Markus Sippo

TIME-BASED EXPIRATION PROBLEM OF SSL/TLS CERTIFICATES

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2021

ABSTRACT

Sippo, Markus
Time-based expiration problem of SSL/TLS certificates
Jyväskylä: Jyväskylän yliopisto, 2020, 65 pp.
Information Systems, Master's Thesis
Supervisor(s): Costin, Andrei

The amount of confidential data in web services is continuously rising, which sets requirements for data encryption during transmission and server authentication. The commonly adopted solution is Transport Layer Security (TLS), which solves both requirements presented above. Technically TLS relies on X.509 certificates to provide features. While X.509 certificates are well-understood topic, the implementation often confuses the domain administrators and errors during the configuration are common. On top of this, certificates have an expiration date, which means that the certificates need to be renewed from time to time. Often, the renewal is either forgotten or neglected by the administrators, which leads to connection issues. This study provides insight on expired certificates and their usage. In addition, this study provides insight on what type of services and businesses are impacted by expired certificates. Most common error cases in TLS implementations were also extracted from the data. The results of this paper indicate that certificate expiration is a common problem, that affects all types of online services, ranging from governmental online services to online shops.

Keywords: Transport Layer Security (TLS), X.509 certificate, expiration

TIIVISTELMÄ

Sippo, Markus
Time-based expiration problem of SSL/TLS certificates
Jyväskylä: University of Jyväskylä, 2020, 65 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja(t): Costin Andrei

Luottamuksellisen datan määrä verkkopalveluissa nousee jatkuvasti, joka asettaa vaatimukset datan salaukselle siirron aikana ja palvelimen tunnistamiselle. Vakiintunut ratkaisu edellämainittuihin vaatimuksiin on Transport Layer Security (TLS). Teknisesti TLS vaatii toimiakseen X.509 varmenteen. Vaikka X.509 varmenteet ovat hyvin ymmärrettyjä, niiden implementointi usein aiheuttaa hämmennystä pääkäyttäjille ja kehittäjille, jonka seurauksena virheet implementoinneissa ovat yleisiä. Tämän lisäksi varmenteilla on voimassaolo päivämäärä, joka tarkoittaa että varmenteet tulee uusia ajoittain. Varmenteiden uusiminen voi unohtua tai se jätetään tarkoituksella tekemättä, joka aiheuttaa usein yhteys ongelmia verkko-palveluihin. Tämä tutkimus tuottaa tietoa vanhentuneiden varmenteiden käytöstä ja niiden yleisyydestä. Tämän lisäksi, tämä tutkimus tuottaa tietoa liittyen palveluihin ja liiketoimintaan, jotka tyypillisimmin kärsivät vanhentuneista varmenteista. Lisäksi tutkimus luokittelee yleisimmät virhetilanteet TLS implementoinneissa. Tutkimuksen tulokset osoittavat, että varmenteiden vanhemeninen on yleinen ongelma, josta kärsivät kaiken tyyppiset verkko-palvelut aina valtioiden verkkosivuista verkkokauppoihin.

Asiasanat: Transport Layer Security (TLS), X.509 varmenne, vanhentunut varmenne

FIGURES

TABLES

TABLE OF CONTENTS

1 INTRODUCTION

Today's internet handles more sensitive information than ever before. This sensitivity of information has set new demands for cyber security in web services across the internet. As more sensitive information is uploaded and processed in a variety of web-services, cyber crime has endless possibilities to exploit this growth. Cyber crime has increased steadily during the twenty-first century.

The global COVID-19 pandemic has changed the way people interact with technology. Changes in working practices and social distancing have made people spend more time online (Lallie et al., 2021). The pandemic has also created unique social and economic circumstances, which have generated more opportunity for cyber crime, which indeed has increased during the pandemic (Lallie et al., 2021).

Online services, such as banking and communication, have reached a huge user base among the whole population. In Finland during the 2020 the usage of the internet grew by 7% in the two oldest age groups (ages 65–74 and 75–89) (Suomen virallinen tilasto (SVT), 2020). This increase also creates opportunity for cyber crime as elderly people have been long recognized to be the most vulnerable targets (Carlson, 2006). In younger age groups the internet has already gained almost 100% usage. The most used online service in 2020 in Finland was online banking. 87% of the population (aged 16-89) have used online banking tools in the last three months (Suomen virallinen tilasto (SVT), 2020).

These modern internet services would not be possible without proper encryption mechanisms in data transit and server authentication. Without server authentication, clients, such as browsers, speaking to web servers could not make sure that the server is the entity it claims to be. In other words, you couldn't be sure that your browser is indeed connecting to your bank's website, instead of a malicious site claiming to be your bank's website and stealing your money.

Without encryption in data transit all data, including usernames and passwords, would be sent in plain text. This is why SSL/TLS encryption is one of the most important tools that enable the services we use every day. Research

topics on the adoption and implementation of TLS are important as they provide insight into what works and what needs further development in the future.

SSL/TLS is probably the most utilized security protocol used for various purposes, such as securing web traffic, email, VPNs and other communications (Bhargavan, Fournet, Kohlweiss, Pironti & Strub, 2013). The usage of SSL/TLS has also seen significant growth after 2014. The mentality of the usage has shifted from securing only the necessary data to securing all traffic in all web services. In early 2014 less than 30% of all pages loaded by Firefox browsers were using HTTPS, whereas the same number in 2021 is between 82-90% depending on the location (Let's Encrypt, 2021).

Despite all the good SSL/TLS brings, it is not a perfect system by any means. The whole architecture has been criticized in academic literature and also successfully targeted in cyber attacks. One of the issues with current architecture is that there are many actors and components in the system with different functions. This creates opportunity for possible attackers and also makes management of the system difficult.

The main focus area of this study was to provide insight on certificate expiration and misconfigurations on the server side. Usage of expired or misconfigured certificates was identified during a scan by sending a request to the web server. The response was then analyzed and the results were saved for further analysis. Complete scan data contains roughly 8500 different domains, which were all pinged three times during the scan. Domains that represented errors were manually inspected to gain more understanding about the particular domain and the type of service it offered to its user base.

Over 84 domains were identified that failed to renew their certificates. Overall, these web services represented various categories, ranging from blogs, online shops, educational sites, government websites and many more. On top of this, many other configuration issues were found in the TLS implementations of these websites. The findings of this paper suggest that TLS implementations remain difficult to implement and both misconfigured and expired certificates remain an issue in the whole ecosystem.

1.1 Research questions

The objectives of this study were to gain insight about various TLS related issues and their rates of occurrence in websites. The research questions were the following.

- **RQ1**: What are the most frequent TLS errors encountered in implementations?
- **RQ2**: How frequently do websites fail to renew their certificates?
- **RQ3**: What kind of services are impacted by poor implementation and management of TLS?

- **RQ4**: How long certificates stay expired, and what are the possible reason behind that?

In addition, one research goal was to showcase that large scale monitoring of TLS implementation for web domains can be achieved. This approach presented in this paper could be further elaborated to provide even more insight and data about TLS implementations. This paper successfully answers the set research questions while leaving a lot of opportunity for further research.

2 OVERVIEW OF THE THEORETICAL BACKGROUND

This chapter describes the theoretical background for this thesis. Firstly, the common protocols are introduced, and their functions are explained. Most of the focus is on TLS protocol and its history and functionalities. Private Key Infrastructure (PKI) and X.509 certificates are introduced as they are key components in the whole certificate architecture. Important historical milestones are elaborated so that the reader can understand the background better.

2.1 HTTP

Hypertext Transfer Protocol (HTTP) is a protocol for transferring data between client and server. It originates from the early 1990's and the development of HTTP protocol provided the foundation for the modern internet. The design of the protocol evolves around hypertext documents, commonly known as HTML.

While the HTTP protocol is still widely used today there are certain security limitations that the protocol alone could not provide. Struggle comes when dealing with confidential data, such as personal information, credit card numbers, passwords, social security numbers, and the list goes on and on. The issue is that HTTP protocol transmits the data in plain text, meaning that anyone could read the delivered message content.

As the modern web has evolved, more and more of sensitive data is processed in the web applications. The need for end-to-end encryption has been identified a long time ago. Without proper encryption the internet could not be used as it is used today. Solution for HTTP to support end-to-end encryption needed to be found. The widespread solution known today is called Transport Layer Security (TLS).

2.2 SSL/TLS

Secure Socket Layer (SSL) was originally developed to provide security to web transactions over HTTP (Clark & Van Oorschot, 2013). SSL needed to solve two major security challenges for web traffic. Firstly, the client data being delivered needed to be encrypted and protected (Clark & Van Oorschot, 2013). Secondly, the client data should be provided only to the intended recipient server (Clark & Van Oorschot, 2013). In other words, the client and server needed to be authenticated. SSL/TLS also supports client authentication, however, its usage is not widely adopted. Thirdly, TLS provides integrity to transmitted data as checksums can be calculated to ensure that the message has not been altered. The fundamental design and security goals of SSL/TLS are discussed in more detail in the later sections of the paper.

Historically, SSL has three different versions, however, all versions of SSL have been deprecated and should not be used any longer as they do not provide adequate security. The latest version SSL 3.0 was deprecated in 2015 (Barnes et al., 2015). Even though these previously mentioned versions have been deprecated, overall TLS provides backwards compatibility (Bhargavan et al., 2013). This is to allow the internet to slowly adopt the newer versions of TLS as the protocol introduces new versions.

Transport Layer Security (TLS) is the successor for the SSL protocol. SSL 3.0 and TLS 1.0 are very similar with only a few differences (Holz, Amann, Mehani, Wachs & Kaafar, 2015). The most recent version of TLS is the version 1.3, which was introduced in 2018 (Rescorla & Dierks, 2018). Both TLS versions 1.2 and 1.3 are supported in today's internet. However, the usage of TLS 1.3 is encouraged since it provides better security overall. Later in this paper SSL/TLS are simply referenced as "TLS" as that is the modern protocol that should be advocated.

Overall, TLS has become a general-purpose protocol used in a variety of applications ranging from e-commerce, VPNs and everything in between (Krawczyk, Paterson & Wee, 2013). TLS is arguably the most used protocol for secure communications today and most likely will be for far in the future.

TLS has two different constituent protocols, the Handshake protocol, and the Record protocol (Krawczyk et al., 2013). The handshake protocol is used for key-establishment and authentication of the web server and the record protocol is used to create a secure channel for transmitted data (Krawczyk et al., 2013). TLS protocol is also flexible, it can be used with a variety of cipher suites (Meyer & Schwenk, 2013).

To conclude TLS is cryptographic protocol that ensures server authentication and data encryption during transmission. TLS has a prominent place in modern internet, as it enables services to deal with sensitive data securely over insecure networks. Next the concept of TLS handshake is introduced, which is used to create a TLS connection.

2.3 TLS Handshake

To establish TLS secured connections, there are certain steps that must be done. TLS handshake is one of these steps.

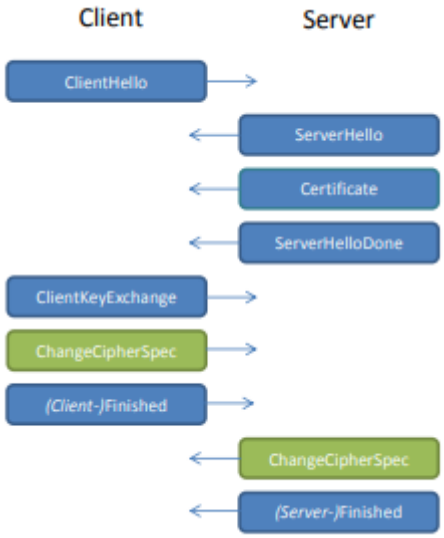

Figure 1 Typical SSL/TLS handshake with RSA key exchange (Meyer et al., 2014).

Figure 1 describes the typical TLS handshake scenario without mutual authentication between client and server (Meyer et al., 2014). The process begins with the client sending a message to the web server. ClientHello message contains all cipher suites that the client supports (Meyer et al., 2014). Cipher suite defines the algorithms that are used for key exchange and encryption (Meyer et al., 2014). There are various different suites to choose from. Some provide better security than others.

Then the web server receives the client message and chooses one cipher suite provided by the client (Meyer et al., 2014). After this the server then responds with a ServerHello message that contains the chosen cipher suite, server's certificate(s) and the server's public key.

The client then verifies the server's certificate chain. If the certificate chain verifies successfully then the process continues. Next, the client generates PreMasterSecret, which is sent in ClientKeyExchange message (Meyer et al., 2014). PreMasterSecret is a random string that is encrypted with the server's public key. The basic principle of asymmetric encryption is that messages encrypted with a public key can only be decrypted with the private key that the server holds and should not disclose to anyone under any circumstances.

Server receives the ClientKeyExchange message and uses its private key to extract the PreMasterSecret chosen by the client. Now both parties have the same PreMasterKey, which was transported in encrypted format and can be

opened only by the private key of the web server. Both parties can now use the same Key Derivation Function (KDF) to generate a master key. As a result, both parties have the same master key, which was transported over an insecure network (internet).

After sending a ClientKeyExchange message both the client and the server are ready to switch to an encrypted mode, which is done by sending a ChangeCipherSpec message (Meyer et al., 2014). Both finished messages (Figure 1) are sent in encrypted format and they contain checksums of all previously sent messages (Meyer et al., 2014). The checksums can be then verified to ensure the integrity of messages.

TLS connection does require a dedicated port (Holz et al., 2015). For instance, the dedicated and well-known port for HTTPS protocol is port number 443. As mentioned previously HTTPS is the main application of TLS used mostly by browsers. Consider the following example where Alice wants to visit Bob's website:

> Concretely, when Alice visits https://www.bob.com, her browser connects to www.bob.com over port 443 where the server responds with a certificate. Then, the browser validates that a trusted certification authority signed it, and checks that it pertains to www.bob.com and no other host. If these checks succeed, the browser uses the public key in the certificate to setup a secure channel with www.bob.com. (Akhawe et al., 2013.)

In the previous example there are multiple things happening behind the scenes. Bear in mind that if all goes well, the end-user (Alice) does not necessarily even know that the connection to Bob's website is secured. When Alice starts the browser and types in Bob's website address, the browser then uses HTTPS protocol to connect to the site. As presented above, HTTPS requires TLS. Then the TLS handshake begins where the server and the client follow the protocols to exchange information, such as server certificate(s) and they exchange keys. Once the handshake is complete Alice has secured connection to Bob's website.

2.4 STARTTLS

STARTTLS (sometimes referred as opportunistic TLS) is another way to utilize TLS between the client and the server. The process begins with regular TCP protocol, without any encryption involved (Holz et al., 2015). After this the client can request the server for capability to upgrade the connection to use TLS. If the server confirms that the secure connection can be used; the TLS handshake follows as usual (Holz et al., 2015). The TLS handshake process is exactly the same as in normal TLS connection (as presented above) (Holz et al., 2015). In

other words, STARTTLS upgrades an already established unencrypted connection between the client and the server to use encryption.

There are some advantages and disadvantages in STARTTLS over regular TLS. Major advantage is that STARTTLS does not require a dedicated port and the connection can be upgraded when necessary (Holz et al., 2015). This gives overall more freedom for the developer to choose where to use secure connections. However, this kind of approach feels a bit outdated in 2021, from a security perspective, as it would make more sense to encrypt every connection. Nevertheless, STARTTLS enables this approach where-ever necessary.

STARTTLS is known to be vulnerable for Man-in-the-Middle attacks (Holz et al., 2015). For example, an attacker could interfere and modify the messages sent to the client to make it appear that TLS connection is not available. In the worst case scenario, the exchange between the client and the attacker could continue in plain text. To remedy this situation, there needs to be clear lines when STARTTLS is needed and any connections without STARTTLS should not be accepted in those situations. However, this does imply that STARTTLS should not be used. Also, regular TLS has its own security limitations. These limitations are discussed later in this paper

2.5 HTTPS

Hypertext Transfer Protocol Secure (HTTPS) is an extension of HTTP protocol, which combines cryptographic protocol TLS to HTTP protocol (Clark & Van Oorschot, 2013). HTTPS is widely used in the modern internet to secure connections between client and the server. HTTPS provides three main features that the added security is based upon. These features are discussed next.

Firstly, HTTPS provides confidentiality (Felt et al., 2017). This means that a successful TLS connection encrypts transmitted data. The encrypted data can be only read by the web server and the client. Without HTTPS the transmitted data would be in plain text, meaning that it could be intercepted and read by anyone.

Secondly, HTTPS provides integrity (Felt et al., 2017). Again, a successful TLS connection encrypts the transmitted data and protects it from being altered by anyone. Any modifications to transmitted data can be detected by inspecting the message digests, however, the CipherSuite selected needs to support this. If the message digest does not match it means that the message has been altered.

Thirdly, HTTPS provides server authentication (Felt et al., 2017). Server authentication is an important feature that allows the client to verify that the web server is actually who the server claims to be. This is done with digital certificates that act as proof of identity. During the TLS connection the server's certificate is verified and the domain needs to match the subject name listed in the certificate. This validation process protects the user against Man-in-the-middle (MITM) attacks. The validation process of certificates is discussed in more detail in later sections of this paper.

The security features of HTTPS presented above are very important. To conclude, HTTPS encrypts data during transmission and forces the identification of web servers. Without identifying the web servers, clients would be vulnerable to MITM attacks, where a perpetrator acts as a web browser and intercepts messages and possibly alters the message content.

HTTPS is mostly used in web browsers, such Google Chrome or Firefox. HTTPS is also the most client-facing component of TLS (Akhawe et al., 2013). All internet users today are most likely using HTTPS despite whether they acknowledge this or not.

Technically HTTPS requires a dedicated port, and the server needs to support it for successful connection establishment. Usually, the port for HTTPS is port 443. Web servers need to present their digital certificates for successful TLS connections to take place.

While the HTTPS is already well established and has a surprisingly long history, its wider adoption has happened surprisingly recently. Figure 1 illustrates the adoption of HTTPS in web pages. The data is collected from Firefox and the statistics are provided by Let's Encrypt. Let's Encrypt is a non-profit Certificate Authority that provides TLS certificates for 260 million websites today. Let's Encrypt is introduced later in this article.

As we can see from the graph (Figure 2) the wide-spread adoption of HTTPS has mostly happened during the 2010s. Of course, the early adopters of HTTPS have started to utilize it way before 2014. However, today's global usage of HTTPS has grown to over 84% from 30% in 2014. Similar adoption rates have been identified in academic research as well (Felt et al., 2017). The adoption of HTTPS is of course a good thing overall. The internet is steadily growing towards 100% HTTPS adoption in browser usage. This will most likely happen during the 2020s.

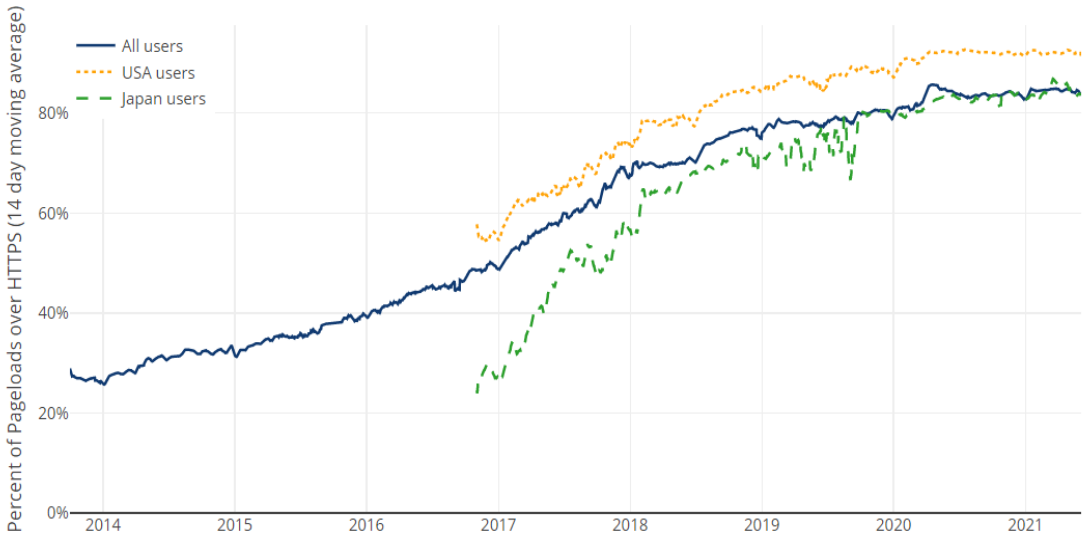

Figure 2 Usage of HTTPS in web pages in Firefox browser between 2014 and 2021 (Let's Encrypt, 2021).

Few things have collaborated to this steadily growing adoption of HTTPS. In 2015 a non-profit organization called "Let's Encrypt" was founded. Let's Encrypt has certainly speeded the adoption of HTTPS.

Let's Encrypt offers validated certificates for all domains free of charge (Aas et al., 2019). This ultimately removed the financial barrier of entry for HTTPS adaption. On top of this, the mentality of encryption has shifted. Before 2014 encryption was used where encryption was considered as a must. This mentality has largely changed after Edward Snowden disclosed the NSA classified documents which revealed widespread surveillance in late 2013. Today, most of the connections and transmitted data are in encrypted format.

2.6 Notable changes in TLS version 1.3

While the purpose and goals of TLS (and SSL) have remained the same between earlier and previous versions, there have been great deal of changes in the newest TLS 1.3 version that overall improve the performance and security of the protocol.

TLS 1.3 was motivated by the security concerns around TLS 1.2. There was a need to deprecate the unsecure algorithms, and also to provide better performance (Dowling, Fischlin, Günther & Stebila, 2021). To achieve above mentioned goals there has been a variety of changes. The development of TLS 1.3 was started in 2014 and it was launched in 2018.

Variety of unsecure and outdated cryptographic algorithms have been also deprecated e.g., SHA-1 and RC4 (Dowling et al., 2021). This change has been stricter than it first sounds, as TLS 1.3 dropped the support of hundreds of cipher suites down to five allowed (Kotzias et al., 2018). To reduce latency the TLS version 1.3 has fewer message flows during the handshake (Dowling et al., 2021). Fewer messages during the handshake improve performance.

Changes have been also made to the key usage and signing. For example, different keys are now used to encrypt the handshake and transmitted data (Dowling et al., 2021). In addition, the whole handshake transcript is signed to provide better authentication (Dowling et al., 2021). The changes above are just some honourable mentions, overall, many changes have been made to the newest version of TLS.

2.7 TLS usage outside HTTPS

In this section other protocols than HTTPS utilizing TLS are introduced. Keep in mind that this is not a comprehensive list, but rather a brief introduction to usage of TLS outside the prominent HTTPS protocol.

Second important application for TLS is email. Surely, both you and I are email users, which makes both of us ultimately rely on encryption provided by

the TLS protocol to secure our emails from others. Most likely the emails that we send rely on Simple Mail Transfer Protocol (SMTP), which is responsible for email delivery (Baumgäertner, Höchst, Leinweber & Freisleben, 2015).

SMTP does not automatically mean that TLS is used. TLS is used if the web server can support it, which it absolutely should do for privacy reasons. In the case that the web server supports TLS, the end-user connects, establishes TLS connection, and authenticates to his/her email provider and transmits the email to the provider (Baumgäertner et al, 2015). The rest of the delivery is handled by the service provider.

The SMTP usually uses the mail transmission port 587 and STARTTLS (Baumgäertner et al, 2015). Or some services might use port 465 and regular TLS. The email provider handles the message transmission to other providers until the correct one is reached. This server-to-server communication should utilize TLS as well (Baumgäertner et al, 2015). It is also important to understand that in email TLS is generally going to prioritize email delivery over security in situations where TLS connection causes issues (Holz et al., 2015).

When the end-user receives email it is usually handled by Post Office Protocol (POP) or Internet Message Access Protocol (IMAP).These protocols actually fetch the emails from the email server (Baumgäertner et al, 2015). These protocols wrap the whole connection using TLS (Baumgäertner et al, 2015). Dedicated ports for POP3S and IMAPS are ports 993 and 995 (Holz et al., 2015). To summarize, TLS is used to secure email in all phases of transmission, while sending email, server-to-server communication and fetching email from the server.

TLS can be utilized with many other protocols, however, the former examples of email and web browsing are probably the most common examples. These other protocols are for example (but not limited to), File Transfer Protocol (FTP) and Lightweight Directory Access Protocol (LDAP). Both of these protocols offer extensions that allow the utilization of TLS. There are also many other protocols that could be used as an example.

Anderson & McGrew (2019) have identified the usage of TLS in various application types, which includes browsers, email, storage, communication, security and more. They also argue that the prominent HTTPS protocol is slowly losing its leader position in TLS usage, as other applications are adopting TLS in increasing numbers (Anderson & McGrew, 2019). However, the finding does not undermine the importance of TLS in HTTPS.

2.8 Public key infrastructure

The public key infrastructure (PKI) is a prerequisite and foundation for security in large networks, distributed systems and for electronic commerce in general (Maurer, 1996). The importance of PKI architecture has been widely recognized over many decades ago. The importance of PKI has not diminished, in fact, one would argue the opposite. Encryption, privacy and sensitive information are

mundane in today's internet and the end-users except that online services can provide confidentiality in their services.

The goal of the PKI is to provide a trust between two parties who do not have an existing trust relationship with one another (Alrawais, Alhothaily, & Cheng, 2015). The communication often is transported over insecure networks and verification of the parties involved in the communication becomes extremely important (Alrawais et al., 2015). PKI is the solution for the issues presented above.

PKI has many different components and actors involved in the architecture, but in this paper the focus is on the process of how PKI works and solves the issues on data encryption and authentication between client and the server.

Maurer (1996) describes the PKI architecture as a distributed database of public key certificates and other information, such as revocation lists. PKI must provide mechanisms for entities to create and retrieve information from the PKI (Maurer, 1996). Information creation means adding new certificates to the PKI structure, and retrieving means a way to retrieve information, such as a certificate. In reality, there are other mechanisms and actors in play and the architecture is more complex.

One of the foundations for which PKI is built upon is public key cryptography. Public key cryptography uses key pairs to encrypt and decrypt messages. These keys are called public and private keys. The key names are almost self-explanatory, private keys should be kept private at all times, whereas, public keys can be made publicly available.

Messages encrypted with a party's public key can be decrypted only with its private key. This means that no other party can read the message, with the assumption that the private key is securely stored. Messages encrypted with a party's private key can be decrypted with the public key, which authenticates the message sender. Successfully decrypted messages with the party's public key indicates that the party who encrypted the message must have the private key.

2.9 X.509 Certificates

X.509 is the Telecommunication Standardization Sector (ITU-T) standard for public key infrastructures (PKI) (Mishari, Cristofaro, Defrawy, Tsudik, 2009). X.509 defines the standard format of public key certificates and many other things such as attribute of certificates and an algorithm for certification path validation (Mishari et al., 2009). In this paper, X.509 public key certificates are referred to as "certificates" or "TLS certificates".

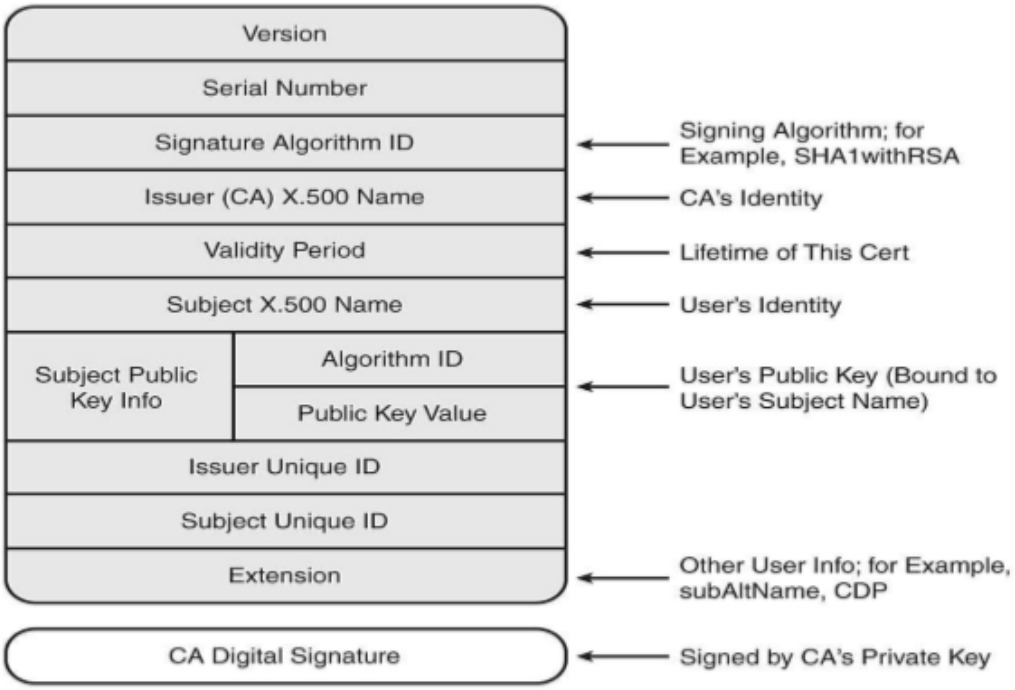

Figure 3 Overall structure of the X.509 Certificate structure Mishari et al., (2009).

Mishari et al., (2009) have described the general structure of X.509 Certificate, which is shown in the Figure 3, and they also described the fields that are found in all X.509 certificates:

1. Version - Describes the X.509 version number
2. Serial number - Describes the serial number assigned by the issuing Certificate Authority.
3. Signature Algorithm ID - Describes the signing algorithm used by the issuer.
4. Issuer - Name of the issuer.
5. Validity Period - Describes the start date and end date for validity. The certificate is only valid during this time period.
6. Subject Name - Describes the name of the entity for which the certificate is issued. For example, the domain name of the website.
7. Extensions - Extensions are optional but quite commonly used, for example Subject Alternative Name (SAN) or Server Name Indication (SNI).
8. Signature - Issuer's signature.

Subject name of certificates holds particular importance because that attribute is used to specify the certificate to an entity. Usually this is done by listing the domain name of the website used in this field, for example. The certificate is then only valid when used in a domain that matches the subject name of the certificate.

Another field with great importance is the signature. The signature is given by the certificate authority signing the certificate with its private key. This

signature is validated whenever a TLS connection is made. The signature allows us to inspect who has issued the certificate.

Validity periods are also important. Each certificate has start and end dates for its validity. Certificates are valid only between the timeframe of these two dates. For the findings of this paper, the end date attribute was in great interest as one of the research goals was to identify domains with expired certificates.

Originally SSL required that servers presented the certificate without knowledge of which domain the client was requesting (Cangialosi, Chung, Choffnes, Levin, Maggs, Mislove, Wilson, 2016). Moreover Cangialosi et al. (2016) describe the issue as follows:

> "This effectively prevented servers from supporting more than one domain per IP address, as a server could only serve a single certificate per IP address, and each certificate could contain only a single Common Name." (Cangialosi et al., 2016).

The problem described above clearly limited implementations and the effectiveness of SSL. To tackle this issue two notable extensions were developed. These extensions are discussed next.

2.9.1 Subject Alternate Names (SAN)

Subject Alternate Names (SAN) extension allows a single certificate to specify multiple domains for which that certificate applies (Cangialosi et al., 2016). Basically, this means that one certificate can have multiple common names (Cangialosi et al., 2016). In other words, one certificate can be used for multiple websites. This is widely used and supported in today's internet.

For example, a certificate could define a SAN list [*.example.com, *test.net], which is included in the certificate itself. This certificate could be then used in all domains that are defined in the SAN list. This is quite convenient and commonly in use in today's internet. Some certificates take the usage of SAN to extreme levels. Akhawe et al. (2013) reported that they found certificates in their data sets that contained more than 540 names in the SAN field.

2.9.2 Server Name Indication (SNI)

Server Name Indication (SNI) is an extension to TLS protocol, which allows the client to specify the domain name that the client wants to connect to (Cangialosi et al., 2016). This solves the issues of hosting multiple domains in the same IP address.

When SNI is used, the client can specify the domain name it is trying to connect to and the server can then use this information to present the correct certificates. Earlier, if there were multiple domains hosted on the same IP ad-

dress the server could not make sure which domain the user was trying to connect to, and thus, could not deduce which certificate should be presented.

However, SNI requires support both from client and server side, but if both parties support it, SNI allows the server to serve multiple certificates from a single IP address (Cangialosi et al., 2016). Most modern browsers support SNI.

3 CERTIFICATE ECOSYSTEM

In this chapter the certification ecosystem and its components are described in detail. Having read this chapter, the reader is familiar with the different actors involved in the certificate ecosystem including certificate authorities, registration authorities and intermediate certificate authorities. Different types of certificates are introduced and the process of certificate requesting, issuing, renewal and revocation is presented.

In this paper the term Certificate Ecosystem means all the necessary actors, certificates and other components required for the overall process on data encryption and server authentication to take place utilizing the X.509 certificates and PKI architecture.

3.1 Certificate Authority (CA)

Each individual certificate must be signed by a higher authority in the certificate hierarchy. So called Certificate Authorities (CAs) are publicly known and trusted parties who are responsible for root certificates (Holz et al., 2015). Root certificates are used to sign other lower level certificates. Different types of certificates, including root certificates, are discussed later in detail. Ultimately, the whole SSL/TLS protocols are based on the trust given to these root certificates and parties responsible for them.

There are many different CAs with their own root certificate and private keys. CAs and their root certificates are equally important and they are allowed to sign certificates to any domain (Durumeric et al., 2013). Basically, this means that a certificate signed by the smallest and least secure CA is technically as valid as the certificate signed by the biggest and most trusted CA.

Durumeric, Kasten, Bailey & Halderman (2013) analyzed organizations that are involved in the certificate ecosystem, and they were able to identify 1832 CA certificates, which were controlled by 683 different organizations, including financial institutions, religious institutions, museums, libraries and

over 130 corporations and financial institutions. Their study was conducted almost ten years ago, but one can argue that it describes the CA ecosystem rather well, it's a mess. In addition to this, more and more organizations have acquired a role in the CA in the ecosystem.

3.2 Intermediate Certificate Authority

Certificate Authorities (CA) with root certificates are strongly discouraged to sign certificates directly. Instead, CAs often sign certificates to intermediate Certificate Authorities. An intermediate certificate authority can then sign end-user certificates or other intermediate certificates for others (Akhawe, Amann, Vallentin & Sommer, 2013).

This approach has many advantages. Firstly, intermediate certificates do not need to have access to the private key of the root certificate in order to sign another intermediate or end-user certificate. The security issue behind this is described in detail later in this paper. Secondly, having intermediate certificates in play reduces bandwidth end-users, since browsers and other clients that use TLS, can cache common intermediate certificates (Akhawe et al., 2013).

Intermediate CAs and their certificates provide flexibility and security to the whole system, however, one issue is that there are no lists publicly available of intermediate CAs (Durumeric et al., 2013). This is one of the issues with intermediate certificates, as they obscure the overview to the whole system. It is common to see chains of certificates where there are multiple intermediate certificates.

To conclude, an intermediate CA is an entity that is allowed to sign certificates directly to end-users, such as websites, but also to other intermediate authorities. Each intermediate CA has its own certificate that needs to be signed by root certificate. Note that there could be other intermediate certificates in the certificate chain before the root certificate on the top of the chain. Intermediate certificates authorities do not control root certificates, which adds security to the system.

3.3 Registration Authority

Before CA issues a certificate to any other party, there are requirements that need to be met. Each certificate applicant needs to prove ownership of the domain that they are requesting the certificate for. For example, a website owner needs to prove that they control the domain. There are different ways to do this ownership verification. This verification step is discussed later in more detail.

Sometimes CAs use so-called Registration authorities (RA) to perform the actual domain verification (Roosa & Schultze, 2013). These parties are often external companies that provide domain verification as a service for the CA.

There could be some security questions related to the use of RAs, however, the usage remains rather rare.

3.4 Certificate types

In this chapter all different types of certificates are introduced. Every subcategory of certificates is based on the X.509 standard and they have similar attributes and structure. However, they differ in usage.

TLS certificate binds a subject name to a public key (Cangialosi et al., 2016). This is common for all different types of certificates, whether a root, intermediate or end-host certificate. The subject is most often a domain name of the website, but technically it could be something else as well, like IP address. This binding allows the authentication of the web server.

3.4.1 Root certificate

Each CA has their own root certificate. This group of trusted parties have their own root certificates, which are self-signed by the party itself (Cangialosi et al., 2016). The private keys which are used to sign root certificates should be under massive security requirements. Should the private keys of the root certificate leak, each intermediate or end-host certificate that leads to the root certificate is also compromised.

These root certificates are preinstalled in most software packages that require TLS, such as browsers, operating systems, and mail clients and many more (Holz et al., 2015). These pre-installed root certificates play a vital role in the whole TLS architecture.

3.4.2 Intermediate certificate

Intermediate certificates are certificates that belong to the intermediate CAs. Intermediate CAs are allowed to issue other intermediate certificates and also end-host certificates. Intermediate certificates need to be ultimately issued by trusted root certificates, even though there can be other intermediate certificates in between the root certificate, in the whole certificate chain.

3.4.3 End-host certificate

End-host certificates are certificates that are actually used in TLS implementation across the internet. These certificates are referred to in academic research as "end-host certificates" or as "leaf certificates". In this paper, the term end-host certificate is used.

Businesses, organizations, and individuals can acquire end-host certificates for their services to allow connections secured by TLS. End-host certificates are often issued by intermediate CAs. Nowadays, end-host certificates are

available as commercial certificates provided by companies and also free ones provided by Let's Encrypt. End-host certificates represent the lowest certificate level in the architecture

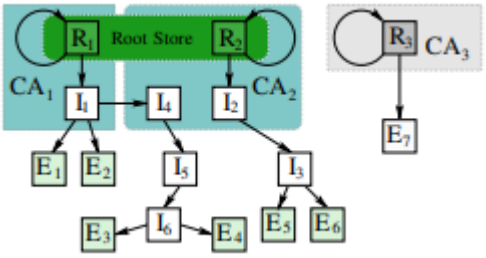

Figure 4 Most relevant features of PKI (Holz et al., 2015).

Holz et al. (2015) provide a clear illustration of the most relevant features of X.509 PKI, the different types of certificates and their usage (Figure 4). R1 and R2 are the root certificates of the CAs. The circle (above CA1 and CA2) illustrates that the root certificate is self-signed by the party itself. The certificates I1-I7 are intermediate certificates. These are parties that are allowed to issue end-user and intermediate certificates, but they do not have their own root certificates. Certificates E1-E7 are end-host certificates. Note that end-host certificate E7 is directly issued by root certificate, which is strongly discouraged. In addition, certificate E7 is signed by untrusted CA as it is not found in the root store (Holz et al., 2015).

3.5 Certificate issuing

Traditionally certificates were issued based on the actions made by the system administrators. The process begins with generation of a private key, then a certificate signing request (CSR) needs to be created, then a CA is selected, and eventually the administrator needs to fulfil the verification requirements made by CA (Aas et al., 2019). On paper, this does not sound overly complicated, but reality suggests that a lot of administrators were confused by this.

Certificates are not issued to everyone without some due diligence. Every certificate requester, whether an organization or individual, should go through some checks before they are issued any certificates. Usually, these checks involve checks if the requesting party can receive emails under the requested domain and email address (Holz et al., 2015; Akhawe et al., 2013). This is considered to indicate ownership over the domain.

3.6 Certificate reissuing

After a service has successfully acquired their first certificate, installed it and adopted its usage, it is only the beginning of the journey. Certificates need to be reissued from time to time for various reasons.

Two primary reasons for certificate reissuing are expired or revoked certificates. In both cases, the administrator must take an action otherwise the TLS connections will not work in their services. When dealing with revoked certificates it is important to understand why the certificate was revoked. If there is any doubt that the private keys are leaked, the administrator must create new public and private key pairs, as creating a new certificate with leaked keys does not solve the issue (Zhang et al., 2014).

If reissuing the certificate is done manually, it means that the administrator must contact the CA and send a new certificate signing request (CSR) (Zhang et al., 2014). Once the certificate is received it might need to be installed manually, which often means replacing the old certificate with new one. However, this is heavily dependent on the implementation. Automatic processes for certificate renewal exist.

3.7 Certificate revocation

Another important function of the CAs is that they are also responsible for certificate revocations (Zhang et al., 2014). A certificate is revoked with utilization of Certificate Revocation List (CRL) (Zhang et al., 2014; Alrawais et al., 2015). To put it simply, CRL is a list of certificates that are revoked, and they should no longer be used. The list also contains the reason behind the revocation and a timestamp (Zhang et al., 2014).

If any certificate would be found on any CRL, the validation process would fail on that certificate. It is also important to understand that if a root certificate or intermediate certificate is revoked, all child certificates referencing the parent certificate would not validate.

3.8 Web hosting with SSL/TLS

Some organizations manage their own web services, including TLS certificates. However, managing your own web infrastructure is time consuming, costly and error prone. As a result, many organizations deploy their web services through Content Delivery Networks (CDN) or through web hosting services (Cangialosi et al., 2016). These service providers are managing big portions of today's internet.

While having your website running on a CDN or web hosting service may provide good security and well-maintained infrastructure, including TLS certificates. This has led to a rather surprising issue. In a study conducted by Cangialosi et al. (2016) they argued that the only way for CDN or web hosting providers to manage TLS certificates for their clients is to have the client's private key. This is called key-sharing. Key sharing is extremely common. Over 76% of all organizations share more than one private key with a third party (Cangialosi et al., 2016).

Currently, hosting providers have accumulated a massive amount of trust in the form of private keys. In their study Cangialosi et al. (2016) found out that a collection of ten hosting providers have private keys for 45.3% of all domains that they observed in their scans. These are staggering numbers. These hosting providers could be targeted by cyber attacks, and if they would be compromised, the consequences would be serious.

Cangialosi et al. (2016) found out that big service providers respond slower but more profoundly to known vulnerabilities than parties who manage their own certificates and private keys. The popularity of service providers makes perfect sense, they often offer adequate security and ease of use for their customers whereas self-managing becomes tedious and expensive. Secure self-management of private keys and certificates also requires know-how and expertise. In the worst-case scenario self-management can be expensive and does not provide security overall.

4 CERTIFICATE CHAIN VALIDATION PROCESS

Having a certificate installed on a domain or server does not suffice. The complete certificate chain needs to be validated for TLS to work correctly. The validation process verifies the whole certificate chain starting from the end-host certificate up until the root certificate. The process ensures that the chain does not contain errors, misconfiguration, lack certificates, is signed by a trusted root, the domain matches the certificate, and more. This validation process is described next in detail.

The validation process begins with the client requesting a connection and thus the TLS handshake begins. The server then responds to the client's requests by sending the certificates needed for chain validation. The server's response usually contains all certificates including, the end-hosts certificate, possible intermediate certificates and the root certificates (Holz et al., 2015). However, the root certificates are usually discarded, because they should be found from the client's root store, whether this is an operating system, browser, or something else (Holz et al., 2015).

Once the web server has provided the needed certificates to the client the validation process begins. During the validation process multiple things are checked to provide a conclusion, whether the web server is legitimate or not. This process includes, building the whole certificate chain starting from the end-host certificate to the root certificate, validating the whole chain to ensure that all certificates in it are valid, and validating the domain name of the end-host certificate to ensure that the certificate truly belongs to the same domain using it.

As an example, the validation process for E4 end-host certificate would require the chain validation until the root certificate R1 is reached (Figure 4). All the following certificates would be validated; E4,I6,I5,I4,I1 and R1. Any deviation in the process would result in failure in the certificate chain validation. The other example of failure would occur in the validation of end-user certificate E7, as the R3 is an untrusted CA.

Errors in the validation process are very common. There are various types of errors that can occur in the certificate validation process. We will next discuss

the most common ones. Akhawe et al (2013) have categorized the possible errors during the whole certificate validation process into three main categories: chain building errors, chain validation errors, and name validation errors. The same approach is used in this paper.

4.1 Chain building errors

The first step in the certificate chain validation process is to check whether the certificate chain starting from end-user certificate to trusted root certificate can be successfully built (Akhawe et al., 2013). Chain building errors happen while building the certificate chain from end-host certificate up to the trusted root certificate. Next these types of errors are discussed in more detail and with examples.

4.1.1 Unknown issuer

In these types of errors, the client is unable to create a chain from the web servers certificate up to the trusted root certificate (Akhawe et al., 2013). The typical situation for this type of error happens when the entity who issued the end host certificate for the domain is not known (Akhawe et al., 2013). In other words, this unknown issuer is not in the root store, and neither is it a known intermediate certificate.

For example, this type of error would happen in the following scenario: the user navigates to a website called "example.com". Example.com has just acquired its TLS certificate from newly founded CA. As the CA is very recent its root certificate has not been yet updated to root stores of operating systems, browsers et cetera. Since the root certificate is yet unknown, the TLS connection will result in an error as the client does not recognize the certificate signed by the particular CA.

4.1.2 Self-signed certificates

Self-signed certificates are also a very frequent occurrence in the validation process. Technically self-signed certificates are not an error, but they are justified in very specific use cases (Holz et al., 2015). Self-signed certificate means that the issuer and the subject of the certificate are the same (Akhawe et al., 2013); Holz et al., 2015).

Self-signed certificates can be used for test environments and testing purposes in general, and in cases where having a real certificate would make things more complicated. Self-signed certificates are also often used in personal devices such routers, music players and more (Akhawe et al., 2013). Nevertheless, self-signed certificates are frequently utilized incorrectly.

An example of self-signed certificates would be if the website "example.com" would issue an X.509 certificate for itself. This would mean that both

fields, issuer and subject name, of the certificate would have the same “example.com” value.

4.1.3 Broken certificate chain

Web servers should present the whole certification chain when requested (Akhawe et al., 2013). However, sometimes the web server does not reply with all needed certificates. Very often web servers present only the end-host certificates, without any intermediate certificates or root certificates required for chain validation (Akhawe et al., 2013).

In addition, web servers may incorrectly present incomplete chains, add extra unnecessary certificates, have certificates in the wrong order, or contain duplicate certificates (Akhawe et al., 2013). As we can clearly see from this example alone, there are many things that can go wrong during the configuration of TLS.

Broken certificates are responsible for a minor amount of the certificate validation errors. These errors can occur for multiple reasons. The certificate chain provided by the server can be missing intermediate certificates all together (Holz et al., 2015). Or the certificate chain can be using CA certificates that are not found in the client's root store (Holz et al., 2015).

All major browsers cache valid intermediate certificates (Akhawe et al., 2013). Browsers will try to use these cached intermediate certificates to build a valid certificate chain alongside with the certificates received from the web server (Akhawe et al., 2013). This means that certificate validation could potentially succeed, if the webserver represents an incomplete certificate chain, but the browser has the needed intermediate certificates in its cache (Akhawe et al., 2013).

One example scenario where the certificate chain would be incomplete would be the following: domain “example.com” acquired its certificate through intermediate CA. The administrators of the “example.com” installed the certificate to their web server, for some reason they did not add the intermediate certificate to the web server. Now when the client requests a TLS connection, the webserver only presents the end-host certificate. In this scenario, the chain validation fails as the client has no idea about the intermediate certificate involved.

There is a possibility that the validation would succeed if the client has seen that intermediate certificate elsewhere and cached the missing certificate. Domain called “another.com” has acquired its certificate through the same intermediate CA. This time the end-user navigates first to “another.com”, which correctly presents all needed certificates. Then the user navigates back to “example.com” but this time the user's browser has the missing intermediate certificate in cache, and thus the certificate verification succeeds.

4.1.4 Multiple paths in chain

In a scenario where the web server provides multiple certificates, the chain validation encounters a problem where there might be multiple available paths

from end-host certificate to the trusted root certificate (Akhawe et al., 2013). There are different approaches to how client-side software handles this type of scenario. For example, client-side software could choose one path and validate it or try multiple paths and see if any of them validates (Akhawe et al., 2013).

4.2 Chain validation errors

The second part of the validation process is to validate the whole certificate chain. This validation happens after the certificate chain has been successfully built (Akhawe et al., 2013). In other words, the web server represented all required certificates, and the root certificate was indeed found from the client's trusted root store (Akhawe et al., 2013). Chain validation errors occur after the chain has been successfully built but some other issue is present in the chain and/or its certificates.

4.2.1 Revocation status

The certificate chain cannot contain revoked certificates. Certificates can be revoked from an entity for various reasons, such as compromised private keys or compromised CA (Alrawais et al., 2015.) The revocation status can be verified by checking the Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP) responders (Alrawais et al., 2015). With these methods the revocation status can be checked upon validation.

If any certificate in the chain would be found from CRL or OCSP, the validation would fail. Revocation of root certificate or intermediate certificate would also result in all end-host certificate verification to fail if the certificate chain uses any revoked certificate.

4.2.2 Encryption key strength and secure algorithms

Encryption key strength plays a vital role in secure communication and in X.509 certificates. When talking about key strength it is dictated by the actual key size that is used. For example, the recommended minimum key length for RSA is 2048 bits (Alrawais et al., 2015). Any smaller key size can compromise the security. Too weak key size will also result in an error TLS handshake and the secure connection cannot be established.

TLS connection also requires secure algorithms to be used. This is especially true when using TLS 1.3 versions, as many older insecure cryptographic algorithms were deprecated. For example, an attempt to establish a TLS connection with the insecure RC4 algorithm does not work in TLS 1.3.

4.2.3 Expired and not yet valid certificates

Each certificate has a start and end date for its validity (Akhawe et al., 2013). If the certificate is being validated outside this timeframe, the validation will result in an error. Not yet valid certificates are a rather rare occurrence, but still possible. Not yet valid errors will occur if the certificate is being used before it is considered valid.

Expired certificates are very common. They occur when certificates are used after the end date for the certificate has been reached but the server still continues to use the old and expired certificate. Renewal of these certificates is often neglected or completely forgotten by the administrators.

One of the key interest areas in this study was to identify domains who failed to renew their certificates before expiration. The number in which this was detected is discussed later in the paper.

Having an expired certificate is bad for security for many reasons. Many CAs do not keep track of the revocation status of expired certificates (Alrawais et al., 2015). In addition, the older the certificate is, the more likely the private key has been compromised (Alrawais et al., 2015). If the private key has been compromised, any party could present itself as the official owner. In addition, if the private key is leaked, renewal of the certificate with the same private key doesn't solve the security problem.

4.3 Name Validation errors

Third part of the validation process is the name validation step. This step ensures that the web server is the same entity which is represented in the provided end-host certificate. This validation step happens after the certificate chain has been successfully built and validated.

4.3.1 Name validation

Name validation happens after the certificate chain has been validated. During name validation the browser checks whether the domain name that the end-user is trying to connect to, matches the domain name specified in the end-host certificate (Akhawe et al., 2013). This is a very important step in the process. The reason being that name validation is the only step that actually can prevent MITM attacks, as attackers can always present perfect chains, but they cannot pass the name validation step (Akhawe et al., 2013).

Worth noting is that the common name field, that usually contains the domain name, can contain wildcard characters. Wildcards are utilized to cover several subdomains with a single certificate (Akhawe et al., 2013). For example, a certificate with common name value *.example.com, would allow "test.example.com" to pass name validation. Wildcard certificates are widely

used as they ease the administration and reduce the number of needed certificates.

To clarify, two things need to happen in order to successfully start a TLS connection between the client and server. Firstly, the server needs to present a complete chain of valid certificates until a trusted root certificate is found at the top of the chain. Secondly, the domain name listed in the end-host certificate needs to match the domain that the user is connecting to.

4.4 Security issues and limitations in TLS

Despite the security benefits that TLS provides to the modern internet it is far from a perfect system. In the past there have been various vulnerabilities and exploits that have led to successful cyber attacks. Some attacks have targeted software bugs, vulnerabilities, or bad design whereas others have exploited the whole PKI model and targeted vulnerable certificate authorities. This chapter provides an overview of the different security issues involved with TLS. Bear in mind that some of these vulnerabilities may have been fixed in the newer implementations of TLS, however, these issues are important to understand to ensure the security in the future.

Over the years SSL/TLS has evolved to different versions starting from SSL 1.0 to modern TLS 1.3. This natural evolution has been crucial since encryption algorithms have evolved massively since the founding of the first version of SSL. In addition, the computing power has grown significantly, which has made many encryption algorithms outdated as they cannot provide adequate security. In other words, this natural evolution of SSL/TLS has been addressing cryptographic weaknesses and design flaws in protocols (Clark & Van Oorschot, 2013; Bhargavan et al., 2013).

While the study conducted by Clark and Van Oorschot (2013) is nearly a decade old the same principles still apply today. Many protocols that were considered secure in 2013 are not secure today. However, the constant cat-mouse play drives the development of cryptography and encryption. It is very interesting to follow the cyber security world today as artificial intelligence, machine learning and quantum computing might offer the next generation of security issues and require competent solutions in the future.

Some attacks against TLS do not even target any specific design flaw or outdated protocol, but rather exploit poorly configured and managed TLS (Bhargavan et al., 2013). This is why investigation on TLS misconfiguration is important.

4.4.1 Issues in CA trust model

As stated before, the whole certificate architecture is as secure as the weakest CA. There are various trust issues in the current model.

The role of CA has faced criticism across the security community. The current CA based implementation of PKI structure is a prime example of weakest-link structure (Perl et al., 2014). The reason behind this is simple, since any CA can issue certificates for any identity/domain/organization, an attacker can target the most vulnerable and poorly managed CA. One compromised CA undermines the trust of the whole system significantly.

Firstly, there is no real benefit from acquiring the certificate from a trustworthy CA (Roosa & Schultze, 2013). All standard certificates work the same on the client machines. This does not drive CAs to compete against one another in the security that they provide. It also means that attackers can target the most vulnerable CAs to enable malicious activity (Perl, Fahl & Smith, 2014).

Secondly, CAs do not hold any legal responsibility for issuing a certificate for malicious domains after possibly failing to recognize them in the domain validation phase (Roosa & Schultze, 2013). Another issue relates to transparency. Often CAs structure their business and operations in a way that creates gaps in auditing and information is undisclosed (Roosa & Schultze, 2013). This creates uncertainty in the CA trust model.

Third issue is the outsourcing of domain validation. The outsourcing of this critical step could lead to potential issues where CAs end up granting the certificate, based on decisions made by the RA. The RAs could potentially have only partial or incomplete ability to conduct the domain verification (Roosa & Schultze, 2013). There have also been successful cyber attacks that have targeted RAs (Roosa & Schultze, 2013).

Browsers, operating systems, and other client-side software also play an important role in CA trust. As browsers, operating systems are so called “trust anchors”, which means that they are the ones to decide which CAs should be trusted and their certificates are preinstalled on client-side software (Dacosta, Ahamad & Traynor, 2012). This has resulted in hundreds of trusted CAs over 50 countries (Dacosta, Ahamad & Traynor, 2012). These numbers are from a decade ago, but the number of trusted CAs has not decreased.

Previously it was mentioned that CAs should not issue end-host certificates directly. There have been incidents relating to this in the past. The good thing is that the number of these cases is in decline, and they are becoming a more and more rare occurrence (Holz et al., 2015). There are clear security reasons why this is considered bad practise.

The reason being that if CA wants to issue certificates it needs to have its private key online (Holz et al., 2015). Compromising CA’s private key would have serious consequences for the whole PKI infrastructure. Corrective measures would require an update to all clients who have that private key stored (Holz et al., 2015). In addition, all certificates signed by CAs certificate, whether signed directly or indirectly, should all be revoked, and new certificates should be requested. Needless to say, that this is a massive endeavour should this happen.

Despite all the problems described above there is also a positive change in CA trust. After the founding of the Let's Encrypt organization there has been

change towards better, this can be seen in two different ways. Firstly, Let's Encrypt provides certificates free of charge to anyone. This has enabled a lot of small business and private users to get their certificate, since the price does not act as a barrier of entry. Secondly, the availability of free certificates drives out businesses that used to sell certificates with the attitude "just to make money" as compared to selling real security services to their clients. The companies that still sell commercial certificates have to compete in a more competitive market, which drives development.

4.4.2 Private key compromise

As described previously, the web server's certificate needs to be signed by the issuer's private key. The private key should not be disclosed to the public under any circumstances, as this allows any third party to disguise themselves as the original domain. Any private key leakage makes the certificates useless in terms of security. There have been incidents of private keys leaking in the past.

In a security vulnerability in OpenSSL that was found in 2014 called "Heartbleed" potentially made the web servers private key visible to attackers (Zhang et al., 2014). The scale of the incident was high. It is estimated that the security flaw was found in 17% of all HTTPS web servers (Zhang et al., 2014).

Zhang et al. (2014) found out that 73% of vulnerable certificates had not been reissued and 87% had not been revoked after three weeks of Heartbleeds disclosure to the public. These numbers indicate that a lot of administration is done by manual processes, revocation decisions are made slowly, or the key leakage is simply neglected.

4.4.3 Ineffective certificate revocation

One common issue in certificate revocation is that clients often download updated CRLs infrequently and use cached versions of CRLs (Zhang et al., 2014). This is mostly done to optimize client performance, however, this results often in a situation where CLRs are not up to date and validation of certificates could potentially validate even with revoked certificates.

On paper certificate revocation sounds rather straightforward, it is very easy to determine when a certificate is revoked, however, knowing when a certificate should be revoked is a much more difficult task (Zhang et al., 2014). This often requires knowledge of certificate architecture as a whole, human interaction and decision making, and the administrator needs to pay attention to the ever-changing cyber world to monitor possible vulnerabilities and so on.

There is evidence that supports the fact that certificate revocation is often neglected by the responsible administrators or certificate owners. Zhang et al. (2014) found in their study that, after a major vulnerability was discovered in OpenSSL, 60% of the reissued certificates were not revoked. In other words, the domain administrators responded to the vulnerability by reissuing a new certificate, however, they did not revoke the old certificate.

4.4.4 Man-in-the-middle attacks against TLS

While one of the key objectives of TLS is to provide server authentication and provide protection against notorious Man-in-the-middle (MITM) attacks, it is still possible to conduct such attacks even though TLS is in use.

Dacosta, Ahamad & Traynor (2012) describe in their work how such attacks are conducted against TLS. They also provide alternative mechanisms for protection against MITM detection, however, that is not the scope of this paper.

For the attacker to create a successful MITM attack against TLS there is a need for illegitimately generated certificates (Dacosta et al., 2012). The first step of the attack is the positioning of the attacher between the client and web server that the client is trying to connect to. Second, the victim makes a request and the attacker provides the illegitimate certificate. If the client accepts the certificate the TLS connection is established between the attacker and the victim (Dacosta et al., 2012.)

Note that MITM attacks against TLS exploit poor end-user behaviour as they might neglect the warnings presented by the browser, which is the only safeguard against the MITM attack. Figure 5 messages 1,4,5 and 8 are the messages required for TLS connection establishment, and after they have been exchanged the attacker and the victim have TLS connection.

In the background the attacker can then create another TLS connection with the web server that the victim was originally trying to connect to. After this connection has been created there are now two TLS connections active, but the client and web server only see one connection (Dacosta et al., 2012). The attacker can then decrypt, encrypt, forward and modify the content of the messages between the victim and the web server (Dacosta et al., 2012).

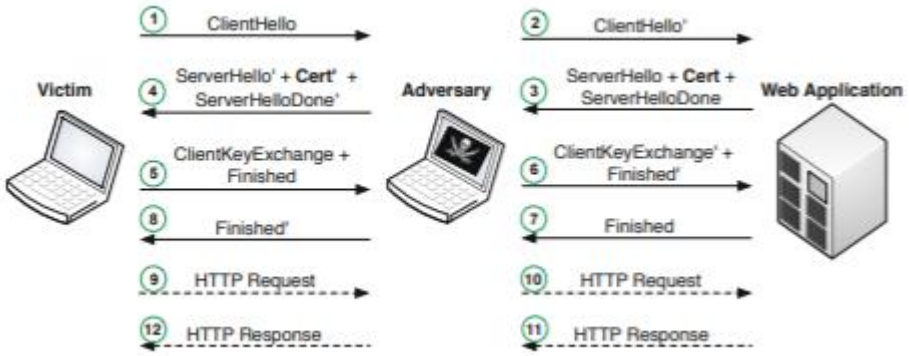

Figure 5 Example of MITM attack agains TLS (Dacosta, Ahamad & Traynor, 2012).

One other issue is in end-user behaviour. Even though end-users are often prompted with warnings about certificate verification failing, they often neglect these warnings and allow the connection (Das & Samdaria, 2014). Once the connection has been accepted by the end-user there will be no other warnings during that session. In addition, the possibly forged certificate can be cached inside the browser's memory (Das & Samdaria, 2014).

The rise of smartphones and mobile apps has also shifted some issues to the mobile world. Mobile application development has increased, and similarly failures to successfully implement TLS to the application have increased. Wang et al., (2020) reported that roughly 11% of Android mobile applications analyzed were vulnerable to MITM and phishing attacks due to an insecure TLS implementation in the application code.

4.4.5 Attacks against TLS protocol or cryptographic algorithms

Meyer and Schwenk (2013) introduced various attacks made against TLS protocol in their article. Most of these very technical attacks are targeted directly towards the TLS protocol or underlying algorithms used. Their article is nearly a decade old and some of the attacks might not be relevant today, however, it provides a perfect example of the variety of ways in which TLS can be attacked against. It is also worth noting that newer TLS versions and better cryptography algorithms that are used today, as compared to 2013, are more secure, however, it is almost certain that they too have vulnerabilities. TLS version 1.3 has removed the support for some of the algorithms, but the constant development and updates are needed as more elaborate attacks are targeted to these encryption algorithms.

Meyer and Schwenk (2013) have divided the attacks they identified into four main categories; attacks on the Handshake protocol, attacks on the Record and Application Data Protocols, attacks on the PKI, and various other types of attacks. Meyer and Schwenk (2013) also include multiple examples of attacks belonging for each of the categories presented above. Many of these security issues have been fixed in the later versions, but they show how many different parts of the architecture can be attacked against.

4.5 State of the certificate ecosystem

Certificate configuration is a difficult task, and it is no surprise that errors during configuration are common. On top of that, certificates have their own lifecycle. This means that they expire, and renewal of certificates is, again, a difficult task for the administrators. This means that the certificate ecosystem is in constant change, as new certificates are added, certificates are revoked, certificates are constantly expired and renewed et cetera. In this chapter the historical state of the ecosystem is discussed. This chapter provides just some examples, which describe the complexity of the certificate ecosystem.

In the early 2010s the certificate ecosystem was already well established, nevertheless, the adoption and usage were limited to services where it required e.g., payment services, other financial applications, and other confidential information. Services that used TLS, most likely used the TLS 1.0 version. Over 90% of connections in 2012 were using the TLS 1.0 version, which was alarming because TLS version 1.2 had been standardized already in 2008 (Kotzias et al.,

2018). The mindset of SSL/TLS encryption in early 2010s was to protect the services where encryption was important. The encryption of all web traffic was not considered as important.

Durumeric et al., (2013) were able to also identify security issues with end-host certificates space; they found that half of the valid end-host certificates use inadequate 1024-bit RSA keys. The good thing is that browsers dropped support for 1024-bit RSA in 2014, which forced entities to renew their certificates with more adequate encryption methods. However, this is a perfect example of forcing security adaption, since in 2013 half of the certificates still used weak 1024-bit RSA. This kind of enforcement is needed continuously as TLS evolves with the modern internet. Another perfect example of this is the deprecation of various insecure cryptographic algorithms in TLS version 1.3 (Dowling et al., 2021).

The distribution of end-host certificates is largely dominated by a handful of large authorities (Durumeric et al., 2013). Durumeric et al. (2013) identified that three organizations control 75% of all end-host certificates. In addition, they identified a party whose compromised private key would force 26% of websites using HTTPS to obtain new certificates (Durumeric et al., 2013). As we can clearly see, an immense amount of trust is set to these large organizations.

They also found out that only 45% og HTTPS servers were correctly configured and 13% of web servers were so badly configured that they could not be used at all by some browsers (Durumeric et al., 2013). These findings illustrate how difficult it is to manage TLS implementations by hand.

Particularly interesting detail was explained in a study conducted by Perl et al. (2014) they found out that 34% of the root certificates are not used to issue any certificates used in HTTPS. They extracted their root certificates from twelve major trust stores, such as Android, iOS, Ubuntu, Windows, Firefox etc. (Perl et al., 2014). They also state that these most-likely unused certificates can be used to launch Man-in-the-middle attacks (Perl et al., 2014). Another interesting finding that Perl et al. (2014) pointed out is that only 28 trusted root certificates can be found from all twelve trust stores that they inspected.

TLS implementations for email remain especially challenging. Baumgäertner et al. (2015) found out that in Germany IP address space only 11% on email servers provided valid certificates. They also argue that the problems with TLS are harder to solve in email as compared to HTTPS. This is because when sending email there is no possibility to promote the end-user with TLS warning and require a decision like in HTTPS (Baumgäertner et al., 2015). In email everything needs to be automatic.

Another study that investigated TLS in email implementations had similar bad news regarding the overall state. The study found out that 15-30% of all servers accept weak cryptographic cyphers, and most of the certificates used are self-signed (Mayer, et al., 2016).

Chung et al. (2016) collected over 80M certificates over three years, and they found out that 88% of these certificates are invalid. This also means that studies that have been conducted on valid certificates only affect roughly 12%

of the whole certificate ecosystem (Chung et al., 2016). These numbers sound quite worrying, but there is a logical explanation why the number of invalid certificates is so high. Chung et al. (2016) state that invalid certificates largely originate from end-user devices that generate new self-signed certificates from time to time.

More recently Certificate Transparency (CT) has seen growth in the TLS ecosystem. Certificate Transparency is an open framework designed for domain owners to monitor certificates assigned to them and to identify fraud certificates. In 2018 33% of connections supported CT (Scheitle et al., 2018). In 2021 it is expected that support for CT has further increased. Scheitle et al. (2018) argue that some information logged in CT is confidential. In addition, they suspect that leaked domain names from CT logs are used to in malicious scans in the internet (Scheitle et al., 2018).

There have also been changes for the good. Kotzias et al., (2018) argue that TLS ecosystem has changed towards better over the last six years, mostly because of removal of weak algorithms, new elliptic curves, TLS version 1.3 and other positive changes. In 2018 the TLS usage was measured to be used in 23.6% of connections (Kotzias et al., 2018).

Let's Encrypt has affected the certification ecosystem significantly after it was founded in 2015. In 2021 Let's Encrypt is largest CA used by millions to get their certificates. Let's Encrypt has had a massive influence on certificate ecosystem overall. Most influential change has been removal of monetary cost of certificates, as Let's Encrypt offers certificates for free to any organization or domain.

This chapter has provided an overview of the state of the certificate ecosystem from past to present. While many improvements have been made to the ecosystem there is still a lot of work to be done. In the upcoming years HTTPS usage will continue to grow until almost 100% usage is achieved. Despite the advances made, misconfigurations and expired certificates still cause security incidents for businesses. Wider adoption of automated processes for TLS implementations could help the ecosystem in the future as manual implementations remain difficult.

4.6 End-user perspective

Everything that we've discussed so far, about certificate chain validation and common errors during that process, happens behind the scenes in the modern internet. Most end-users are not even aware that such processes or systems even exist. Once the user is confronted with an TLS error the first time, it could be a confusing event. Most likely these errors are shown inside a web browser as that is the most client-facing component of TLS (Akhawe et al., 2013). In this chapter the end-user perspective is discussed.

4.6.1 World of false positives

Assume a scenario where an end-user like you (however, maybe not as tech savvy) is browsing the internet. The end-user tries to proceed to a website. Behind the scenes HTTPS protocol is used, and the process of TLS handshake and certificate validation begins. For some arbitrary reason the website's certificate validation results in an error and the browser prompts a security warning for the end-user. The error is shown because the browser cannot distinguish between cyber attacks or harmless server misconfiguration (Akhawe et al., 2013). The end-user has two options, either to continue to the website and ignore the warning or he/she may decide not to continue to the website.

Figure 4 illustrates the end-user view of TLS error in Google Chrome browser while browsing to a site with an expired certificate (Badssl, 2021). This warning allows the end-user to decide whether to proceed to the website or terminate the connection. In Google Chrome, the end-user needs to click "Advanced" in the bottom left corner to see the "allow" option. Figure 5 illustrates the next view in which the user can actually proceed to the possibly unsafe website after clicking "proceed to expired.badssl.com (unsafe)" in the bottom right corner. This logic is fairly similar in all major browsers with some minor differences.

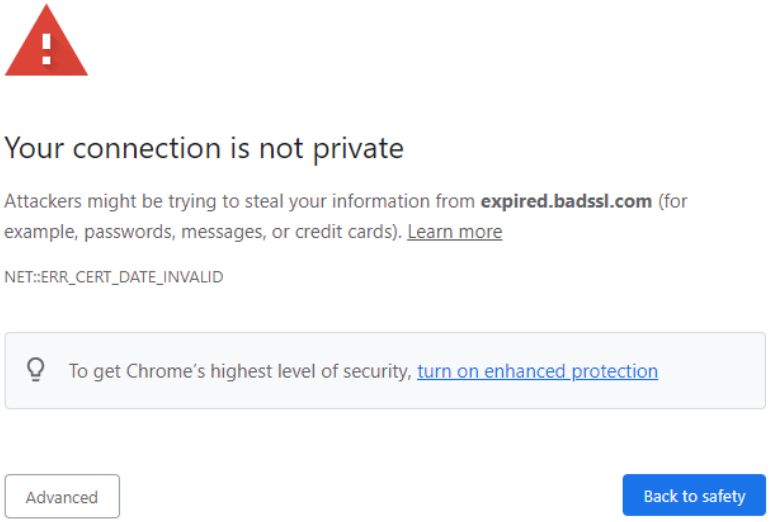

Figure 6 Example of end-used warning in Google Chrome browser. (Badssl, 2021).

As stated earlier, misconfigurations on server side are common. Direct cyber attacks targeted against the end-user, like Man-in-the-middle attacks, are way rarer. This results in a lot of false positive warnings during browsing (Akhawe et al., 2013). Studies have shown that end-users often neglect the warnings and click them through to proceed to the website anyway (Akhawe et al., 2013). It is obvious that ignoring alert messages quickly undermines the security that TLS provides. This kind of behaviour is often learned while encountering enough false positives while browsing the web (Akhawe et al., 2013; Acer et al., 2017).

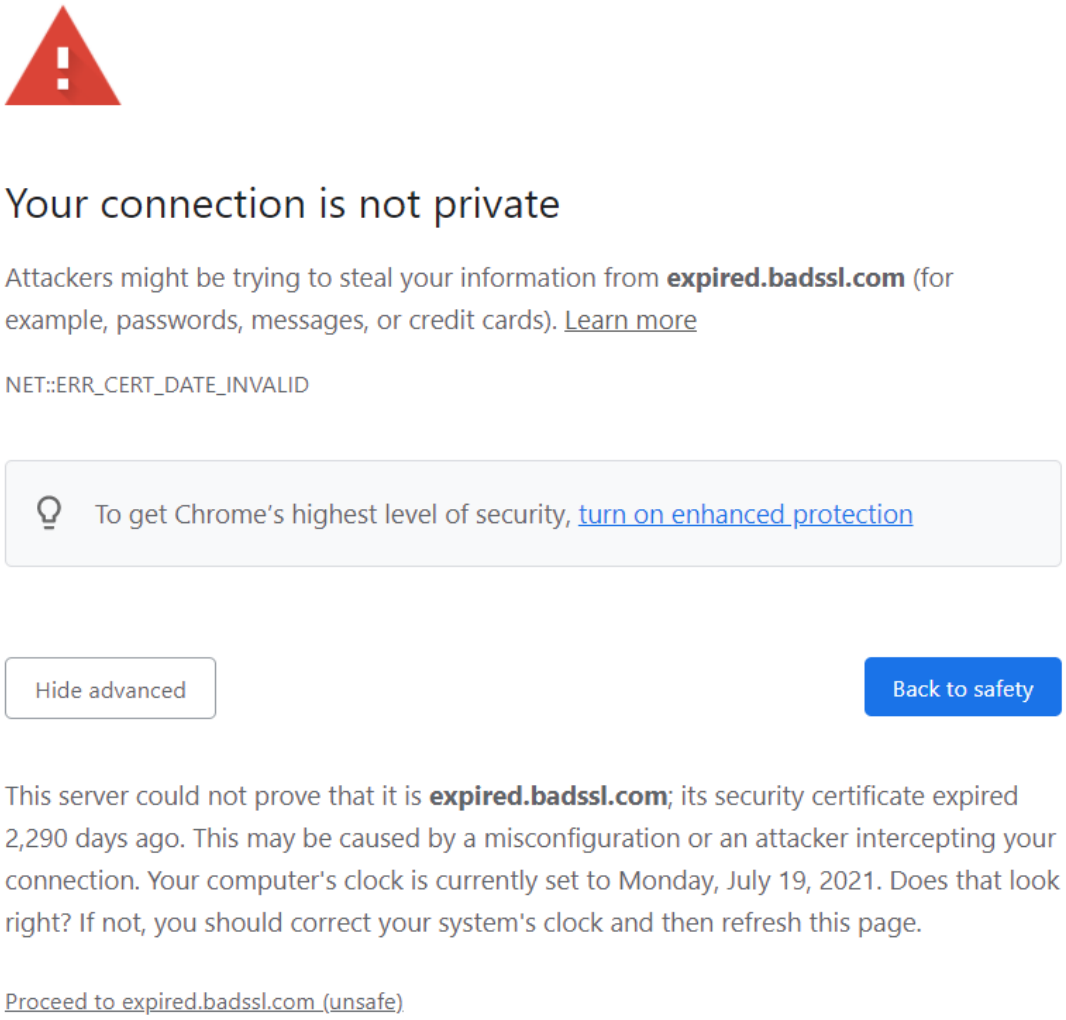

Figure 7 Second example of end-user view while proceeding to unsafe site (Badssl, 2021).

The question is how often these false positives occur compared to real attacks. Akhawe et al. (2013) monitored connections and they found that 1.54% all connections presented a false warning to the end-user. They also argue that real MITM attacks are way more rare that 1.54% of all connections (Akhawe et al., 2013). This means that false positives are actively training the end-users to click away the warnings, which heavily undermines the importance of TLS.

The false positives are also and unfortunately triggered by other events than misconfigurations on the server side. Acer et al., (2017) argue that more than half of the errors captured in Google Chrome browser were caused by either client side issues or network issues. Two most frequent issues were insufficient intermediates or incorrect client clocks (Acer et al., 2017).

There is also evidence that supports that all end-users do not understand the strongly worded warnings made by the browsers (Roosa & Schultze, 2013). Some arguments have been even made against the allowance of choice that individuals have when encountering a certificate related error while browsing.

Users could be prevented from connecting to a unsecure site rather than trying to tailor a choice which allows insecure behaviour (Roosa & Schultze, 2013). However, this seems a rather radical option but it illustrates the issue of confused end-users.

4.7 Administrator perspective

This chapter provides a perspective on how administrators have to manage their certificates, if no third party service providers are used and all certificates are managed manually. After reading this chapter the reader should understand why managing certificates this way is laborious and challenging. Often, manual administration also results in misconfiguration and security issues.

4.7.1 Obtaining and installing certificates

The traditional way of obtaining and installing the certificates is to do everything manually. The first thing that is required when acquiring a certificate is to generate a private key and then to generate a certificate signing request (CSR) (Aas et al., 2019). The CA requires administrators to complete the verifications steps (Aas et al., 2019). The verification is done to make sure that the administrator really has ownership of the domain in question.

Once the CA has granted the administrators the required certificate, it needs to be installed on the servers. Oftentimes, the administrators are blindly following instructions as they are doing this (Aas et al., 2019). The lack of understanding while installing the certificates often results in misconfiguration or other malfunctions.

While it is important to know how the certificates were requested and managed previously, it does not mean that this would be the only way. Fortunately, there are more elaborate tools to obtain and manage certificates. These tools are introduced later in this paper.

4.7.2 Certificate monitoring and renewal

Once the certificate has been successfully installed to a domain the job is not complete from administration perspective. All certificates have a lifecycle meaning that they will expire in the near future. This means that the administrators need to have a procedure in place to keep track of their certificates, and once the expiration date is closing in, the certificate needs to be renewed. Basically, this means requesting a new certificate and installing it again to the server.

This does not sound too complicated on paper, but in reality, certificates often expire before they are renewed. Aas et al. (2019) argue that before foundation of Let's Encrypt, around 20% of trusted certificates were let to expire before renewal.

In addition, the administrators should pay attention to the cyber world around them. There are many incidents that would require actions on admins, such as compromised CA or exposed vulnerabilities in TLS or other components of the architecture. The harsh truth is that this type of administration is a lot of work and requires commitment. For businesses this often means extra expenses, or the administrative work is away from something else.

4.7.3 Unclear TLS error messages

Administrators are the ones that have to deal with unclear TLS error messages, when debugging a problem in TLS implementation. Oftentimes the error messages are not clear and do not provide enough information to solve the problem. Especially name-constrained certificate messages are especially poorly understood among administrators (Ukrop, Kraus & Matyas, 2020). The error messages also affect the administrator's ability to identify flawed certificates, and thus these messages can have security implications as well. The author of this paper was also left with confusion while investigating various vague error messages that were found in the scan data.

4.8 How Let's Encrypt helps administrators

Before Let's Encrypt was founded all certificates were offered by commercial businesses. The price for a single certificate could range from tens of dollars into thousands of dollars a year, depending on the applicant, type of the certificate and the type of services provided with the certificate. In 2015 the average price for one-year certificate for one domain from five largest CA's was $178 (Aas et al., 2019). Lower cost one did exist, but they often came with some sort of limitations (Aas et al., 2019). This meant that applying encryption to all web traffic would have a high barrier of entry, as applying the encryption would be costly and complex (Aertsen, Korczyński, Moura, Tajalizadehkhoob & Van Den Berg, 2017). This was especially true for any non-commercial web sites, for example, blogs, personal websites, and hobbyist projects.

While commercial certificates are still widely available online there are several reasons why domain administrators should consider Let's Encrypt. Let's Encrypt was founded in 2015 and since then it has experienced significant growth, into the largest CA in the web (Aas et al., 2019). Let's Encrypt is a non-profit organization that offers X.509 certificates for anyone (who passes the domain validation that is).

The main benefits of Let's Encrypt are that it is free, open, and automated CA (Aas et al., 2019). Let's Encrypt also offers automated tools to install, configure, apply, and renew the HTTPS certificates, e.g., "Certbot" and "ACME" protocol (Aas et al., 2019). Certbot automated most of the tedious web server configuration tasks that are needed for HTTPS deployment (Aas et al., 2019). ACME is a protocol designed to automate the whole relationship between the

certificate applicant and the CA (Aas et al., 2019). Before ACME this relationship was based on manual messaging and often led to confusion among the administrators.

Let's encrypt offers a wide support for different web server software, such as Apache and Nginx (Aas et al., 2019). Certbot can also automatically modify the web server's configuration files, which is often the most tedious task of TLS deployment on the web server side.

There are also web servers that can directly integrate with Let's Encrypt and provision certificates automatically, for example Caddy (Aas et al., 2019). The web server has an integrated ACME client that handles all the certificate related activities and can provision the certificates without any user interaction (Aas et al., 2019). On top of this, Let's Encrypt supports integrations to web control panels, such as cPanel, Plesk and Webmin (Aas et al., 2019). This functionality is often utilized by hosting providers to deploy smaller sites with HTTPS (Aas et al., 2019). This capability has resulted in huge adoption numbers. Around 18% of Let's Encrypt's certificates are issued by cPanel plugin (Aas et al., 2019).

Let's Encrypt has also support for web hosting services to automatically provision HTTPS for their customers, again, without any user interaction (Aas et al., 2019). In 2019 there were over 200 web hosting providers, such as Wordpress and Google, that were utilizing the automatic provisioning of the certificates (Aas et al., 2019). This functionality has allowed quick and automated deployment of encryption in a large number of domains (Aertsen et al., 2017). Let's Encrypt certificates expire every 90 days, but thanks to the automation, nearly 70% of domains remain active after the first issuance (Aertsen et al., 2017).

It becomes evident that Let's Encrypt has done a lot of good for web encryption in general, especially by offering automated tools for certificate management and by offering the certificates free of charge for anyone.

5 METHODOLOGY

This chapter describes the methodology used in the paper. The research questions will also be clarified. Description of the data collection and data analysis part will guide the reader to understand how the study was conducted. Also, the technical architecture of the data collection is introduced at a high-level.

5.1 Research questions

The objectives of this study were to gain insight about various TLS related issues and their rates of occurrence in websites. The research questions were the following.

- **RQ1**: What are the most frequent TLS errors encountered in implementations?
- **RQ2**: How frequently do websites fail to renew their certificates?
- **RQ3**: What kind of services are impacted by poor implementation and management of TLS?
- **RQ4**: How long certificates stay expired, and what are the possible reason behind that?

5.2 Data description

The data set used in this paper comes from Rapid7 Open Data, which is a project that offers internet wide surveys across different services and protocols (Rapid7 Open Data, 2021a). The datasets offered are made public to enable security research (Rapid7 Open Data, 2021a).

The dataset used in this paper is from the Sonar project, which produces SSL/TLS certificate datasets each week Rapid7 Sonar (2021b). The data sets contain new certificates extracted from internet wide scans on IPv4 address space from port 443. The SSL-Certificates project provides new data sets on a weekly basis. One of the data files containing X.509 certificates, was selected to be used as the base data set.

The base data set contained roughly 80k base certificates. The base certificate data was then loaded into a MySQL database.

5.3 Data collection and analysis

The data collection was done during one month during June and July in 2021. During this time, we analysed roughly 8500 certificates. This is roughly 10% of the whole base set of certificates. The data analysis was done programmatically as such large data sets become impossible to analyse manually. In this study we wanted to track domains whose certificates were about to expire. This was done by following procedure.

1. All certificates in the base data set were scanned daily and those certificates that were about to expire in the next 30 days were flagged.
2. Domain name was extracted for each flagged certificate.
3. The domain was pinged three times. The ping was done by python program, which sent a request to the domain. The pings were conducted on four different ports 443, 465, 993, 995. Timeout value for pings was set to 2.5 seconds per port. The ping tried to establish an SSL/TLS connection with the server.
    a. First ping was done immediately after flagging approximately 30 prior to expiration.
    b. Second ping was done roughly a week before the certificate was about to expire.
    c. Third ping was done after the certificate had already expired.
4. The results of each ping were saved.
    a. Some pings resulted in connection timeout as the server did not respond in time.
    b. Some servers gave valid responses and TLS connections were established.
    c. Sometimes TLS errors occurred. These errors were also saved for further analysis.

The results of the pings could be then analysed, and the aim was to identify a wide range of TLS errors. Our findings are discussed later in this paper.

5.4 Technical architecture of the scan

This chapter provides technical background, which helps to understand the main concepts and technologies behind the scan. The software used in this paper, was specifically designed for pinging domains and validating their responses and saving the results for further analysis.

All the code used in the data collection was written in Python. Python provides fast development speed and has well-established libraries for connectivity. The software used tries to connect to a domain, if a TLS error occurs and the connection attempt fails, TLS related error is returned. The error messages were used to classify the issues and provide statistics and information about the domains certificate and TLS implementation.

The connection attempts were done by utilizing two python modules that provide needed functionalities, ssl and socket modules. Ssl module uses underlying OpenSSL which is responsible for the required cryptography in the TLS implementations, for example validation of the certificate chain is done with this module. Socket module provides a low-level networking interface that can be utilized with conjunction with previously mentioned ssl module. Figure 6 provides a code example of these modules in action. The example is from python documentation, but the source code used in the scan uses similar logic.

```
import socket
import ssl

hostname = 'www.python.org'
context = ssl.create_default_context()

with socket.create_connection((hostname, 443)) as sock:
    with context.wrap_socket(sock, server_hostname=hostname) as ssock:
        print(ssock.version())
```

Figure 8 Example of connection establishment using socket and ssl modules. (Python SSL module documentation, 2021).

The scans were conducted on several different ports, 443,465,993,995. The timeout value of 2,5 seconds was used in all pings and all ports. The results of pings were saved in a dictionary that contains information from different ports for each scan, month before, week before and after expiration. This dictionary was wrapped into a python class that provides certain utilities. Results were then saved to a MySQL database as base64 encoded string. Encoding ensures that the data is not corrupted, or any conflicts are not encountered during the insertion and updates in the database side. Once the data was analysed the python programs converted the encoded string back into objects that could be analysed. All of the ping data was backed up utilizing the MySQL native tool to backup database tables.

Docker containers were utilized to provide easily accessible and configurable architecture, which ensures that the environment is easy to set-up. Several docker containers were utilized for different purposes.

5.5 Inspection of the expired or badly configured domains

Closer inspection for all domains that were found expired or misconfigured during the scan was done. Categorization was done by what kind of services these domains provided. In addition, we tried to find out whether these domains offered registration to their sites and whether they used sensitive information, like usernames, email addresses, passwords, credit card or other private information. This type of personal information should always be protected by TLS encryption during data transit.

Another purpose of this inspection was to get a clear picture how many of these expired/misconfigured domains could be considered as commercial businesses. Commercial websites are interesting in the sense that they are providing a service online or selling products for exchange of money. This means that incidents relating to TLS will have some sort of effect on the business. Also, commercial sites need to provide a way for money transactions. This makes them targets for cyber attacks.

The hypothesis is that if the site's certificate was either expired or badly configured, there is a high probability that the domain IT processes are manual and there might be other areas where industry best practices are not being followed. The closer inspection will also provide data on what kind services these badly managed domains usually are. The results of the scan are discussed in the next section.

The inspection was carried out by hand and each web site was evaluated individually. The evaluation is based on the observance made by the author of this paper. This also limits the results as other inspectors could have different opinions. Any investigated domain names are not shared in this paper for privacy reasons.

5.6 Reproducibility

All the code used in this paper, including the docker files to set up the architecture, is given based on request. All raw data collected in the scan made in this study, is available as a .sql file, which can be used to create a copy of the collected data inside MySQL database. This table and its data was analysed further to provide the results.

6 RESULTS

This chapter describes the results of the study in detail. The overall statistics and numbers about the scan and its data are introduced to provide a scale for the study. Numbers regarding TLS version usage are also presented. Important TLS error messages that were encountered in the scan data are both introduced and categorized to provide information about the occurrence of each error. Lastly, the results of the manual inspection of web sites with either expired or misconfigured certificates are presented.

6.1 Overall statistics

Table 1 shows some high-level data from the collected dataset. During the 30-day scan time 8470 domains were monitored and connected to occasionally. 7925 (93.57% of all connection attempts) of these domains responded successfully to all pings during the scan period. This means that the TLS connection was established and thus the certificate must be properly configured and installed.

Connection failures happened occasionally while connecting to domains. In port 443 connection failures happened 188 times (2.21% of connection attempts). Connection failure category contains all errors that were related to connectivity. Sometimes the connection was timed out as the web server did not respond in time. The timeout value in the scan was 2.5 seconds. Occasionally, the web server refused the connection attempt. There were also web servers that initially responded to the pings, but during the 30-day windows the web server stopped responding, in which case it was impossible to measure the TLS certificate renewal in that domain.

	count	%
Total connection attempts	8470	
Successful	7925	93.57%

Connection failure	188	2.21
TLS error before scan	179	2.11%
TLS error during the scan	178	2.10%

Table 1 Connection attempts and distribution

In terms of encountered TLS errors, the results can be split into two different categories. First category are the TLS errors that occurred in all pings during the scan. This means that the domain and the certificate associated with that domain was invalid during the whole scan. These types of errors were found on 179 domains, which represents 2.11% of all connection attempts.

The second category are the domains, which had valid certificates at the start of the scan. However, after the last ping of our scan the certificate was no longer valid. Usually this meant that the certificate had either expired, or the certificate was tried to be renewed but it was misconfigured. Expired certificates were the most frequent reason behind this. There were 178 domains that belonged to the second category, which represents 2.10% of all connection attempts.

In total, there were 357 TLS errors present in the scan data, which counted for 4.21% of all connection attempts. These TLS errors will be analysed in more detail later section of the paper.

6.2 TLS version usage

If the web-server responded to the connection attempt and the TLS connection was indeed established, the web server provided the TLS version that it was using at the time. This lets us derive some data about the adoption of TLSv1.2 and TLSv1.3 to be more specific. Any lower TLS version would not be able to connect in the scan. Any versions of SSL or TLS 1.1 or lower are becoming very rare occurrences, these versions accounted for less than 0.1% connections (Anderson & McGrew, 2019).

The TLS connection was considered successful if the connection attempt did not return any errors from the port 443. Any domains that returned error in any state of the scan were omitted from this data. In other words, the TLS usage data is derived from connection attempts that were successful during the whole scan.

	Count	%
Connections with TLS 1.2	2927	36.93%
Connections with TLS 1.3	4998	63.07%
Total	7925	

Table 2 TLS Version usage between versions 1.2 and 1.3

During the scan a total of 7925 domains were successfully scanned. TLS version 1.2 was used roughly in 37% of connections whereas TLS version 1.3 was used in roughly 63% of the connections. Table 2 illustrates these numbers in detail. Even though the sample size is rather small it can be clearly seen that the adoption of the latest TLS version 1.3 is being already widely adopted. From a security perspective TLS version 1.3 is superior to earlier versions.

6.3 TLS errors encountered

As expected, there are various TLS related errors that occurred during the scan. The classification of these errors was done to provide data about the different issues encountered and their frequencies. The error message is returned from the python SSL library, which handles the TLS handshake and connection attempt used in the python program.

Next, we will discuss all the errors that were encountered during the analysis of the scan results. Each classified error message is presented along with the explanation behind it.

> "[SSL ERROR]: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has expired"

The error message above was used to categorize domains that presented expired certificates, which means a certificate that has exceeded its last validity date. This error message is rather self-explanatory.

> "[SSL ERROR]: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: self signed certificate"

This error message was received when the server provided a certificate that was self-signed, meaning that it was issued and signed by the same entity. Self-signed certificates were successfully detected during the scan. As with expired certificates, this error message is also self-explanatory.

> "[SSL ERROR]: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate"

The error above was used to differentiate errors that occurred because of an unknown issuer. These situations happen when the domain certificate is issued

by a party that is not found in the root store or it is not a known intermediate certificate that the client has seen before.

> "[SSL ERROR]: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: Hostname mismatch, certificate is not valid for"

The above error presents a situation where the certificate is registered to a different domain than the web server which actually presents the certificate. For example, if the certificate would be registered to example.com but the domain presenting that certificate would be forexample.com this would result in the same situation.

> "[SSL ERROR]: [SSL: DH_KEY_TOO_SMALL] dh key too small"

The above error is related to the weak key size used by the web server. Diffie-Hellman key exchange is used to securely agree on shared keys. However, in this case the key size is too small and considered insecure, which is why the connection attempt with TLS results in this error.

There is demonstrated evidence that shows that weaknesses in the Diffie-Hellman key exchange have been exploited in the past. Adrian et al. (2015) introduced the so-called Logjam attack in their study. Logjam attack exploited exactly these types of situations where too small key size was used by the web-server. However, there was also a security flaw associated with TLS that would let MITM attacks downgrade the used keys to smaller and insecure ones (Adrian et al., 2015).

6.3.1 Unclear error situations

In the scan data there were TLS errors presented that were ambiguous. These error messages were not totally clear to the author even after some investigation. Fortunately, the number of unclear errors is proportionately low. This kind of error messaging also troubles the administrators who need to investigate similar situations in production installations of TLS. For each of the error messages below there is an explanation presented based on a best guess made by the authors. However, each of them needs to be taken with a grain of salt, as the true root cause might lie somewhere else.

Unfortunately, these error messages could not be investigated in great detail. This is because the whole stack trace of the error was not saved, which leaves the authors only with the error message represented here. These types of situations are also very hard to replicate in a lab environment.

"[SSL ERROR]: [SSL: TLSV1_ALERT_INTERNAL_ERROR] tlsv1 alert internal error"

The error message above remained unclear to the author, even after some hours of investigation.

"[SSL ERROR]: [SSL: UNSUPPORTED_PROTOCOL] unsupported protocol"

Unsupported protocol error message indicates that there is a conflict between support of TLS versions and protocols used by the web server. In the scan, few of these scenarios were found. This could either indicate that the TLS version is too low, for example TLS version 1.0. Or this could also indicate that an insecure cipher suite was used.

"[SSL ERROR]: EOF occurred in violation of protocol"

After investigating the error message above, the best explanation of this error message is not actually in the TLS implementation of the server. In fact the issue seems to be in the Domain Name System (DNS). While it is unclear why the error message is typed the way it is, the DNS issues were put on the table after closer investigation of the domains presenting this error.

"[SSL ERROR]: [SSL: WRONG_SIGNATURE_TYPE] wrong signature type"

As discussed earlier, all the error messages that were received are not totally intuitive and clearly describe what is the issue. This demonstrates one of the issues with TLS. It is rather difficult to configure as there are lots of things that can go wrong. The administrators are often left confused by TLS implementations when something goes wrong. In addition, the reader must also bear in mind that there are numerous errors that simply were not encountered in this scan and thus are not represented in this paper.

6.4 TLS error statistics and classification

To gain more understanding about the TLS errors the errors were extracted from the dataset and classified based on the error message. This provides accurate numbers of their occurrence. As previously, the TLS errors are divided into two categories. First category are the TLS errors that were occurring during the

start of the scan until the end. Meaning that error has started to occur before the data collection took place.

The second category are the errors that were collected from the domain who initially responded to the first pings with successful messages. However, after the last scan TLS connection was not working anymore. Meaning that something had happened to the domain during the data collection. These domains in the second category hold particular interest, as they act as hard evidence of bad administration on the server side.

Error	Count	%
Expired	5	2.79%
Self-signed	5	2.79%
Local issuer missing	44	24.58%
Hostname mismatch	102	56.98%
EOF occurred	5	2.79%
Wrong signature type	7	3.91%
DH key too small	6	3.35%
Unsupported call	3	1.68%
Other	2	1.12%

Table 3 First category; improper TLS configured throughout the scan

Table 3 shows the data in the first category. These errors were visible in all pings on the domain during the whole scan, which means that the error or misconfiguration was already happening before the scan started in. Together there are 179 events in this category, which adds up to 2.11% of all scans.

By far the most common occurrence was hostname mismatch in the configuration. There were 102 errors that represent roughly 57% of all events in this category. In these scenarios the domain is trying to utilize a certificate that has not been registered to that domain name.

The second most common reason for TLS error in the first category was the missing local issuer. In total there were 44 of these events, which count for roughly 24,6% of the errors in the first category.

There were also 5 domains that were using an expired certificate at the beginning of the scan. Another 5 domains were using self-signed certificates. There were also 6 errors relating to weak key sizes used by the server. Few error messages were caused by unknown reasons, as the error message didn't provide much information about the issue. Fortunately, these errors happen rather infrequently, and they count for a small number of error cases.

Error	Count	%
Expired	84	47.19%
Self-signed	4	2.24%
Local issuer missing	9	5.05%
Hostname mismatch	10	5.61%

EOF occurred	7	3.93%
Other	64	35.95%

Table 4 Second category; improper configuration after the last scan attempt

Table 4 illustrates errors in the second category that were encountered for the first time during the scan. Meaning that in the first ping, 30 days prior to certificate expiration, the web server had presented a valid certificate and the TLS connection was made successfully. However, after the certificate was supposed to expire, based on the date information on the certificate, the domain was pinged again. This time, however, the connection was no longer working and instead TLS connection attempt resulted in an error.

In total, there were 178 events that were classified as error situations. As finding expiring domain is one of the research questions, one can consider the amount of expired sites found in the scan as success. The total number of 84 domains (roughly 47%) were found to let their certificate expire without renewal in time. This means that users of these 84 domains were prompted by browser security warnings relating to the expired certificate. It is hard to evaluate what kind of effects this warning might have caused to these domains. Most likely some domains were affected more than others. This is why manual inspection of all these domains was conducted, as they are very interesting when evaluating the business impact. The results of the inspection are discussed in the next section.

While the expired certificate was by far the most common explanation for the TLS error in the second category there were also other TLS related errors present in the scan results. These results indicate that the domain's certificate has been tried to renew, since the connection attempts were successful in the first two scans but then they failed in the last one. This means that the certificate is misconfigured during the renewal process. Self-signed certificates (4 events), local issuer missing (9 events), and hostname mismatches (10 events) are all examples of misconfiguration.

There were also 64 events which we classified as "other" events. These domains gave various responses to pings, which makes it hard to fit into one category. There were domains that had TLS error occurring in the first two pings, but then the issue was fixed, and error disappeared in the last scan. In other words, some domains had fixed their TLS implementation during the scan.

Mixture of connection issues and TLS errors were also present, for example, one domain had a connection failure in two of the first scans and in the last scan it responded with TLS error for expired certificates. There were multiple similar scenarios present in the data.

There were also domains with connectivity issues during the first two pings, but then the connection in the last ping was successful. The common nominator in this group is that the data is somehow incomplete or hard to fit into one category and therefore categorized as "other". This category is not particularly interesting for the findings of this study.

6.5 Expired domain inspection

Each expired or badly configured domain was inspected for deeper understanding about the domain and the type of web service that was affected by poorly configured or expired TLS certificate. This inspection was done manually. The results of these inspections are discussed in this chapter.
Total of 84 domains were found with expired certificates. Note that these domains were correctly configured in the beginning of our scan, but they failed to renew the certificate after it expired. These 84 domains represent various web services, ranging from online businesses, to blogs, porn, gaming, non-profit organizations, restaurants, dentists, online learning platforms, news sites or other media, governmental agencies, and scamming sites. There were also several sites that did not offer English as a language option, and it was nearly impossible to determine what the site was used for.

Alarmingly, 26 domains were still using the expired certificate over 30 days later when the manual inspection was done in August. There are few options that could explain this. One explanation is that these services might not be used anymore. But that raises the question why then pay for web hosting in the first place. It is not free to host services online. Second option is that most of the user base have already set an exception for the site in their browser and they do not see the warning anymore. However, having an expired certificate for over a month shows just pure negligence on the domain admin side.

Out of 84 expired web sites, a total of 12 online businesses were found. These businesses either sold products, offered services, or provided a platform for other commercial purposes. Two businesses are still using the expired certificate, more than 30 days after expiration. The business areas ranged from IT services, online shops for various merchandise, online learning platforms, leadership coaching, crypto currency and blockchain consulting and even web hosting.

These 12 identified businesses have in common that they have had the TLS incident, which has affected the business in some way. For the identified online shops this could mean direct financial losses as they might have lost customers during the time of the incident. Reputation and trust losses are also possible, sites selling IT services, web hosting, and cryptocurrency and blockchain consultation could have had this happen to them.

One of the domains that failed to renew the certificate belonged to the United States government. This is the perfect example that even big organizations and governmental agencies can indeed fail in the TLS administration and management. There is also evidence that implies deeper issues with governmental websites. Acer et al., (2017) argue that governmental websites are responsible for high number of server-side errors, as 65% of the most visited websites with TLS warnings are run by governments. These numbers are indeed worrying, as governmental websites are used for various sensitive purposes, like tax decisions, around the world. It also trains the end-user to accept possi-

ble errors when visiting these services, which undermines the whole security (Acer et al., 2017).

Two of the identified online businesses offered web hosting services. One of them sold "SSL Certificate services", along with other services, to their customers and advocated for its security benefits. Ironically, the website's own certificate had expired, and the issue remains to be fixed after almost 30 days of expiration. One business sold internet of things solutions relating to smart homes. Their website tried to convince the consumer about the security of their services, however, their TLS certificate is still expired today (again over 30 days after expiration).

There were also few domains that were clearly unfinished or somehow broken. Some sites clearly were still under construction, which partly explain issues with TLS. Also, few scamming sites that were most likely used for malicious purposes were found. Both of these categories contained a small number of domains out of the total of 84 expired domains.

There were also clear business sites that advertised their services online, such as restaurants, dentists, museums and more, but these services did not aim to make money online. As the provided service or product was sold "not online". These sites did not count as "online businesses", however, they too might have been affected by the TLS incident.

6.6 Results from ports 465,993,995

The scan collected data also from ports 465, 993, 995 in order to inspect email implementations of TLS. The certificates used in the scan were collected from port 443. This resulted in a high number of connection issues with these ports, for example, almost 49% of the servers scanned did not respond or allow the connection to port 465. Most likely the reason for this is that these web servers are not used as email servers.

Only 6.16% (522 successful connections) of the connection attempts were successful. The low number of successful connections also resulted in very few domains that could be inspected during the 30-day period and as a result of this, only 8 domains that let their certificate to expire were found. Similarly, other configuration issues were found on very low numbers.

There were also other data related issues regarding email ports. This ultimately made the author believe that the data collected does not represent the majority of email servers, and thus that data was not further analysed. In the future email servers could be targeted in a similar scan to gain more insight from TLS in email implementations.

7 DISCUSSION

Encryption and privacy have gained good momentum in the last decade. In addition, the foundation and wide adoption of Let's Encrypt has contributed to high numbers of TLS usage in today's internet. However, there are still a lot of improvements to be made in the cryptography space, including TLS.

In this study, a subset of X.509 certificates were scanned during a 30-day scan period. The focus was on the certificates that were about to expire during the scan. The main focus was to identify domains who failed to renew the certificate in time or domain who misconfigured the domain during renewal. Closer inspection was made to these domains to gain insight about it and the service it offered.

Both expired and misconfigured were found during the scan. In total, any kind of TLS error was found in 4.21% of all connection attempts. Despite the improvements made in the TLS landscape overall, certificates still expire without proper renewal at a steady rate. In addition, misconfiguration is still a frequent encounter in TLS connections. Both issues could be solved with proper certification management and tools, such as Let's Encrypt.

To get a better understanding, how many domains expire or are misconfigured in a year, we can get a rough idea by doing some simple calculations. In one of the most comprehensive certificate ecosystem studies, Durumeric et al., (2013) collected 42.4 million unique X.509 certificates. If we take this number as the amount of certificates in the whole ecosystem (the true size of the ecosystem is way larger) and multiply it with 0,0421 (TLS error % represented in this paper) the result would be roughly 1,8 million domains expire or are misconfigured each year.

Of course, this kind of calculation has to be taken with a grain of salt, as the certificate ecosystem contains way more certificates, but the true size is hard to estimate. But it gives a scale of how much these events could occur, millions in any case. The numbers are also influenced by the data and findings of this paper, which has its own limitations. The limitations of this paper are discussed at the end of this chapter.

Impacts that expired certificates can have for businesses could be high. This could potentially come from direct financial losses, e.g., an online shop could lose customers during the incident. This kind of direct impact is easier to evaluate, however, that evaluation is out of scope in this study.

Studies have been conducted about the trust among IT professionals towards flawed certificates, which indicate that IT professionals tend to trust expired certificates. The trust is influenced by time elapsed after the expiration and the reputation of the domain (Ukrop, Kraus & Matyas, 2020.) This means that expired certificates are seen as a quite common problem and they are not considered that severe, even among IT professionals.

Secondly, expired certificates could potentially impact businesses in terms of trust issues among the customer base. This could be especially applicable if the domain with expired certificate is dealing with Information security, finance, healthcare, or other high trust business areas. Loss of trust in the user base is far harder to evaluate and measure. In addition, the business implications could range from none to extremely high.

Misconfigured TLS implementations can potentially expose the domain to a variety of security risks. The errors classified as misconfiguration in this paper are self-signed certificates, hostname mismatches, unknown issuers, and other more infrequent errors. Studies have shown that self-signed certificates are trusted based on the context, as administrators can expect the use of self-signed certificates on a known server (Ukrop, Kraus & Matyas, 2020). Hostname mismatches were poorly understood by administrators (Ukrop, Kraus & Matyas, 2020).

Misconfigured or expired certificates might also train end-users to accept security warnings to the websites. This could expose the end-user to MITM attacks, as the end-user might expect a warning from the site because he/she has encountered one before. This might make the end-user to ignore the warning and proceed to a malicious web site.

There is also a possibility that, domains with overall bad IT and/or security practises, are more likely to be represented in our data, as they would be more likely to not renew or possibly misconfigure their certificates. Maybe bad practises in terms of certificate management are a sign of bad practises elsewhere in other IT services. However, this has not been tested in this study and leaves room for further research.

Potentially, certificate information on the domain could be used by hackers to identify domains with bad certificate practises in mass scale. This paper proves that this kind of monitoring can be done rather easily on a large scale. Better and more elaborate scanning tools could be developed for this purpose to gain more information about the domain.

The domains whose certificate ended up expiring/misconfigured during the scan, could be further analyzed, or even penetration tested, for other security issues or vulnerabilities. However, this raises some ethical questions and would require at least permission from the domain owners.

Changes in TLS version usage were also identified. In 2018 only 23.6% of connections used the TLS 1.3 version (Kotzias, et al., 2018). The dataset in this paper indicates that TLS version 1.3 adoption has increased significantly to be over 63%.

7.1 Further research

The findings of this study leave a lot of room for further research. As proved in this paper domains can be actively monitored for expired or incorrectly configured certificates.

One area of further research has to do with the business impact that misconfigured or expired TLS certificates can cause to businesses. In this paper, few well-known TLS related incidents have been described and some online businesses have evidently been under similar issues in our scan, however, most likely on a smaller scale. This offers great topics for further research.

Another topic for further research is to investigate whether bad certificate management is an indication of poor security and IT practises in general. In addition, poorly managed domain administrators could be contacted and asked about the current practises of their TLS implementations and management. However, there could be rather low willingness to participate in this kind of study amongst the admins.

7.2 Limitations

Limitations of this paper revolve around data collection and analysis. The scanner that was built for purposes of data collection and analysis, is by no means perfect and could potentially contain bugs (as all software does). These factors combined might affect our data in some minor ways.

Secondly, the sample size is rather small, containing roughly 8500 certificates. This is because pinging the web servers is rather time consuming, as many connection attempts result in a timeout failure. In addition, we wanted to be able to manually inspect the domains with improper configuration, and if the scale of the data would grow significantly this would eventually become too laborious and impossible to do. However, this leaves room for future research as more profound and robust ways of pinging and monitoring the web servers could be developed. This paper acted as a proof of concept, demonstrating that web servers can be monitored in large scale and their implementation of TLS could be evaluated.

Third limitation relates to networking. The data quality would improve if the scans would be made on a more “internet facing” machine. This would make more web-servers reachable and give better results overall. However, the

amount of connection issues in the scan remained surprisingly low, and it is now expected that this would have any significant impact on the data quality.

Acer et al., (2017) argue that networking issues are responsible for a major part of the error messages on TLS implementations. This also sets some limitations to our data collection, however, manual inspection was done to the websites which confirmed most of the results.

Lastly, as shown in this paper TLS is far from easy to set up and configure. This is also true for large scale monitoring of domains and their certificates. The configuration used in this paper might change if the underlying configuration would be changed. For example, changing the version of OpenSSL library in the operating system, could potentially change the way that certificates are validated in some scenarios. This is natural, as new versions of libraries might behave differently for various reasons.

Despite these limitations, the results and findings are expected to provide an accurate overview of the TLS ecosystem and prove that the problems with expired certificates do exist.

8 CONCLUSION

This paper investigated the TLS certificates and implementations in HTTPS web servers. The research goals were to find the most frequent issues in TLS implementations and provide data on how often web services fail to renew their TLS certificates, and to gain understanding of what kind of services are affected by expired or misconfigured certificates. Possible reasons behind expiration were discussed.

The results prove that certificates are still left to expire, while solutions for automatic management of certificates exist. Similarly, many web services were found with misconfigured certificates. Both previously mentioned issues are signs of bad administration on the server side. In addition, these incidents might affect businesses and the end-users using these services. However, monetary costs of these incidents are very difficult to evaluate.

Categorization of web services with expired certificates imply that improper certificates are not only an issue of small business or hobbyist sites. The results of this study shows that bad certificate management is an issue across the internet services, ranging from small personal websites to commercial online businesses, and even government websites

REFERENCES

Aas, J., Barnes, R., Case, B., Durumeric, Z., Eckersley, P., Flores-López, A., ... & Warren, B. (2019). Let's Encrypt: An automated certificate authority to encrypt the entire web. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (pp. 2473-2487).

Acer, M. E., Stark, E., Felt, A. P., Fahl, S., Bhargava, R., Dev, B., ... & Tabriz, P. (2017). Where the wild warnings are: Root causes of Chrome HTTPS certificate errors. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1407-1420).

Aertsen, M., Korczyński, M., Moura, G. C., Tajalizadehkhoob, S., & Van Den Berg, J. (2017). No domain left behind: is Let's Encrypt democratizing encryption?. In Proceedings of the applied networking research workshop (pp. 48-54).

Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J. A., ... & Zimmermann, P. (2015). Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 5-17).

Akhawe, D., Amann, B., Vallentin, M., & Sommer, R. (2013). Here's my cert, so trust me, maybe? Understanding TLS errors on the web. In Proceedings of the 22nd international conference on World Wide Web (pp. 59-70).

Alrawais, A., Alhothaily, A., & Cheng, X. (2015). X. 509 check: A tool to check the safety and security of digital certificates. In 2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI) (pp. 130-133). IEEE.

Anderson, B., & McGrew, D. (2019). TLS beyond the browser: Combining end host and network data to understand application behavior. In Proceedings of the Internet Measurement Conference (pp. 379-392).

Badssl (2021) “https://expired.badssl.com/”

Barnes, R., Thomson, M., Pironti, A., & Langley, A. (2015). Deprecating secure sockets layer version 3.0. In IETF RFC 7568.

Baumgäertner, L., Höchst, J., Leinweber, M., & Freisleben, B. (2015). How to Misuse SMTP over TLS: A Study of the (In) Security of Email Server Communication. In 2015 IEEE Trustcom/BigDataSE/ISPA (Vol. 1, pp. 287-294). IEEE.

Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., & Strub, P. Y. (2013, May). Implementing TLS with verified cryptographic security. In 2013 IEEE Symposium on Security and Privacy (pp. 445-459). IEEE.

Carlson, E. L. (2006). Phishing for elderly victims: as the elderly migrate to the Internet fraudulent schemes targeting them follow. Elder LJ, 14, 423.

Chung, T., Liu, Y., Choffnes, D., Levin, D., Maggs, B. M., Mislove, A., & Wilson, C. (2016). Measuring and applying invalid SSL certificates: The silent majority. In Proceedings of the 2016 Internet Measurement Conference (pp. 527-541).

Clark, J., & Van Oorschot, P. C. (2013). SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In 2013 IEEE Symposium on Security and Privacy (pp. 511-525). IEEE.

Dacosta, I., Ahamad, M., & Traynor, P. (2012, September). Trust no one else: Detecting MITM attacks against SSL/TLS without third-parties. In European symposium on research in computer security (pp. 199-216). Springer, Berlin, Heidelberg.

Das, M. L., & Samdaria, N. (2014). On the security of SSL/TLS-enabled applications. Applied Computing and informatics, 10(1-2), 68-81.

Dowling, B., Fischlin, M., Günther, F., & Stebila, D. (2021). A cryptographic analysis of the TLS 1.3 handshake protocol. Journal of Cryptology, 34(4), 1-69.

Durumeric, Z., Kasten, J., Bailey, M., & Halderman, J. A. (2013). Analysis of the HTTPS certificate ecosystem. In Proceedings of the 2013 conference on Internet measurement conference (pp. 291-304).

Felt, A. P., Barnes, R., King, A., Palmer, C., Bentzel, C., & Tabriz, P. (2017). Measuring {HTTPS} adoption on the web. In 26th {USENIX} Security Symposium ({USENIX} Security 17) (pp. 1323-1338).

Al Mishari, M., De Cristofaro, E., El Defrawy, K., & Tsudik, G. (2009). Harvesting SSL certificate data to mitigate web-fraud. arXiv preprint arXiv:0909.3688.

Holz, R., Amann, J., Mehani, O., Wachs, M., & Kaafar, M. A. (2015). TLS in the wild: An Internet-wide analysis of TLS-based protocols for electronic communication. arXiv preprint arXiv:1511.00341.

Kotzias, P., Razaghpanah, A., Amann, J., Paterson, K. G., Vallina-Rodriguez, N., & Caballero, J. (2018). Coming of age: A longitudinal study of tls deployment. In Proceedings of the Internet Measurement Conference 2018 (pp. 415-428).

Krawczyk, H., Paterson, K. G., & Wee, H. (2013). On the security of the TLS protocol: A systematic analysis. In Annual Cryptology Conference (pp. 429-448). Springer, Berlin, Heidelberg.

Lallie, H. S., Shepherd, L. A., Nurse, J. R., Erola, A., Epiphaniou, G., Maple, C., & Bellekens, X. (2021). Cyber security in the age of covid-19: A timeline

and analysis of cyber-crime and cyber-attacks during the pandemic. Computers & Security, 105, 102248

Let's Encrypt (retrieved July 2021; https://letsencrypt.org/stats/)

Maurer, U. (1996). Modelling a public-key infrastructure. In European Symposium on Research in Computer Security (pp. 325-350). Springer, Berlin, Heidelberg

Meyer, C., Somorovsky, J., Weiss, E., Schwenk, J., Schinzel, S., & Tews, E. (2014). Revisiting SSL/TLS implementations: New bleichenbacher side channels and attacks. In 23rd {USENIX} Security Symposium ({USENIX} Security 14) (pp. 733-748).

Meyer, C., & Schwenk, J. (2013). SoK: Lessons learned from SSL/TLS attacks. In International Workshop on Information Security Applications (pp. 189-209). Springer, Cham.

Mayer, W., Zauner, A., Schmiedecker, M., & Huber, M. (2016). No need for black chambers: Testing TLS in the e-mail ecosystem at large. In 2016 11th International Conference on Availability, Reliability and Security (ARES) (pp. 10-20). IEEE.

Cangialosi, F., Chung, T., Choffnes, D., Levin, D., Maggs, B. M., Mislove, A., & Wilson, C. (2016). Measurement and analysis of private key sharing in the https ecosystem. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 628-640)

Perl, H., Fahl, S., & Smith, M. (2014). You won't be needing these any more: On removing unused certificates from trust stores. In International Conference on Financial Cryptography and Data Security (pp. 307-315). Springer, Berlin, Heidelberg.

Python SSL module documentation (2021). https://docs.python.org/3/library/ssl.html

Rapid7 Open Data (2021a) https://opendata.rapid7.com/about/

Rapid7 Sonar (2021b) https://opendata.rapid7.com/sonar.ssl/

Rescorla, E., & Dierks, T. (2018). The transport layer security (TLS) protocol version 1.3.

Roosa, S. B., & Schultze, S. (2013). Trust darknet: Control and compromise in the internet's certificate authority model. IEEE Internet Computing, 17(3), 18-25.

Scheitle, Q., Gasser, O., Nolte, T., Amann, J., Brent, L., Carle, G., ... & Wählisch, M. (2018). The rise of Certificate Transparency and its implications on the Internet ecosystem. In Proceedings of the Internet Measurement Conference 2018 (pp. 343-349).

Suomen virallinen tilasto (SVT) (2020): Väestön tieto- ja viestintätekniikan käyttö [verkkojulkaisu].

ISSN=2341-8699. 2020. Helsinki: Tilastokeskus [viitattu: 18.7.2021].

Saantitapa: http://www.stat.fi/til/sutivi/2020/sutivi_2020_2020-11-10_tie_001_fi.html

Ukrop, M., Kraus, L., & Matyas, V. (2020). Will you trust this TLS certificate?. Perceptions of people working in IT (extended version). Digital Threats: Research and Practice (DTRAP), 1(4), 25.

Wang, Y., Xu, G., Liu, X., Mao, W., Si, C., Pedrycz, W., & Wang, W. (2020). Identifying vulnerabilities of SSL/TLS certificate verification in Android apps with static and dynamic analysis. Journal of Systems and Software, 167, 110609.

Zhang, L., Choffnes, D., Levin, D., Dumitraş, T., Mislove, A., Schulman, A., & Wilson, C. (2014, November). Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. In Proceedings of the 2014 Conference on Internet Measurement Conference (pp. 489-502).