How to Misuse SMTP over TLS: A Study of the (In)Security of Email Server Communication

Lars Baumgärtner, Jonas Höchst, Matthias Leinweber, Bernd Freisleben

Department of Mathematics & Computer Science, University of Marburg
Hans-Meerwein-Strasse 6, D-35032 Marburg, Germany
Email: {lbaumgaertner, hoechst, leinweberm, freisleb}@informatik.uni-marburg.de

***Abstract*—Electronic mail is one of the oldest and widely used services on the Internet. In this paper, an empirical study of the security properties of email server communication within the German IP address space range is presented. Instead of investigating end-user security or end-to-end encryption, we focus on the connections between SMTP servers relying on transport layer security. We analyze the involved ciphers suites, the used certificates and certificate authorities, and the behavior of email providers when communicating with improperly secured email servers. Conclusions drawn from this analysis lead to several recommendations to mitigate the security issues currently present in the email system as it is deployed in the Internet.**

I. Introduction

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are application-layer protocols to encrypt data segments transferred on the underlying transport layer of the Internet Protocol Suite. The communicating entities use X.509 certificates and thus rely on asymmetric cryptography to authenticate themselves and to exchange symmetric session keys to encrypt data flowing between the communicating entities. The use of X.509 certificates requires certificate authorities (CA) and a public key infrastructure (PKI) to verify the relation between a certificate and its owner, as well as to generate, sign, and administer the validity of certificates.

Several versions of TLS and SSL protocols are used in applications such as the WWW, electronic mail (email), and Voice-over-IP. The current version of TLS, TLS 1.2, was defined in RFC 5246 and released in August 2008, TLS 1.3 is currently available as a draft version. The most recent version of SSL, SSL 3.0, was released in 1996 (see RFC 6101).

The focus of this paper is the use of TLS in SMTP [1] [2], the Simple Mail Transfer Protocol, responsible for the delivery of email. In Figure 1, the process of sending and receiving email is outlined. Alice connects to her provider via SMTP on *Mail Submission Port* 587. Using *StartTLS*, she can encrypt the connection, as long as the email provider has this option enabled. After authenticating herself, she submits her email for Bob to her provider's email server. Her provider then looks up the DNS MX record for Bob's email address. In the next step, Alice's provider connects to Bob's provider using the *Mail Transfer Port* 25. Neither Alice nor Bob are able to review the connection properties the providers are using for the email transfer. Finally, Bob connects to his provider via the provider's web page or protocols such as POP3 or IMAP, and retrieves the email from his provider. Even if the email body may be encrypted by Alice using a client-side end-to-end encryption protocol such as Pretty Good Privacy (PGP) [3], meta-data such as sender, receiver and subject names may be visible to others, if the server-to-server connection is not encrypted properly. To secure the server-to-server connection, SMTP has been combined with TLS to encrypt email delivery and exchange between the participating entities [4]. Usually, the end user has no influence on this part except for his/her own mail submission to his/her provider's email server.

Recent revelations by Edward Snowden show that various government agencies actively and passively gather as much information from communication in the Internet as they can. Furthermore, since many corporate processes are coordinated using email within a company or with its costumers, the security of email is important for avoiding corporate espionage. Although consumers often communicate via Facebook, Whatsapp or Google Talk, email is typically used for banking, tax and online shopping related information that may be quite valuable for criminals, governments or other entities.

In this paper, we present the results of a study of the security properties of SMTP over TLS conducted within the German IP address space (about 100 million IP addresses). We take a look at the involved cipher suites, the used certificates, CAs, and the general availability of TLS within the detected SMTP servers. Since most private email correspondence is managed by a few big email providers, we also analyze the behavior of their Mail Transfer Agents (MTAs) when communicating

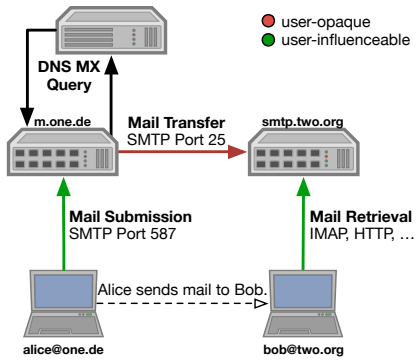

Fig. 1. Email transfer and TLS usage.

with improperly secured email servers. The results of our investigation lead to recommendations and best practices to solve some of the identified security issues.

This paper is organized as follows. Section II discusses related work. Section III presents the results of our empirical study. Advice on how to increase security of SMTP servers is given in Section IV. Finally, Section V concludes the paper and outlines areas for future work.

II. Related Work

The security properties of the TLS/SSL landscape have been investigated in several works. The used certificates, the lengths of the private keys and the supported cryptographic functions bear significant security risks, as indicated by attacks such as POODLE[1], BEAST[2] and LUCKY THIRTEEN [5].

Lee et al. [6] have investigated cryptographic cipher suites, key lengths and support for the insecure version SSL 2.0 in TLS/SSL servers. Attacks on the RC4 stream cipher [7] and the MD5 hash function [8] have been presented in other publications. In their study on the certificate ecosystem used in the WWW, Eckersley and Burns [9] have shown that only around 40% of the investigated web servers had a valid certificate chain. In 2011, Holz et al. [10] have presented their analysis of the SSL landscape and the use of X.509 PKIs based on active and passive gathering of certificates, indicating that only 18% of the provided certificates were accepted without warning when validating them with the Mozilla Root Store. Ristic and Small [11] have presented an overview of SSL usage in the WWW. In 2013, a similar study has been published by Durumeric et al. [12] to analyze signing CAs, key lengths and cryptographic algorithms. Fahl et al. [13] have conducted a mass audit of mobile Android applications to identify security issues in the use of TLS/SSL.

Giesen et al. [14] have published an approach to increase the security of recent mechanisms for TLS renegotiation. This hardening prevents Man-in-the-Middle attacks in some instances and minimizes the attack surface of applications using TLS. Focusing on TLS certificate management, Szalachowski et al. [15] have presented a solution based on the idea of publicly verifiable logs as made popular by Laurie et al. [16] with Certificate Transparency for PKIs. The reference implementation for a HTTP environment aids in securing the PKI landscape as a whole. Ryan's work [17] also enhances Certificate Transparency and integrates it with end-to-end email encryption in an attempt to make it easier accessible for users and avoid some of the cumbersome quirks of PGP. This approach helps to improve certificate management, removes several trust issues and includes end-to-end encryption transport channel security. Even if certificate validation is performed accordingly, it still leaves an attack surface. A general overview of past attacks on SSL in the context of the WWW and issues with the certificate trust model has been shown by Clark and Oorshot [18]. However, the authors have web browsers and HTTPS traffic in mind, and their solutions are tied to this use case.

Huang et al. [19] have published a study to detect forged SSL certificates in the wild. They have analyzed over 3 million real-world SSL connections to Facebook. Even though the used detection mechanisms are limited, about 0.2% of the connections were detected using forged certificates. Validating certificates is an error-prone task, and various SSL libraries have different default behaviors. Automated tests have been performed by Brubaker et al. [20] to reveal major flaws in common libraries and how they are used in software like web browsers, giving false or at least misleading feedback to the user. Slow deployment rates of security fixes for SSL related code has been identified as a key problem by Bates et al. [21]. The proposed approch hooks SSL verification code to non-browser applications to give them increased security without tampering with the actual source code. These extra validations impose a 20 ms overhead and work out-of-the-box with 94% of Ubuntu's most popular packages.

None of the cited works is concerned with the use of TLS/SSL in SMTP server communications. However, email is still an integral part of today's business communications. Recently, Facebook[3] has published some interesting observations regarding their email system. For example, *STARTTLS* is adopted by 76% of the unique MX hostnames that Facebook is in contact with. Moreover, 58% of the notification emails sent are successfully encrypted. The study concludes that general support for encryption is available, but proper certificates and certificate validation are missing.

III. An Empiricial Study of SMTP over TLS

Our study of the email TLS landscape is based on the German IPv4 address space. In particular, we scanned 116,824,576 IP addresses to investigate the TLS properties of German SMTP servers. Using nmap[4], we first checked whether the relevant ports were open and then analyzed TLS versions, cipher suites, certificates, CAs, and email provider strategies.

SMTP uses port 25 as its main port. Port 465 has been suggested for SMTP over TLS, called SMTPS (Simple Mail Transfer Protocol Secure) [4], but was later revoked; nevertheless some email providers still use this port. Port 587 is used to transfer email from the user to the provider's server.

Nmap also performs a reverse Domain Name System (DNS) lookup to find the domain names belonging to the referenced IP addresses. It is common to check the reverse DNS name when an email is sent to a server, thus the domain names should be set. These domain names are used later to check the validity of TLS certificates and to make sure that the IP in question is under the authority of the domain owner; emails from servers without valid reverse lookups should not be accepted.

A. TLS Usage

From the scanned hosts reached at the 116,824,576 IP addresses, 656,295 hosts offer SMTP services on at least one of the following ports: 25, 465 or 587. Figure 2 shows that

[1] https://poodlebleed.com/ssl-poodle.pdf

[2] http://www.hit.bme.hu/ buttyan/courses/EIT-SEC/abib/04-TLS/BEAST.pdf

[3] https://www.facebook.com/notes/protect-the-graph/the-current-state-of-smtp-starttls-deployment/1453015901605223

[4] Network Mapper - https://nmap.org

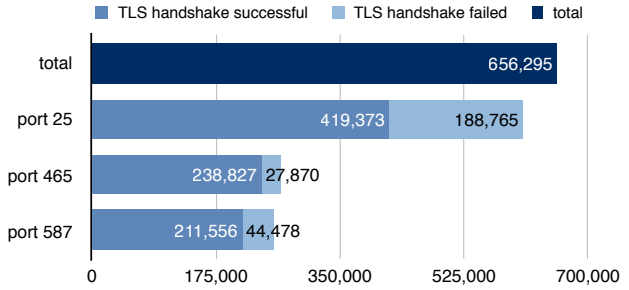

Fig. 2. SMTP and TLS usage among the scanned hosts.

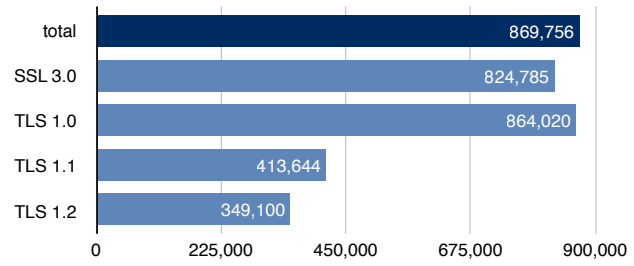

Fig. 3. SSL and TLS versions used in the scanned services.

only 68.96% (419,373) of the 608,138 hosts that offer SMTP services on port 25 perform a successful TLS handshake on this port. On the mail submission port 587, 82.63% (211,556 of 256,034) complete a TLS handshake successfully. While on port 465, which has TLS enabled by default, a TLS handshake nevertheless fails in 10.45% of the attempts. It is likely that server administrators use this port for different services not related to SMTPS. The results presented below are either based on the 656,295 hosts that offer SMTP services on at least one of the three ports or on the 869,756 services (see Fig. 3) with a successful TLS handshake on at least one of three ports.

B. TLS/SSL Versions

TLS and SSL have a long history of organic growth, leading to a non-uniform use of their different versions. SSL 2.0 had certain design flaws, ultimately leading to an insecure protocol. SSL 3.0 was considered secure, until the POODLE attack went public in October 2014. Our SMTP server scans took place in May 2014, five months before the POODLE attack was released to the public. On these grounds, the collected data allows us to observe the potential impact of the attack.

In Figure 3, the TLS/SSL versions used in the scanned SMTP servers are visualized. While the insecure version SSL 2.0 is not used at all, SSL 3.0 is enabled in 94.83% of the servers. The most popular TLS version 1.0 is supported by 99.34% of the servers, whereas the versions TLS 1.1 and TLS 1.2 are supported by less than half of the servers. BEAST's target was TLS 1.0, so there is no reason to hold back the newer versions. POODLE is also a good argument to use TLS 1.2 instead of SSL 3.0 – SSL 3.0 is only used for the sake (or curse) of compatibility.

C. TLS Cipher Suites

In a TLS handshake, the client offers a list of supported cipher suites to the server, which then picks one to secure the connection. To obtain all cipher suites supported by a server, the client has to iterate over the cipher suites and then tries to establish a connection using the selected cipher suite. This functionality is provided by the nmaps *ssl-enum-cipher* script[5]. Performing this test requires many connections to the tested servers and therefore creates quite some traffic in the network.

[5]https://nmap.org/nsedoc/scripts/ssl-enum-ciphers.html

If an attacker is able to influence the TLS handshake by selecting a cipher suite that (s)he is able to break, this attack is called a cipher suite *downgrade attack*. Even if backward compatibility is a reason to offer potentially insecure cipher suites, they open up a huge attack vector. The cipher suite classification of the nmap *ssl-enum-ciphers* script provides three categories of cipher suites, as described below.

Broken Cipher Suites: Broken cipher suites do not support protection against passive or active eavesdropping. Therefore, all anonymous cipher suites with an unauthorized Diffie-Hellman key exchange belong to this category. It is easy for an attacker to intercept and modify the messages between client and server and decipher the data. Furthermore, cipher suites using no encryption at all or just authentification are also considered as broken.

Weak Cipher Suites: The category of weak cipher suites mainly consists of historical cipher suites. Examples are the *export ciphers* that emerged as a consequence of the US export rules on cryptographic systems before 1999, and cipher suites based on the legacy Data Encryption Standard (DES). Weak cipher suites can be deciphered in a brute force manner by powerful hardware in less than a day[6].

Strong Cipher Suites: All remaining cipher suites are strong cipher suites. Although there are known weaknesses in RC4 [22] and MD5 [23], they are nevertheless used with some workarounds in many cases. With RFC 7465, attempts are made to remove RC4 completely [24].

Another noteworthy property of cipher suites is Perfect Forward Secrecy [25]. It ensures that a session key derived from a set of long-term keys can not be compromised if one of the long-term keys is compromised in the future. Thus, an attacker cannot decipher past messages even with the server's private long-term key.

Figure 4 illustrates the used cipher suites. Almost every server offers strong cipher suites, but only 27.66% of the servers are hardened in the sense that they offer strong cipher suites only. It is remarkable that 85.28% of the servers offer Perfect Forward Secrecy (PFS). A widely discussed issue is the use of Elliptic Curve Cryptography (ECC), since many of the ECC algorithms are suspected to have been constructed under the influence of US intelligence services[7]. In our data set, 22,32% of the servers support ECC ciphers (ecdh).

[6]http://www.sciengines.com/company/news-a-events/74-des-in-1-day.html

[7]http://archive.wired.com/politics/security/commentary/securitymatters/2007/11/securitymatters_1115

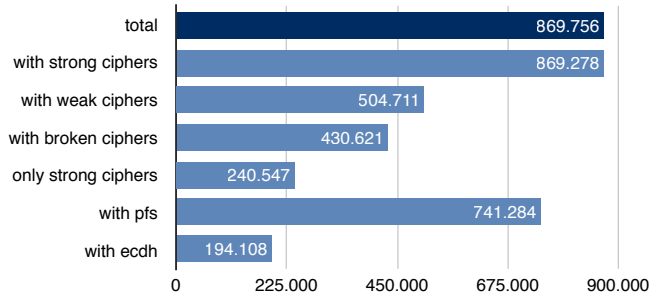

Fig. 4. Use of cipher suites.

Name	Usage	Share
DH_anon_AES_256_CBC_SHA	426,848	49.08%
DH_anon_3DES_EDE_CBC_SHA	426,835	49.08%
DH_anon_AES_128_CBC_SHA	426,779	49.07%
DH_anon_RC4_128_MD5	424,526	48.81%
RSA_DES_CBC_SHA	416,324	47.87%
RSA_EXPORT_RC4_40_MD5	410,627	47.21%
RSA_EXPORT_RC2_CBC_40_MD5	409,893	47.13%
RSA_EXPORT_DES40_CBC_SHA	407,480	46.85%
DHE_RSA_DES_CBC_SHA	382,903	44.02%
DHE_RSA_EXPORT_DES40_CBC_SHA	368,247	42.34%

Fig. 5. Shares of weak and broken cipher suites.

In Figure 5, we present the top ten most used weak and broken cipher suites. Many servers accept weak cipher suites without the need to. Looking at the FREAK attack[8] published in March 2015, more than half of the server' connections can be downgraded to weak or broken cipher suites.

D. TLS Certificates

We used the nmap *ssl-cert* script[9] to retrieve the certificates used by the servers. This script tries to perform a TLS handshake with the given server/port and saves the Privacy-enhanced Electronic Mail (PEM) certificate in the scan report. In addition, it parses the fields of the certificate and stores them in the report. The script also supports the STARTTLS command for an active SMTP session.

The validity of a certificate is based on three major properties: issuance by a valid CA, time period of validity and a matching Common Name (CN). In addition, we investigate other issues of certificates, such as multiple uses of the key pair or short private keys. Lastly, all retrieved certificates are categorized using these properties.

1) Multiple Use of Key Pairs: X.509 certificates are uniquely identified by their fingerprint using a MD5 or the *Secure Hash Algorithm 1* (SHA1) sum; current practice is to use a SHA256 hash nowadays. Although 656,295 hosts offer SMTP services in our data set, we only found 218,239 unique certificates. The reason is the presence of shared hosters using a single wildcard certificate for their hosts. This may not be a problem unless any user is able to retrieve the corresponding private key.

[8]https://freakattack.com

[9]https://nmap.org/nsedoc/scripts/ssl-cert.html

Type	Count	Share
not yet valid	161	0.07 %
expired	56,078	25.68 %
valid	161,993	74.20 %
never valid	99	0.05 %

Fig. 6. Time period of validity of the retrieved certificates.

Type	Count	Share	is_ca	is_server
server	79,832	36.58 %	0	1
ca	705	0.32 %	1	0
self-signed	137,694	63.09 %	1	1
invalid	1	0.01 %	0	0

Fig. 7. Self-signed, server and CA certificates.

Another group of repeatedly used certificates are the certificates delivered with the servers' operating systems. Those are recognizable by their subject's CN, which often contains the operating system name, phrases like localhost or a domain ending in **.local*. These configurations have to be considered insecure, because multiple users have access to the private key and therefore can decipher the TLS connections of other users.

We also found private keys that are used multiple times in different certificates. At a first glance, this does not involve a decrease of security. However, if the private key is lost, one has to revoke not only one, but all certificates issued for this private key. There is no acceptable reason for a system administrator to use the same key for multiple certificates.

2) Time Period of Validity: To investigate the time of validity, we categorized the certificates into four groups, as outlined in Figure 6. A certificate has two timestamps, defining the time period during which the certificate is valid. 25.80% of the certificates were not valid. We suspect these are hosts that are not maintained anymore. Although expired certificates may not be a security problem, they are an indicator that email is treated with a low priority. Expired certificates prevent secure communication with these servers if strict certificate validation is turned on at the email sender's side. There are 99 certificates which were never valid, since the *not-after* date of use was lower than or equal to the *not-before* date of use.

3) Certificate Issuance: We analyzed who issued and signed the obtained certificates. Figure 7 shows that 63.09% of the certificates were self-signed certificates, i.e., certificates that are signed by the same entity whose identity they certify, by signing with their own private key. We did not look at them in more detail, since the signature is not meaningful in these certificates, and they clearly represent a security issue.

To build the *chain of trust* for the non-self-signed certificates, we used the Ubuntu 14.04 trusted root certificates as a trust anchor. We were able to reach 24,641 certificates by building a trust chain with the root and the retrieved intermediate certificates. This set of certificates corresponds to 11.29% of the 218,239 found certificates and can be considered trustworthy according to the signature.

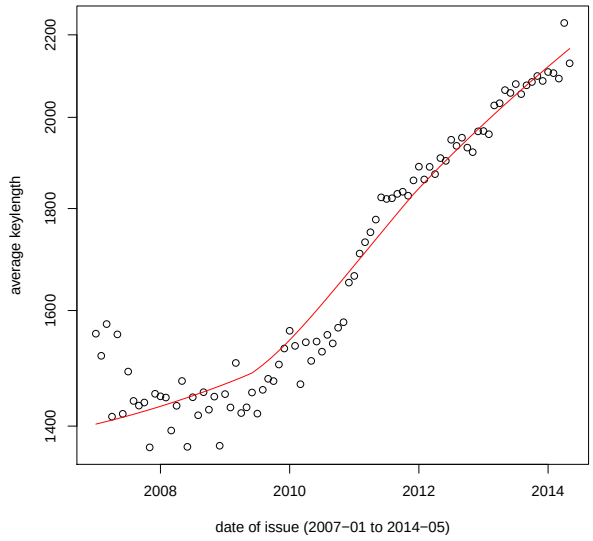

Fig. 8. Change of the average key lengths over time.

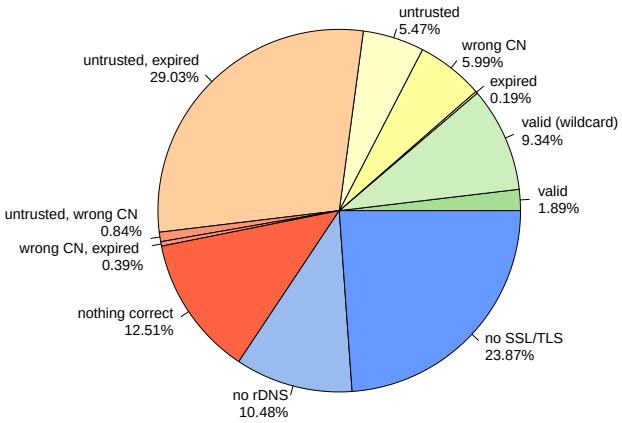

Fig. 9. Categorization of services using the main security properties.

4) Key Lengths: Figure 8 shows the growth of key lengths in relation to the dates of issuance of their certificates. It is apparent that the key lengths are growing steadily, using an average of more than 2,230 bits at the time of our scans. The average lowest key length stems from 2007 with 1366 bits. It is remarkable that a 1023-bit RSA number has been factored in May 2007[10], and that key lengths will become an issue when more powerful hardware becomes available.

5) Categorization: To summarize our findings regarding certificates, we use the three main validity properties to form eight disjoint categories of certificates. Since not every server has reverse DNS entries, we were not able to get the names of 10.48% of the servers. 23.87% did not offer TLS. Figure 9 indicates that only 11.23% of all scanned email services had valid certificates in all concerns. Considering the certificate subject's CN, we can split up the category of valid certificates further. A certificate is not only accepted as valid, if the CN matches the domain name, but if it has a wildcard pattern in the form of **.domain.tld* matching the domain name (e.g., mail.domain.tld). Using this categorization, only 1.89% of the email services use a certificate only valid for this specific host.

[10]http://phys.org/news98962171.html

Fig. 10. Root certificates in relation to their signed server certificates.

	Count	Share	Common Name
1	7,085	28.58 %	StartCom Certification Authority
2	4,309	17.38 %	AddTrust External CA Root
3	4,278	17.26 %	GeoTrust Global CA
4	2,632	10.62 %	thawte Primary Root CA
5	2,079	8.39 %	GlobalSign Root CA
6	1,243	5.01 %	DFN-Verein PCA Global - G01
7	640	2.58 %	UTN-USERFirst-Hardware
8	515	2.08 %	COMODO Certification Authority
9	285	1.15 %	Go Daddy Root Certificate Authority
10	279	1.13 %	Deutsche Telekom Root CA 2

Fig. 11. Top 10: Most popular CAs, measured by their issued certificates.

E. Certificate Authorities

A widely criticized problem of the TLS trust model are CAs. Over the last years, more and more CAs are trusted by software vendors. Figure 10 outlines the obtained trusted root CAs in relation to their signed server certificates of our test set. The ten most popular CAs shown in Figure 11 sign 94.16% of the server certificates. To have 99% coverage, one needs to trust the 23 most popular CAs. The idea that "if a CA can sign for one domain, it can sign for any domain" leads to a loss of security. Examples such as the DigiNotar hack[11] or the TurkTrust incident[12] show that this is a problem of practical relevance. Since the default configuration remains unchanged in many settings, the operating system and application vendors should act more responsibly in this respect.

F. CA Topologies

To check the validity of a certificate, a user builds a so called chain of trust. In addition to its own certificate, the server

[11]https://www.eff.org/deeplinks/2011/08/iranian-man-middle-attack-against-google

[12]http://web.archive.org/web/20130926134541/http://turktrust.com.tr/en/kamuoyu-aciklamasi-en.2.html

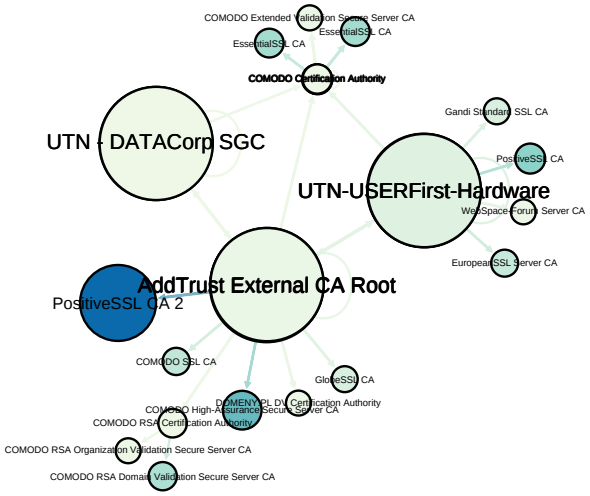

Fig. 12. CA Topology of the Comodo CA.

can deliver additional intermediate certificates. The client then builds a chain of trust as follows: The issuer of a non-root certificate is identified by the issuer's properties of this certificate, and the signature is obtained by using the issuer's private key. To get a valid chain of trust, the last certificate needs to be in the list of trusted root certificates of the client. If no path from the server certificate to any of the trusted root certificates can be found, the certificate is considered as untrusted. This is often the case when a private CA for in-house deployment or self-signed certificates are used.

Using OpenSSL[13], we built a key-value database representing the graph of signatures and the hierarchy of CAs. Two CAs showed peculiarities. First, StartSSL uses two almost identical root certificates, only differing in the date of issuance and the serial number. We used the trusted root certificate set as shipped by default with Ubuntu 14.04. Second, as shown in Figure 12, the Comodo CA uses more than just the CA and intermediate certificates. Comodo cross-certifies its root keys using the other root certificates. This results in multiple signed key pairs and therefore in multiple certificates for every key. If one of those root certificates is revoked, there are signing chains starting at other root certificates available. Comodo enables a kind of fail-safe strategy for their customers. Even if one root certificate is removed from the set of trusted certificates, the customers' servers have other intermediate certificates to build a trusted chain. Although cross-certification is legitimate according to the X.509 standard, it implies a less secure CA, if the private key is lost.

G. Email Provider Strategies

Finally, we investigated the connections of the providers' email servers to other email servers. For this purpose, we registered email accounts at various free email providers popular in Germany and sent emails to our prepared test email server. This allowed us to investigate the connection details of the providers' outgoing connections. For this part of our study, we used Google Mail[14], web.de[15], GMX[16], Freenet[17], Yahoo Mail[18], Microsoft Outlook.com/Hotmail/Live[19], Apple iCloud Mail[20] and T-Online[21].

The idea of this investigation was to find situations that a potential attacker with access to the connection between two email servers can exploit to read email. The attacker could just passively eavesdrop on the connection or actively perform a Man-in-the-Middle attack. To simulate an attacker that attempts to change cryptographic properties, we changed the servers' TLS properties to test the behavior of the email providers. In particular, we examined the following settings:

- Expired certificate (not security critical)
- 512-bit RSA key certificate
- Certificate with wrong CN
- Only anonymous ciphers (no certificate)
- Only weak or broken ciphers

We found different security settings and thus more and less difficult situations to attack the email providers. All providers forward email to servers with invalid certificates and still communicate plaintext if a server does not support TLS at all. With one exception (Google), we were even able to trick the email providers to send their data without any encryption. The results of the experiments are summarized in Figure 13.

1) Invalid Certificates: The main security properties of a certificate are a trusted issuer, the period of validity and the domain name matching the common name. All email providers ignored these features, completed the TLS handshake and submitted the data. A Man-in-the-Middle could provide this kind of certificate without any effort and thus see the data. We used multiple certificates to check these properties. None of the providers complained about the often changing certificates, implying that certificate pinning is not used at all or at least not automatically for new or uncommon servers.

2) Default Cipher Suites: All examined email providers use strong cipher suites in their connections, with one exception. Freenet offered, in addition to several strong cipher suites, three cipher suites with an anonymous key exchange. The anonymous key exchange methods do not use a certificate and therefore do not provide any authentication of the server. An attacker would love this situation, since his or her effort to read the email data is minimized.

3) Short RSA Keys: To examine the use of short cryptographic keys, we created a certificate with a 512-bit RSA key. The email providers acted in different ways: Web.de, GMX, Yahoo and iCloud accepted the certificate, completed the TLS handshake and transmitted the email data. Google Mail and Microsoft Outlook/Hotmail rejected the certificate and continued with multiple unsuccessful retries. Freenet and

[13]https://www.openssl.org

[14]https://mail.google.com
[15]https://www.web.de
[16]https://www.gmx.de
[17]https://email.freenet.de
[18]https://mail.yahoo.com
[19]https://login.live.com
[20]https://www.icloud.com
[21]https://email.t-online.de

Provider	Untrusted Certificate	512-bit RSA Key	Anonymous Ciphers	Weak Ciphers	No STARTTLS
Google Mail	encrypted	no delivery	no delivery	no delivery	unencrypted
Web.de	encrypted	encrypted	unencrypted	unencrypted	unencrypted
GMX	encrypted	encrypted	unencrypted	unencrypted	unencrypted
Freenet	encrypted	unencrypted	encrypted	unencrypted	unencrypted
Yahoo	encrypted	encrypted	unencrypted	unencrypted	unencrypted
Outlook/Hotmail	encrypted	no delivery	unencrypted	unencrypted	unencrypted
iCloud	encrypted	encrypted	unencrypted	unencrypted	unencrypted
T-Online	encrypted	unencrypted	unencrypted	unencrypted	unencrypted

Fig. 13. Email provider strategies for connections to other email servers.

T-Online rejected the certificate and closed the connection. Immediately afterwards, they opened another connection to transfer the email without using any encryption.

4) Weak Cipher Suites: We also checked the behavior of the email servers when our server only supported weak ciphers. As mentioned above, the providers offer only strong ciphers in their client hello message. Therefore, the TLS handshake could not be completed. However, all providers except Google Mail reconnect after this failed TLS handhake and transfer the email without using encryption.

5) No STARTTLS: In the last test, we completely disabled the STARTTLS command on port 25. After the SMTP session is initiated, an email server offers a list of features it supports. STARTTLS is a means to encrypt the session after it is established. All tested email providers continued with the non-encrypted transfer of an email when this feature was disabled. Therefore, an attacker only needs to modify the SMTP command list provided by the server. Modern routing and firewall hardware, e.g., as provided by Cisco, has exactly this mechanism built in to inspect mail traffic[22].

6) Implications: Considering the results presented above, an interesting situation happens: If an attacker is able to inject a few TCP packets into the connection, the TLS handshake is aborted, and the email provider reconnects and transfers the email without any transport security. Google seems to use a non-secured TLS policy in the sense that Google transfers email using TLS only, but does not insist on correct certificates.

Over the last years, several methods to avoid downgrade attacks in web browsers and other applications have been developed. Initiatives such as the SSL Observatory[23] and tools such as HTTPS Everywhere[24] try to protect users against leakage of their private data. Email providers need to catch up and deploy similar security standards in their infrastructures.

IV. Advice for Email Providers

The SMTP TLS landscape has major flaws across most email providers. In this section, we give some advice to email providers for obtaining secure configurations for their servers. Our advice is meant to harden a TLS server at the cost of compatibility, which we think is one reason for weak configurations. In addition to the relevant TLS factors, we suggest measures to prevent simple downgrade attacks.

We give the following recommendations to network administrators and email providers to harden their TLS configurations:

1) Update the TLS stack frequently.
2) Use TLS 1.0 (or higher) instead of SSL 3.0 (or lower).
3) Support strong cipher suites only.
4) Perform smart TLS certificate checks.
   - Time of validity
   - CA issuance
   - Key lengths
5) Accept/use TLS enabled email transfer only.

The first advice is that the TLS stack of the system needs to be updated frequently to keep the system secure.

The second advice is that currently only TLS 1.0 and the higher TLS versions can be considered secure, since SSL 3.0 is affected by the POODLE attack.

The third advice is to support strong cipher suites only. Older cipher suites with MD5 hashing or 3DES encryption should not be used, since they are more likely to be broken in the near future. As with security updates for software and the TLS stack, email providers should pay close attention to the effects of new attacks on the list of secure cipher suites and remove insecure cipher suites accordingly[25].

The fourth advice is to perform appropriate certificate checks regarding period of validity, CA issuer and key lengths and use a certificate pinning mechanism to detect bogus certificates[26]. A list of the trusted certificates for every server must be maintained. When a TLS handshake is in progress, it can be checked whether the certificate has changed without revocation or whether it has expired.

The final advice is to either at least warn users and let them decide to use a potentially insecure connection, as done in web browsers, or decline non-encrypted email transfer, rather than just transferring email without encryption. One possibility to achieve this would be to set a special email header in the *Mail User Agent* (MUA) indicating that an email should only be delivered over encrypted links, with strictly validated peers or just anybody and any connection. Email should only be delivered over links with an equal or higher security rating than the one provided in the header. Although

[22] https://www.cisco.com/web/about/security/intelligence/asa_esmtp_starttls.html

[23] https://www.eff.org/observatory

[24] https://www.eff.org/de/https-everywhere

[25] https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/

[26] https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning

there is no guarantee that the server respects the user's wishes, if the big email providers followed this proposal and enough users set a flag for strict validation and encryption, smaller providers would be forced to use valid certificates and strong ciphers. Thus, unencrypted communication could possibly be eliminated after some time. We also recommend to consider Ristic's TLS deployment best practices[27] for more details on TLS configuration hardening.

Furthermore, email providers are mostly left alone when it comes to tools for securing and testing their setups. Qualys provides a service for web server administrators to check their setup for common misconfigurations[28]. This kind of service would be of great benefit for email providers. Standardized test infrastructures could also be utilized by central instances like CERT, the German BSI or large websites with lots of email traffic, such as Facebook, Youtube or Github, to automatically generate emails to MTA owners in case of security concerns. This, of course, means that every entity must support the common mailbox names as suggested by RFC2142 [26].

V. Conclusion

In this paper, we have presented an empirical study of the security properties of SMTP over TLS within the German IP address space. We have shown that even though many email servers support strong cipher suites, weak or broken cipher suites are still present. Furthermore, only few certificates provided by the email servers are valid (about 11%). The behavior of email providers differs significantly with respect to handling various TLS configurations. Based on our results, we have given practical advice on securing SMTP setups to avoid some of the identified issues. Unfortunately, an ultimate solution offering perfect security is simply not possible with the current PKI landscape. This topic is often discussed in the context of HTTP(S), but with email the problems may be even harder to fix, since decision making happens automatically without user interaction.

There are several areas for future work. For example, appropriate best practices are needed to ensure transport layer security of email traffic. A simple plain-text fallback as used by many providers significantly reduces the protection level of SMTP communication. Furthermore, validation and verification of peers and their certificates must be simplified. Finally, staying up to date with broken, weak and strong cipher suites is another challenging task, since there is no central entity giving advice for regular updates.

References

[1] J. Postel, "RFC 821: Simple Mail Transfer Protocol," http://tools.ietf.org/html/rfc821, August 1982.

[2] J. Klensin, "RFC 5321: Simple Mail Transfer Protocol (SMTP)," http://tools.ietf.org/html/rfc5321, October 2008.

[3] S. Garfinkel, *PGP: Pretty Good Privacy*. O'Reilly Media, 1995.

[4] P. Hoffman, "RFC 3207: SMTP Service Extension for Secure SMTP over Transport Layer Security," http://ietf.org/rfc/rfc3207.txt, 2002.

[5] N. J. Al Fardan and K. G. Paterson, "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols," in *IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 526–540.

[6] H. K. Lee, T. Malkin, and E. Nahum, "Cryptographic Strength of SSL/TLS Servers: Current and Recent Practices," in *7th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2007, pp. 83–92.

[7] A. Klein, "Attacks on the RC4 Stream Cipher," *Designs, Codes and Cryptography*, vol. 48, no. 3, pp. 269–286, 2008.

[8] M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D. Molnar, D. A. Osvik, and B. De Weger, "Short Chosen-prefix Collisions for MD5 and the Creation of a Rogue CA Certificate," in *Advances in Cryptology-CRYPTO 2009*. Springer, 2009, pp. 55–69.

[9] P. Eckersley and J. Burns, "Is the SSLiverse a Safe Place?" in *Chaos Communication Congress*, https://lb1.eff.org/files/ccc2010.pdf, 2010.

[10] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL Landscape: A Thorough Analysis of the X.509 PKI Using Active and Passive Measurements," in *2011 ACM SIGCOMM Conference on Internet Measurement*. ACM, November 2011, pp. 427–444.

[11] I. Ristic and M. Small, "A Study of What Really Breaks SSL," *Hack in the Box*, vol. http://blog.ivanristic.com/Qualys_SSL_Labs-A_Study_of_Really_Breaks_SSL-HITB_Amsterdam_2011.pdf, May 2011.

[12] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS Certificate Ecosystem," in *2013 Conference on Internet Measurement*. ACM, October 2013, pp. 291–304.

[13] S. Fahl, M. Harbach, T. Muders, M. Smith, L. Baumgärtner, and B. Freisleben, "Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security," in *2012 ACM Conference on Computer and Communications Security*. ACM, 2012, pp. 50–61.

[14] F. Giesen, F. Kohlar, and D. Stebila, "On the Security of TLS Renegotiation," in *ACM Conference on Computer & Communications Security*. ACM, 2013, pp. 387–398.

[15] P. Szalachowski, S. Matsumoto, and A. Perrig, "PoliCert: Secure and Flexible TLS Certificate Management," in *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 406–417.

[16] B. Laurie, A. Langley, and E. Kasper, "RFC 6962: Certificate Transparency," 2013.

[17] M. D. Ryan, "Enhanced Certificate Transparency and End-to-end Encrypted Mail," *Proceedings of NDSS 2014, The Internet Society*, 2014.

[18] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements," in *IEEE Symposium on Security and Privacy*. IEEE, 2013.

[19] L. S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing Forged SSL Certificates in the Wild," in *IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 83–97.

[20] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov, "Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations," in *IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 114–129.

[21] A. Bates, J. Pletcher, T. Nichols, B. Hollembaek, D. Tian, K. R. Butler, and A. Alkhelaifi, "Securing SSL Certificate Verification through Dynamic Linking," in *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 394–405.

[22] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," in *Selected Areas in Cryptography*. Springer, 2001, pp. 1–24.

[23] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions," in *Advances in Cryptology–EUROCRYPT*. Springer, 2005, pp. 19–35.

[24] A. Popov, "Prohibiting rc4 cipher suites," *Computer Science*, vol. 2355, pp. 152–164, 2015.

[25] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and Authenticated Key Exchanges," *Design, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, Jun. 1992.

[26] D. Crocker, "RFC 2142: Mailbox Names for Common Services, Roles and Functions," http://www.ietf.org/rfc/rfc2142.txt, 1997.

[27] https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices.pdf

[28] https://www.ssllabs.com/