Characterizing the HTTPS Trust Landscape: A Passive View from the Edge

Gustaf Ouvrier† Michel Laterman‡ Martin Arlitt‡ Niklas Carlsson†
† Linköping University, Sweden, firstname.lastname@liu.se
‡ University of Calgary, Canada, firstname.lastname@ucalgary.ca

Abstract

Our society increasingly relies on web-based services like online banking, shopping, and socializing. Many of these services heavily depend on secure end-to-end transactions to transfer personal, financial, and other sensitive information. At the core of ensuring secure transactions are the HTTPS protocol and the "trust" relationships between many involved parties, including users, browsers, servers, domain owners, and the third-party Certification Authorities (CAs) that issue certificates binding ownership of public keys with servers and domains. This article presents an overview of the current trust landscape and provides statistics to illustrate and quantify some of the risks facing typical users. Using measurement results obtained through passive monitoring of the HTTPS traffic between a campus network and the Internet, we provide concrete examples and characterize the certificate usage and trust relationships in this complex landscape. By comparing our observations against known vulnerabilities and problems, we highlight and discuss the actual security that typical Internet users (such as the people on campus) experience. Our measurements cover both mobile and stationary users, consider the involved trust relationships, and provide insights into how the HTTPS protocol is used and the weaknesses observed in practice. While the security properties vary significantly between sessions, out of the 232 million HTTPS sessions we observed, more than 25% had weak security properties.

I. Introduction

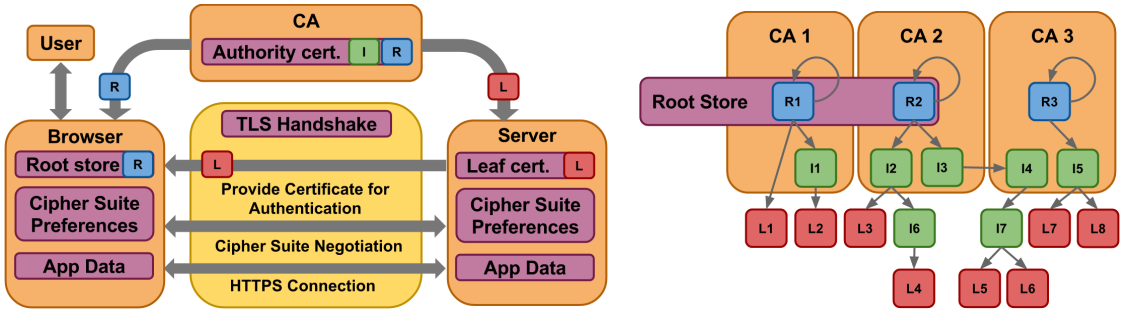
We are living in an information society in which organizations and individual users frequently must rely on the security and privacy offered by the HTTPS protocol. With HTTPS, any type of data that can be exchanged between a client and a server using regular Hypertext Transfer Protocol (HTTP) requests and responses are transferred over an end-to-end connection encrypted using Transport Layer Security (TLS) or its predecessor Secure Sockets Layer (SSL). With increased value of the information exchanged over the Internet, it is perhaps not surprising that HTTPS usage is increasing [1]. HTTPS can provide secure end-to-end transfers of money and other sensitive information, and is often used by authentication-based services such as online banking, shopping sites, and social networking services. With increased awareness of wiretapping and manipulation of network traffic, HTTPS has also become common among services that have not traditionally used secure end-to-end connections, including streaming services such as YouTube and Netflix that account for a majority of current Internet traffic volumes. This trend has been further augmented by free solutions from the Let's Encrypt project [1].

In addition to the TLS/SSL protocol suite, the security of HTTPS relies on a set of complex "trust" relationships. For example, consider a simple scenario (Figure 1(a)) in which a user accesses a single-server website using HTTPS. In this scenario, the user must trust his/her browser, the server that the browser will exchange information with, a certificate presented by the server to the client, the third-party Certification Authority (CA) who vouches that the certificate belongs to the corresponding server/domain, as well as the strength of the keys, hashes, and cryptographic algorithms (determined by a pairwise negotiated cipher suite) used for the connection establishment and data exchange.

Naturally, not all relationships are equally trustworthy. In particular, typical web sessions involve a diversity of servers, CAs, certificates, cryptographic algorithms, keys, and hashes. Measurements are therefore important to understand the practical vulnerabilities that a user is exposed to. In this article, we first untangle the trust relationships in this landscape and then present a data-driven characterization of the trust landscape and the security risks observed in practice for the different relationship types. All results and findings are discussed in the context of known vulnerabilities and previous measurement studies.

For this study, we passively collected and analyzed all HTTPS usage between a university campus network and the Internet for a week-long period, capturing more than 232 million HTTPS sessions. We develop a novel session-based labeling methodology that allows us to perform per-device and per-OS type analysis without requiring IP addresses to be recorded. The dataset includes traffic from both mobile and stationary users, and provides an overview of the actual security that typical Internet users (on campus) experience when browsing the web, as well as the trust relationships that are often invisible to the end user.

Our characterization shows that many per-session properties of mobile and stationary clients are almost identical, and highlights current weaknesses in each of the relationships in the landscape, including that clients often use older browser versions, that there is a skewed popularity among the CAs, and that there is a lack of adherence to best practices by all involved parties. For example, ciphers are frequently used by both clients and servers long after their weaknesses have become public knowledge, certificates often have long validation periods (providing more time to crack them), and CAs often issue certificates that allow their signing ability to be further delegated. Furthermore, both multi-domain and wildcard certificates (that allow many domains and sub-domains to share the same certificate, respectively) are very popular, suggesting that



(a) Relationships and involved parties (b) Certificate landscape example

Fig. 1. Example trust relationships in an HTTPS scenario.

website owners often prioritize convenience and economy over security. Overall, our per-session quality evaluation shows that a significant portion of the sessions have relatively weak security.

II. Background and Related Work

We begin with an overview of a functionally successful HTTPS session and describe the role of the involved parties in each trust relationship.

A. Certificate Authorities

At the center of the HTTPS landscape are the Certificate Authorities (CAs). The CAs are "trusted" third-party organizations that are responsible for issuing digital certificates and administering their validity. Before issuing a certificate, CAs are required to verify the identity of the website and make sure that the requester is the owner. When the CA issues a certificate it vouches for the identity of the website by digitally signing it using one of its authority root certificates. This creates a cryptographic binding between the website's identity and the public key contained inside the certificate, which can be validated using the CA's root certificate.

A web browser contains a root store, which is a selection of root certificates that it trusts. Furthermore, a browser generally trusts all certificates L that can be validated using a root certificate R from its root store. The reason for the root store is to relieve users, who often do not even know about certificates, from having to specify which CAs they trust themselves.

B. Transport Layer Security (TLS) Handshake

When establishing a new HTTPS connection, the client (i.e., the browser) and server must complete a TLS handshake in which they agree upon details for the session, including which TLS/SSL version to use and which cryptographic algorithms (i.e., cipher suite) to employ. During the handshake process (Figure 1(a)) the server presents an X.509 digital certificate (L), which is used to authenticate the server to the client. This certificate contains information about the identity of the server (e.g., a website domain name), a validity period, a public key, and is signed by a CA with root certificate (R). Given a certificate, the client checks that the identity matches the target server, that the certificate is in its validity period, and that the CA's digital signature is valid. If validation is successful, a session key is determined and an end-to-end encrypted communications channel is opened.

C. Certificate Landscape Example

Each CA controls a handful of authority certificates, called *root certificates*, which are validated by them directly. The root certificates are used to issue further certificates, which can be either another authority certificate, called an *intermediate certificate*, or an end-entity certificate, called a *leaf certificate*. Every new intermediate certificate can issue further intermediate certificates resulting in a chain of trust, referred to as a certificate chain. Figure 1(b) shows a simple certificate landscape example. Here, R1, R2 and R3 are root certificates to their respective CAs, while I1, I2, I3 and I5 are intermediate certificates directly signed by the root certificates. L1-L8 are leaf certificates for individual example websites, vouched for by the corresponding CAs. The case of I3 signing I4 is called a cross signing and can be useful when one CA resides in the root store and the other does not. For example, the Let's Encrypt project (https://letsencrypt.org/, last accessed: Mar. 2017) uses cross signing by IdenTrust (large CA) so that they can reach more users, long before they themselves are included in all root stores.

In our example only R1 and R2 reside in the root store, so a website with certificates L1-L6 will be considered trusted by the browser, while L7-L8 are untrusted. The use of intermediate certificates helps spread the workload of identity checking and signing procedures, as globally operating CAs can delegate such tasks to local intermediate CAs. Most importantly, using

easily removable/changeable intermediate certificates for online business signing purposes also allows the private keys of root certificates to be kept offline. However, since every intermediate certificate can be used to issue a valid certificate for any domain, each intermediate certificate comes with its own risks and attack surface [2].

D. Trust Relationship Breakdown

To untangle the trust relationships, consider our original single-server scenario (Figure 1(a)). First, the user must trust the browser, its implementation of HTTPS, and that the browser is up-to-date against the latest known security vulnerabilities. Second, the browser (and implicitly the user) needs to trust all CAs in the browser's root store. If a single trusted CA is compromised and starts generating certificates for non-trusted servers, this significantly compromises the security that a browser provides. With many available CAs, each with their own strengths and weaknesses, there are significant differences in which CAs distinct browsers select to trust. Disparate web services may also select different CAs for their certificates.

Third, the browser needs to trust the server that it communicates with. This trust is often build around X.509 certificates signed by the CAs in the root store or by other trusted entities to which the CAs have delegated part of this responsibility. These chained trust relationships are further complicated by (chained) certificates often being valid for different time periods and being difficult to invalidate when trust relationships are broken.

Finally, the browser must trust the cipher suite negotiated between the browser and server during the TLS handshake, where the cipher suites determine a combination of a key exchange, encryption, and Message Authentication Code (MAC) algorithm. Ciphers have different cryptographic strengths, and many ciphers in use today have known vulnerabilities and can therefore pose a significant risk to the confidentiality of the information transferred between the two parties.

E. Related Work

Both active and passive measurements have been used to analyze particular aspects that go into a secure HTTPS connection.

Many of these works have examined attacks targeting particular aspects of the connection establishment, including the key exchange [3], targeted ciphers [4] and MACs [5], to compromise the security of the HTTPS connections. This article characterizes and discusses the security experienced by regular users within the context of some of these attacks.

We also use passive measurements by Holz et al. [6] (published in 2011) and by Durumeric et al. [7] (published in 2013), together with our own measurements, as reference points for a longitudinal discussion. In addition to complementing these studies with more recent data points and a tutorial-style overview of the landscape, we also present complementary new analyses based on our novel session-based labeling, for example, which allows us to compare and contrast the heterogeneous security offered to both mobile and stationary devices. The remaining references are used to support claims.

III. Methodology and data collection

Our dataset was collected by passively monitoring the Internet traffic to/from the University of Calgary, Canada, at the university's multi-Gbps ingress/egress link. We used the Bro (https://www.bro.org/, last accessed: Mar. 2017) network security monitor to log specific information about the non-encrypted part of the TLS/SSL handshake, including all digital certificates sent.

We filter sessions based on IP prefix and focus on the traffic with servers located outside the campus. To distinguish between mobile and stationary users, we map the user-agent strings observed in HTTP sessions to IP addresses for five-minute rolling windows. Assuming that HTTP and HTTPS sessions from the same IP address within a window use the same user-agent, this allows us to create a mapping between user agents (seen in the HTTP data) and HTTPS sessions. After making the mapping, the IP addresses are removed. The methodology leverages the fact that typical web sessions, even when only visiting a single website, involve requests to many servers; some accessed with HTTP and some with HTTPS. By extracting OS and browser related information from the user-agent, this novel methodology allows us to perform per-device and per-OS type analysis, while preserving user privacy. Due to limited observed differences in certificate usage, in this article, we primarily focus on the distinction of mobile and stationary users.

Our dataset was collected during a week-long period (Oct. 11-17, 2015). In total, we observed 232,640,189 sessions using TLS/SSL, 67,664 unique certificates, and 552,387,188 certificate exchanges. Of the sessions, 67.6% contained (one or more) certificates, while the rest were session resumptions. Using known user-agent strings, we identified 46,913,633 mobile HTTPS sessions (53.5% iPhone/iPad/iPod and 45.6% Android) and 109,549,848 sessions from stationary clients. Both subsets have similar resumption ratios (29.9% and 32.1%, respectively). Of the 67,664 unique certificates, 750 were authority certificates, and the remaining 98.9% were leaf certificates.

IV. Trust Relationship Analysis

Older browser versions: The browser plays a key role in the HTTPS landscape. Despite the popular browsers using automatic updates and/or frequent reminders to upgrade to the latest versions, many clients still use outdated browsers. To illustrate this, Table I shows the fraction of sessions associated with different browser versions. Here, the latest officially released

TABLE I
BROWSER USAGE AND VERSION DISTRIBUTION (OCT. 11-17, 2015).

Browser	Browser's share of all sessions	Breakdown of browser version usage			
		Up-to-date	One behind	Two behind	Older
Chrome	51.48%	22.40%	64.68%	1.82%	11.12%
Safari	22.55%	0.62%	41.52%	10.36%	47.50%
Firefox	18.98%	0.00%	0.00%	66.90%	33.10%
Internet Explorer	6.72%	2.20%	45.76%	13.48%	38.56%

stable version is considered "up-to-date" and versions that do not have the latest security update are considered behind. We do not take into account Beta or developer versions. With the exception of Chrome, where 64.7% of the sessions are (only) a single version behind, the majority of sessions are using browsers that are at least two versions behind. These results highlight a significant delay in the rollout of new security updates.

Skewed usage towards a few CAs: With a few exceptions, any organization with control of an authority certificate can issue certificates for any domain. If a single authority certificate is compromised the whole system becomes vulnerable [8]. Although our dataset includes 750 authority certificates, we find that the vast majority of non-self-signed leaf certificates are issued by only a handful of organizations. The top-five organizations signing leaf certificates are Comodo CA Limited (with 22.9% of the sessions), Go Daddy (18.1%), GeoTrust (16.3%), DigiCert (9.4%), GlobalSign (6.8%). Part of this skew is due to rich-get-richer effects as buyers often select popular CAs as these may be less likely to be removed from root stores [8].

To meet increasing market demands many identity checks have become less stringent over time. Extended Validation (EV) certificates were introduced to help restore the resulting waning user trust in certificates. EV certificates are intended to follow stricter issuing criteria needed by organizations where secure communication is essential. Interestingly, when considering EV certificates, the skew is both higher and towards different CAs than for regular certificates, with Symantec Corporation (56.2%) and DigiCert (27.6%) making up the majority of the observed EV sessions (and 37.9% of the certificates). While the top issuers are somewhat different than for leaf certificates in general, we note that EVs only are observed in 4.94% of the leaf certificates and 6.27% of all sessions.

Some trust in Symantec may be inherited through the acquisition of Verisign's authentication business unit in 2010. Unfortunately, even the most highly used (and trusted) CAs can be compromised or make mistakes that degrade the overall security. For example, recently it was discovered that Symantec had issued test certificates for 76 domains they did not own (including for Google domains) and another 2,458 unregistered domains.[1] Google has since demanded that Symantec logs its certificates in publicly auditable Certificate Transparency (CT) logs [9].

Weak cryptography in certificates: Despite being susceptible to known attacks and CAs no longer signing new certificates with SHA1, SHA1 (with RSA encryption) is the second most used signature algorithm in our dataset. SHA1 is responsible for signing 24.9% of the leaf certificates and 50.3% of the authority certificates. The recommended SHA256 (with RSA encryption) signing algorithm, has replaced SHA1 more so far for leaf certificates (72.3% SHA256) than authority certificates (42.8% SHA256). Although we observed improvements compared to the 98.7% share of SHA1 that Durumeric et al. [7] observed in 2013, there is still a long way to go. While decisions by Mozilla, Microsoft, and Google, for example, to phaseout SHA1 (e.g., not showing a padlock symbol or to various degrees blocking SHA1 usage) may speed up this process, there have been setbacks in the outphasing as some of the browser companies have softened their decisions, including Mozilla re-enabling support for SHA1 in Firefox.[2] Service providers such as Facebook and Twitter have also suggested a delay in the phaseout of SHA1, due to concern that millions of users with older devices would lose access to their services. In addition, we observed 10 authority certificates (1.33%) and 97 leaf certificates (0.14%) still using MD5, almost seven years after Sotirov et al. [5] successfully created a rogue CA certificate. Also, we observed a non-negligible number of authority (1.33%) and leaf (5.61%) certificates using weak 1,024-bit RSA keys, which NIST recommended to stop using in 2013 [10].

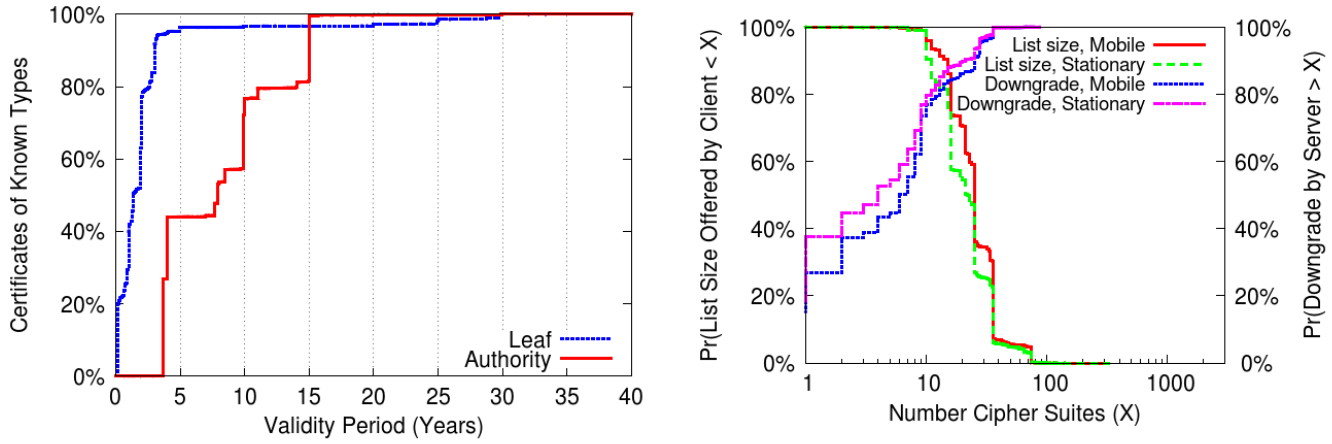
When inspecting EV certificates further, we did not find any certificates using weak keys (length less than 2,048-bit), but did not discover any significantly stronger keys either. Most EV certificates were signed using SHA256 (84.6%) or SHA1 (15.2%), with SHA1 being observed in 25.3% of the sessions.

Lack of path-length constraints: The maximimum path length of a certificate is decremented for each non-self-issued certificate in the path. It limits the length of a potential certificate chain and the trust delegation that is possible. Despite providing an extra measure of protection from misuse and helping to mitigate mistakes like issuing authority certificates instead of leaf certificates, we observed that 26.7% of all authority certificates did not specify any path length constraints.

Wildcard and multi-domain certificates: Wildcard certificates (valid for all subdomains; e.g., **.domain.com*) were used in over 70% of sessions. While the wildcard feature is convenient for administrators, it also poses the risk of validating rogue or buggy hosts [11]. Since the private key for a certificate must be stored on each machine, the attack surface increases with

[1]Ryan Sleevi, Sustaining Digital Certificate Security (Google Security Blog). https://security.googleblog.com/2015/10/sustaining-digital-certificate-security.html Published: Oct. 28, 2015; Last accessed: Mar. 2017.

[2]SecurityWeek News, Mozilla Re-Enables Support for SHA-1 in Firefox, http://www.securityweek.com/mozilla-re-enables-support-sha-1-firefox. Published: Jan. 7, 2016; Last accessed: Mar. 2017.



(a) CDF of certificate validity periods

(b) Cipher suite list size and downgrades

Fig. 2. Selected statistics for certificates and cipher suite downgrades.

the number of machines and domains covered by a certificate. We also observed significant usage of the Subject Alternative Name (SAN) extension, which allows multiple domain names to be specified for a certificate. For example, 68% of the unique certificates have at least two domain names, 20% of all certificate observations are for the 9% of certificates with at least 10 domain names, and 134 certificates (0.2%) had more than 100 subjects. Of these, the top-five belongs to GlobalSign (514 and 510 subjects), Google (two different certificates with 503 subjects), and Technische Universitaet Muenchen (429 subjects). In the top-10 we observed another four universities and another Google certificate.

Long validity period durations: Due to continuous advances in computer and cryptographic technologies, certificates valid for an extended time period can quickly become viable targets for attack. Using shorter validity periods is therefore a good practice. Figure 2(a) shows the cumulative distribution function (CDF) of the validity periods lengths of observed certificates. Typically, authority certificates have longer lifespans (e.g., 4, 10, and 15 years) than leaf certificates (e.g., 1, 2, or 3 years), but we also identified lifespans of up to 37 years. This is far beyond the predicted security lifespan for many certificates, according to NIST's current prediction that 112-bit security will be acceptable through 2030 [10].

We also validated the certificate chains during each non-resumption HTTPS session using the Mozilla root store. While most sessions were validated successfully (94.8%), a non-negligible share (4.2%) were not. About 1% of sessions contained self-signed certificates and in a few cases the certificate was outside its validity period (21,819 expired and 4,537 not yet valid).

Server downgrades of TLS/SSL version: The security of HTTPS heavily depends on which TLS/SSL version and cipher suite is used, where the cipher suite is a combination of a key exchange, encryption, and MAC algorithm. These details are negotiated during the TLS handshake. The browser first informs the server about the highest TLS/SSL protocol it supports and sends a list of supported cipher suites, ordered by preference. The server determines the final protocol and selects a cipher suite from this list. While the server typically chooses the version offered by the browser, we find that in 4% of the sessions the server downgraded to a lower version. Overall, the usage is almost completely divided between TLSv1.2 (80%) and TLSv1.0 (19%). Less than 0.1% of the sessions used SSL (v3 and v2).

Known weaknesses in the key exchange: The key exchange algorithm in the protocol is used to derive the shared session key. In 2015 alone, two attacks were discovered targeting the key exchange algorithm. The FREAK and Logjam attacks exploit bugs in the TLS/SSL implementation to downgrade sessions of servers that still support RSA-EXPORT and DHE-EXPORT grade ciphers, respectively [3]. Such downgrades allow an attacker to passively eavesdrop on the session. In total, we identified 428 instances of TLS_RSA_EXPORT and 27 instances of TLS_DHE_EXPORT being used. In general, however, we find that a majority of sessions use Elliptic Curve Diffie-Hellman (e.g., 62.11% use TLS_ECDHE_RSA and 20.32% use TLS_ECDHE_ECDSA) and 83.65% of all sessions have "perfect forward secrecy" (i.e., previously recorded sessions would not be compromised by the long-term keys being compromised in the future). While elliptic curves have long been recommended by security experts (and still are), it should be noted that ECDHE-based solutions in particular have recently come under scrutiny due to the influence that the National Security Agency (NSA) of the United States has in their design [3].

Weak encryption and MACs: While AES-128 and AES-256 are currently the most commonly used encryption algorithms, we discovered many cases where ciphers with weaker encryption are both offered and selected. For example, despite being prohibited from use in TLS [12] and viable attacks against RC4 being published in 2013 [4], the RC4 cipher is still offered (54.7%) and used (7.6%) in many sessions. On a positive note, the overall numbers are down from what Holz et al. [6] observed in 2011. Fewer concerns have been raised regarding MACs for data integrity and authenticity. This is consistent with the relatively shorter lifetime of session MACs (compared with certificate MACs) and the relatively strong (session)

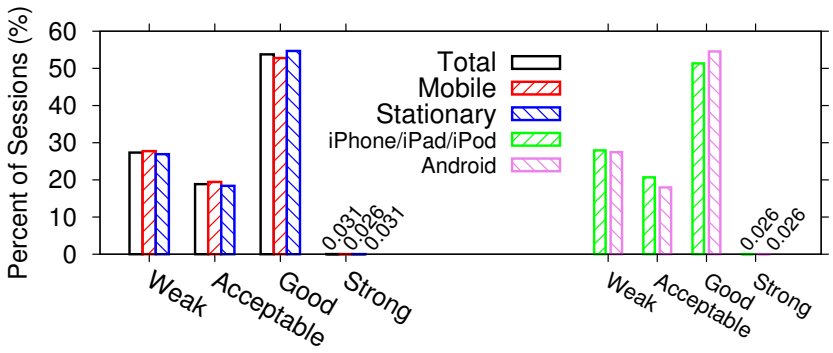


Fig. 3. Session quality evaluation based on four-level security classification.

MACs that we observed, including SHA256 (48.4%), SHA1 (31.0%), and SHA384 (14.8%). While MD5 (5.9%) is no longer acceptable in situations where collision resistance is required, such as for digital signatures, it is not urgent to stop using MD5 in HMAC-MD5 schemes [13]. The MD5 is almost exclusively used together with the RC4 cipher.

Downgrade of ciphers: Referring to the CDFs in Figure 2(b), in more than 99% of all sessions the clients offer at least nine cipher suites and often substantially more. As these lists often include weak ciphers, these results suggest that clients often prioritize compatibility (by giving servers many options) over security. Unfortunately, servers typically do not pick the clients' top candidate and sometimes perform substantial downgrades. For example, the most preferred option is only chosen in 36% of all sessions and in 20% of the cases an option outside the top-10 are selected.

To gain additional insights to the downgrades, we looked closer at in what position the RC4 cipher was when chosen by the server. In more than 80% of the cases it is outside the top-10 and in 60% of the cases it is outside the top-25, suggesting that the choice to use RC4 may be the result of poorly configured servers. We have also noticed cases where the servers appear to prioritize user experience over security, by not turning away visitors not supporting strong security options.

Mobile vs stationary users: In general, we have observed very small differences in per-session statistics and corresponding distribution when comparing mobile and stationary users. For example, consider the distributions shown in Figure 2(b). Also the cipher suites (offered and used) and the certificate chain length distributions are almost identical for mobile and non-mobile users, with both types typically seeing a chain length of two (44.8% and 45.2%) or three (47.4% and 47.8%). In both cases we observed chain lengths of up-to 21. This can perhaps partially be explained by the client classes (as an aggregate within the class) producing a highly correlated communication patterns. For example, a more detailed analysis of the visited websites shows that the relative popularities of the visited domains (as measured by their popularity ranks) are very similar for the mobile and non-mobile users on campus. Perhaps the largest difference between mobile and stationary sessions were observed for RC4 usage. Here, we observe a slightly higher usage of RC4 among mobile sessions (9.23%), compared to the stationary sessions (6.92%). However, in general our observations are consistent and very similar across the two classes.

V. Session Quality Evaluation

We conclude our analysis by evaluating the observed HTTPS session qualities. Figure 3 summarizes our results using a four-level classification. Here, a session is classified as *Acceptable* if it (i) uses protocol version TLSv1.0 or better, (ii) uses a NIST approved encryption cipher with at least 112-bit security strength, (iii) has a version-3 leaf certificate with a validity period of at most 25 months (2 years and a grace period), (iv) uses a signature algorithm SHA1 or better, and (v) uses public keys with at least 112-bit security strength. Sessions that do not satisfy these minimum requirements are classified as *Weak*. *Good* sessions further require that the certificate uses stronger signature algorithm than SHA1 and that either a key exchange algorithm with "perfect forward secrecy" or encryption with at least 128-bit security strength. Finally, for *Strong* sessions, we further restrict the protocol version to TLSv1.1 or better, the certificate validity period to at most 13 months, and the public key to use at least 128-bit security strength. This last class is included to illustrate roughly where we have observed the upper bound region of what is used in practice (hence the much smaller usage observed here).

During our measurements, a majority of the sessions were classified as *Good* (53.8%), an additional 18.9% as *Acceptable*, but a very insignificant portion as *Strong* (0.03%). We again did not find any significant differences between the mobile and stationary client devices, or when comparing Apple and Android devices within the mobile category. While the bulk of sessions has at least *Acceptable* quality, the many *Weak* (27.3%) sessions are concerning.

VI. Concluding Remarks

Using passive measurements we have characterized and discussed current shortcomings in the different trust relationships that affect the security of online users. We highlighted risks associated with the lack of adherence to best practices, including the slow outphasing of weak protocols and similarly slow adoption of new versions. For example, while modern browsers may be quick with security updates and patches, users typically use far from the latest versions. There is a significant skew in the organizations signing certificates, with differences between regular leaf certificates and EV certificates suggesting substantial differences in websites' trust in different CAs. Wildcard and multi-domain certificates are very popular, suggesting that websites often prioritize convenience or cost over security. Many certificates are valid for extended durations and do not limit path lengths to prevent authority certificates from further delegating signing ability. We have also seen that some clients offer broken ciphers (e.g., RC4) and servers sometimes choose them over better options. In general, we have observed limited differences between mobile and stationary clients, suggesting that our characterization is invariant to which of these two types of users is selected. Finally, our per-session quality evaluation showed that a significant portion (over 25%) have weak security quality. These results highlight that many browsers and servers prioritize user experience over security.

References

[1] M. Aertsen *et al.*, "No domain left behind: Is Let's Encrypt democratizing encryption?" *CoRR*, vol. arXiv:1612.03005v1 [cs.CR], Dec. 2016.
[2] B. Amann *et al.*, "No attack necessary: the surprising dynamics of SSL trust relationships," in *Proc. ACSAC*, Dec. 2013, pp. 179–188.
[3] D. Adrian *et al.*, "Imperfect forward secrecy: How Diffie-Hellman fails in practice," in *Proc. ACM CCS*, Oct. 2015, pp. 5–17.
[4] N. AlFardan *et al.*, "On the security of RC4 in TLS," in *Proc. USENIX Security*, Aug. 2013, pp. 305–320.
[5] A. Sotirov *et al.*, "MD5 considered harmful today," in *Proc. Annual Chaos Communication Congress*, Dec. 2008.
[6] R. Holz *et al.*, "The SSL landscape: a thorough analysis of the X.509 PKI using active and passive measurements," in *Proc. IMC*, Nov. 2011, pp. 427–444.
[7] Z. Durumeric *et al.*, "Analysis of the HTTPS certificate ecosystem," in *Proc. IMC*, Oct. 2013, pp. 291–304.
[8] H. Asghari *et al.*, "Security economics in the HTTPS value chain," in *Proc. WEIS*, June 2013.
[9] J. Gustafsson *et al.*, "A first look at the CT landscape: Certificate transparency logs in practice," in *Proc. PAM*, Mar. 2017.
[10] E. Barker *et al.*, "Recommendation for key management, part 1: General (revision 3)," *NIST Special Publication 800-57*, Jul. 2012.
[11] P. Saint-Andre, S. Santesson, and J. Hodges, "Representation and verification of domain-based application service identity within internet public key infrastructure using X.509 (PKIX) certificates in the context of transport layer security (TLS)," RFC6125, Mar. 2011.
[12] A. Popov, "Prohibiting RC4 cipher suites," RFC7465, Feb. 2015.
[13] S. Turner and L. Chen, "Updated security considerations for the MD5 message-digest and the HMAC-MD5 algorithms," RFC6151, Mar. 2011.

Gustaf Ouvrier is an M.Sc. student at Linköping University, Sweden. His research interests include security, Internet measurements, and traffic characterization.

Michel Laterman received his M.Sc. in computer science from the University of Calgary where he worked on network traffic monitoring. He currently works as a software developer at SAP.

Martin Arlitt is a principal research scientist at Hewlett Packard Enterprise, where he has worked since 1997. His general research interests are workload characterization and performance evaluation of distributed computer systems. His 80+ research papers have been cited over 9,600 times (according to Google Scholar). He has 31 granted patents and more pending. He is an ACM Distinguished Scientist, a senior member of the IEEE, and an adjunct assistant professor at the University of Calgary.

Niklas Carlsson is an Associate Professor at Linköping University, Sweden. He received the M.Sc. degree in engineering physics from Umeå University, Sweden, and the Ph.D. degree in computer science from the University of Saskatchewan, Saskatoon, SK, Canada. His research interests include design, modeling, characterization, and performance evaluation of distributed systems and networks.