elliptic curves values [56, 57]. The different combinations of these parameters in a Client Hello message (see §2) can reveal the client from which a given TLS flow has emanated. Matching client fingerprints against the Notary dataset allows us to analyze the evolution of TLS usage of specific applications, e.g., to see how they react to discovered TLS vulnerabilities.

In this work, a TLS client fingerprint is the concatenation of four features extracted from the Client Hello: *(i)* the cipher suite list, *(ii)* the list client extensions, *(iii)* Supported Elliptic Curves (EC), and *(iv)* the Supported EC Point Formats extension. All features are stored in the order they appear in the Client Hello.

Some Google software like Chrome uses a feature called GREASE [38] that adds a list of invalid values to these fields to improve server tolerance of future new extensions. We identify and remove these values from handshakes.

Our feature selection is similar to the feature selection of previous work, albeit slightly more constrained because a few items are not available in our passive dataset. Prior work has included additional fields like the client TLS version, compression methods, and signature algorithms [22,45]. We plan adding these fields to the Notary in the future. Fingerprints with more fields allow a more specific identification of client software. To determine the impact we take the fingerprints of [22] and apply our more restrictive methodology to them. Originally 2.4% of the fingerprints collide; with our methodology this increases to 7.3%. Thus, our collection gives slightly less distinct results.

Each fingerprint in our database maps to a program or library and the version range that the fingerprint covers. When a collision with a different kind of software or library occurs we remove the fingerprint from the database; it cannot uniquely identify a client. When a collision between a specific software and a library occurs we assume that the software uses the library. Due to this, e.g., Chrome on Android is just identified as "Android SDK", conflict resolution is performed manually.

We build our fingerprint database from several sources: we use the data of previous research studies [45, 71], use BrowserStack [2] to gather the fingerprints of browsers and mobile devices, compile multiple versions of OpenSSL to gather their data, and manually identify TLS clients by examining the hosts they connect to.

4.0.1 TLS Client Fingerprint Coverage. As discussed in §3.1, the Notary gradually incorporated new TLS-related features since its initial deployment in February 2012. The fields necessary for fingerprinting have been introduced in February 2014. As a result, we can only match our fingerprints to 191.9G (60%) total flows containing the necessary information. When applied on those 191.9G TLS connections, our 1,684 fingerprints allow us to identify the TLS client originating 69.23% of them. Table 2 summarizes our fingerprint database. The whole Notary dataset contains 69,874 unique TLS connection fingerprints. The coverage follows a power law distribution: the 10 most common fingerprints explain 25.9% of the total Notary traffic. These 10 fingerprints are associated to popular web browsers such as Chrome and Safari and OS-provided libraries (mainly Android and iOS). The most common unlabeled fingerprint is responsible for only 1% of remaining traffic. This indicates that obtaining more fingerprints associated with unpopular TLS clients may not translate in significantly improving our coverage.

Type (Examples)	**№ FPs**	**Coverage**
Libraries (OpenSSL, MS CryptoAPI)	700	46.49%
Browsers (Chrome, Firefox)	193	15.63%
OS Tools and Services (Apple Spotlight)	13	2.29%
Mobile apps (Facebook, Hola VPN)	489	1.35%
Dev. tools (Flux, git)	12	0.88%
AV (Avast, Bluecoat Proxy, Kaspersky)	44	0.85%
Cloud Storage (Dropbox)	29	0.71%
Email (Apple mail, Thunderbird)	33	0.58%
Malware & PUP (Zbot, InstallMoney)	49	0.48%
All	1,684	69.23%

Table 2: Fingerprint summary. The table reports number of unique fingerprints, and amount of matching Notary connections for each class.

4.1 Software use over time

As far as we are aware, our work is the first time that TLS fingerprinting is used on a large-scale dataset covering several years. A few interesting questions that arise in this scenario are: How long do we see specific TLS fingerprints? Do fingerprints commonly only appear for short periods of time (which might indicate frequent client updates) or are a lot of them unchanged for years?

In our dataset we have 69,874 usable fingerprints (for which all needed features that we are using are present), starting from Oct. 2014. We define the duration a fingerprint was seen as the time between the first time and the last time it was seen. The maximum duration in our dataset is 1,235 days (3 years, 4 months). The median duration a fingerprint was seen is 1 day, the mean 158.8 days, the 3rd quantile 171 days and the standard deviation is 302.31 days.

It is interesting that a lot of fingerprints are only seen very briefly and do not reappear later. Our dataset reveals an extreme bias with 42,188 of the 69,874 fingerprints only appearing on a single day. These 42,188 fingerprints are only responsible for 801,232 of the 191B total connections for which we have fingerprints. We are not sure which software these fingerprints originate from. One possibility is that there is software that does not send its ciphersuites in a fixed order (due to a bug, perhaps), causing an explosion of fingerprints.

Looking at the other end reveals a different picture. There are 1,203 fingerprints that we see for more than 1,200 days. These

fingerprints are responsible for 21.75% of connections (for which we have fingerprints). We were able to identify the software for 343 of these (responsible for 18.08% of connections). The top software identified is iPad Air (library), Safari, Android SDK, as well as Chrome, Firefox, and the MacOs Mail App.

This indicates that significant numbers of connections are caused by software that has either not been updated since 2014 or not changed its fingerprints since then. The latter seems unlikely since it takes significant effort to not change the fingerprint at all, especially since doing so prevents adoption of new features.

5 VULNERABILITY ANALYSIS

In this section, we analyze the TLS ecosystem evolution by examining negotiated TLS parameters in TLS connections in the Notary dataset, as well as client TLS configurations. We view our analysis through the prism of the disclosed TLS attacks listed in §2.2. We examine if and how server and client configurations change, and how traffic patterns shift.

5.1 Legacy SSL Versions

Old protocol versions, especially SSL 2 and 3, are no longer considered secure [78, 81]. Our passive traffic shows that there is still a marginal number of connections that use SSL 2 or 3. For example, we saw 1.2K SSL 2, as well as 360.1K (< 0.01% of total) SSL 3 connections in February 2018. We have not observed SSL 3 in a significant amount of connections since mid-2014 and never observed significant numbers of SSL 2 connections.

All observed SSL 2 connections terminate at servers of a single University; some of them on the Nagios port, a piece of system monitoring software. SSL 3 paints a more varied picture—we see connections to 1,789 different servers indicating use of it; 30 servers receive more than 1,000 connections. The 4 servers receiving more than 50,000 connections belong to Symantec and Wayport.

Looking at Censys data for server support shows that in September 2015, more than 45% of servers still supported SSL 3. The number has decreased since then—as of the beginning of May 2018, less than 25% of servers support SSL 3. However, considering that TLS 1.0 was standardized in 1999 (Table 1) and that SSL 3 has been considered badly broken since the POODLE attack in 2014 (on top of the RC4 attacks), this still seems like a large number. Censys does not scan using SSL 2 and we have no information about its support.

5.2 CBC Attacks

There are three publicly known attacks against CBC-mode (see §2): BEAST (2011), Lucky 13 (2013), and POODLE (2014). Each CBC-mode attack requires different countermeasures. BEAST could only be fixed on the client-side, for example using record splitting. However, it was also recommended to switch to newer TLS versions (TLS 1.1 is immune to the attack) or to avoid CBC-mode on the server side. Because of lack of widespread support for TLS 1.1 and TLS 1.2 at the time, this would mean switching to RC4 cipher suites. In the case of Lucky 13, complex patches were rolled out on both clients and servers. Again, TLS 1.2 was not widely available at the time, so switching to authenticated encryption (AEAD) was not practical. However, Lucky 13 in combination with the RC4 attacks

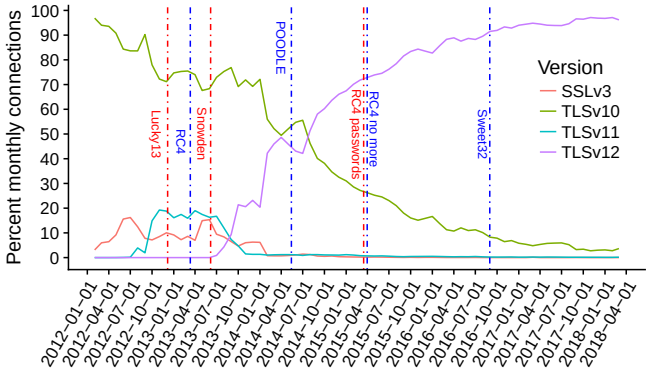

Figure 1: Negotiated SSL/TLS Versions. Vertical lines show dates of high-profile attacks.

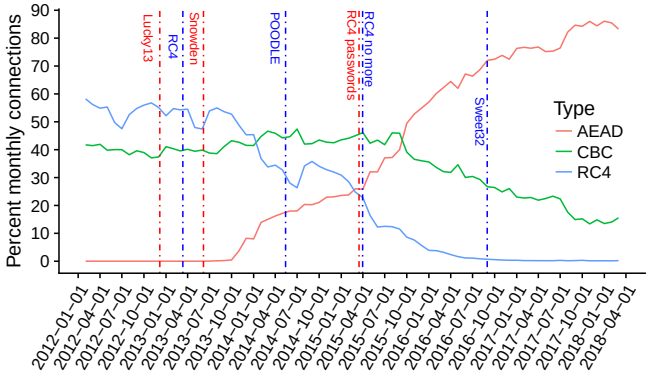

Figure 2: Negotiated connections using RC4, CBC or AEAD cipher suites.

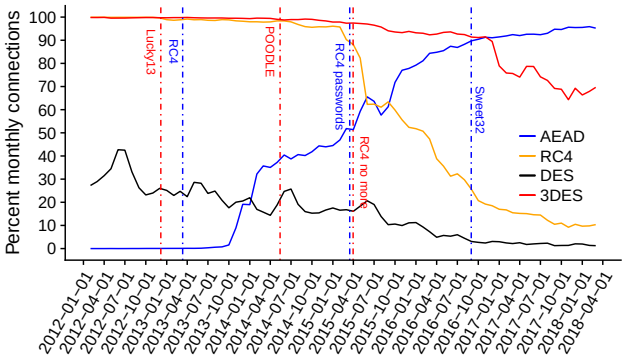

Figure 3: Connections with client advertising support for RC4, DES, 3DES or AEAD cipher suites. Total CBC-mode is always above 99%.

can be seen as having spurred the adoption of TLS 1.2. The best defense against POODLE is to disable SSL 3 completely (since it also enables attacks against later versions of TLS due to TLS fallback behavior).

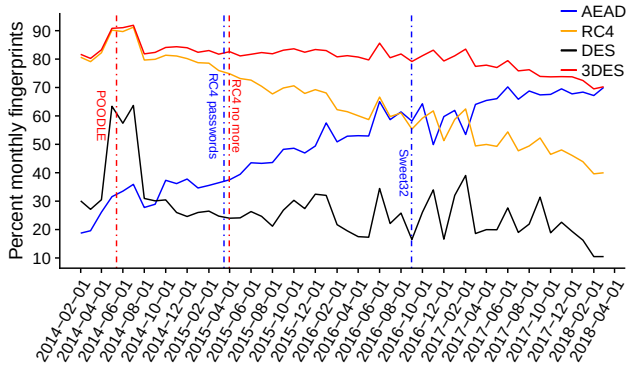

Figure 4: TLS fingerprints with support for RC4, DES, 3DES or AEAD cipher suites. CBC-mode support is near universal.

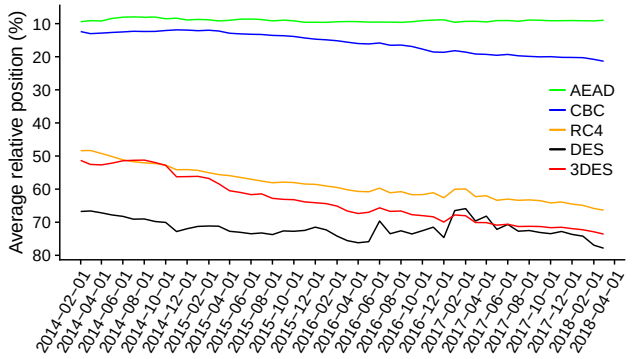

Figure 5: Average relative position of the first AEAD, CBC, RC4, DES or 3DES client-advertised cipher.

Figure 1 shows that TLS 1.0 stayed in use for years after BEAST became public, though there is a steady decline over the period of our study from nearly 100% in January 2012 to 2.8% in February 2018. Interestingly, there is a period of time from mid-2012 to late 2013 where TLS 1.1 gains traction; this may have been in response to BEAST. The TLS 1.1 traffic drops off quickly as TLS 1.2 takes off in late 2013. Figure 2 shows both that RC4 traffic did not increase significantly post-BEAST, and also that CBC-mode remained popular right up until August 2015, when the amount of CBC-mode traffic finally starts to show a decline. This is all suggestive that client-side patches for BEAST were seen as sufficient mitigation for the attack (which anyway required a specific browser vulnerability to gain the fine control over the placement of chosen plaintext blocks to make the attack feasible [26, 41]).

We also see a drop off in SSL 3 traffic in Figure 1. However, this predates the publication of the POODLE attack, regarded as the "SSL 3 killer". Thus we cannot directly say that POODLE influenced the move away from SSL 3, though this could be the case if information about the attack was circulating prior to publication as is often the case.

Browser	Ver.	Date	№ CBC ciphersuites
Firefox	27	04/02/2014	Reduced from 29 to 17
	33	14/10/2014	Reduced to 10
	37	31/03/2015	Reduced to 9
	60 beta	14/03/2018	Reduced to 5
Chrome	29	20/08/2013	Reduced from 29 to 16
	31	12/11/2013	Reduced to 10
	41	03/03/2015	Reduced to 9
	49	02/03/2016	Reduced to 7
	56	25/01/2017	Reduced to 5
Opera	15	02/07/2013	Increased from 25 to 29
	16	27/08/2013	Reduced to 16
	18	19/11/2013	Reduced to 10
	28	10/03/2015	Reduced to 9
	30	09/06/2015	Reduced to 7
	43	07/02/2017	Reduced to 5
Safari	7.1	18/09/2014	Increased from 28 to 30
	9	01/09/2016	Reduced to 15
	10.1	19/07/2017	Reduced to 12

Table 3: Changes in the number of CBC ciphersuites offered by major browsers.

Concerning the impact of Lucky 13 on CBC-mode usage, we do not see a clear shift in traffic patterns. This may be because all major implementations patched against the attack and this was considered sufficient mitigation. We point to the rise in TLS 1.2 traffic and the use of AEAD cipher suites as a longer-term reaction to the sequence of attacks on CBC-mode (including Lucky 13) and RC4.

During our observation period basically all of the TLS clients in our dataset support CBC-mode. However, while clients still supported CBC-mode, they gradually "downgraded" support for it over time. Table 3 lists the changes in the TLS configurations of 4 major browsers as regards as their support of CBC-mode cipher suites. It shows that Firefox, Chrome, and Opera significantly downgraded their support between August 2013 and February 2014. Safari did so in September 2016. On the other hand, while clients are reducing the number of CBC-mode cipher suites they offer, Figure 5 shows that they still place CBC-mode quite high in their list of preferences, with little change in the relative position of the first offered CBC-mode cipher suite over time. Figure 5 shows the relative position over time of various client-advertised ciphers. Ciphers found at the top (AEAD, and CBC) are (on average) placed in the beginning of the client advertised cipher list. Even today, as Figure 3 and Figure 4 show, almost all clients still offer CBC-mode cipher suites. Presumably this is done in order to be able to establish secure connections with servers that have not been updated to TLS 1.2 yet. According to Censys, server side support of CBC ciphers has also dropped significantly, with the percentage of servers choosing CBC-mode cipher suites over other cipher suites offered going down from 54% in September 2015 to 35% in May 2018, with the highest drop happening between late-2016 and mid-2017

Browser	Ver.	Date	№ RC4 ciphersuites
Firefox	27	04/12/2014	Reduced from 6 to 4
	36	24/02/2015	Fallback only[1][5]
	38	12/05/2015	Only available for whitelist [5]
	44	26/01/2016	Removed completely
Chrome	29	20/08/2013	Reduced from 6 to 4
	43	19/05/2015	Removed completely
Opera	15	02/07/2013	Increased from 2 to 6
	16	27/08/2013	Reduced to 4
	30	09/06/2015	Removed completely
IE/Edge[2]	13	20/05/2015	All RC4 ciphersuites removed
Safari	6	25/02/2012	Reduced from 7 to 6
	9	30/09/2015	Reduced to 4
	10.1	20/09/2016	Removed completely

[1] RC4 only offered if connection without RC4 failed.
[2] Not removed in Windows XP.

Table 4: Changes in the support of RC4 ciphersuites by major browsers.

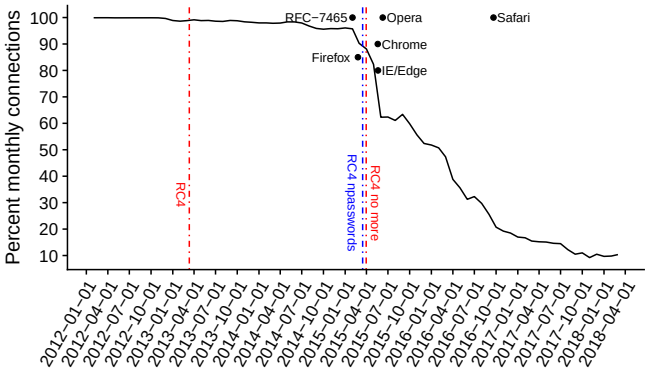

Figure 6: Percentage of connections where the client advertises RC4 cipher suites. Lines represent the dates of high profile attacks; black dots the dates that browsers dropped RC4 support.

5.3 RC4 Attacks

Multiple attacks targeting RC4 were found in recent years [9, 48, 85]. It has been recommended since at least 2015 that RC4 should no longer be used [67]. Note that this was a change from earlier recommendations: after CBC-mode problems were discovered, RC4 was often recommended as an alternative cipher.

Figure 2 shows the percentage of connections that negotiate an RC4 cipher suite and captures the drop of RC4 usage from 60% in August 2013 to almost zero in March 2018. A similar observation can be made for Figure 6 that shows the percentage of connections in which clients advertise RC4 cipher suites. Table 4 shows the dates at which each browser reduced and completely removed their support for RC4.

A big drop in clients advertising RC4 connections can be observed at the beginning of 2015 in Figure 6, correlating in time with the decision of Chrome, Firefox and IE/Edge to completely remove support for RC4. However, matching our fingerprints to Notary traffic reveals that a residual number of clients continued to advertise RC4 for some time after browsers officially dropped it, indicating a user population that does not quickly update. This is supported by our findings in §4.1.

Interestingly, the drop in advertised RC4 cipher suites, driven mostly by changes in major browsers, comes almost 18 months after the beginning of the reduction in the number of TLS connections that negotiate RC4 seen in Figure 2. This demonstrates that removal of RC4 started on the server side and only later happened on the client side. This is further illustrated in Figure 4 which shows that the removal of RC4, when counted by distinct TLS fingerprints, happened much more slowly. Indeed, 39.9% of the observed fingerprints still support RC4 as of March 2018.

Censys does not contain an active scan that tests for just RC4 support. However, it shows that even today (May 2018) 1,342,659 servers (3.4%) will choose RC4 given an older Chrome cipher list (down from 11.2% in September 2015). Curiously some server configurations will choose RC4 even if they support much more secure ciphers. For example, the website of one of the largest banks in Iran, `bankmellat.ir`, will use RC4 despite much stronger ciphers being offered. When removing RC4 from the list, it will switch to a modern AEAD cipher. SSL Pulse [70] tests RC4 support for about 150K websites with selection based on Alexa's list of most popular sites. 19.1% of servers still support RC4 cipher suites (down from 92.8% in their first scan in October 2013). They also report that 1 site still supports *only* RC4 (down from 4,248 or 2.6% of sites surveyed in October 2013).

5.4 Heartbleed

As mentioned in §2, Heartbleed was an OpenSSL vulnerability in the processing of the Heartbeat extension that could leak process memory, including private keys. The group that discovered Heartbleed estimates that over 66% of servers on the Internet might have been vulnerable to it before disclosure [80]; however, this number is just based on the number of Apache and NGinx servers, and these could have been using other libraries. Some sites might also have deactivated this feature. Durumeric et al. [44] estimated that at least 23.7% of servers were vulnerable at the time when Heartbleed became public. In their first scan in 2014, they found 5.9% of servers to be vulnerable and 11.4% of servers supporting the Heartbeat extension.

Heartbleed was extensively featured in the media and received a quick response from server operators; the number of vulnerable servers dropped to less than 2% in a month [44]. Looking at Heartbleed today, according to Censys scans, 0.32% of servers were still vulnerable in May 2018, pointing to a significant long tail effect. 34% of servers now support the Heartbeat extension. While purely passive observation cannot show if hosts are still vulnerable to Heartbleed, we can monitor whether the Heartbeat extension is still being actively used in the wild: 3% of the observed TLS connection negotiations still use it (i.e., the extension is offered by client and acknowledged by the server) in May 2018. We find this odd in view of the fact that the Heartbeat extension is a DTLS-specific feature that is used to keep connections alive and detect the path

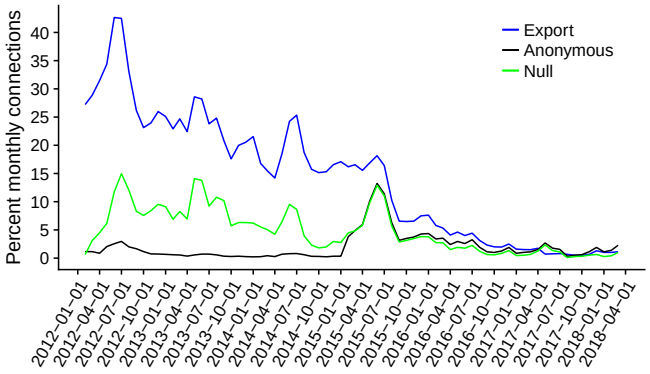

Figure 7: Percentage of monthly connections where the client advertises Export, NULL, or Anonymous ciphers.

MTU [77], both features that are not needed in TCP based TLS connections.

5.5 FREAK, Logjam and Export Ciphers

Web servers supporting export ciphersuites are vulnerable to the FREAK [17] and Logjam [6] attacks (see §2). Looking at our passive data, we can conclude that export ciphers are basically not negotiated in the wild: only 677 connections were negotiated with these ciphers in 2018. However, we cannot infer from this that FREAK and Logjam attacks were not mounted in practice against clients. This is because our data gathering is done close to the clients, and at the clients, the TLS handshakes do not contain export cipher suites even if they are undergoing a FREAK or Logjam attack.

A closer look at the connections supporting export cipher suites reveals that the connections either terminated at servers of a university using anonymous export cipher suites (the port number suggests Nagios servers) or are associated with Interwise products (according to the certificate served), a company providing voice & videoconferencing services. For the university Nagios connections, secure cipher suites were offered by the client; the server chose export ciphersuites. In the case of Interwise, the clients offered a non-export cipher suite (RC4_128_SHA) and the server chose an export RC4 cipher suite (EXP_RC4_40_MD5). This deviates from the TLS protocol specification. Our logs indicate that at least some of the sessions were successfully established (both sides sent a Change Cipher Spec). We are unsure about what exactly is behind this unexpected behavior, but this again deviates from normal TLS negotiation.

Figure 7 shows that client support for export ciphers has decreased significantly over the last few years. In 2012 (more than 10 years after the restrictions were lifted) it still was advertised in 28.19% of connections (2018: 1.03%). The steady downward trend can be attributed to a growing awareness of the dangers of supporting weak cipher suites, as exemplified by FREAK and Logjam.

5.6 Sweet32, DES and 3DES

Sweet32 is a birthday attack against 64-bit block ciphers such as DES and 3DES [18] (see §2). Looking at our passive data, in 2018 0.3% of connections negotiated a 3DES cipher suite. By comparison, in 2012 (June-August) 1.4% of connections negotiated 3DES. Generally, usage always has been relatively low with the highest peaks reaching 5%.

Browser	Ver.	Date	№ 3DES ciphersuites
Firefox	27	04/02/2014	Reduced from 8 to 3
	33	14/10/2014	Reduced to 1
Chrome	29	20/08/2013	Reduced from 8 to 1
Opera	16	27/08/2013	Reduced from 8 to 1
Safari	6.2	18/09/2014	Reduced from 7 to 6
	9.0	30/09/2015	Reduced to 3

Table 5: Changes in the number of 3DES ciphersuites offered by major browsers.

On the other hand, Figure 3 shows that almost all clients advertised 3DES up to the end of 2016, a few months after the disclosure of the attack. The figure today still stands at more than 69%. Figure 4 backs this up, showing that even today more than 70% of fingerprinted clients offer 3DES. Table 5 shows how 3DES support has changed in popular browsers; notably, all major browsers still support 3DES. We find this set of findings remarkable given the poor performance of the algorithm (OpenSSL's inbuilt speed test shows it to be 10 times slower than AES-128, even without hardware support), its extreme age (it was developed in the 1970s), and the wide availability of more secure, faster options. However, it is perhaps justifiable to keep 3DES as a "cipher of last resort" for clients connecting to out-dated servers. In connection with this, active scan data from Censys indicates that the popularity of the only 3DES cipher offered by the scan has reduced over time, with the percentage of the servers choosing it (despite its placement at the bottom of the list and much stronger cipher suites being offered) dropping to 0.25% in May 2018 from 0.54% in August 2015. This suggests a long tail on the server side, but one large enough for clients to justify their continuing to offer 3DES. Censys scans do not offer any DES cipher suites.

6 NOTABLE ECOSYSTEM FINDINGS

In this section we look at other characteristics of the TLS ecosystem that do not directly have to do with attacks. We discuss the use of weak as well as strong ciphers and examine how TLS 1.3 is already being used.

6.1 NULL Cipher Suites

NULL ciphers provide integrity but no confidentiality—data is sent in the clear and can be read by any observer. Figure 7 reveals that a relatively large number of connections and software offer at least one NULL cipher in the handshake. For most of these connections and fingerprints we could not find any matching software: these connections do not originate from any Browser known to us, nor any known library in its default configuration. However, we do identify two Android applications, *Craftar Image Recognition*, and *Lookout Personal* that advertise support for them. The latter is an identify theft protection application.

Our hypothesis concerning the source of these connections is supported when we look in detail at connections that are actually

established using NULL ciphers. Examining our dataset reveals that connections where the server chooses a NULL cipher are rare; over all of our dataset 2.84% of connections were established using a NULL cipher. We examined all connections using a NULL cipher in 2018 (42.3M connections or 0.42% of connections). Nearly all these connections (99.99%) were identifiable as GRID traffic that is used for data transfers between scientific institutions, where TLS is only used for mutual authentication of client and server.

While using NULL ciphers in GRID computing environment might make sense, it is alarming that a not insignificant number of the client fingerprints we see (8% of fingerprints and 0.46% of connections in 2018) offer a NULL cipher. If an attacker were able to perform a downgrade attack they might be able to force an unencrypted connection for such connections.

There is also the NULL_WITH_NULL_NULL cipher suite offering neither integrity nor confidentiality. Since 2012 we have seen a total of 198.3K connections using this cipher suite, with 198 in 2018. They terminate at the Nagios servers mentioned in §5.5.

6.2 Anonymous cipher suites

Anonymous cipher suites provide record layer protection (integrity and confidentiality) but the key establishment process is not authenticated. There are 19 such cipher suites, all identifiable by the keyword "Anon" in their name. Typically, a Diffie-Hellman key exchange is used to establish a symmetric cipher but the server does not present a certificate. Figure 7 shows the percentage of connections and unique TLS fingerprints that advertise anonymous cipher suites. As in the case of NULL ciphers, while there is a significant number of fingerprints that include anonymous cipher suites, as well as a few percent of connections that advertise them, we could not determine the vast majority of applications responsible for this. However, among those that we can identify we find a security scanner (i.e., *Shodan*) and Android security applications (i.e., *Lookout Personal*, *Kaspersky*). Once again we assume the main reason might be poorly written client applications. We do not have an explanation for the spike in clients offering such cipher suites in mid-2015, where the figure jumped from 5.8% to 12.9% in the space of two months. This spike correlates in time with a spike in NULL cipher suites being offered.

Similarly to the case of NULL ciphers, only a minuscule number of connections successfully negotiate the use of an anonymous cipher suite: 0.17% of connections in the whole dataset and 0.60% in 2018. Examining the successful connections reveals that nearly all of them are caused by Nagios. Nagios uses anonymous cipher suites in combination with its own authentication scheme that is performed after the TLS connection is established.

The fact that so many clients offer anonymous cipher suites is alarming. This is because client/server combinations offering/accepting such cipher suites are trivially vulnerable to MITM attack, even if the client also offers non-anonymous cipher suites in preference to the anonymous ones in the Client Hello.

6.3 Strong Cipher Suites

6.3.1 Forward secrecy. In TLS handshakes that use RSA key transport to establish keying material, the symmetric keys used in the record protocol can be recovered by anyone in possession of the server's private key after observing the handshake. If an attacker recorded complete TLS sessions, then a later compromise of the server's private key would compromise all sessions. Forward secrecy (FS) decouples the symmetric record protocol keys from the long-lived server key. In TLS this is achieved by performing a DHE or an ECDHE key exchange.

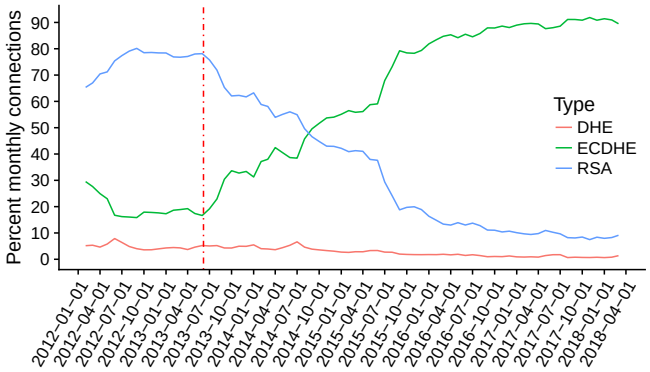

Figure 8: Negotiated RSA and forward secret connections. Dotted line shows the date of first Snowden revelations.

Figure 8 shows the number of connections that we see using RSA, DHE and ECDHE key exchanges. It shows that there has been a significant shift in the last few years away from RSA-based key establishment; now the vast majority of connections use ECDHE; meanwhile DHE never found much use.

This result is especially interesting when also examining client support for FS cipher suites: even at the beginning of our measurement period in 2012 more than 80% of clients supported FS cipher suites, and this quickly increased to nearly 100%. Thus servers chose to not negotiate FS cipher suites for a long time even when clients supported them. We find this interesting because making the switch does not require issuing new RSA keys and certificates to the server. Moreover, ECDHE exchanges are in general faster to perform for servers than RSA-based ones, so there is a performance incentive to switch.

The Snowden revelations in June 2013 raised awareness of the public of the importance of forward secrecy [15]. Figure 8 shows that the Snowden revelations coincide with the start of a significant shift to use of FS cipher suites. While it looks like there was already a small shift towards FS cipher suites before, there was a tremendous shift immediately after these revelations. This correlation in time does not necessarily mean causality.

TLS also supports static DH and ECDH key exchanges that are not forward secret. We also see some use of these: DH is used in 0.00% of connections (2018: 4 total), ECDH in 0.27% (2018: 20.2K total). DH connections in 2018 terminate at two now unreachable servers; ECDH nearly exclusively at Splunk servers on port 9997.

6.3.2 Authenticated Encryption with Associated Data. Authenticated Encryption with Associated Data or AEAD provides message integrity in addition to message confidentiality in a single cryptographic transform. In TLS, AEAD is supported starting with TLS 1.2, for example with AES in Galois Counter Mode (AES-GCM). Happily, because of widespread support at the CPU instruction

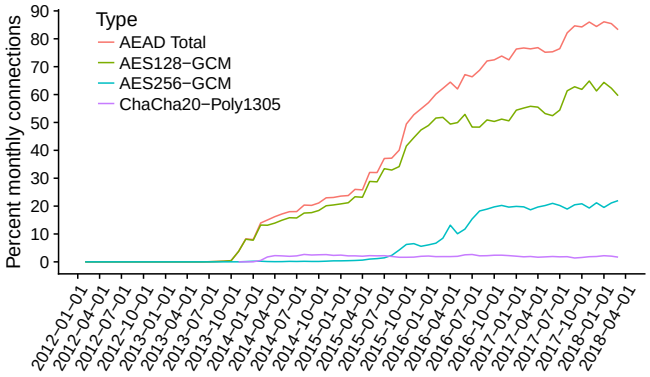

Figure 9: Percentage of connections that use AES-GCM(128/256), ChaCha20-Poly1305, and number of connections using any AEAD cipher.

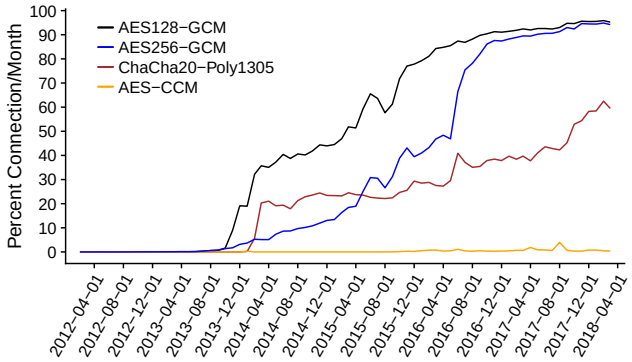

Figure 10: Connections advertising AES-GCM ChaCha20-Poly1305, and AES-CCM.

Browser	Ver.	Date	Protocol Support
Firefox	27	04/02/2014	TLS 1.1/1.2 supported
	37	31/03/2015	SSL 3 fallback removed
	60	16/05/2018	TLS 1.3 supported
Chrome	22	25/09/2012	TLS 1.1 supported
	29	20/08/2013	TLS 1.2 supported
	39	18/11/2014	SSL 3 fallback removed
Internet Explorer	11	01/11/2013	TLS 1.1/1.2 supported
Opera	16	27/08/2013	TLS 1.1 supported
	27	22/01/2015	SSL 3 fallback removed
Safari	7	22/10/2013	TLS 1.1/1.2 supported
	9	30/09/2015	SSL 3 support removed

Table 6: Browser TLS version support.

level for the cryptographic operations involved, GCM is roughly twice as fast as CBC-mode cipher suites on mainstream CPUs. The ChaCha20-Poly1305 AEAD scheme is a good alternative where such instructions are not available, e.g., on mobile devices.

Figure 9 shows the number of connections negotiated with AEAD and Figure 10 shows the percentage of connections that advertise such ciphers. We see a sharp up-tick in the use of AEAD from late 2013 onwards. This is time at which significant client support for TLS 1.2 became available. Table 6 provides a timeline of protocol support for the major browsers. While many clients now offer ChaCha20-Poly1305, it is clear that the vast majority of AEAD use today is of AES-GCM, with 128-bit keys dominating 256-bit keys. We saw ChaCha20-Poly1305 being used in 1.7% of connections in March 2018, and we see little AES-CCM traffic—it was offered in 0.3% connections across our dataset.

6.3.3 Elliptic Curve Cryptography. As shown in Figure 8, connections using Elliptic Curve Cryptography are becoming more and more common. During the duration of our measurement, the top 5 used curves are secp256r1 (84.4%), secp384r1 (8.6%), curve 25519 (6.7%), sect571r1 (0.2%), and secp521r1 (0.1%). Curve 25519 is seen as being independent of NSA influence and has become more popular over time, especially since mid-2017. It is currently the second most used curve and was used in 22.2% of connections in February 2018.

6.4 TLS 1.3

The standardization process for TLS 1.3 started in 2014 [39] and reached its final stages in the IETF process. The latest, and possibly final, draft is draft 28 [72].

TLS 1.3 represents the most radical change to the TLS ecosystem since the switch from SSL 2 to SSL 3. It touches all parts of the protocol. The TLS 1.3 handshake is quite distinct from earlier releases, and much more of it is encrypted (including certificates). TLS 1.3 significantly reduces the number of cipher suites from the several hundred permitted in earlier releases to just 5. In particular, CBC-mode and RC4 cipher suites from earlier TLS versions are no longer permitted.

TLS 1.3 is of special interest to us since it shows how the ecosystem has changed over the past years. While it took TLS 1.2 six years after standardization to be used in more than 50% of connections, we already see significant uptake of TLS 1.3 today, before the RFC is completely ratified.

In April 2018, in 23.6% of the observed connections, the clients indicated that they support TLS 1.3. This is a marked increase from earlier months (9.8% in March and 0.5% in February). This is probably caused by TLS 1.3 being enabled by new versions of Chrome and Firefox for a subset of users [21]. In fact, TLS 1.3 support has been gradually rolled out by Firefox and Chrome. TLS 1.3 was added to Firefox 52.0 (released in March 2017) but it was disabled by default [64]. It was not until version 60.0, recently released in May 2018, when Firefox uses TLS 1.3 by default [65]. In the case of Chrome, TLS 1.3 was temporally enabled by default in 2017 after the release of Chrome 56 but it was removed [25].

The number of negotiated sessions is still much lower—only 1.3% of observed connections successfully negotiated it in April 2018. This is not surprising since supporting TLS 1.3 at the moment usually requires compiling new versions of libraries & custom setup procedures.

The TLS 1.3 version negotiation mechanism is different from earlier versions: the Client Hello still lists 1.2 as the offered version; however, there is a new extension that contains a list of all TLS

versions that the client actually supports. This change to the mechanism is to make TLS 1.3 easier to deploy in the face of middleboxes that block connections offering 1.3 as the version number. Since each draft of the TLS 1.3 specification has a unique version number this gives us some insight into the deployment of TLS 1.3 over time. Curiously the version that we saw most commonly advertised by clients in the extension has not been officially assigned—0x7e02 in 82.3% of connections with the extension. This is one of the several experimental Google TLS 1.3 variants. The most commonly advertised "official" version of TLS 1.3 that we encountered was draft 18 (13.4%).

7 DISCUSSION

This section discusses the evolution of TLS and its security based on our observations.

7.1 Ecosystem Improvements

Our measurements show that the security and the deployment of TLS have both changed radically in the last few years. In 2012, more than 90% of connections were using TLS 1.0. Nearly all connections used either RC4 or CBC-mode cipher suites. More than 60% of connections used non-forward-secret ciphers. At that point in time, TLS 1.2 had already been standardized for nearly 4 years (August 2008). AEAD cipher suites for TLS similarly were standardized in 2008. Diffie-Hellman (DH) forward secret cipher suites were part of TLS 1.0 (standardized in 1999); in 2006 support for ECDH and ECDHE was standardized [19]. So more secure options existed; they simply were not being used.

Today more than 90% of connections use TLS 1.2 with a forward secret, AEAD cipher suite. Furthermore we already see significant support for TLS 1.3 even though the standardization process has not yet concluded. Where in the past there was a gap of years between standardization of a technique and its deployment, today this time gap can be practically zero. This points towards the fact that encrypted communication is seen as a much more important topic today than it was even as recently as 2012; indeed more than half of all HTTP traffic is now protected using TLS. This is due in no small part to the influence of a few large vendors (such as CloudFlare, Facebook, and Google) becoming more actively involved in standardization bodies like IETF and vendor forums like the CA/Browser forum. These organizations have also pushed at first experimental and then production deployment of new versions of TLS, new cipher suites, and new protocol features.

7.2 Backwards Compatibility

Despite these positive advances, as our data shows, there are still many long tail effects in TLS deployment.

In particular, support for older, sometimes insecure, ciphers still is common: for example more than 20% of servers still support SSL 3 which was superseded by the standardization of TLS 1.0 in 1999. 3DES is still offered by clients in 69% of connections as a "cipher of last resort". Nearly 40% of application fingerprints that we see per day still support RC4.

This is a problem in several ways: first, downgrade attacks have surfaced time and again. Thus even supporting old cipher suites might open users up to attack—even if they have taken precautions to prevent downgrades. Second, supporting old features leaves room for misconfigurations. As mentioned in §5.3 we found servers that prefer old ciphers even when they support newer ones. Just offering an older cipher as a client might result in the server choosing it over more secure choices, perhaps as a result of poor server-side configuration.

There are multiple reasons for supporting old cryptographic primitives. Browsers are hesitant to remove support for old cipher suites fearing that they might break compatibility with websites. If a web browser shows warning messages or refuses to connect to a site where a different web browser still works, this might encourage users to switch browsers [3]. Additionally, today a significant number of smartphones, embedded systems, and IoT devices (e.g., printers and even smart light bulbs) support TLS [69]. While this represents an important step towards better security on the part of device vendors, many do not then provide security updates to their devices. This results in devices being used, sometimes for years, with abandoned and outdated software that only supports older and weaker cipher suites [71].

On the other hand, servers are hesitant to remove support for old cipher suites for the reverse reasons: old clients. For example, there are still millions of devices running Android 2.3 (Gingerbread) (Google claims 0.3% of devices connecting to the Play Store [50]; there are more than 2 billion Android devices in total [75]). Android 2.3 only supports TLS 1.0, and supports neither ECDHE nor AEAD cipher suites [49].

Solving this dichotomy is difficult. The solution that Firefox employed for RC4 has merit: removing insecure cipher suites by default but either allowing them as a fallback or (even better) just for whitelisted servers. Compliance with industry standards has also been a driver for change here. One can particularly highlight PCI-DSS, which is applicable to any enterprise that processes credit card payments. While the PCI DSS requirements with regards to SSL and TLS have been slow to harden, they did ban SSL in June 2016 and will require the use of TLS 1.1 (and preferably TLS 1.2) from June 2018 onwards [66].

7.3 Misconfigurations, Poor Implementations

While we mentioned legitimate reasons to keep supporting older cipher suites and protocols in some cases, it should be noted often there are no legitimate reasons to do so. Due to the complexities involved and the radical speed of change in the last years it is inevitable that developers and administrators are not always well-versed on the state of the art. Web servers that choose outdated cipher suites despite supporting much stronger ones or client software that offers NULL ciphers along with non-NULL ciphers are strong indicators of software developers not having caught up with proper TLS usage. In addition, some suggested attack mitigations may have also generated confusion for users—for example, one fix for the BEAST attack in 2011 was to switch from CBC-mode to RC4, but this was then followed in 2013 by attacks on RC4, which then may have prompted users to upgrade to TLS 1.1 and switch back to CBC-mode, and ultimately to make the harder leap to TLS 1.2.

Our data also contains a small number of hosts that, against TLS specification, choose cipher suites that were not offered by the client. Among these are a number of hosts that choose GOST

cipher suites, a Soviet and Russian government block cipher, and anonymous NULL cipher suites. While none of the standard TLS clients finish the handshake in these cases, it shows that there are an alarming number of systems in that might be running custom TLS implementations with questionable security.

Improving the accessibility and usability of TLS libraries and other software packages that use TLS can help mitigate some of these issues: providing a strong general-purpose default configuration by TLS libraries and making TLS software configuration more stream-lined and accessible in general encourages people to use trusted and usable TLS libraries instead of implementing their own. Making backwards compatibility features opt-in can also greatly reduce the possibility of misconfiguration through use of default settings.

7.4 Impact of Security Research

Our work shows that the impact of security research on the TLS ecosystem is sometimes spectacular, but sometimes also quite slow. In a few cases—especially those that attracted a lot of media attention or had significant impact—the ecosystem changed very quickly. The best example is Heartbleed (§5.4). The beginning of the Snowden revelations also clearly correlates with a change to forward secret ciphers (§6.3.1).

On the other hand, while RC4 use started to drop soon after attacks were announced (§5.3), it still took several years for RC4 usage to reduce significantly. Only with increasingly strong attacks did browser vendors eventually opt for full disablement in mid-2015. It can be argued that the first RC4 attack in 2013 was rather easy to dismiss as infeasible in practice; only when a direction of travel was established did the ecosystem begin to shift. There were also persistent rumors concerning the capabilities of Nation State Adversaries with regards to breaking RC4, though it is unclear whether these were relevant for RC4 as used in TLS.

With the CBC attacks (§5.2), there was no clear change in traffic (though we may posit that the temporary uptick in TLS 1.1 traffic was due to BEAST). In fact, CBC traffic increased slightly over time post-BEAST and despite Lucky 13, until TLS 1.2 implementations became widespread. This is attributable to the lack of better available options at the time, the ability to patch against both BEAST and Lucky 13 attacks, and the fact that both attacks are actually difficult to mount in practice. In contrast, no single-sided patch was available for the RC4 attacks, making them much harder to address.

Server support for SSL 3 is still embarrassingly high (§5.1), despite the severity of POODLE and RC4 attacks which apply to it. This can be attributed to modern web browsers never proposing to use SSL 3, modern web servers supporting newer versions of TLS, and the removal of version fallback from modern web browsers. However, we have seen several times how maintaining support for legacy features weakens the whole TLS ecosystem, especially with export cipher suites and FREAK/Logjam. It may only be a matter of time before an exploit is found which exploits widespread server-side support for SSL 3. For example, one could imagine a DROWN-style attack making use of some as yet undiscovered vulnerability in legacy code for RSA decryption in SSL 3 implementations to forge RSA signatures on TLS 1.3 handshakes, cf. [16, 59, 62].

7.5 Dataset Bias

The ICSI SSL notary contains SSL/TLS connections collected from North American universities and research institutions. Although this may introduce a potential geographical bias, given the size of the dataset we believe it is representative of many properties of real-world SSL/TLS activity.

Censys active scans are performed on the IPv4 address space without the use of the Server Name Indication (SNI) extension. We do not consider the lack of IPv6 scans to be an important limitation, since IPv6 adoption, although increasing over the years, is still limited. The lack of SNI does not introduce any bias in our study, since our analysis does not include server certificates.

8 RELATED WORK

A large body of work has examined different facets of the TLS and HTTPS ecosystem. A large amount of work has focused on the PKI [7, 11, 24, 46, 54], including revocation [86, 87]. Most recently, a number of PKI papers have focused on Certificate Transparency [23, 52, 76, 84]. More closely related to our work, a smaller set of work examined properties of specific parts of TLS. Durumeric et al. [43] examined email delivery security using active scans. Touching on a lot of different topics, the work also includes a snapshot of key exchanges/ciphers used for email delivery. Holz et al. [53] performed a more focused analysis of how TLS is used for electronic communication protocols (Mail, IMAP, IRC, XMPP). SSL Pulse [70] provides current statistics of the quality of SSL/TLS across the Alexa top pages. Similarly the ICSI SSL Notary page [58] gives up-to-date high level statistics about the dataset used in this study.

TLS handshake fingerprinting has been proposed and used by different parties over the years. Ristić [73] proposed to use an Apache module to create cipher suite lists. Majkowski implemented fingerprinting for p0f [61]. Most closely related to our work, HusÃąk et al. [55] did a study in which they linked HTTP user-agent to TLS cipher suite lists. They obtained 12,832 user-agent/cipher-suite links. Using this list they performed a 7-day measurement at the uplink of Masaryk University. They determined that the top 31 cipher suites were enough to cover 90% of traffic. Anderson et al. [13] examined how malware uses TLS and how their TLS handshakes differ from other applications. Recently Durumeric et al. [45] studied the prevalence of HTTPS interception using TLS fingerprinting.

For a discussion of TLS attack papers, please see §2.

9 CONCLUSIONS

Our longitudinal study shows that the TLS ecosystem has changed radically in the last six years, largely for the better: weak algorithms like DES and RC4 are going the way of the dodo, to be replaced by stronger AEAD designs; export, anonymous and NULL cipher suites are in retreat; forward secrecy is increasingly *de rigeur*; new elliptic curves are on the uptake; and TLS 1.3 is showing signs of rapid adoption. There is, and will probably continue to be, a long tail of software and devices that advertise and select legacy cipher suites; we have extensively discussed the reasons for this messy state of affairs. We have examined, through the lens of attacks, how and why these changes have come about, using datasets gathered from large-scale active and passive scans in concert with the largest database of TLS client fingerprints assembled to date.

We conclude by noting that the datasets we used reveal many other fascinating insights which lack of space precluded us from discussing here. As examples, we are able to track the response to the TLS renegotiation attack through the deployment of the RIE extension; we can see the very limited take up of the "Encrypt-then-MAC" extension as a response to the Lucky 13 attack; and we can see the rapid evolution of TLS 1.3 as it begins to be deployed experimentally by vendors. As the ecosystem continues to develop, we will continue to track it in all its fascinating and complex glory.

ACKNOWLEDGMENTS

We thank the Censys team for their support throughout writing this paper. We also thank Matías Spatz for his help on collecting TLS fingerprints for Windows. At last, we thank Dave Levin and the anonymous reviewers for their insightful comments and feedback.

This work was supported by the US National Science Foundation under grants CNS-1528156, CNS-1564329, and OAC-1348077. This work was also supported by the UK EPSRC under grants EP/M013472/1, EP/K035584/1 and EP/P009301/1. This research was partially supported by the Regional Government of Madrid through the N-GREENS Software-CM project S2013/ICE-2731, by the Spanish Government through the DEDETIS grant TIN2015-7013-R, and by the European Union through the ElasTest project ICT-10-2016-731535. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors or originators, and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Bro network monitoring system. https://www.bro.org/.
[2] Browserstack. https://www.browserstack.com.
[3] Bugzilla - Allow RC4 only for whitelisted hosts. https://bugzilla.mozilla.org/show_bug.cgi?id=1124039#c2.
[4] zgrab: A banner grabber, in go. https://github.com/zmap/zgrab.
[5] Mozilla Security Blog - Deprecating the RC4 cipher. https://blog.mozilla.org/security/2015/09/11/deprecating-the-rc4-cipher/, 2015.
[6] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.
[7] D. Akhawe, J. Amann, M. Vallentin, and R. Sommer. Here's My Cert, So Trust Me, Maybe?: Understanding TLS Errors on the Web. In *Proc. of the International Web Conference (WWW)*, 2013.
[8] M. R. Albrecht and K. G. Paterson. Lucky Microseconds: A Timing Attack on Amazon's s2n Implementation of TLS. In *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2016.
[9] N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt. On the security of RC4 in TLS. In *Proc. USENIX Security Symposium*, 2013.
[10] N. J. AlFardan and K. G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *Proc. IEEE Symposium on Security and Privacy (S&P)*, May 2013.
[11] J. Amann, R. Sommer, M. Vallentin, and S. Hall. No Attack Necessary: The Surprising Dynamics of SSL Trust Relationships. In *Proc. Annual Computer Security Applications Conference*, 2013.
[12] J. Amann, M. Vallentin, S. Hall, and R. Sommer. Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. Technical Report TR-12-014, ICSI, Nov. 2012.
[13] B. Anderson, S. Paul, and D. McGrew. Deciphering malware's use of tls (without decryption). *Journal of Computer Virology and Hacking Techniques*, Aug 2017.
[14] G. I. Apecechea, M. S. Inci, T. Eisenbarth, and B. Sunar. Lucky 13 strikes back. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.
[15] T. Arcueri. Imperfect Forward Secrecy: The Coming Cryptocalypse, July 2016. https://tonyarcieri.com/imperfect-forward-secrecy-the-coming-cryptocalypse.
[16] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, E. Käsper, S. Cohney, S. Engels, C. Paar, and Y. Shavitt. DROWN: Breaking TLS Using SSLv2. In *Proc. USENIX Security Symposium*, 2016.
[17] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P. Y. Strub, and J. K. Zinzindohoue. A Messy State of the Union: Taming the Composite State Machines of TLS. In *Proc. IEEE Symposium on Security and Privacy (S&P)*, May 2015.
[18] K. Bhargavan and G. Leurent. On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.
[19] Blake-Wilson, S. and Bolyard, N. and Gupta, V. and Hawk, C. and Moeller, B. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS), 2006. RFC 4492.
[20] R. Bricout, S. Murphy, K. G. Paterson, and T. van der Merwe. Analysing and exploiting the Mantin biases in RC4. *Des. Codes Cryptography*, 86(4):743–770, 2018.
[21] M. Brinkmann. Mozilla starts to enable TLS 1.3 on Firefox Stable, Apr. 2018. https://www.ghacks.net/2018/04/13/mozilla-starts-to-enable-tls-1-3-on-firefox-stable/.
[22] L. Brotherston. TLS fingerprinting. http://www.virustotal.com://github.com/LeeBrotherston/tls-fingerprinting.
[23] L. Chuat, P. Szalachowski, A. Perrig, B. Laurie, and E. Messeri. Efficient Gossip Protocols for Verifying the Consistency of Certificate Logs. In *2015 IEEE Conference on Communications and Network Security (CNS)*, 2015.
[24] J. Clark and P. van Oorschot. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In *Proc. IEEE Symposium on Security and Privacy (S&P)*, 2013.
[25] B. Coat. ProxySG, ASG and WSS will interrupt SSL connections when clients using TLS 1.3 access sites also using TLS 1.3. http://bluecoat.force.com/knowledgebase/articles/Technical_Alert/000032878, 2017.
[26] CVE-2011-3389. https://nvd.nist.gov/vuln/detail/CVE-2011-3389, 2011.
[27] CVE-2013-2566. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-2566, 2013.
[28] CVE-2012-4929. https://nvd.nist.gov/vuln/detail/CVE-2012-4929, 2012.
[29] CVE-2013-0169. https://nvd.nist.gov/vuln/detail/CVE-2013-0169, 2013.
[30] CVE-2014-0160. https://nvd.nist.gov/vuln/detail/CVE-2014-0160, 2014.
[31] CVE-2014-3566. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566, 2014.
[32] CVE-2015-0204. https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2015-0204, 2015.
[33] CVE-2015-2808. https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-CVE-2015-2808, 2015.
[34] CVE-2015-4000. https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-CVE-2015-4000, 2015.
[35] CVE-2015-7575. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7575, 2015.
[36] CVE-2016-0800. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-0800, 2016.
[37] CVE-2016-2183. https://nvd.nist.gov/vuln/detail/CVE-2016-2183, 2016.
[38] D. Benjamin. Applying GREASE to TLS Extensibility. IETF Draft, 2016.
[39] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, bis 00 (pre-draft), Apr. 2014. https://tools.ietf.org/html/draft-ietf-tls-rfc5246-bis-00.
[40] Dierks, T. and Rescola, R. The Transport Layer Security (TLS) Protocol Version 1.2, 2008. RFC 5246.
[41] T. Duong and J. Rizzo. Here come the ⊕ ninjas. *Unpublished manuscript*, 2011.
[42] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. Halderman. A Search Engine Backed by Internet-Wide Scanning. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.
[43] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. Halderman. Neither Snow Nor Rain Nor MITM...: An Empirical Analysis of Email Delivery Security. In *Proc. ACM Int. Measurement Conference (IMC)*, 2015.
[44] Z. Durumeric, J. Kasten, D. Adrian, J. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson. The matter of Heartbleed. In *Proc. ACM Int. Measurement Conference (IMC)*, 2014.
[45] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. Halderman, and V. Paxson. The Security Impact of HTTPS Interception. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2017.
[46] Z. Durumeric, E. Wustrow, and J. Halderman. Zmap: Fast internet-wide scanning and its security applications. In *Proc. USENIX Security Symposium*, volume 2013, 2013.
[47] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why Eve and Mallory love Android: An analysis of Android SSL (in) security. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2012.
[48] C. Garman, K. G. Paterson, and T. van der Merwe. Attacks only get better: Password recovery attacks against RC4 in TLS. In *Proc. USENIX Security Symposium*, 2015.

[49] Google. Android Developer Portal: SSLSocket. https://developer.android.com/reference/javax/net/ssl/SSLSocket.

[50] Google. Android Distribution dashboard. https://developer.android.com/about/dashboards/.

[51] M. Green. Attack of the week: FREAK (or factoring the NSA for fun and profit). https://blog.cryptographyengineering.com/2015/03/03/attack-of-week-freak-or-factoring-nsa/, 2017.

[52] J. Gustafsson, G. Overier, M. Arlitt, and N. Carlsson. A First Look at the CT Landscape: Certificate Transparency Logs in Practice. In *Proc. Passive and Active Measurement (PAM)*, 2017.

[53] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar. TLS in the wild: An Internet-wide analysis of TLS-based protocols for electronic communication. In *Proc. Network and Distributed System Security Symposium (NDSS)*, Feb. 2016.

[54] 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 *Proc. ACM Int. Measurement Conference (IMC)*, 2011.

[55] M. HusÃąk, M. CermÃąk, T. JirsÃŋk, and P. Celeda. Network-Based HTTPS Client Identification Using SSL/TLS Fingerprinting. In *Proc. International Conference on Availability, Reliability and Security*, Aug 2015.

[56] IANA. Transport Layer Security Parameters. https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml, 2017.

[57] IANA. Transport Layer Security (TLS) Extensions. https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xml, 2017.

[58] ICSI Certificate Notary. https://notary.icsi.berkeley.edu, 2017.

[59] T. Jager, J. Schwenk, and J. Somorovsky. On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1 v1.5 Encryption. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2015.

[60] D. Kaminsky, M. L. Patterson, and L. Sassaman. PKI layer cake: New collision attacks against the global X. 509 infrastructure. In *International Conference on Financial Cryptography and Data Security*. Springer, 2010.

[61] M. Majkowski. SSL fingerprinting for p0f. https://idea.popcount.org/2012-06-17-ssl-fingerprinting-for-p0f/, 2012.

[62] C. Meyer, J. Somorovsky, E. Weiss, J. Schwenk, S. Schinzel, and E. Tews. Revisiting SSL/TLS implementations: New bleichenbacher side channels and attacks. In *Proc. USENIX Security Symposium*, 2014.

[63] B. Möller, T. Duong, and K. Kotowicz. This POODLE bites: exploiting the SSL 3.0 fallback. *Security Advisory*, 2014.

[64] Mozilla. Firefox 52.0 - Release notes. https://www.mozilla.org/en-US/firefox/52.0/releasenotes/, 2017.

[65] Mozilla. Firefox 60.0 - Release notes. https://www.mozilla.org/en-US/firefox/60.0/releasenotes/, 2018.

[66] PCI Security. Are You Ready for 30 June 2018? Saying Goodbye to SSL/early TLS. https://blog.pcisecuritystandards.org/are-you-ready-for-30-june-2018-sayin-goodbye-to-ssl-early-tls, May 2017.

[67] Popov, A. Prohibiting RC4 Cipher Suites, 2015. RFC 7465.

[68] T. project. Tor. https://www.torproject.org/, 2018.

[69] Qualys. SSL Labs database of user agent capabilities. https://www.ssllabs.com/ssltest/clients.html, 2018.

[70] Qualys. SSL Pulse. https://www.ssllabs.com/ssl-pulse/, 2018.

[71] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill. Studying TLS Usage in Android Apps. In *Proc. ACM Int. Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2017.

[72] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, Draft 28, Mar. 2018. https://tools.ietf.org/html/draft-ietf-tls-tls13-28.

[73] I. Ristić. HTTP client fingerprinting using SSL handshake analysis. https://blog.ivanristic.com/2009/06/http-client-fingerprinting-using-ssl-handshake-analysis.html, 2009.

[74] I. Ristic. Is BEAST still a threat? https://blog.qualys.com/ssllabs/2013/09/10/is-beast-still-a-threat, 2013.

[75] J. Rossignol. Google Says There Are Now More Than 2 Billion Monthly Active Android Devices, May 2017. https://www.macrumors.com/2017/05/17/2-billion-active-android-devices/.

[76] M. D. Ryan. Enhanced Certificate Transparency and End-to-End Encrypted Mail. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

[77] Seggelmann, R. and Tuexen, M. and Williams, M. Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension, 2012. RFC 6520.

[78] Y. Sheffer, R. Holz, and P. Saint-Andre. Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS), May 2015. RFC 7525.

[79] J. Somorovsky. Systematic Fuzzing and Testing of TLS Libraries. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[80] Synopsys. The Heartbleed Bug. http://heartbleed.com/.

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

[82] L. Valenta, S. Cohney, A. Liao, J. Fried, S. Bodduluri, and N. Heninger. Factoring as a service. In J. Grossklags and B. Preneel, editors, *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, volume 9603 of *Lecture Notes in Computer Science*, pages 321–338. Springer, 2016.

[83] N. Vallina-Rodriguez, J. Amann, C. Kreibich, N. Weaver, and V. Paxson. A Tangled Mass: The Android Root Certificate Stores. In *Proc. ACM Int. Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2014.

[84] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. Halderman. Towards a complete view of the certificate ecosystem. In *Proc. ACM Int. Measurement Conference (IMC)*, 2016.

[85] M. Vanhoef and F. Piessens. All your biases belong to us: Breaking RC4 in WPA-TKIP and TLS. In *Proc. USENIX Security Symposium*, 2015.

[86] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When Private Keys Are Public: Results from the 2008 Debian OpenSSL Vulnerability. In *Proc. ACM Int. Measurement Conference (IMC)*, 2009.

[87] L. Zhang, D. Choffnes, D. Levin, T. Dumitras, A. Mislove, A. Schulman, and C. Wilson. Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. In *Proc. ACM Int. Measurement Conference (IMC)*, 2014.