No Need for Black Chambers: Testing TLS in the E-mail Ecosystem at Large

Wilfried Mayer, Aaron Zauner, Martin Schmiedecker
SBA Research, Austria
{*wmayer*|*azauner*|*mschmiedecker*}*@sba-research.org*

Markus Huber
FH St. Pölten, Austria
markus.huber@fhstp.ac.at

Abstract—**TLS is the most widely used cryptographic protocol on the Internet today. While multiple recent studies focused on its use in HTTPS and the adoption rate of additional security measures over time, the usage of TLS in e-mail-related protocols is still lacking detailed insights. End-to-end encryption mechanisms like PGP are seldomly used, and as such today's confidentiality in the e-mail ecosystem is based entirely on the encryption of the transport layer. However, a large fraction of e-mails is still transmitted unencrypted, which is highly disproportionate with the sensitive nature of e-mail communication content. A well-positioned attacker may be able to intercept plaintext communication content as well as communication metadata passively and at ease.**

We are the first to collect and analyze the complete state of today's e-mail-related TLS configuration, for the entire IPv4 address range. Our methodology is based on commodity hardware and open-source software, and we draw a comprehensive picture of the current state of security mechanisms on the transport layer for e-mail by scanning cipher suite support which was previously considered impossible due to numerous constraints. We collected and scanned a massive dataset of 20 million IP/port combinations of all e-mail-related protocols (SMTP, POP3, IMAP). Over a time span of approx. three months we conducted more than 10 billion TLS handshakes. Additionally, we show that securing server-to-server communication using e.g. SMTP is inherently more difficult than securing client-to-server communication, and that while the overall trend points in the right direction there are still many steps needed towards secure e-mail.

I. Introduction

E-mail has become one of the most fundamental and important services of the Internet and is used by billions of users every day. While the implementations and the underlying protocols have (for the most part) endured the test of time, there is hardly any way for users to assess the security and confidentiality of e-mails in transit. Even if the client-to-server connection is secured, server-to-server communication relies on plaintext communication as fallback leaving a large fraction of transmitted e-mails unencrypted and unauthenticated. This is a serious limitation, as they are passively observable along the transmission path and users are mostly left in the dark whether their e-mails will be transmitted between servers in plaintext or not. While the recently proposed use of opportunistic encryption [17] is good from the privacy viewpoint, it offers no protection against active attackers. Recent data from Google [32] shows that almost three years after the revelations by Edward Snowden approximately 31% of inbound and 17% of outbound e-mails are still transmitted in plaintext. While PGP would protect e-mails from end to end, its adoption is marginal.

If somehow the usage of TLS[1] between all links during e-mail transit could be enforced, this would not only provide authentication, but also confidentiality thanks to the use of encryption. However, due to the decentralized nature of e-mail there is no way to upgrade all servers or somehow enforce link encryption. In fact, we are still just beginning to understand the consequences of large fractions of e-mails being transmitted unencrypted.

In this paper we present a methodology to analyze the overall cryptographic primitives for e-mail in-depth and at scale. While many of the problems of e-mail are connected to the design of popular e-mail protocols, we quantify and evaluate the adoption of well-known security primitives on a large scale, namely the support of TLS for e-mail protocols on the Internet on a cipher suite base. Previous large-scale studies on usage and prevalence of TLS focused either on the protocol itself or its usage and implications for secure web browsing using HTTPS. Other recent studies focused on passive data and active scanning of a small subset of possible cryptographic primitives for the e-mail ecosystem. We are the first to present our findings on TLS usage and deployment in e-mail protocols using active probing on cipher suite level, with more than ten billion TLS handshakes conducted. In particular, the contributions of this paper are as follows:

- We conducted IPv4-wide scans on a number of important ports for e-mail regarding TLS, providing a holistic measurement on e-mail transport layer security.
- We compare the results with previous work on HTTPS and find that TLS in e-mail is inherently less secure.
- We analyze the quality and distribution of supported ciphers, arguing that it's still a long way to secure e-mail transport.

The remainder of this paper is organized as follows: Section II gives a brief introduction to TLS and how it is used in e-mail protocols. In Section III we describe our scanning methodology as well as the different inputs we used for seeding our scanning activity. Section IV presents our results of multiple months of scanning activity, while Section V discusses our findings, interprets the results and compares

[1] In this paper we use the term TLS to refer to all incarnations of the TLS and SSL protocols, if not specified otherwise.

them with other protocols that use TLS. Section VI discusses mitigation strategies and methods how the overall security for e-mail can be increased. Section VII discusses related work, before we conclude in Section VIII.

II. Background

Today Transport Layer Security (TLS) serves as the underlying foundation for securing many of the most widely used protocols on the Internet, including HTTP and e-mail. TLS provides confidentiality, integrity and authenticity of data transferred on the wire. A general overview of TLS, and more specifically on its usage in HTTPS, is given in [56], [12]. In recent years, the TLS protocol and specific software implementations of the protocol have been under close scrutiny by security researchers, and numerous attacks have been discovered: BEAST [18], POODLE [51], CRIME [57], BREACH [31] or Lucky13 [3]. They exploit the incorrect usage of block cipher modes, compression and shortcomings in specific versions of TLS. RFC 7457 [40] provides a good overview of known attacks against TLS. Heartbleed [22] on the other hand was a devastating bug in OpenSSL which allowed attackers to read content from the server process heap memory, including login credentials and the private key used by the server. MS14-066 [50] (dubbed "Winshock") allowed a remote attacker code execution on a system by supplying a malicious ECDSA certificate during the handshake. Other papers analyzed, at large, the unsound use of cryptographic primitives like the Diffie-Hellman key exchange parameters [2] or problematic issues with RC4 [4] or RSA in TLS [60], [35], [48]. Lastly it was shown that specific implementations were vulnerable to numerous timing-based side channel attacks [49]. Other publications had a special focus on the usage of TLS on smartphone apps as well as other non-browser software [26], [30], [27], [33]. Formal verification of TLS is another research area which received an increased momentum [6], [7] as well as the verified implementation of the TLS protocol [10], [8], [44].

The e-mail ecosystem makes use of numerous ports and protocols for transmission: SMTP is used for transmitting e-mails between servers and runs usually over TCP port 25 [45], [37]. Clients use either the IMAP4 or POP3 protocols to retrieve e-mails from a message delivery server, and SMTP for e-mail submission. IMAP uses TCP ports 143 or 993 [53], [54], POP3 uses TCP ports 110 or 995 [13], [54], and SMTP Message Submission runs on TCP port 25, 587 [29] or 465. Although IANA revoked the usage of port 465 for SMTPS in 1998, it is still heavily used for this purpose. All client-facing protocols are available on two different TCP ports, because there are two distinct ways to establish a TLS session: implicit TLS where the connection is established with a TLS handshake, and in-band upgrade of the used plaintext protocol via STARTTLS. An overview of commonly ports is given in Table I. The complete path an e-mail takes is shown in Figure 1: submitted by the client to the mail submission agent (MSA), the e-mail passes an arbitrary number of relaying mail transfer agent (MTA) servers before it finally reaches the recipient's MTA. Note that the user is only able use or enforce TLS on the edge of this path – she has no control or knowledge whether the MSA or mail delivery agent (MDA) communicate using TLS, or if the MTAs in between make use of it.

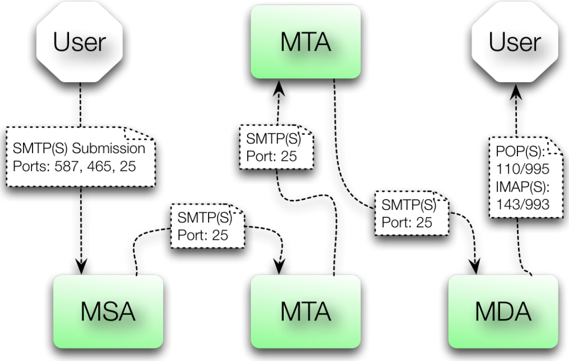

Fig. 1. Typical e-mail flow on the Internet from submission to delivery via multiple hops. *MSA: Mail Submission Agent, MTA: Mail Transfer Agent, MDA: Mail Delivery Agent.*

A. *TLS Handshake and STARTTLS*

To initiate a TLS-encrypted connection, a TLS handshake is necessary. A "PreMasterSecret" is established with a `ClientKeyExchange` message which is then encrypted using the public key of the server certificate. Client and Server use the "PreMasterSecret" together with random numbers (usually a timestamp and random bytes) to establish a common "MasterSecret". During this handshake a specific cipher suite is negotiated with `ChangeCipherSpec` messages. A cipher suite is represented by a two-byte value and determines the cryptographic primitives to be used in the connection [14], [11]. It consists of methods for the key exchange, encryption and message authentication. For example `TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA` (IANA-assigned values: `0x00, 0x16`) is the cipher suite combining an ephemeral Diffie-Hellman key exchange with RSA authentication, 3DES in EDE-CBC mode for encryption and SHA-1 for message authentication (HMAC). Historically, TLS implementations used their own nomenclature for cipher suites as IANA only standardized TLS parameters in 2006 with the introduction of TLS 1.1.

The flow of STARTTLS messages is specific to the underlying protocol to be secured and not a general way for in-band upgrade. Different RFCs [37], [54] specify these details.

The term "opportunistic encryption" [17] is commonly used when referring to TLS in e-mail protocols. Opportunistic encryption means that connections are usually not authenticated (certificates not validated) and vulnerable to an active attacker, but enable more wide-spread use and deployment of security protocols like TLS and efficiently secure against passive adversaries.

III. Methodology

In this section, we describe the source of our input data, the methodology and tools we used, and our considerations regarding the impact of our scans. In the beginning we started with the following requirements: the performance of our methodology should be sufficient, i.e. running on commodity hardware with a fast Internet connection. Our goal was to scan more than a million hosts in less than a week, with low additional load for the target systems. Handling network connections is what these systems do (by design), and we took specific precautions that our connections would have no negative impact on the performance or operations. Furthermore our scans should be complete, i.e. all cipher suites should be covered as well as all the important TLS parameters should be collected, i.e. certificates, the certificate chain up to the root CA which signed the certificate, RSA primes and DH-group. For our evaluation we wanted to cover all seven ports which are used by default for e-mail.

A. Input Dataset

The source of our scan consists of a list of all input IP addresses together with the application-specific TCP port. To only scan IPs with running services, we used existing and publicly available service discovery datasets from scans.io[2] as a source for our cipher suite scans. These datasets consist of all addressable IPv4 hosts which completed a full TLS handshake at a given time. We used results originating from March to April 2015 and tightened the set by only selecting hosts which completed the handshake without any error. This resulted in 18,430,143 IP/port combinations which were used for our scan. In June 2015 we verified these IP/port combinations with the results of banner grabbing and certificate collection scans performed with masscan[3] by our own team. This scan resulted in 23,367,809 possible IP/port combinations. Nearly 70% of these combinations were already included in our initial input data. To allow observations based on the domain of a host, we also scanned MX records for all domains listed in the Alexa Top 1 Million ranking. We stored these records together with the resolved IP addresses to allow the interpretation of our scan results with the popularity of a domain and to combine them with the Google Transparency Report on safer e-mail [32].

B. Scanning the Cipher Suite Support

The specification of the TLS handshake protocol [16] requires that the client specifies a set of supported cipher suites in the `ClientHello` handshake message and the server then picks one. The response is either a `ServerHello` or `Alert` message which can be interpreted as an accepted or rejected cipher suite. We used the tool *sslyze*[4] to perform a scan during which every cipher suite is tested individually. Seven different cipher suites were tested for SSLv2 and 136 cipher suites for SSLv3, TLSv1, TLSv1.1 and TLSv1.2 each. This deliberately also includes combinations of TLS version and cipher suite which are not specified in the corresponding RFCs and thus violate the standard, e.g. SSLv3 with ECDHE cipher suites. In total we established at least 551 connections per IP to fully test the supported cipher suites for each IP/port combination. We stored the acceptance status as either *accepted*, *rejected*, *preferred* or *error*. If the handshake was successful, we also stored key exchange parameters for the Diffie-Hellman key exchanges. In the case that the TLS connections were not successful, we stored the given alert message or error reason. Additionally we stored the complete certificate chain per IP.

To optimize *sslyze* for our needs we made certain changes. *sslyze* was written as a command line tool. To use it efficiently with millions of possible targets we embedded the tool into a distributed environment and integrated it with a queuing system (AMQP). As such, our methodology can be easily used in a distributed environment, with many simultaneous hosts that perform scanning activities and work together on an input list of IPs to be scanned. We only used one host with one IP for the scans (Ubuntu Linux 14.04, Intel Core i7, 4x 3.10 GHz, 32GB DDR3 RAM, 100MBit/s Network Bandwidth). We consider our results to be a lower bound for assessing the usage of TLS in e-mail, and the scan time can easily be decreased by adding more hosts or bandwidth.

sslyze scans all cipher suites on a per host/port basis and thus fulfills our requirements listed above. All cipher suite tests are queued consecutive within one process. It then maximizes parallelization by using 15 threads per process and by using more than one process. This intense scan behavior may be seen as a Denial-of-Service attack and may disrupt normal service behavior, because all connection attempts are started in a very short timeframe. To solve this issue, we had to spread the cipher suite tests over time. Also, the availability of hosts may change during a scan, which can increase the chance of incomplete results due to hosts that go offline during that timeframe. We therefore serialized the behavior of the scan, spreading the total scan time to several minutes for one host/port, and increased the total number of scan processes. This yielded a good balance between minimizing the scanning impact on the target systems to be scanned – as not all connections are opened in direct succession – while completing the entire list of cipher suites to be scanned in reasonable time. With this methodology we were able to scan between 12,000 and 30,000 IP/port combinations per hour. Our bottleneck was identified to be the number of sockets provided by the operating system, in particular the time until a socket can be reused – spreading the load over multiple machines can easily help to overcome this obstacle. After scanning was completed for each port, we transformed the output and analyzed it using Google BigQuery.

C. Ethical Issues

Similar to related work on active probing on the Internet, we took specific measures to inform not only about our intentions, but also about our scanning methodology and how people can be excluded from future scans. On our scanning host

[2] https://scans.io

[3] https://github.com/robertdavidgraham/masscan

[4] https://github.com/nabla-c0d3/sslyze

we served a simple webpage to inform about our scans, both over HTTP and HTTPS. We furthermore created a specific abuse address as single point of contact and included it in the WHOIS abuse database. Lastly we set a reverse DNS entry to identify us as a scanning host and to point to the webpage. We were in direct contact with our upstream ISP's network administrators to answer complaints and inquiries and made sure that all requests were answered in a timely manner. To prevent connection- or resource-wise Denial of Service for the hosts to be scanned, we tested our scanning methodology on devices with constrained hardware, in addition to spreading the number of connections over time. These devices were limited in the overall computation resources. We chose them to reflect, appropriately, the smallest possible devices on the Internet that could be used to run a full e-mail stack, similar to a very small cloud instance (VPS) or an embedded system (router or appliances). We neither took specific measures to prevent our connections from being logged nor to stay below the radar of vigilant system administrators.

IV. Evaluation

This section presents the data we obtained through our scans. In total we conducted 20,270,768 scans over seven different TCP ports (Table I). The complete data set will be published upon publication. The scans were performed between April and August 2015. Thereof, 18,381,936 were valid results, meaning that at least one TLS handshake was completed. 1,888,832 of the scans were invalid with no TLS session established at all (77.9% *could not connect (time-out)*, 17.5% *connection rejected*, 3% *SMTP EHLO rejected*, 1.6% *STARTTLS rejected*). Since we established 551 TLS handshakes with different cipher suites and TLS versions per host and port, this resulted in more than 10.2 billion TLS handshakes conducted. 89.78% of these handshakes got rejected, 8.26% accepted and 1.95% resulted in an error. The high percentage of rejected ciphers is due to the fact that e-mail servers typically do not support all TLS versions available between SSLv2 and TLS 1.2, and not every cipher suite for each supported TLS version.

A. Supported Cipher Suites

Table II shows the percentage of unique hosts that support a specific TLS protocol version. A host supports a TLS version if at least one handshake with any chosen cipher suite for this version is accepted by the server. SMTP is remarkable in that it has a relatively large acceptance of SSLv2 and SSLv3. The support of SSLv2 for client-to-server connections is generally low, except for e-mail submission on port 587 where 15% of all tested IPs accept cipher suites from SSLv2. TLSv1 is with more than 90% clearly the most supported version for all studied e-mail protocols. This means that 9 out of 10 e-mail servers that speak TLS can use TLSv1, whereas approximately 50–66% accept the newer versions TLSv1.1 and TLSv1.2. It also shows some TLS version support for combinations of interest, e.g., SSLv2 and SSLv3, without any support for TLSv1, TLSv1.1 and TLSv1.2 (which is close to non-existing) or modern configurations without support of SSLv2 or SSLv3 (deployed by up to one third of all hosts for e-mail delivery protocols). Only 8% of all SMTP hosts have this configuration. Hosts that exclusively use TLSv1.1 and TLSv1.2 are also close to non-existing.

In Table III we show the acceptance rate of selected cipher suites for TLSv1 (as it is by far the most supported version of TLS across all protocols). For many cipher suites we can identify a big gap in the acceptance rate of weak cipher suites for SMTP and client protocols. E.g., DES-CBC-SHA is only supported by 6% of all IMAP and POP hosts, but 75% of SMTP hosts. The percentage for submission lies at 29%. To better understand these results, we cluster the cryptographic primitives of all accepted cipher suites in the next subsection.

B. Cryptographic Primitives

Table IV compares the different percentages of key exchange methods used. Static RSA is supported by the majority of hosts. 15%–17% of POP3 and IMAP deployments use static RSA as the only key exchange algorithm. Just a few (7%–9%) of all POP3 and IMAP hosts offer ECDHE for key exchange, but close to 75% support DHE as key exchange mechanism. Both methods guarantee perfect forward secrecy. A high number of deployments accept weak anonymous Diffie-Hellman and export grade handshakes.

Table V presents the percentage of hosts that support an encryption method with a given key size. 3DES and AES in CBC mode are currently the most widely adopted encryption methods. Because of several weaknesses [4] the use of RC4 in TLS was prohibited in February 2015 [55]. We observe that it is still supported by 82.3% to 86.5% of all hosts. However, less than 0.9% exclusively support RC4. AES in GCM is supported by 49.2% to 63.3% hosts. Still 15%–18% of all hosts support export ciphers (DES-40, RC4-40, RC2-40) for POP3 and IMAP, 73%–75% for STMP and 28%–30% for submission.

C. Key Exchange Parameters

The recently discovered Logjam attack against TLS [2] was possible, because many servers employ weak Diffie-Hellman parameters. We analyzed used group sizes, since 512-bit and 768-bit groups are considered very weak. Common used primes are more valuable for precalculation attacks, so we tried to uncover some of them.

1) Weak Diffie-Hellman Parameters: We analyzed the use of different Diffie-Hellman parameters and thereafter categorized all cipher suites into two sets: cipher suites with export-restricted algorithms and all other cipher suites. Because the Diffie-Hellman key exchange for export cipher suites is restricted to 512 bit, almost all use this size. For SMTP, twelve hosts used a different group size, whereas 1,045,456 hosts used the 512-bit group. For non-export cipher suites, 1024-bit parameters are clearly most commonly used. Still, 6%–7% of POP3 and IMAP use 768-bit parameters. Weak DH parameters were deprecated in OpenSSL.[5] 2048-bit parameters are rarely

[5] https://www.openssl.org/blog/blog/2015/05/20/logjam-freak-upcoming-changes/

TABLE I
OVERVIEW OF OUR SCAN RESULTS

Port	Protocol	Source of Input	Timeframe	Total Scans	Valid Results	Invalid Results	Reoccured
25	SMTP with STARTTLS	scans.io UMich	05/05 – 05/20	1,825,657	1,558,796	266,861	1,611,051
110	POP3 with STARTTLS	scans.io UMich	05/20 – 06/04	4,123,819	3,553,201	570,618	3,572,513
143	IMAP with STARTTLS	scans.io UMich	06/04 – 06/06	859,470	764,211	95,259	765,511
465	SMTP over TLS	scans.io Rapid7	06/07 – 06/21	3,123,874	2,851,598	272,276	2,681,955
587	Submission with STARTTLS	masscan results	07/27 – 08/03	1,840,629	1,656,901	183,728	-
993	IMAP over TLS	scans.io UMich	04/27 – 05/05	4,254,564	4,047,385	207,179	3,897,348
995	POP3 over TLS	scans.io UMich	04/17 – 04/27	4,242,755	3,949,844	292,911	3,775,292
	Σ			20,270,768	18,381,936	1,888,832	

TABLE II
PROTOCOL VERSION SUPPORT

TLS version	25	110	143	465	587	993	995
SSLv2	41.7%	2.9%	3.7%	4.2%	15.0%	8.7%	9.3%
SSLv3	82.6%	35.3%	38.6%	36.1%	64.0%	53.7%	53.8%
TLSv1	91.6%	98.9%	98.9%	94.8%	92.4%	98.2%	97.9%
TLSv1.1	55.7%	50.5%	54.1%	65.6%	49.6%	57.2%	55.1%
TLSv1.2	56.1%	50.7%	54.2%	65.2%	48.8%	57.7%	55.6%
only SSLv2&3	0.2%	0.0%	0.0%	0.0%	0.0%	0.1%	0.1%
only TLSv1–1.2	8.0%	36.4%	37.2%	44.7%	17.9%	32.8%	31.9%
only TLSv1.1–1.2	0.1%	0.1%	0.1%	0.0%	0.5%	0.1%	0.1%

TABLE III
PERCENTAGE OF SELECTED CIPHER SUITE ACCEPTANCE FOR TLSV1

Cipher Suite Name (OpenSSL)	25	110	143	465	587	993	995
AES256-SHA	89.0%	91.5%	93.1%	92.8%	82.7%	89.4%	88.4%
CAMELLIA256-SHA	57.3%	54.4%	57.2%	65.3%	49.5%	55.3%	53.7%
DES-CBC-SHA	75.0%	5.9%	5.6%	6.7%	29.3%	6.1%	6.2%
DES-CBC3-SHA	89.4%	95.9%	97.1%	92.6%	90.8%	93.4%	93.1%
DHE-RSA-AES256-SHA	83.6%	72.0%	73.5%	85.4%	63.5%	72.5%	71.8%
IDEA-CBC-SHA	21.3%	45.0%	46.4%	46.0%	28.1%	45.3%	45.3%
SEED-SHA	52.8%	47.6%	49.8%	52.4%	35.3%	48.3%	47.2%
RC4-SHA	84.9%	85.1%	85.4%	81.6%	78.8%	81.7%	81.9%
ECDHE-RSA-AES256-SHA	39.8%	8.1%	8.4%	11.3%	24.1%	8.6%	7.0%
EXP-RC4-MD5	72.7%	17.9%	16.4%	5.2%	28.0%	14.1%	14.0%
EXP-DES-CBC-SHA	72.7%	17.6%	16.1%	5.2%	27.7%	14.0%	13.9%

TABLE IV
KEY EXCHANGE METHOD SUPPORT

	25	110	143	465	587	993	995
RSA	92.3%	99.1%	99.0%	95.0%	93.0%	99.1%	99.0%
DHE-RSA	85.9%	74.8%	74.7%	86.4%	65.0%	73.9%	73.7%
ECDHE-RSA	41.2%	8.4%	8.7%	11.5%	25.2%	8.9%	7.2%
ADH	64.3%	7.8%	6.3%	14.5%	26.0%	5.3%	5.5%
AECDH	35.9%	1.7%	1.7%	7.3%	14.5%	1.7%	1.6%
EXP-RSA	75.2%	18.1%	16.5%	5.8%	28.9%	15.7%	16.1%
EXP-DHE-RSA	71.6%	11.8%	10.3%	3.6%	18.7%	9.2%	9.4%
EXP-ADH	63.2%	7.5%	6.0%	0.1%	12.8%	5.0%	5.2%
Only RSA	2.7%	15.9%	16.2%	6.2%	14.5%	17.1%	17.4%

deployed and 4096 are close to non-existing. One exception are parameters used on TCP port 465. 69.5% of all hosts use a 2048-bit group and only 30.39% use a 1024-bit group. We explain this difference with the default configuration of Exim, a popular MTA daemon which uses a 2048-bit DH group (*2048-bit MODP Group with 224-bit Prime Order Subgroup* by default.

2) Reuse of Common Primes for Diffie-Hellman: For SMTP we found one 512-bit prime which is used by 996,792 distinct IPs (64% of all 1,557,288 valid scans) and also one 1024-bit prime which is used by 1,077,736 distinct IPs (69.2%). These two primes are statically included in the postfix source code. We also rediscovered these primes (512 bit: 2,066 hosts, 1024 bit: 512,391 hosts) at DH handshakes on port 465. 1,673,271 distinct IPs on port 465 (58.7%) make use of the specific DH group used by Exim as described above. On port 587, 539,322 IPs (32.7%) use the 1024-bit and 212,898 IPs (12.9%) use the 512-bit prime used by postfix. 218,163 (13.2%) use the Exim 2048-bit group and 85,548 (5.2%) use a 1024-bit prime included in the nginx source code. 99,278 IPs on port 993 (2.38%) and 94,555 IPs on port 995 (2.4%) use the same 1024-bit nginx prime. 512-bit primes are not shared for these protocols. Only 721 IPs (IMAPS) and 733 IPs (POP3S)

TABLE V
ENCRYPTION METHOD SUPPORT

Encryption algorithm	25	110	143	465	587	993	995
AES-128 (CBC)	91.3%	93.5%	94.0%	93.5%	84.3%	92.2%	91.3%
AES-256 (CBC)	91.6%	93.4%	93.8%	93.9%	84.5%	93.4%	92.5%
3DES-168	91.6%	97.4%	97.5%	93.3%	92.3%	96.2%	96.3%
RC4-128	86.3%	86.5%	85.7%	82.3%	79.6%	83.4%	83.7%
CAMELLIA-128	58.7%	55.9%	57.8%	66.0%	51.3%	58.5%	56.9%
AES-128 (GCM)	54.9%	49.3%	52.8%	63.3%	46.4%	55.2%	53.4%
AES-256 (GCM)	55.4%	49.2%	52.7%	63.0%	46.6%	55.1%	53.0%
SEED-128	53.8%	48.6%	50.0%	52.8%	35.8%	49.1%	47.9%
IDEA-128	21.6%	45.9%	46.5%	46.3%	28.4%	45.8%	45.9%
DES-56	76.1%	5.8%	5.6%	6.8%	29.8%	6.5%	6.8%
RC4-40	74.9%	18.1%	16.5%	5.8%	28.8%	15.4%	15.7%
RC2-128	40.9%	2.7%	2.2%	4.1%	14.4%	5.6%	6.3%
NULL	0.1%	0.00%	0.01%	0.02%	0.02%	0.05%	0.04%
Only AES-CBC	0.16%	0.11%	0.07%	0.06%	0.11%	0.22%	0.37%
Only RC4	0.05%	0.45%	0.45%	0.87%	0.10%	0.78%	0.80%

share the most used 512-bit prime. Primes used in server deployments for POP and IMAP are more diverse. The most used 1024-bit prime is used by 27,429 IPs (POP) and 6,291 IPs (IMAP), which are only 0.8% of all distinct IPs each. The most commonly shared 512-bit prime is only used by 0.05% of all IPs.

3) Elliptic Curve Diffie-Hellman: One alternative is using Diffie-Hellman key exchanges based on elliptic curve cryptography. We examined commonly used curves specified for the use in TLS [9]. For SMTP-related protocols the secp256r1 curve is used for up to 99.2% of all ECDH handshakes. For POP3 and IMAP ports the usage differs, as three different curves are in use, whereas secp384r1 is by far the mostly used curve. Additionally, the curves sect163r2, secp224r1 and secp256k1 occur, but only for less than 0.03% of all handshakes.

D. TLS Certificates

The subject of certificate validation for HTTPS has been investigated by various other studies [42], [24], [21]. In our study we analyzed 2,115,228 different unique leaf certificates for all e-mail-related protocols. We used certificates collected with our cipher suite scans.

1) Self-signed certificates: We used the Mozilla NSS Truststore (as of 04/2015) to validate all certificates. For client-to-server protocols the percentage of self-signed certificates is very high. Over 50% of all distinct IPs use self-signed certificates, only 33%–37% are correctly validated. For SMTP the number of self-signed certificates is even higher. 65% of all distinct IPs we scanned use a self-signed certificate (expired self-signed certificates included). This is no surprise, since SMTP deployments do not validate certificates by default. Around 1% of all valid, certificates are valid but expired. An overview is given in Table VII.

2) RSA Public Key Size: We find that over 99% of all leaf certificates use RSA public keys. We analyzed the public key size of these certificates and find that over 90% of all trusted leaf certificates are equipped with 2048-bit public keys. Less than 10% use 4096-bit public keys and less than 0.1% use 1024-bit keys. This situation changes if we consider self-signed certificates. 1024-bit keys are often used (15%–40%), followed by 2048-bit keys.

3) Common Leaf Certificates: Many certificates are found on more than one IP. Table VIII shows the number of IPs that use the same certificate. The table includes the top certificates per port. We identified certificates that are used by a massive number of IPs, usually in same subnets. The certificate with the SHA1 fingerprint `b16c...6e24` was provided on 85,635 IPs in two different /16 IP ranges and on two IPs in two completely different IP ranges.

We also found self-signed certificates that do not share the same fingerprint, but subject attributes are the same (*CommonName, OrganizationalUnitName, OrganizationalName*). We identified entities that are responsible for a large number of self-signed certificates. These entities are either software administration tools like *Parallels Panel* or *Plesk*, or default configurations of e-mail software like *Courier* and *Dovecot*. They also play an important role in reducing the use of self-signed certificates in the future. An overview of top entities is shown in Table IX.

4) Weak RSA Keys: Remember that the security of the RSA scheme relies on the difficulty of an attacker factoring (i.e. finding) the random, large primes p and q given their product, the public RSA modulus N. If the factorization of N is known to an attacker, the private key can be easily obtained from a RSA public key. Of 30,757,242 collected RSA certificates from all e-mail protocols and related ports we've extracted 2,354,090 unique RSA moduli and were able to recover 456 greatest common divisors (hence RSA private keys). We used techniques and tooling provided by the authors of [35] for the computation of vulnerable moduli. A similar analysis was performed on data sets for HTTPS in [21].

E. Domains

Our scans are based on IPv4 addresses. The responsible e-mail server address for a specific domain is stored in MX records of the domain name system. The influence of each SMTP server deployment to the global state of TLS security is not evenly distributed, because a single SMTP server can be

TABLE VI
DH Parameter Size for non-export cipher suites and curves used for all cipher suites

Size of Prime	25	110	143	465	587	993	995
512 bit	0.08%	0.04%	0.05%	0.05%	0.05%	0.07%	0.05%
768 bit	0.02%	6.31%	6.52%	0.00%	0.03%	7.02%	7.43%
1024 bit	99.10%	92.11%	91.64%	30.39%	75.39%	92.14%	91.77%
2048 bit	0.79%	1.50%	1.75%	69.50%	24.48%	0.71%	0.70%
4096 bit	0.01%	0.00%	0.01%	0.01%	0.04%	0.01%	0.00%
DH Curve							
secp256r1	99.20%	27.67%	25.52%	99.73%	94.63%	28.77%	23.26%
secp384r1	0.22%	65.70%	67.35%	0.10%	3.84%	63.82%	69.65%
secp521r1	0.12%	0.94%	1.01%	0.12%	0.93%	1.02%	0.93%
sect571r1	0.46%	5.69%	6.13%	0.02%	0.59%	6.38%	6.13%

TABLE VII
Truststore results and public key size of certificates

Result of Validation	25	110	143	465	587	993	995
ok	20.66%	32.87%	34.11%	36.20%	37.57%	35.81%	37.43%
self-signed certificate	65.20%	52.13%	53.05%	53.96%	38.61%	51.57%	50.03%
unable to get local issuer certificate	10.05%	11.31%	11.01%	8.30%	20.50%	11.09%	11.19%
validation error	1.24%	2.49%	0.53%	0.24%	0.99%	0.07%	0.08%
RSA Public Key Size							
512	0.29%	0.13%	0.13%	0.06%	0.10%	0.10%	0.05%
1024	19.97%	17.57%	16.37%	9.85%	19.01%	16.27%	15.29%
2048	75.74%	79.49%	80.45%	86.68%	76.75%	80.28%	81.58%
4096	3.78%	2.72%	2.96%	3.34%	4.02%	3.27%	3.00%

TABLE VIII
Common leaf certificates

Common Name (Issuer CN)	Fingerprint	Port	IPs
*.nazwa.pl	b16c…6e24	25	40,568
(nazwaSSL)		465	81,514
		587	84,318
		993	85,637
		995	85,451
*.pair.com	a42d…768f	25	15,573
(USERTrust RSA Organization …)		110	60,588
		143	13,186
		465	63,248
		587	61,933
		993	64,682
		995	64,763
*.home.pl	8a4f…6932	110	126,174
(RapidSSL SHA256 CA - G3)		143	26,735
		587	125,075
*.home.pl	c4db…a488	993	128,839
(AlphaSSL CA - SHA256 - G2)		995	126,102

TABLE IX
Top self signed certificiates for all protocols

Name	Size	IPs
Parallels Panel - Parallels	2048	306,852
imap.example.com - IMAP server	1024	261,741
Automatic…POP3 SSL key - Courier …	1024	87,246
Automatic…IMAP SSL key - Courier …	1024	83,976
Plesk - Parallels	2048	68,930
localhost.localdomain - SomeOrganizationalUnit	1024	26,248
localhost - Dovecot mail server	2048	13,134

responsible for many domains and assigned to many different public IP-addresses. We analyzed MX records for the domains listed in the Alexa Top 1 Million ranking. Although this ranking is based on web traffic, we can identify common used SMTP servers. We checked 978,842 domains and resolved all MX records to IPs. 12.35% of the domains had no MX record at all. From the remaining 857,910 domains 16.01% used MX records attributed to Google. *secureserver.net* was included in 4.10% and *qq.com* in 1.22% of all domain MX records. On the contrary, 527,680 MX record names are only used by one domain. This gives us a basic idea on how much some players influence the global state of TLS security for SMTP.

Google started to publish statistics of encrypted e-mail traffic as part of their transparency report [32]. They state that 31% of inbound and 17% of outbound e-mail traffic from Google is still unencrypted. Also, Twitter recently published data stating that only 5% of their e-mail traffic is unencrypted.

These transparency reports are important sources of information. Due to the decentralized infrastructure of the e-mail transport system, we only get a very limited view of the global state of encryption. More transparency reports would help to clarify this even more.

F. Plaintext Authentication

A commonly observed issue with e-mail are servers supporting plaintext authentication without support for STARTTLS. Despite having nothing to do with cryptography per se, this is an important issue for transport security as connection authentication data sent via plaintext channel provides an easy target for any attacker. Some application layer e-mail protocols offer so-called `CAPABILITIES` when connecting as a client to a server, which may specify the way in which a username and password exchange is performed. In more detail:

1) `AUTH-PLAIN` in SMTP, POP3 and IMAP: `AUTH-PLAIN` should only be offered after a connection has been upgraded via STARTTLS. If a server offers plaintext authentication on a plaintext connection, neither server nor user will notice if STARTTLS stripping is performed and

TABLE X
HOSTS THAT OFFER AUTH PLAIN

Port	no STARTTLS	STARTTLS	Total Hosts
25	12.90%	24.21%	7,114,171
110	4.24%	63.86%	5,310,730
143	4.38%	66.97%	4,843,513
587	15.41%	42.80%	2,631,662

password data sent in plaintext over the wire. Table X outlines the number of hosts that offer AUTH-PLAIN and support STARTTLS, but allow plaintext authentication before an upgrade to TLS, as well as hosts that do not offer STARTTLS at all.

2) Improper use of LOGINDISABLED in IMAP: The idea behind the LOGINDISABLED mechanism is that when a server announces the LOGINDISABLED capability, a client won't send any plaintext credentials until the connection is upgraded via STARTTLS. In practice, some mail user agents (MUAs) fail to honor the IMAP4 specification and continue to send plaintext information if a server explicitly discourages this behavior. Thus, even for passive attackers, sniffing of user credentials becomes a triviality. 342,124 IPs or 7.06% support LOGINDISABLED and 4,501,389 IPs or 92.94% do not offer the LOGINDISABLED capability specific to the IMAP protocol.

V. DISCUSSION

The results from our scans cast a disillusioning light on the use of TLS in e-mail. Not only are weak encryption mechanism and cipher suites like export grade ciphers supported by a considerable fraction of e-mail servers, we also found that the recent increase in HTTPS certificate security (moving certificates from 1024 to 2048 bit) went totally unnoticed for all e-mail-related ports, IPv4-wide. Furthermore, millions of hosts are currently misconfigured to allow AUTH-PLAIN over unencrypted connections resulting in the risk of transmitting user credentials unencrypted. Even if each of these systems has only one user, this leaves close to 20 million users vulnerable to attack.

We found that more than 650,000 SMTP deployments accept export grade ciphers with TLSv1.1 and TLSv1.2 (with at least OpenSSL affected, implementation-wise) although this behavior is explicitly forbidden (MUST NOT) in RFC 4346 [15].

We also found troublesome issues despite the cryptographic primitives used (if used at all). More than half of all certificates we observed were self-signed, meaning that an active attacker could exchange the certificate in a man-in-the-middle scenario. Since the user interface of most e-mail clients lack a convenient way to check the authenticity of certificates, this leaves clients without the most basic protection mechanisms – which are well deployed and well understood for the HTTPS PKIX ecosystem. Compared to previous work on HTTPS [24], [21] we did not analyze the certificate authorities who issued the certificates at length. While HTTPS enables the client to authenticate the final server serving the content, this is different for e-mail due to the distributed nature of e-mail transmission protocols. Only the first hop is observable to the client, and thus only a fraction of servers along the path of an e-mail. Lastly, we find this to be rather a policy issue, i.e. who is trusted by the software vendors as certificate authority, then a trust issue, i.e. do I trust CA X to issue certificates for a given domain Y [58]. Furthermore, there are now effective countermeasures available for HTTPS which are transparent for users, like HPKP [25] for dynamic public key pinning, or certificate pinning within the browser to effectively prevent man-in-the-middle attacks. No such mechanisms exist so far for e-mail protocols and server implementations, and as such we purposely didn't evaluate the issuing certificate authorities.

A. Who Leads the Way?

The overall security of the e-mail ecosystem cannot be improved by a single actor. Overall security in e-mail transmission can only improve slowly if the majority of transmitting hosts aligns. We identified multiple issues that need to change not only in the long-term, but already in the near future. We found that big providers of e-mail infrastructure often lead the way regarding TLS in e-mail. For one, the usage of TLS in e-mail has increased since the documents by Edward Snowden on NSA dragnet surveillance were published, with Yahoo and Microsoft *live.com* enabling their server infrastructure to enforce TLS wherever possible. Secondly, big service providers like Gmail are in the position to give deep insight into the server-to-server use of TLS. With the release of more and more transparency reports (most recently by Twitter), we can draw a more complete picture of TLS use and identify global actors that handle large volumes of e-mail and do not support TLS. Based on this data, it would be great to have a list of the most important domains regarding legit e-mail volume, similar to the often cited Alexa list of popular websites.

B. Limitations

Since there is no comprehensive list and global ranking of e-mail domains regarding their e-mail volume and overall popularity, this is a clear drawback of our approach. We treated and weighted all scanned IPs equally. Major e-mail providers that handle an overproportional volume of clients are treated equally to the single-user, single-domain self-hosted server. However, we tried to counter this form of bias in our observations and findings by using for one the most popular e-mail-related domains as published by Google, as well as correlate it with the most popular websites, even though there seems to be no direct correlation between popular websites and popular e-mail domains. A full list of popular SMTP hosts would have been very helpful, but without any data openly available we had to resort to unweighted statistical analysis. Another limitation is that our findings are only showing a point-in-time snapshot and as such are only of short-living value.

C. Future Work

For future work we intend to keep our scans actively running and will publish the results, since the volatility in cipher configurations and certificates demands regular scan intervals. Long-term trends and reaction times to security incidents like Heartbleed [22] or the weak certificates issued with Debian's faulty PRNG [60] back in 2008 are still not fully understood, and data is needed to show the improvements of technical mechanisms to enhance the security in e-mail as well as the awareness of e-mail server administrators. We also plan to keep improving our scanning infrastructure, to allow for long-term data collection and to provide open data for analysis.

VI. Mitigation Strategies

This paper primarily focuses on TLS-enabled e-mail deployments. At the moment, many deployments exist which do not support TLS at all, so the first step toward a more secure e-mail ecosystem is to enable TLS. Foster et al. [28] give good recommendations regarding the support of TLS, the certificates in use, and the adoption of SPF, DKIM and DMARC. These recommendations are mainly addressed to server administrators.

Various approaches are technically feasible to increase the security of TLS in e-mail, despite trying to increase the TLS configuration of each and every individual server. The most important issue however is to increase the awareness for security and secure configurations among administrators. Furthermore, mitigation can be put into two different categories: (i) Various forms of pinning could be used (either on the host itself or on the network layer) as the certificate information is transmitted prior to the encrypted communication, and the number of e-mails transmitted to particular servers is usually much larger compared to the number of its certificate changes. *Tack.io*[6] is a draft from 2013 which could be used to add trust assertions to certificates and would be especially valuable to protocols making use of STARTTLS. (ii) Another class of defenses includes all methods where signed certificates are transmitted over additional communication channels, most notably DNSSEC or DANE [38]. Both rely on including signed assurances in DNS name records that can be verified by the client to not have been manipulated during communication. Notary services like *ICSI Certificate Notary* [5], Convergence[7] or the SSL Observatory from the EFF are just a few examples of readily-available and deployed approaches where the trust is distributed among independent entities. Defending against active attackers is much harder however, as the attacker is always in the position to suppress communication like the STARTTLS command at the beginning of the transmission, as showed by Durumeric et al. [20]. Such stripping attacks can only be prevented if there is no fallback mechanism to plaintext, which is currently unfortunately the default between MTAs. In IETF a new draft is being worked on, called DEEP ("Deployable Enhanced Email Privacy")[8]. If standardized and implemented, it would offer extensions to SMTP, POP3 and IMAP for so-called "latches" which are similar to what HSTS [36] does for HTTP, and follow-up connections will use the same or better security parameters as previous transactions with a given server. The extensions also offer mail user agents (MUA) the possibility to indicate security levels to end users.

Lastly, operators of smaller e-mail servers often do not change default configurations. They should be incentivized to enable important security primitives or to enforce TLS for communication. Guidelines to improve security for various software products exist [1], [39], [52], but these suggestions are not automatically implemented. If software products embed secure configurations by default, this could change the overall primitives in TLS usage. Smart update mechanisms could enforce, or at least offer, the use of stronger cryptographic primitives. A promising project for exterminating the usage of self-signed certificates is *Let's encrypt*[9] which is a fully automatic CA to issue trusted certificates for free. This and other suggestions were also made by Holz et al. [41].

VII. Related Work

1) Large-scale scanning: In 2008 Heidemann et al. [34] started an Internet-wide discovery scan of edge hosts. They were the first to census the IPv4 address space, but were limited to host discovery, not containing protocol- or service-specific results. First projects to cover TLS-related scanning included the EFF (Electronic Frontier Foundation) Observatory [24] which collected 1.3 million unique certificates of the HTTPS/TLS public key infrastructure. Holz et al. [42] verified flaws in the PKI of TLS with a larger data set obtained by using nmap as well as passive scan data. In 2013 Durumeric et al. developed zmap, a tool for fast Internet-wide scanning [23]. This resulted in the HTTPS ecosystem study [21], on certificates and certificate authorities, having a nearly complete dataset. These studies are the inspiration for our work, but primarily focused on the public key infrastructure of HTTPS. Unlike our study, detailed cipher suite scans were not conducted. The results of port scans originating from the University of Michigan were used as input data for our scan. Durumeric et al. [19] recently introduced a search engine for Internet-wide scans. This engine also features community-defined, pluggable scanning modules and user-defined annotations for existing scanning results. To the best of our knowledge, this data does not contain complete cipher suite scans at the moment.

2) Evaluating HTTPS: In 2007 Lee et al. [47] surveyed 19,000 TLS-enabled servers and inspected the supported TLS versions and different cryptographic primitives for HTTPS servers. Several newer studies for different deployed security mechanisms exist. Huang et al. [43] surveyed forward secrecy parameters and measured latencies of different cipher suites, Kranch and Bonneau [46] conducted a study of HTTP Strict Transport Security and public key pinning deployments. Weissbacher et al. [59] analyzed the adoption of CSP (Content Security Policy). These datasets are all limited to the Alexa Top 1 Million ranking and, especially in case of CSP and HSTS, not relevant for e-mail security. The acceptance of

[6] https://tack.io

[7] https://convergence.io

[8] https://datatracker.ietf.org/doc/draft-ietf-uta-email-deep

[9] https://letsencrypt.org/

cipher suites which support forward secrecy is analyzed in our study. Heninger et al. [35] performed Internet-wide scans to identify vulnerable RSA and DSA keys. Due to entropy problems of the random number generators used, they were able to obtain 0.5% of all RSA private keys for HTTPS. We revisited their work and analyzed RSA public keys used in TLS over e-mail related protocols. Adrian et al. [2] investigated the security of Diffie-Hellman key exchanges using an Alexa Top 1 Million scan. This scan was also performed for HTTPS, not considering the e-mail ecosystem. We investigated common primes and Diffie-Hellman parameters, as well as usage statistics on elliptic curve Diffie-Hellman exchanges for the e-mail ecosystem.

3) E-mail subsystem: Very recently three other publications focused on specific security aspects of the e-mail ecosystem. Foster et al. [28] surveyed major e-mail providers with different measurement techniques. The support of the e-mail related SPF, DKIM and DMARC as well as the fundamental support of TLS was examined. Only the major e-mail providers identified by the Alexa Top 1 million domains and data occurring in the Adobe leak were observed. They did not examine in-depth cipher suite usage and did not cover cipher-suite-based parameters of TLS deployments for all IPv4-wide e-mail-related servers. Good recommendations to increase the security of the e-mail ecosystem were proposed, although cipher-suite-based TLS configurations were not considered. Durumeric et al. [20] examined the support of SPF, DKIM, DMARC and STARTTLS for e-mail transfer with SMTP. The set of scanned e-mail servers is also built upon the Alexa Top 1 million domains. With an additional data set from Gmail, they also analyzed the use of inbound and outbound TLS traffic on the Gmail servers. This also included the used cipher suites for inbound traffic. They did not cover client-to-server use cases for e-mail submission and retrieval and did not study all accepted cipher suites for all e-mail-related deployments. Concurrently with our work, Holz et al. [41] analyzed the use of TLS for communication protocols. Their dataset included an active scan with zmap and OpenSSL as well as passively collected traffic of one university network. Although they conducted analyses similar to ours, they did not perform any in-depth cipher suite scan to analyze cryptographic primitives accepted in the wild.

VIII. Conclusion

In this paper we showed and evaluated a scalable approach to assess the overall security in e-mail protocols by inspecting the underlying TLS primitives. We evaluated our methodology on all different ports in use for e-mail by actively scanning the entire range of IPv4. We conducted more than 10 billion TLS handshakes and discovered multiple flaws that prevent the use of TLS as an effective countermeasure against passive attackers. Overall, the state of TLS in e-mail transmission is, unsurprisingly, worse than compared to HTTPS, as 15%–30% of all servers accept weak export grade ciphers and the majority of certificates is self-signed. On the positive side we were able to show that RC4 and the use of SSLv2 and SSLv3 for backward compatibility can be considered almost obsolete. Disabling them has hardly any negative impact on the total number of reachable SMTP servers.

Acknowledgments

We would like to thank the HPC team at the Gregor Mendel Institute which allowed us to use MENDEL for parts of the evaluation. Thanks to Hanno Böck for discussion and feedback on Cryptography related subjects, RSA Weak-Keys and TLS. Thanks to Viktor Dukhovni for valuable feedback and insights into OpenSSL specifics, MTA operational details and discussion on opportunistic e-mail encrytion in general. Thanks to Wolfgang Breyha for discussion on applied e-mail security.

The research was funded by COMET K1, FFG - Austrian Research Promotion Agency and under grant 846028.

References

[1] Applied Crypto Hardening. Online at https://bettercrypto.org, 2015.

[2] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, , and P. Zimmermann. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *22nd ACM Conference on Computer and Communications Security*, Oct. 2015.

[3] N. J. Al Fardan and K. G. Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 526–540. IEEE, 2013.

[4] N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. Schuldt. On the Security of RC4 in TLS. In *22nd USENIX Security Symposium*, pages 305–320, 2013.

[5] B. Amann, M. Vallentin, S. Hall, and R. Sommer. Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. Technical Report TR-12-014, ICSI, Nov. 2012.

[6] M. Avalle, A. Pironti, and R. Sisto. Formal Verification of Security Protocol Implementations: A Survey. *Formal Aspects of Computing*, 26(1):99–123, 2014.

[7] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue. A Messy State of the Union: Taming the Composite State Machines of TLS. In *IEEE Symposium on Security and Privacy*, 2015.

[8] K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, and P. Strub. Implementing TLS with Verified Cryptographic Security. In *IEEE Symposium on Security and Privacy*, pages 445–459, 2013.

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

[10] 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*, pages 114–129. IEEE, 2014.

[11] P. Chown. Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS). RFC 3268 (Proposed Standard), June 2002. Obsoleted by RFC 5246.

[12] 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*, pages 511–525, 2013.

[13] M. Crispin. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501 (Proposed Standard), Mar. 2003.

[14] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), Jan. 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176.

[15] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), Apr. 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746, 6176.

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

[17] V. Dukhovni. Opportunistic Security: Some Protection Most of the Time. RFC 7435 (Informational), Dec. 2014.

[18] T. Duong and J. Rizzo. Here Come The $\oplus$ Ninjas. *Ekoparty*, 2011.

[19] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. A search engine backed by Internet-wide scanning. In *22nd ACM Conference on Computer and Communications Security*, pages 542–553. ACM, 2015.

[20] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman. Neither snow nor rain nor MITM... an empirical analysis of email delivery security. In *15th ACM Internet Measurement Conference*, 2015.

[21] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS Certificate Ecosystem. In *13th ACM Internet Measurement Conference*, pages 291–304, Oct. 2013.

[22] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman. The Matter of Heartbleed. In *14th ACM Internet Measurement Conference*, Nov. 2014.

[23] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *22nd USENIX Security Symposium*, Aug. 2013.

[24] P. Eckersley and J. Burns. An Observatory for the SSLiverse. DEF CON 18 https://www.eff.org/files/defconssliverse.pdf, July 2010.

[25] C. Evans, C. Palmer, and R. Sleevi. Public key pinning extension for http (hpkp). RFC 7469 (Draft), 2015.

[26] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security. In *19th ACM Conference on Computer and Communications Security*, pages 50–61, 2012.

[27] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith. Rethinking SSL Development in an Appified World. In *20th ACM Conference on Computer and Communications Security*, pages 49–60, 2013.

[28] I. D. Foster, J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko. Security by any other name: On the effectiveness of provider based email security. In *22nd ACM Conference on Computer and Communications Security*, pages 450–464. ACM, 2015.

[29] R. Gellens and J. Klensin. Message Submission for Mail. RFC 4409 (Draft Standard), Apr. 2006. Obsoleted by RFC 6409.

[30] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software. In *19th ACM Conference on Computer and Communications Security*, pages 38–49, 2012.

[31] Y. Gluck, N. Harris, and A. A. Prado. Breach: Reviving the CRIME attack. Technical report, 2013.

[32] Google. Transparency Report - Email encryption in transit. https://www.google.com/transparencyreport/saferemail/. visited 2015-09-16.

[33] B. He, V. Rastogi, Y. Cao, Y. Chen, V. Venkatakrishnan, R. Yang, and Z. Zhang. Vetting SSL Usage in Applications with SSLINT. 2015.

[34] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister. Census and Survey of the Visible Internet. In *8th ACM Internet Measurement Conference*, pages 169–182, Oct. 2008.

[35] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In *21st USENIX Security Symposium*, pages 205–220, Aug. 2012.

[36] J. Hodges, C. Jackson, and A. Barth. HTTP Strict Transport Security (HSTS). RFC 6797 (Proposed Standard), Nov. 2012.

[37] P. Hoffman. SMTP Service Extension for Secure SMTP over Transport Layer Security. RFC 3207 (Proposed Standard), Feb. 2002.

[38] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard), Aug. 2012.

[39] R. Holz. Recommendations for secure use of TLS and DTLS. In *IETF, RFC 7525*, 2015.

[40] R. Holz. Summarizing known attacks on TLS and DTLS. In *IETF, RFC 7457*, 2015.

[41] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar. TLS in the wild: an Internet-wide analysis of TLS-based protocols for electronic communication. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2016. arXiv preprint arXiv:1511.00341 http://arxiv.org/abs/1511.00341.

[42] 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 *11th ACM Internet Measurement Conference*, pages 427–444, Nov. 2011.

[43] L.-S. Huang, S. Adhikarla, D. Boneh, and C. Jackson. An Experimental Study of TLS Forward Secrecy Deployments. *Internet Computing, IEEE*, 18(6):43–51, 2014.

[44] D. Kaloper-Meršinjak, H. Mehnert, A. Madhavapeddy, and P. Sewell. Not-quite-so-broken TLS: lessons in re-engineering a security protocol specification and implementation. In *24th USENIX Security Symposium*, 2015.

[45] J. Klensin. Simple Mail Transfer Protocol. RFC 5321 (Draft Standard), Oct. 2008.

[46] M. Kranch and J. Bonneau. Upgrading HTTPS in Mid-Air: An Empirical Study of Strict Transport Security and Key Pinning. Feb. 2015.

[47] H. K. Lee, T. Malkin, and E. Nahum. Cryptographic Strength of SSL/TLS Servers: Current and Recent Practices. In *7th ACM Internet Measurement Conference*, pages 83–92, Oct. 2007.

[48] A. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter. Ron was wrong, Whit is right. Technical report, IACR, 2012.

[49] C. Meyer, J. Somorovsky, E. Weiss, J. Schwenk, S. Schinzel, and E. Tews. Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks. In *23rd USENIX Security Symposium*, pages 733–748, 2014.

[50] Microsoft. MS14-066: Vulnerability in Schannel Could Allow Remote Code Execution (2992611). https://technet.microsoft.com/en-us/library/security/ms14-066.aspx, 2014.

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

[52] Mozilla Wiki. Security/Server Side TLS. https://wiki.mozilla.org/Security/Server_Side_TLS, 2015.

[53] J. Myers and M. Rose. Post Office Protocol - Version 3. RFC 1939 (INTERNET STANDARD), May 1996.

[54] C. Newman. Using TLS with IMAP, POP3 and ACAP. RFC 2595 (Proposed Standard), June 1999.

[55] A. Popov. Prohibiting RC4 Cipher Suites, Feb. 2015.

[56] I. Ristić. Bulletproof SSL and TLS. *Feisty Duck*, 2014.

[57] J. Rizzo and T. Duong. The CRIME Attack, 2012.

[58] C. Soghoian and S. Stamm. Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL. In *Financial Cryptography and Data Security*, pages 250–259. Springer, 2012.

[59] M. Weissbacher, T. Lauinger, and W. Robertson. Why is CSP Failing? Trends and Challenges in CSP Adoption. In *Research in Attacks, Intrusions and Defenses*, pages 212–233. Springer, 2014.

[60] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When Private Keys are Public: Results from the 2008 Debian OpenSSL Vulnerability. In *9th ACM Internet Measurement Conference*, pages 15–27, Nov. 2009.