Hochschule München
Fakultät für Informatik und Mathematik

Sicherheitsanalyse der TLS-Konfiguration von SMTP-Installationen

Security Analysis of TLS Configurations of SMTP Deployments

Bachelorarbeit
Abgabe am 1. Mai 2015

Autor: Thomas Maier
Matrikelnummer: 02946810

Prüfer: Professor Dr.-Ing. Hans-Joachim Hof
MuSe - Munich IT Security Research Group
Betreuer: Dipl.-Inf. (FH) Thomas Schreck
Siemens CERT

Erklärung

Hiermit erkläre ich, dass ich die Bachelorarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum Unterschrift

Inhaltsverzeichnis

1 Einleitung

Bereits vor ca. 2.500 Jahren wurden Nachrichten im antiken Griechenland mithilfe der *Skytale von Sparta* verschlüsselt [1]. Immer wieder wurden neue Verschlüsselungsverfahren entwickelt, die zur Geheimhaltung von Informationen genutzt wurden. Das unbefugte Erlangen von Information wurde mit Anbruch des Informationszeitalters leichter, da Daten, die über das Internet übertragen werden, an Knotenpunkten abgehört werden können. Bei der Überwachungsaffäre 2013 war zu sehen, dass auch das Mitlesen von großen Datenmengen möglich ist [2]. Mit dem Protokoll *Secure Sockets Layer* (SSL), in späteren Versionen *Transport Layer Security* (TLS) genannt, können Anwendungsdaten für den Transport verschlüsselt werden.

Wie sicher SSL/TLS bei der Übertragung von Mails mit dem *Simple Mail Transfer Protocol* (SMTP) implementiert wurde, ist nicht bekannt. Daher wird in dieser Bachelorarbeit die SSL/TLS-Konfiguration von SMTP-Servern geprüft, um eine repräsentative Aussage über den aktuellen Stand der Sicherheit in Hinsicht auf den Transport von Mails zu treffen. Um diese Analyse zu ermöglichen, wird im ersten Schritt ein Werkzeug entwickelt, mit dem anschließend SSL/TLS-Konfigurationen im Internet gesammelt werden. Im zweiten Schritt werden die bei der Datenerhebung gesammelten Daten analysiert und ausgewertet.

Am Anfang der Arbeit werden in **Kapitel 2** Grundlagen erklärt, die für das Verständnis der nachfolgenden Kapitel notwendig sind. Anschließend werden in **Kapitel 3** verwandte Arbeiten vorgestellt, die bei der Recherche untersucht werden. In **Kapitel 4** wird beschrieben, wie die Datenerhebung entworfen, entwickelt, getestet und ausgeführt wurde. Die dabei gesammelten Daten werden im nachfolgenden **Kapitel 5** analysiert und bewertet. Der Inhalt und die Ergebnisse dieser Arbeit werden am Ende in **Kapitel 6** zusammengefasst. Ferner wird in diesem Kapitel beschrieben, wie die Arbeit in Zukunft fortgeführt werden kann.

2 Grundlagen

Die Sicherheitsanalyse von Protokollen ist Kern dieser Arbeit. Die dafür notwendigen Grundlagen werden in diesem Kapitel beschrieben. Begonnen wird dabei mit einem allgemeinen Blick auf Public-Key-Infrastrukturen, die wichtig für den Nachweis von Identitäten sind. Zur Verschlüsselung von Informationen auf dem Transportweg wird nachfolgend SSL/TLS erklärt. Schlussendlich wird auf verschiedene Protokolle zur Identitätsverifikation eingegangen. Nachdem verschiedene Protokolle zur Identitätsverifikation erklärt wurden, werden Portscanner erklärt.

2.1 Public-Key-Infrastrukturen

In einer Public-Key-Infrastruktur (PKI) werden Zertifikate verwendet, um die Identität des Inhabers mit dessen öffentlichem Schlüssel zu verknüpfen. Nachfolgend werden PKIs, wie im Buch *Sicherheit in Kommunikationsnetzen* [3] beschrieben, zusammengefasst.

Der Inhaber eines Schlüssels beantragt die Zertifizierung bei einer Zertifizierungsstelle (*Certificate Authority*, CA). Weist dieser Antragsteller seine Identität nach, stellt die CA ein Zertifikat C (*certificate*) aus. Es enthält Eigenschaften (*attributes*) des Inhabers A_I, z.B. seinen Namen, seine Adresse oder dessen Organisation. Außerdem beinhaltet es die Eigenschaften A_K des Schlüssels K_{pub} bzw. dessen Fingerprint. Jede CA hat ein Schlüsselpaar (R_{priv}, R_{pub}) (*root key*), mit dem andere Zertifikate signiert werden. Für diese Eigenschaften wird also von der CA eine Signatur S_C mit deren privatem Schlüssel R_{priv} erstellt.

$$S_C := enc(R_{priv}, A_I \| A_K) \tag{2.1}$$

In der Formel 2.1 ist zu sehen wie die Eigenschaften konkateniert und anschließend mit dem privaten Schlüssel der CA signiert werden. Das fertige Zertifikat C (incl. der Signatur S_C) wird dem Antragsteller ausgehändigt und bei einem öffentlichen Verzeichnisdienst hinterlegt.

Erstellt der Antragsteller, der nun Inhaber des Zertifikats ist, eine Signatur S_M über seine Daten M oder deren Fingerprint, so kann damit die Integrität der Daten nachgewiesen werden, indem die Signatur mit dem öffentlichen Schlüssel K_{pub} verifiziert wird.

$$S_M := enc(K_{priv}, M) \tag{2.2}$$

$$M := dec(K_{pub}, S_M) \tag{2.3}$$

Zuerst wird der Inhalt wie in der Formel 2.2 signiert, um den Prozess danach wie in Formel 2.3 umzukehren.

Ist der öffentliche Schlüssel K_{pub} im Zertifikat enthalten und wurde die Signatur S_C von der CA ausgestellt, so ist zusätzlich zur Integrität auch die Authentizität der Nachricht M nachgewiesen. Dieses System setzt ein bestehendes Vertrauensverhältnis gegenüber der CA voraus, das sicherstellt, dass der öffentliche Schlüssel R_{pub} von der CA stammt.

$$A_I \| A_K := dec(R_{pub}, S_C) \tag{2.4}$$

Durch den Nachweis mit Formel 2.4 wird verifiziert, dass die Signatur von der CA erstellt wurde.

Der PKI-Standard X.509 (auch “PKIX” genannt) wurde von der internationalen Fernmeldeunion (*International Telecommunication Union*, ITU) entwickelt. Er enthält Spezifikationen zur Struktur von Zertifikaten und deren Signaturen [4].

Wie bereits beschrieben, basiert das System von Public-Key-Infrastrukturen darauf, dass der signierenden CA im Vornherein vertraut wird. Die X.509-PKI hat sich für die Verifikation von Zertifikaten als Standard durchgesetzt. Software-Pakete beinhalten häufig bereits beim Download vom Anbieter die öffentlichen Schlüssel bekannter CAs [5].

Um ein signiertes Zertifikat zu bekommen, muss ein *Certificate Signing Request* (CSR) als Antrag bei einer CA gestellt werden [6]. Nach der Prüfung der Identität durch die CA wird ein X.509-Zertifikat ausgestellt.

Ein selbstsigniertes Zertifikat zeichnet sich dadurch aus, dass es mit dem eigenen privaten Schlüssel und nicht von einer CA signiert wurde.

Jedes Zertifikat verfügt über einen Gültigkeitszeitraum, der in Form eines Start- und Endzeitpunkts angegeben wird. Je kürzer der Gültigkeitszeitraum festgelegt wird, desto höher ist die Sicherheit des Zertifikats einzustufen. Wird ein Zertifikat gestohlen, kann es nur innerhalb dieses Zeitraums genutzt werden. Das *CA/B Forum* [7] forderte CAs dazu auf, Zertifikate nur noch für maximal drei Jahre auszustellen [8–10]. Auch beim Webbrowser *Google Chrome* [11] wurde entschieden, dass ab dem ersten Quartal 2014 keine Zertifikate mehr akzeptiert werden, die länger als fünf Jahre gültig sind [12]. Da SSL/TLS unabhängig vom Anwendungsprotokoll verwendet wird, können diese Empfehlungen auch auf Mailserver übertragen werden.

Gesperrte Zertifikate werden auf einer *Certificate Revocation List* (CRL), also einer Sperrliste, veröffentlicht. Auf dieser Liste bleibt es, bis die zeitliche Gültigkeit des Zertifikats abgelaufen ist [4]. Um den Status von X.509-Zertifikaten abzufragen, wurde das Online Certificate Status Protocol (OCSP) entwickelt. Es wird benutzt, um festzustellen, ob ein Zertifikat innerhalb des Gültigkeitszeitraums als ungültig gemeldet wurde [13].

In RFC 5280 [4] ist der Aufbau von X.509-Zertifikaten genau spezifiziert.

Feld	**Beschreibung**
`version`	X.509-Version (1, 2 oder 3)
`serialNumber`	Seriennummer des Zertifikats
`signature`	Vom Aussteller verwendeter Signatur-Algorithmus
`issuer`	Zertifikat-Aussteller
`validity`	Zeitraum in dem das Zertifikat gültig ist
`subject`	Zertifikat-Inhaber
`subjectPublicKeyInfo`	Öffentl. Inhaber-Schlüssel und Schlüsselalgorithmus (z.B. RSA)
`issuerUniqueID`	Eindeutige Nummer der CA (ab X.509-Version 2)
`subjectUniqueID`	Eindeutige Nummer des Inhabers (ab X.509-Version 2)
`extensions`	X.509-Erweiterungen (ab X.509-Version 2)

Tabelle 2.1: Inhalt eines X.509-Zertifikats

Zusätzlich zu den Feldern in Tabelle 2.1 enthält das Zertifikat die Signatur (vgl. Formel 2.1).

2.2 Transport Layer Security (TLS)

Bereits 1994 wurde *Secure Sockets Layer* (SSL) von *Netscape Communications* [14] entworfen, um die Sicherheitsziele Vertraulichkeit, Authentizität und Integrität beim Transfer von Anwendungsdaten über ein zuverlässiges Transportprotokoll zu erfüllen [15]. Damit sollte unter anderem eine Authentifizierung des Servers und optional des Clients über ein zuverlässiges Transportprotokoll ermöglicht werden. Durch die hohe Flexibilität von SSL ist es möglich, den Datentransfer eines beliebigen Anwendungsprotokolls zu verschlüsseln.

Nach der Veröffentlichung von SSL 3.0 [16] übergab Netscape die Weiterentwicklung des Protokolls an die *Internet Engineering Task Force* (IETF). Ab diesem Zeitpunkt

wurde das Protokoll unter dem Namen *Transport Layer Security* (TLS) weitergeführt [17].

Da TLS die Weiterentwicklung von SSL ist, wird in dieser Arbeit stellvertretend für alle Versionen nur von TLS gesprochen. Falls Unterschiede herrschen, wird explizit darauf hingewiesen.

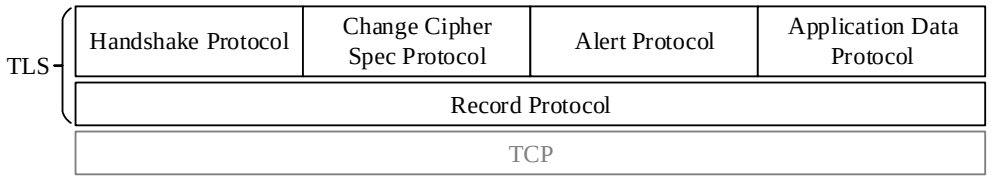


Abbildung 2.1: TLS im TCP/IP-Protokollstack

Das Protokoll TLS befindet sich im ISO/OSI-Referenzmodell auf der Sitzungsschicht und hängt somit direkt über der Transportschicht. TLS verschlüsselt Daten von Anwendungsprotokollen wie dem *Hypertext Transfer Protocol* (HTTP) oder dem *Simple Mail Transfer Protocol* (SMTP), bevor diese an die Transportschicht übergeben werden. TLS besteht wiederum aus zwei Schichten (vgl. Abb. 2.1). Die untere Schicht, das Record Protocol, ist für die symmetrische Verschlüsselung sowie für die Sicherstellung von Integrität und Authentizität zuständig. Die dazu notwendigen Schlüssel werden beim TLS-Verbindungsaufbau (*Handshake Protocol*) ausgehandelt [3]. Auf die anderen TLS-Protokolle in Abb. 2.1 wird nicht weiter eingegangen, da sie für diese Arbeit nicht relevant sind.

Cipher Suites

Durch das Konzept der Cipher Suites ist TLS ein generisches Protokoll. Cipher Suites geben an, wie sich die Kommunikationsteilnehmer verhalten und welche Verfahren angewendet werden [18]:

`TLS_<1>_<2>_WITH_<3>_<4>_<5>_<6>`

<1> Verwendeter Schlüsselaustauschalgorithmus
<2> Verwendeter Authentifizierungsalgorithmus
<3> Verwendeter Verschlüsselungsalgorithmus
<4> Länge des verwendeten Schlüssels
<5> Modus der Verschlüsselung

Genannte

<6> Cipher Suites bis TLS 1.1: *Hash-based Message Authentication Code* (HMAC)
zur Sicherung der Integrität mithilfe einer Hash-Funktion
Cipher Suites ab TLS 1.2: *Pseudorandom Function* (PRF)
zum Generieren von Zufallszahlen (HMAC nicht in Cipher Suite enthalten)

Punkte, die nicht unbedingt in einer Cipher Suite enthalten sein müssen, werden im Abschnitt 2.2.1 genauer erklärt.

Um nachfolgend ein Beispiel anhand dessen Bestandteile zu erklären, wird zuerst das Konzept *Perfect Forward Secrecy* (PFS) erklärt: Wird PFS beim Schlüsselaustausch verwendet, so wird es in Cipher Suites mit einem `E` für "ephemeral" hinter dem Algorithmus angegeben. Beim Schlüsselaustausch werden also "kurzlebige" Sitzungsschlüssel generiert, um die Kommunikation besser zu schützen. Wird der private Schlüssel oder einer der Sitzungsschlüssel kompromittiert, so kann trotzdem nicht die komplette Kommunikation abgehört werden. Auch Kommunikation in der Vergangenheit bleibt weiterhin geschützt.

Die nachfolgende Cipher Suite ist ein gutes Beispiel zum besseren Verständnis, da es für jedes bereits genannte Verfahren einen Algorithmus einsetzt. Sie wird wie oben tabellarisch anhand der einzelnen Verfahren erklärt.

`TLS_DHE_RSA_WITH_AES_128_CBC_SHA`

`DHE` Der Schlüssel wird mit dem Diffie-Hellman-Verfahren unter Verwendung von PFS ausgehandelt.
`RSA` Zur Authentifizierung wird das asymmetrische Verfahren RSA eingesetzt.
`AES` Als Verschlüsselungsalgorithmus wird AES verwendet
`128` Die Schlüssellänge beträgt 128 Bit.
`CBC` AES wird im *Cipher Block Chaining Mode* (CBC) verwendet.
`SHA` Als HMAC bzw. PRF wird SHA-1 benutzt.

Beim Aufbau einer TLS-Verbindung wird die verwendete Cipher Suite ausgehandelt.

Verbindungsaufbau

Der Aufbau von TLS-Verbindungen wird in RFCs als *TLS Handshake Protocol* spezifiziert. Zum einen kann der TLS Handshake direkt auf den *TCP Handshake* folgen, zum anderen ist es möglich, vorerst eine Klartext-Verbindung aufzubauen.

Im zweiten Fall wird der Server-Anwendung der Beginn des TLS Handshake signalisiert, indem der Client die Zeichenkette `STARTTLS` an den Server schickt [3].

Wie der Verbindungsablauf aussieht, hängt von der in den ersten beiden Nachrichten ausgehandelten Cipher Suite ab.

In Abb. 2.2 ist ein beispielhafter Handshake zu sehen, bei dem sich der Server mit einem X.509-Zertifikat authentifiziert. Es wird angenommen, dass zum Schlüsselaustausch RSA verwendet wird [3]. Der Client verbindet sich zum Server und initialisiert mit dem `ClientHello`-Paket die TLS-Verbindung [19], die unter Anderem eine Zufallszahl R_C enthält.

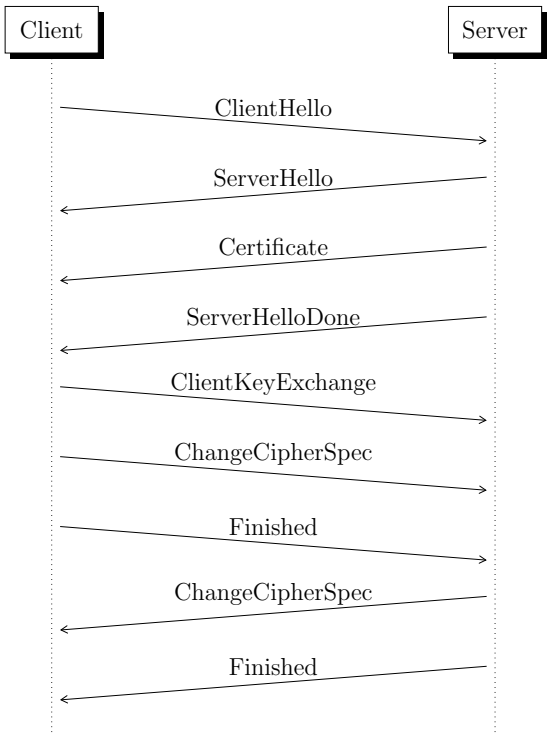


Abbildung 2.2: TLS Handshake [3]

Feld	Beschreibung
`ProtocolVersion`	Aktuellste vom Client unterstützte TLS-Version
`Random`	Vom Client generierte Zufallszahl R_C zur Erzeugung von Schlüsselmaterial
`SessionId`	ID einer bestehenden Sitzung, falls der Client die Sitzung wiederaufnehmen will, ansonsten bleibt dieses Feld leer
`CipherSuites`	Liste aller vom Client unterstützten Cipher Suites
`CompressionMethods`	Liste aller vom Client unterstützten Kompressionsverfahren
`Extensions`	TLS-Erweiterungen, die der Client unterstützt bzw. benutzen will

Tabelle 2.2: Inhalte des ClientHello-Pakets beim TLS-Handshake

Nachdem das `ClientHello`-Paket mit den Inhalten aus Tabelle 2.2 gesendet wurde, wählt der Server aus den vom Client unterstützten Verfahren aus. Er antwortet entsprechend mit einem `ServerHello`-Paket [19]. Auch der Server wählt eine Zufallszahl R_S aus und sendet sie.

Feld	Beschreibung
`ProtocolVersion`	Aktuellste TLS-Version, die sowohl vom Client, als auch vom Server unterstützt wird
`Random`	Auch der Server generiert eine Zufallszahl R_S, die später in die Schlüsselberechnung einfließt
`SessionId`	ID einer bestehenden Sitzung, falls eine Sitzung wiederaufgenommen werden soll, ansonsten generiert der Server eine neue Session-ID
`CipherSuite`	Vom Server ausgewählte Cipher Suite, die sowohl Client als auch Server unterstützen.
`CompressionMethod`	Ausgewähltes Kompressionsverfahren, das von Client und Server unterstützt wird
`Extensions`	TLS-Erweiterungen

Tabelle 2.3: Inhalte des ServerHello-Pakets beim TLS-Handshake

Nach dem `ServerHello`-Paket mit den Inhalten aus Tabelle 2.3 authentifiziert sich der Server mit einem X.509-Zertifikat beim Client. Anschließend beendet der Server diesen Protokollschritt mit dem `ServerHelloDone`-Paket [20].

Da in diesem Beispiel RSA zum Schlüsselaustausch verwendet wird, sendet der Client danach im `ClientKeyExchange`-Paket eine von ihm generierte Zufallszahl, das Pre-Master Secret Pre. Beide Kommunikationsteilnehmer bilden nun aus dem Pre-Master Secret, den beiden Zufallszahlen im `ClientHello`-/`ServerHello`-Paket und der Zeichenkette "master secret" das Master Secret Mas mithilfe einer Pseudozufallsfunktion PRF.

$$Mas := PRF(Pre, R_C, R_S, "master\ secret") \tag{2.5}$$

`ChangeCipherSpec` ist das letzte unverschlüsselte Paket, mit dem dem Gegenüber mitgeteilt wird, dass alle Nachrichten ab dem nächsten Paket mit dem Master Secret aus Formel 2.5 verschlüsselt werden. Mit dieser Nachricht und dem verschlüsselten `Finished`-Paket schließen Server und Client das TLS Handshake Protocol ab.

2.2.1 Problematiken von TLS

Bei TLS wird die Authentizität mit X.509-Zertifikaten gewährleistet, indem deren Echtheit über eine CA erfolgreich verifiziert wird [19]. Zur Feststellung der Vertrauenswürdigkeit einer CA sind mehrere Faktoren zu betrachten.

Von einer CA wird erwartet, dass sie ausreichend gegen Kompromittierungen bzw. Angriffe geschützt ist. In der Vergangenheit war zu sehen, dass dies oft nicht der Fall war. Große Aufmerksamkeit wurde z.B. 2011 den sehr verbreiteten CAs Comodo [21] und DigiNotar [22] zuteil. In beiden Fällen wurden gefälschte Zertifikate ausgestellt, die anschließend für Man-in-the-Middle-Angriffe benutzt werden konnten. Das größte Problem an der X.509-PKI ist, dass eine CA jede beliebige Identität bestätigen kann. Stellt ein Angreifer ein Zertifikat aus, das seinen öffentlichen Schlüssel mit einer fremden Identität verknüpft, so kann diese Fälschung technisch nicht ohne weitere Hilfsmittel festgestellt werden. Somit wird der Identitätsnachweis durch die X.509-PKI komplett außer Kraft gesetzt [23].

Ein zweiter Faktor ist der Einfluss von Behörden auf CAs, der je nach Land ausgeübt werden kann. In Frankreich wurde 2013 ein gefälschtes Google-Zertifikat in Umlauf gebracht. Die Fälschung wurde von einer CA ausgestellt, die in Verbindung zu einer Behörde für Sicherheit von Informationssystemen steht [24].

Das Vertrauen von Benutzern in die verwendete Software ist als dritter Faktor zu nennen. Beim *SSL Observatory Project* [5] wurde festgestellt, dass gewisse Web-Browser ohne Zustimmung des Benutzers über 600 CAs vertrauen. Schafft es ein Angreifer, dass eine dieser CAs ein gefälschtes Zertifikat ausstellt, so vertraut der Browser auch diesem Zertifikat [25]. Diese *Trust Stores*, eine Ansammlung von "vertrauenswürdigen" Zertifikaten, werden auch in anderen Software-Paketen wie Mailservern mitgeliefert.

Schwache Cipher Suites

Nachdem der Aufbau und Sinn von Cipher Suites bereits geklärt wurde, wird im Folgenden etwas genauer auf schwache Cipher Suites eingegangen.

NULL Cipher Suites: Initial wird eine CipherSuite ohne Schlüsselaustauschalgorithmus oder Datensicherungsalgorithmen verwendet, da diese beim Verbindungsaufbau noch nicht festgelegt wurden [19]:

`TLS_NULL_WITH_NULL_NULL`

Diese Cipher Suite kann aber bei schlechter Konfiguration auch bei der Kommunikation selbst verwendet werden. Es gibt auch andere schwache Cipher Suites, die `NULL` nur an bestimmten Stellen enthalten. Mit dem nachfolgenden Beispiel wird der Schlüssel mit dem asymmetrischen Verfahren RSA ausgetauscht, allerdings findet danach keine Verschlüsselung statt:

`TLS_RSA_WITH_NULL_MD5`

Cipher Suites mit Verfahren wie `DH_anon` im folgenden Beispiel handeln den Schlüssel ohne Authentifizierung, also anonym, aus, womit die Identität des Gegenübers nicht festgestellt wird:

`TLS_DH_anon_WITH_RC4_128_MD5`

Kommunikation, die mit diesen Cipher Suites stattfindet, ist anfällig für Man-in-the-Middle-Angriffe [26].

Export Cipher Suites: Durch Regulierungen durften kryptographische Algorithmen ab einer festgelegten Stärke nicht mehr aus den USA exportiert werden [27]. Für diesen

Zweck wurden Export Cipher Suites für TLS standardisiert, die nur schwache Algorithmen enthalten. Nachfolgend ist eine solche Cipher Suite, die den Verschlüsselungsalgorithmus RC2 mit einer Schlüssellänge von 40 Bit verwendet, zu sehen. Dieser wurde bereits gebrochen [28]:

`TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5`

2.2.2 Simple Mail Transfer Protocol (SMTP) und TLS

Das *Simple Mail Transfer Protocol* (SMTP) wurde 1982 entwickelt, um effizient und zuverlässig Nachrichten zu übertragen [29]. Da zu dieser Zeit keine Bedrohungen präsent waren, wurde der Schwerpunkt nicht auf Sicherheit gelegt. So waren alle Mail-Server *Open Relays* [30], d.h. sie haben jede Mail auf Port 25 ohne Authentifizierung in Empfang genommen und verarbeitet [31]. Mit der Zeit wurde dieses Standardverhalten ausgenutzt, um Spam-Mails oder Schadcode in großen Mengen zu verbreiten [30]. Um dieses Problem einzudämmen, wurde mit RFC 2476 der Port 587 geschaffen [32], für den festgelegt wurde, dass Mails nur nach einer Authentifizierung durch den Benutzer angenommen werden. Damit wurde das Versenden einer Mail durch einen Benutzer und der Transport von Mails zwischen Servern technisch voneinander getrennt. Der Transport von Mails zwischen SMTP-Servern geschieht auch heute noch über Port 25. Um SMTP per TLS zu verschlüsseln wurde von der Internet Assigned Number Authority (IANA) der Port 465 reserviert. Der Port wurde allerdings nie mit einem RFC standardisiert [31]. Er wurde schon nach kurzer Zeit abgeschafft, als die SMTP-Erweiterung `STARTTLS` standardisiert wurde [33].

Der Port 25 wird heute vor allem zur Kommunikation zwischen SMTP-Servern verwendet. Obwohl der Port 465 längst nicht mehr benutzt werden sollte, wird er aufgrund von Altsystemen immer noch häufig unterstützt. Zum Mail-Versand steht der Port 587 zur Verfügung [31].

Problematiken von TLS bei SMTP

Mails werden noch immer größtenteils unverschlüsselt übertragen [34].

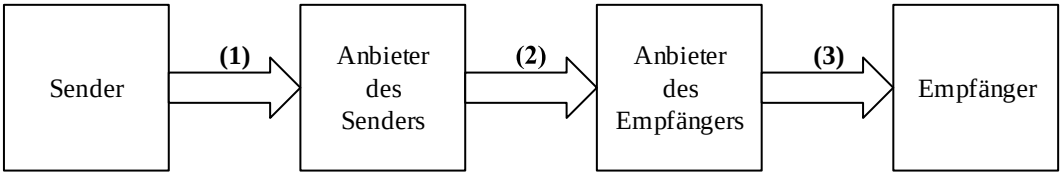


Abbildung 2.3: Übertragung einer Mail

Wie in Abb. 2.3 zu sehen ist, findet diese Übertragung in mehreren Schritten statt. Der Absender einer Mail hat zwar häufig die Möglichkeit den Transport zum Anbieter **(1)** zu verschlüsseln, allerdings hat er keinen Einfluss darauf wie die Mail anschließend zum Empfänger gelangt. Dazu wäre erforderlich, dass der Anbieter des Absenders die Mail verschlüsselt überträgt **(2)**. Außerdem müsste die Mail auch vom Anbieter des Empfängers zum Empfänger **(3)** verschlüsselt übertragen werden. Empfänger und Sender können also nicht wissen, ob die Mail beim kompletten Transport verschlüsselt wurde.

Hinzu kommt, dass TLS nicht von allen SMTP-Servern unterstützt wird [35]. Daher ist es nötig, dass eine Mail auch ohne Verschlüsselung zum SMTP-Server des Empfängers übertragen werden kann.

Ein weiteres Problem ist die Verifikation des Zertifikats über die X.509-PKI auf dem Transportweg einer Mail **(2)**. Dies ist nur möglich, wenn der CA vertraut wird, die das Zertifikat auch signiert hat. Würden Mailserver die Authentizität des Gegenübers mit Trust Stores sicherstellen, hängt das positive Ergebnis einer Verifikation davon ab, ob das Zertifikat des Kommunikationspartner von einer vertrauten CA signiert wurde. Falls es sich aber um ein selbstsigniertes Zertifikat handelt oder um eine unbekannte CA, stellt sich die bisher nicht beantwortete Frage, was mit der Mail gemacht wird. Man-in-the-Middle-Angriffe können nicht erkannt werden, weil die Identität des Gegenübers nicht festgestellt werden kann. Aus diesem Grund nehmen Mailserver Mails unabhängig von verwendeten Zertifikaten an.

2.3 Verifikationsprotokolle

Um Man-in-the-Middle-Angriffe zu vermeiden, muss die Authentizität der jeweiligen Kommunikationspartner bei jeder Übertragung festgestellt werden. Wie in 2.1 erwähnt, hat sich die X.509-PKI zur Authentifizierung von Teilnehmern etabliert. Zusätzlich zur X.509-PKI wurden weitere Techniken zur Authentifizierung von Teilnehmern entwickelt, auf die im Folgenden eingegangen wird.

2.3.1 Inter Mail Provider Trust (IMPT)

Die Deutsche Telekom und United Internet haben mit der Initiative *E-Mail made in Germany* (EmiG) festgelegt, dass zwischen Teilnehmern des Verbundes Mails nur verschlüsselt übertragen werden [36]. Später haben sich der Initiative auch weitere deutsche Partner angeschlossen [37]. Innerhalb des Verbundes wird Nutzern zugesichert, dass Mails zwischen den sendenden und empfangenden Providern mit TLS verschlüsselt werden. Findet Kommunikation mit Providern statt, die nicht im Verbund sind, wird versucht mit `STARTTLS` zu verschlüsseln. Falls dieser Provider keine `STARTTLS`-Verschlüsselung unterstützt, wird die Mail beim Transport nicht verschlüsselt [38].

Für EmiG wurden verschiedene Möglichkeiten geprüft, um die Authentizität von SMTP-Servern zu prüfen. Da andere Protokolle zu diesem Zeitpunkt noch nicht einsatzbereit waren, wurde das Verfahren *Inter Mail Provider Trust* (IMPT) entwickelt. Das Verfahren setzt voraus, dass jeder Teilnehmer von EmiG eine Infrastrukturbeschreibung zur Verfügung stellt, die z.B. Domainnamen, IP-Adressen und TLS-Zertifikate enthält. Zur Zertifizierung eines neuen Partners wird diese Beschreibung geprüft und anschließend an alle anderen EmiG-Partner verteilt. Beim Senden einer Mail werden Zertifikate, wie bei TLS-Verbindungen üblich, über die X.509-PKI validiert. Zusätzlich wird dabei geprüft, ob eine für IMPT nötige Infrastrukturbeschreibung des Servers vorliegt [39].

Bei der Umsetzung von EmiG wurde auch das Protokoll *DNS-based Authentication of Named Entities* (DANE) zu Zwecken der Verifikation geprüft. Zum Zeitpunkt der Entwicklung von EmiG war DANE allerdings noch nicht in populären Mailservern implementiert, weshalb dieser Standard nicht in Frage kam [39].

2.3.2 DNS-based Authentication of Named Entities (DANE)

Hinter DANE [40] steckt die Idee, TLS-Zertifikate unter Verwendung der *DNS Security Extensions* (DNSSEC) zu validieren.

DNS wurde als dezentrales System entworfen, d.h. jede Zone kann unabhängig von anderen auf einem Nameserver verwaltet werden. Die DNS-Zonen-Konfiguration beinhaltet verschiedene Records. Beispielsweise wird der Record-Typ `A` benutzt, um einer Domain IP-Adressen zuzuordnen. Mit dem Record-Typ `NS` wird auf Nameserver verwiesen, die darunterliegende Zonen verwalten [41].

Mit Records, die zur Umsetzung von DNSSEC standardisiert wurden, wurde auf Basis der DNS-Hierarchie eine PKI aufgebaut. In den folgenden Absätzen wird kurz auf die DNS-Problematik eingegangen, die Grund für die Entwicklung von DNSSEC war. Darauf folgt eine kurze Erklärung des für DANE notwendigen Record, um auch X.509-Zertifikate über DNSSEC zu verifizieren.

Man in the Middle bei DNS

DNSSEC wurde entwickelt, um die Authentizität und Datenintegrität von DNS zu sichern und damit Angriffen vorzubeugen [42, 43].

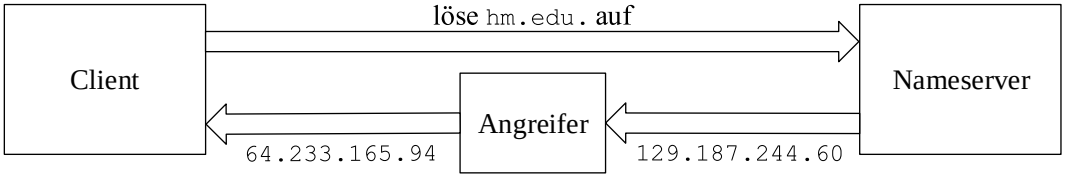


Abbildung 2.4: Man-in-the-Middle-Angriff bei einer DNS Response

Man stelle sich vor, das DNS-Antwort-Paket wird vom Nameserver zum DNS-Client geschickt. Ein Angreifer fängt das Paket ab, verändert dessen Inhalt und sendet es weiter. Beispielsweise könnte er bei der Auflösung von "`hm.edu.`" die IP-Adresse von `129.187.244.60` auf `64.233.165.94` verändern. Da der Client wie in Abb. 2.4 die falsche IP-Adresse erhält, verbindet er sich anschließend unbewusst auf den falschen Server.

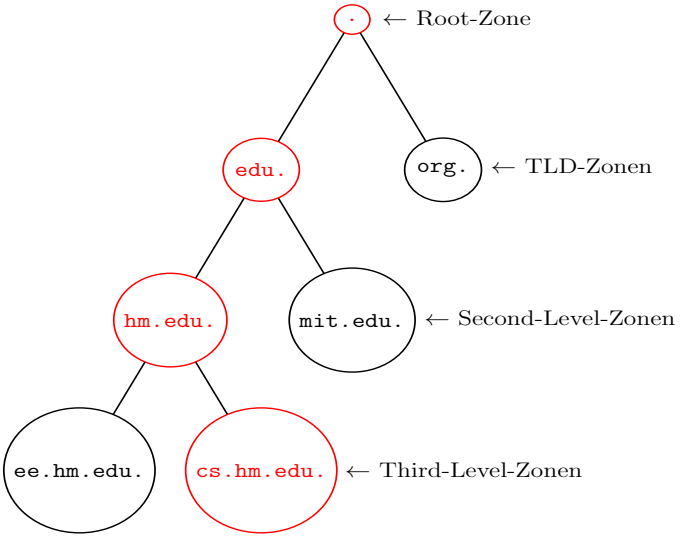


Abbildung 2.5: Hierarchische Darstellung von DNS

Um Veränderungen von Datenpaketen auf dem Weg zum Ziel festzustellen, werden digitale Signaturen benutzt. Zudem wird die DNS-Hierarchie aus Abb. 2.5 genutzt, um eine Public-Key-Infrastruktur herzustellen [42].

Mit der Begründung, dass DNS-Daten öffentlich seien, wurde bei der Entwicklung von DNSSEC explizit nicht auf die Vertraulichkeit der Daten geachtet. Stattdessen war ausschließlich die Sicherstellung der Integrität und Authentizität von gesendeten DNS Records die Motivation für die Entwicklung [44].

Record-Typ	**Payload**
`DNSKEY`	Öffentlicher Schlüssel dieser Zone
`RRSIG`	Signatur für ein Record Set
`DS`	Hash-Wert eines öffentlichen Schlüssels einer untergeordneten Zone

Tabelle 2.4: Überblick über DNSSEC Records

Die für DNSSEC notwendigen DNS Records aus Tabelle 2.4 wurden im RFC 4033 [45] definiert und werden nachfolgend genauer beschrieben.

DNSKEY Record

Da DNSSEC mit digitalen Signaturen, also mit asymmetrischer Kryptographie, arbeitet, wird pro Zone ein Schlüsselpaar generiert. Mit dem privaten Schlüssel werden Records signiert. Anschließend können diese Signaturen mit dem öffentlichen Schlüssel verifiziert werden. In Listing 2.1 ist ein DNSKEY Record der Domain `fedoraproject.org.` zu sehen.

```
1 fedoraproject.org.  49  IN  DNSKEY  256 3 5 AwEAAcCWNQWl5pCI3iOOP2r8nStL60Zjb/2
      JQLQytamVap0L44z0YWft u7pu0hx3cnIM1ejQOsEwbg2/10IyC+38cYqJDXbSdFg1zGztOS5xNz7r 9
      hzSRK5N2jkycdJ/BoByJ4Y+XGpDqfG4I97++8sIzSrw60TmGAKTvM9v iL3ByeCN
```

Listing 2.1: Ein DNSKEY Record der Domain fedoraproject.org.

Ein DNSKEY Record kann als Payload entweder einen *Key Signing Key* (KSK) oder einen *Zone Signing Key* (ZSK) enthalten. Der ZSK wird üblicherweise zur Signatur aller Record Sets genutzt. Nur die DNSKEY Record Sets werden mit dem KSK signiert. Mit diesem Schlüssel kann mithilfe der DNS-Hierarchie in Abb. 2.5 eine Vertrauenskette (*Trust Chain*) aufgebaut werden.

RRSIG Record

Eine Ansammlung mehrerer DNS Records des gleichen Resource-Record-Typs werden als Record Set bezeichnet. Jedes Record Set kann mit dem Record-Typ RRSIG (*Resource Record Signature*) signiert werden. In Listing 2.2 sind vier A Records zu sehen, die mit einem RRSIG Record signiert wurden.

```
1 fedoraproject.org.  53  IN  A 209.132.181.16
2 fedoraproject.org.  53  IN  A 213.175.193.206
3 fedoraproject.org.  53  IN  A 85.236.55.6
4 fedoraproject.org.  53  IN  A 152.19.134.142
5 fedoraproject.org.  53  IN  RRSIG A 5 2 60 20150227030514 20150128030514 7725
      fedoraproject.org. N8ASD7YWfa84/U9ECVV4vM9Olgoi238ReXH7Q4+tGX4npcZ9g4BpGJWX
      ABI5TeZNndPAJT+AKffim1xCjgQ9KvF1ZGVaAiYCQqlQgQ/mwYPijI5h
      dkxuvj9TuU29Sm01BkleesIUe4sWAo1LPCozhwYktOM7y8jnuiTD45kF BRI=
```

Listing 2.2: A Record mit zugehörigem RRSIG Record der Domain fedoraproject.org.

DS Record

Der DS Record (*Delegation Signer*) beinhaltet den Hashwert eines öffentlichen Schlüssels, der darunterliegenden Zone, um auf diese zu verweisen. DNSSEC wurde so konzipiert,

dass mithilfe der DNS-Hierarchie in Abb. 2.5 eine Trust Chain aufgebaut werden kann. Dafür stellt die höherliegende Zone einen oder mehrere DS Records der darunterliegenden Zone zur Verfügung.

Beispielhaft sind in Listing 2.3 die DS Records der Zone `fedoraproject.org.` zu sehen. Sie stammen von einem Nameserver der `org.`-Zone und sind demnach mit dem privaten Schlüssel der `org.`-Zone signiert (vgl. RRSIG Record).

```
1 fedoraproject.org.  73332 IN  DS  63721 5 2 0
      DF6AEC61BE8F2579A1D0A7C974A6C20E692B94EB070F9212E8AF618 805DCDC3
2 fedoraproject.org.  73332 IN  DS  301 5 2 17
      D5A4A61B88AF2F6F7BA9B6F952354B4C4E1487040168949D64D3F7 5DE59968
3 fedoraproject.org.  73332 IN  DS  16207 5 2
      A7C9BF5AFE374C9650ED678F3D36931A7DE9256B86A7BC34D6DEED7D 4E492E5E
4 fedoraproject.org.  73332 IN  RRSIG DS 7 2 86400 20150215155354 20150125145354 53348
       org. jlNSXre2gS7zJxBaBNhI3KYV7OOr17TBuZDgiOinLkaM4bfyslqWB4z/ q0srYNu+
      EtzDlZeH6i3LUSorrtYLrt4JDxH5VHtTvQqB6OqlI2lhAUmV
      vtBfBmcbUEKEDaj2XGPkKHYSbcqaSUa7pld34xTsRjNQiBiOH5pqvQDU Vzo=
```

Listing 2.3: DS Record mit zugehörigen RRSIG Record der Domain fedoraproject.org.

Verifikation mit DANE

Die beschriebene DNSSEC-PKI kann als Alternative oder zusätzlich zur X.509-PKI verwendet werden. Zu diesem Zweck wurde `TLSA` als neuer DNS Record eingeführt, der wie andere Records in der Zone angelegt wird. Er enthält das TLS-Zertifikat oder einen Fingerprint des Zertifikats, welcher wiederum per DNSSEC (mit dem RRSIG Record) signiert wird.

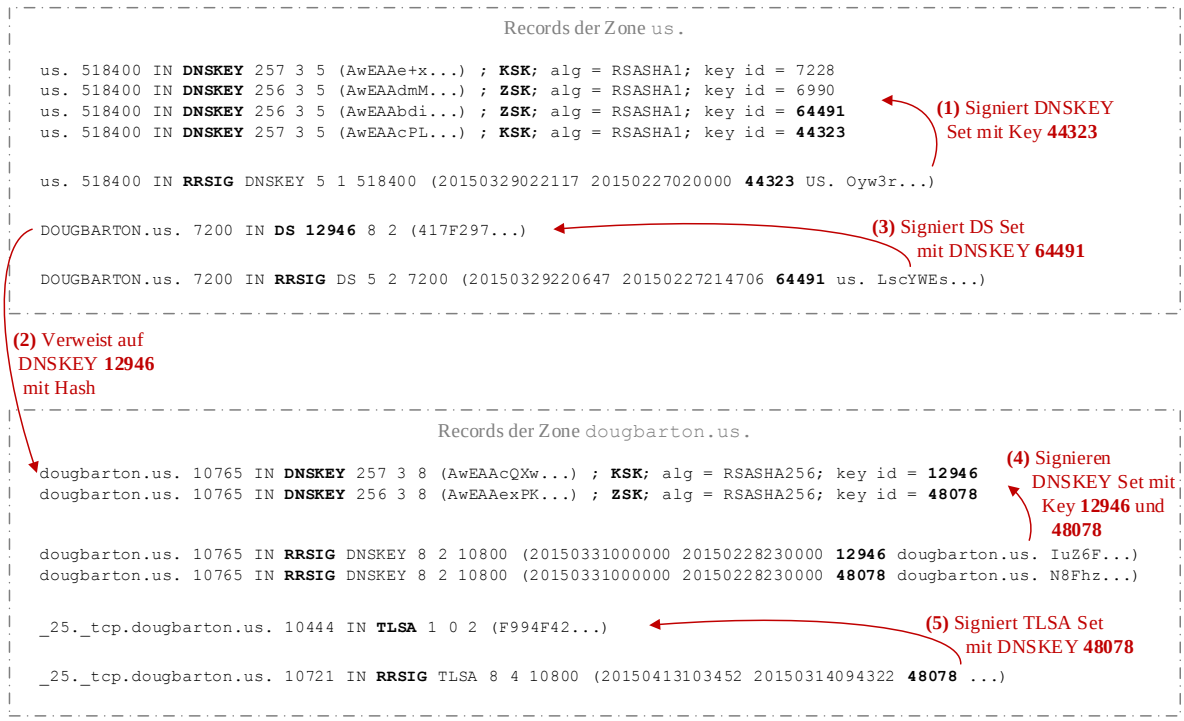


Abbildung 2.6: Beispielhafte Trust Chain von DANE/DNSSEC

In Abb. 2.6 ist zu sehen, wie die Trust Chain der Domain `dougbarton.us.` verifiziert wird, um die Echtheit des Mailserver-Zertifikats nachzuweisen. Der *Trust Anchor* ist die oberste Instanz einer Trust Chain. In diesem Fall muss der DNSKEY `44323` im vornherein als Trust Anchor bekannt sein, um das Vertrauensverhältnis zu darunterliegenden Zonen abzuleiten.

Diese Zone verfügt über mehrere DNSKEY Records, die ein Record Set bilden. Ist die Signatur des DNSKEY Record Sets im darauf folgenden RRSIG Record korrekt **(1)**, so kann auch allen anderen DNSKEY Records vertraut werden. Der DS Record darunter enthält einen Hash, der aus dem DNSKEY Record der Zone `dougbarton.us.` generiert wurde **(2)**. Auch dieser DS Record wurde mit einem der Schlüssel der Zone `us.` **(3)** signiert, das heißt, dass diesem DNSKEY Record der Zone `dougbarton.us.` vertraut werden kann.

Wie in der TLD-Zone wird als Nächstes auch das DNSKEY Record Set der Second-Level-Zone mit Signaturen versehen **(4)**. Auf der Abbildung ist auch die unterschiedliche Verwendung von ZSK und KSK erkennbar. Nach dem TLSA Record folgt dessen Signatur

mit dem ZSK im entsprechenden RRSIG Record **(5)**. Der TLSA Record beginnt nach RFC 6698 [40] mit der Bezeichnung des TLS-Dienstes:

`_<Port>._<Transport Protocol>.<Domain>`

Darauf folgen die für DNS Records üblichen Parameter *Time to Live* (TTL), die Record Class und der Record-Typ `TLSA` [41]. Direkt dahinter wird der Inhalt und die Bedeutung des TLSA Record genauer beschrieben [40].

Feld	**Beschreibung**
`Certificate usage`	Dieses Feld beschreibt die Art des X.509-Zertifikats, um das es sich handelt. Dabei wird festgelegt, ob es sich um das Zertifikat einer CA, ein TLS-Zertifikat, einen Trust Anchor oder um ein selbstsigniertes Zertifikat für eine Domain handelt.
`Selector`	Mit dem Selector wird festgelegt, wie auf das verwendete Zertifikat verwiesen wird. In Abb. 2.6 steht an dieser Stelle eine `1`, was bedeutet, dass es sich beim Payload des TLSA Record nur um den öffentlichen Schlüssel des Zertifikats handelt.
`Matching Type`	Der Matching Type definiert die Art des Payloads. Der Matching Type klärt den Inhalt des Payloads. Es handelt sich entweder um den kompletten mit dem Selector gewählten Inhalt oder um einen Hash des Inhalts.
`Certificate Association Data`	Hierbei handelt es sich um den Payload, dessen Art mit den vorherigen Feldern festgelegt wurde.

Tabelle 2.5: Inhalte eines TLSA Record

In Tabelle 2.5 ist der Payload eines TLSA Record zu sehen.

DANE hat einige signifikante Vorteile gegenüber der X.509-PKI. Anders als bei X.509 werden Schlüssel ohne Umwege direkt an die DNS-Einträge geknüpft. Bei der X.509-PKI werden Schlüssel lediglich einer nicht weiter bestätigten Zeichenkette zugeordnet. Bei DANE muss der Schlüssel einer Domain von der direkt darüberliegenden Zone signiert werden. Gegenüber der X.509-PKI hat die unterzeichnende Instanz also nur die Möglichkeit seine eigenen Subzonen zu beeinflussen. Für andere Domains bzw. deren Subzonen besteht im Fall eines Angriffs damit keine Gefahr [40].

Unterstützt ein Provider DANE, gibt es zwei Möglichkeiten falls die Verifikation fehlschlägt. In Anwendungsprogrammen kann dem Benutzer mitgeteilt werden, dass es sich

um eine potenziell unsichere Verbindung handelt. Alternativ kann die Verbindung aus Sicherheitsgründen komplett abgebrochen werden [46]. Ein solcher Verbindungsabbruch ist vor allem bei Systemen wie SMTP sehr interessant, weil bei Mails auf dem Transportweg keine Benutzerinteraktion möglich ist. Wird beim Transport einer Mail festgestellt, dass der TLSA Record nicht zum verwendeten TLS-Zertifikat passt, besteht die Möglichkeit den weiteren Transport zu unterbinden. Diese Technik wird bereits von Mailservern wie Postfix unterstützt [47].

2.4 Portscanner

Um aktive Scans zu ermöglichen, müssen vor der Sammlung über TLS-Konfigurationen verfügbare SMTP-Server gefunden werden. Ein Portscanner schickt mithilfe des *Transmission Control Protocol* (TCP) ein SYN-Paket an den Port einer bestimmten IP. Läuft ein Service auf diesem Port, so wird eine Verbindung akzeptiert, also antwortet das System mit einem SYN-ACK-Paket.

Als Portscanner kommen verschiedene Software-Pakete in Frage. Ein vor allem für Systemadministratoren interessanter Netzwerkscanner ist *nmap* [48], der umfangreiche Konfigurationsmöglichkeiten bietet. Allerdings wurde bei Testläufen festgestellt, dass der Scanner *ZMap* [49] um den Faktor 1300 schneller ist. Die Software *Masscan* [50] wurde entwickelt, um nach offenen Ports in großen IP Ranges zu suchen. Bei einer Gegenüberstellung wurde festgestellt, dass Masscan gegenüber ZMap einige Nachteile hat. Zum einen ist ZMap performanter als Masscan, zum anderen werden zusammengehörige IP Ranges durch Ziel-Randomisierung von ZMap nicht überlastet [51].

3 Verwandte Arbeiten

Eine groß angelegte Untersuchung aktueller TLS-Konfigurationen von SMTP-Installationen konnte bei der Recherche nicht gefunden werden. In verschiedenen Projekten gab es aber Untersuchungen in Bezug auf das Protokoll HTTPS.

Mit der Veröffentlichung *Cryptographic Strength of SSL/TLS Servers: Current and Recent Practices* [52] wurden Webserver auf deren TLS-Unterstützung geprüft. Betrachtet wurde die Unterstützung verschiedener SSL und TLS-Versionen, sowie die statistische Verteilung der verwendeten kryptographischen Algorithmen in Bezug auf Schlüsselaustausch, Authentifizierung, Hash-Funktionen und Verschlüsselung. Außerdem wurde getestet, welche Mechanismen vom Server ausgewählt werden und welche Schlüssellängen verwendet werden. Im Gegensatz zur vorliegenden Arbeit wurden nicht Mailserver, sondern ausschließlich die 19.000 bekanntesten Webseiten geprüft.

Es gab außerdem einige Projekte, die Zertifikate passiv und aktiv gesammelt haben. Beispielsweise wurden im Rahmen des *SSL Observatory Project* [5] Zertifikate von Webservern analysiert. Auch andere Veröffentlichungen [53, 54] betrachten nur das Protokoll HTTPS.

Größere und vor allem regelmäßige HTTPS-Scans wurden vor allem von zwei Projekten gemacht. Das Entwickler-Team des Portscanners ZMap veröffentlicht Scan-Ergebnisse, die teilweise sogar täglich durchgeführt wurden [55]. Darunter sind auch Scans, die TLS-Zertifikate von Webservern sammeln oder TLS-Schwachstellen testen.

Ein weiteres Projekt, das sich nur auf die Analyse der TLS-Konfiguration von Webservern konzentriert, ist *Qualys SSL Labs* [56]. Auf der Webseite werden verschiedene Werkzeuge und Schriften angeboten. Eines dieser Werkzeuge ist der *SSL Server Test*, mit dem es möglich ist, einzelne Webserver auf dessen TLS-Konfiguration zu prüfen. Dieses Werkzeug beschränkt sich zwar nur auf das Protokoll HTTPS, testet TLS aber umfangreich auf

dessen Konfiguration [57]. Auf diesen Überprüfungen basierend gibt der *SSL Server Test* Empfehlungen zur Verbesserung der TLS-Konfiguration. Unter anderem werden dabei verwendete Zertifikate und unterstützte TLS-Versionen herausgefunden. Diese und weitere Überprüfungen werden mithilfe des *SSL Server Rating Guide* eingestuft [58]. So lässt sich eine vergleichbare Aussage über den Stand der Sicherheit eines Webservers treffen.

Mit dem *SSL Server Test* können ausschließlich Webserver auf deren TLS-Konfiguration geprüft werden. Außerdem ist es nur möglich, einzelne Server über die Weboberfläche zu prüfen. Große Mengen an TLS-Scans wären zwar mit den *SSL Labs APIs* [59] möglich, jedoch ebenfalls nur für Webserver.

Im Rahmen der *SSL Labs* werden außerdem regelmäßige Scans von TLS-Konfigurationen zu Statistiken zusammengefasst. Diese Daten werden in Form des *SSL Pulse* veröffentlicht [60] und werden in Kapitel 5 zum Vergleich benutzt.

Mit der OpenSource-Software *SSLyze* [61] ist es möglich, einen oder mehrere Server eines beliebigen Services auf dessen TLS-Konfiguration zu prüfen. Da SSLyze leicht erweiterbar und durch parallele Verarbeitung auch für groß angelegte Scans geeignet ist, wird es für diese Bachelorarbeit verwendet.

4 Datenerhebung

Für die durchzuführende Datenerhebung wird im Folgenden eine Umgebung entwickelt, welche SMTP-Server in bestimmten IP Ranges scannt. Bei diesen Scans werden SMTP- bzw. TLS-Verbindungen mittels SMTP-Ports aufgebaut, um die Zielsysteme hinsichtlich deren Transportsicherheit zu prüfen. Ein Zielsystem wird im Folgenden als eine IP mit dem entsprechenden Port definiert.

4.1 Anforderungen

Bei der Datenerhebung müssen nachfolgende Anforderungen erfüllt werden. Zuerst müssen vorher definierte IP Ranges nach IPs durchsucht werden, die einen offenen SMTP-Port (25, 465 oder 587) haben. Anschließend sollen Zertifikate gesammelt und über die X.509-PKI und DANE verifiziert werden. Zudem sollen TLS-Verbindungen aufgebaut werden, um Aussagen über verwendete TLS-Versionen, verwendete Cipher Suites und die Anfälligkeit auf die Heartbleed-Schwachstelle zu treffen. Da eine hohe Anzahl an Zielsystemen in begrenzter Zeit gescannt wird, sollte die Laufzeit möglichst linear ansteigen. Die gesammelten Daten sollen in einer Datenbank persistiert werden, die ein einfaches Einfügen und Suchen ermöglicht.

4.2 Ablauf

Aus den genannten Anforderungen wurde ein Scanning-Prozess entworfen.

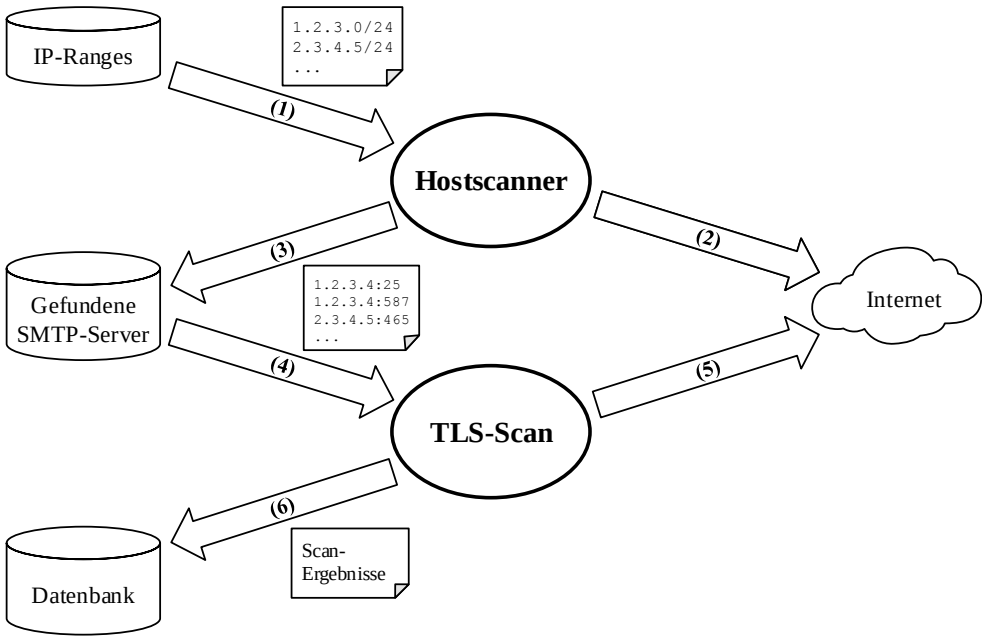


Abbildung 4.1: Überblick über einen Scan

Bei einem neuen Scan wird der **Hostscanner**, wie in Abb. 4.1 zu sehen, mit den IP Ranges gestartet **(1)**, in denen nach SMTP-Servern gesucht wird **(2)**. Alle gefundenen SMTP-Server werden abgespeichert **(3)** und dem **TLS-Scan (4)** übergeben. Im Rahmen des TLS-Scans werden Verbindungen zu SMTP-Servern aufgebaut und anschließend deren TLS-Konfiguration gescannt **(5)**. Die Ergebnisse werden in einer Datenbank abgelegt **(6)**.

4.3 Hostscanner

Wie in Kapitel 2.4 erläutert, ist es mit ZMap möglich, große IP Ranges sehr schnell nach offenen Ports zu durchsuchen. Den Anforderungen entsprechend werden Hosts wie in Abb. 4.1 **(2)** mit den folgenden drei SMTP-Ports gesucht: Port 25 für den SMTP-Transport und die Ports 465 bzw. 587 für SMTP-Submissions. Da ZMap darauf optimiert wurde,

große Netzwerke nur nach genau einem Port zu durchsuchen [62], wird parallel für alle Ports ein eigener ZMap-Prozess gestartet.

```
1 #!/bin/sh
2
3 IP_RANGES_FILE=$1
4 zmap -p 25 -w $IP_RANGES_FILE -b blacklist -o ../../results/zmap_output_25 &
5 zmap -p 465 -w $IP_RANGES_FILE -b blacklist -o ../../results/zmap_output_465 &
6 zmap -p 587 -w $IP_RANGES_FILE -b blacklist -o ../../results/zmap_output_587 &
7 wait
8 python format_and_prepare_results.py ../../results/ ../../results/hosts_to_scan
```

Listing 4.1: Hostscanner

Das Shellscript wird mit einer Datei als Parameter gestartet, in der alle IP Ranges stehen, die gescannt werden sollen. In Listing 4.1 werden nun zuerst die drei ZMap-Instanzen gestartet. Danach wird gewartet, bis alle Instanzen abgeschlossen sind (Zeile 7). Wichtig ist hier, dass bei jedem ZMap-Aufruf eine Blacklist-Datei übergeben wird. Diese ist nötig, um Netzwerke auszuschließen, die von der IANA für andere Zwecke reserviert wurden [63]. In den Dateien mit dem Namensmuster `zmap_output_<Port>` sind nun alle Host-IPs zu finden, die den jeweiligen Port geöffnet haben.

```
1  import os
2  import sys
3
4  # Get filenames
5  results_dir = sys.argv[1]
6  hosts_to_scan = sys.argv[2]
7  filenames = next(os.walk(results_dir))[2]
8
9  # Read files
10 output_path = hosts_to_scan
11 with open(output_path, 'w') as output_file:
12     for filename in filenames:
13         if 'zmap_output_' in filename:
14             file_path = results_dir + '/' + filename
15             with open(file_path) as input_file:
16                 port = str(filename).replace('zmap_output_', '')
17                 for line in input_file:
18                     if not 'saddr' in line:
19                         host = line.replace('\n', '')
20                         host = host + ':' + port
21                         output_file.write(host + '\n')
```

Listing 4.2: Aufbereitung der Hostscanner-Ergebnisse

Im Listing 4.2 (`format_and_prepare_results.py`) ist zu sehen, wie alle Ergebnis-Dateien ausgelesen und in der Datei `hosts_to_scan` im Format `<IP>:<Port>` aufgelistet werden. Wie in Abb. 4.1 zu sehen ist **(4)**, kann diese Datei nun dem TLS-Scan übergeben werden.

4.4 TLS-Scan

Für die Rolle des TLS-Scan in Abb. 4.1 wird die Software SSLyze aus zwei Gründen für diese Arbeit verwendet. Zum einen enthält SSLyze bereits einige TLS-Überprüfungen, zum anderen ist dessen Erweiterbarkeit ein großer Vorteil für Anpassungen. Standardmäßig kann SSLyze Scan-Ergebnisse als Text auf der Kommandozeile oder als Datei im XML-Format ausgeben.

Beim Start von SSLyze werden, nach dem Parsen der Kommandozeilenparameter, alle Zielsysteme (im Format `<Host>:<Port>`) auf ihre Verfügbarkeit getestet. Dabei wird für Port 25 und 587 eine SMTP-Verbindung aufgebaut, über die zuerst der SMTP-Befehl `EHLO` geschickt wird, um zu testen, ob der Server SMTP-Erweiterungen unterstützt [64]. Wird dieser Befehl vom Server angenommen, wird als Nächstes `STARTTLS` gesendet, um zu testen, ob TLS unterstützt wird. Falls einer dieser Befehle nicht angenommen wird, kann das Zielsystem nicht getestet werden [33]. Auf Port 465 ist zum Test der Verfügbarkeit lediglich eine TLS-Verbindung notwendig, um festzustellen, dass dieses Zielsystem gescannt werden kann.

Alle Zielsysteme, die diese ersten Überprüfungen bestanden haben, werden im nächsten Schritt auf deren TLS-Konfiguration geprüft.

Da Scans parallel abgearbeitet werden, eignet sich SSLyze besonders für die massenhafte Überprüfung von Zielsystemen.

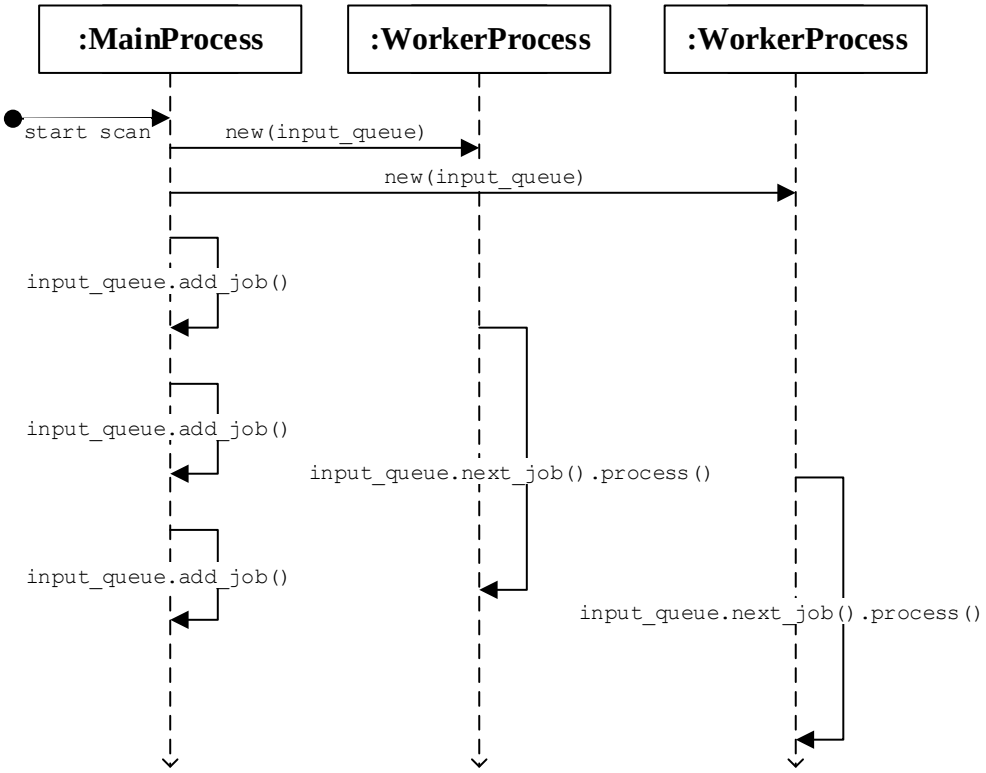


Abbildung 4.2: Parallele Verarbeitung mit SSLyze

In Abb. 4.2 ist zu sehen, wie mithilfe der Python-API `multiprocessing` [65] nach dem Start des Hauptprozesses mehrere parallele Prozesse gestartet werden. Der Hauptprozess befüllt nun die `input_queue`, die vorher an die Unterprozesse übergeben wurde. Diese arbeiten die Queue anschließend parallelisiert ab. Dieses Queue-System wird sowohl für die Verfügbarkeitsprüfung von Zielsystemen als auch für die darauf folgenden TLS-Überprüfungen mit Plugins verwendet.

4.4.1 Plugins

Neue Scans können bei SSLyze durch das Erstellen eines neuen Plugins hinzugefügt werden. Beim Start wird SSLyze mit den zu verwendenden Plugins und den Hosts, die überprüft werden sollen, aufgerufen. Mit dem folgenden Aufruf wird beispielsweise geprüft, ob das Zielsystem `8.8.8.8:465` anfällig für die Heartbleed-Schwachstelle ist, indem der Plugin-Parameter `--heartbleed` angegeben wird:

```
python sslyze.py --heartbleed 8.8.8.8:443
```

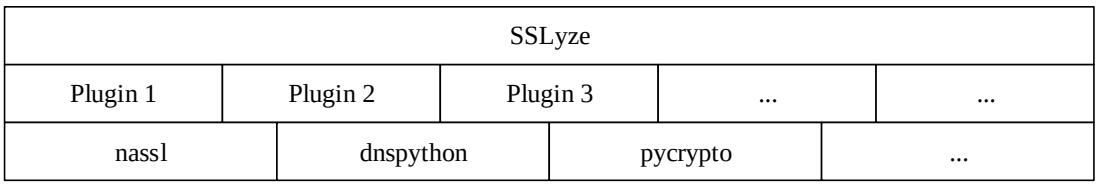


Abbildung 4.3: Software-Schichten von SSLyze

SSLyze besteht aus mehreren Schichten, in der die zweite, zu sehen in Abb. 4.3, aus Plugins besteht. Diese nutzen wiederum verschiedene Python-Bibliotheken wie `dnspython` oder `pycrypto`.

Die Python-interne TLS-Bibliothek [66] eignet sich nicht für SSLyze, weil sie nur für den Aufbau und die Verwendung von TLS-Verbindungen entwickelt wurde. Daher spielt die Bibliothek *Nassl* [67] eine besonders große Rolle. Sie wurde für SSLyze entwickelt, um TLS-Verbindungen transparent aufzubauen und zu verwalten. Mit *Nassl* wurde eine Schnittstelle geschaffen, die es erlaubt, tiefer in die Vorgänge des TLS-Protokolls einzugreifen.

SSLyze beinhaltet folgende Plugins, die für diese Arbeit verwendet werden können:

Plugin	**Beschreibung**
`CertInfo`	Liest das Zertifikat aus, validiert es gegen die Trust Stores von Mozilla, Microsoft, Apple und Java
`Heartbleed`	Prüft, ob der Server anfällig für die Heartbleed-Schwachstelle ist
`OpenSSLCipherSuites`	Prüft, welche TLS-Versionen mit welchen Cipher Suites unterstützt werden

Tabelle 4.1: Verwendete SSLyze-Plugins

Zusätzlich zu den Plugins in Tabelle 4.1 wurde ein weiteres Plugin entwickelt, das für die Datenerhebung verwendet wird.

4.4.2 Anpassungen

Für diese Arbeit wurde SSLyze um zwei Funktionen erweitert. Da DANE gegenüber TLS-Diensten einige Vorteile bietet, wurde ein Plugin zur Verifikation von DANE entwickelt. Als zweite Erweiterung wurde SSLyze um eine Datenbankunterstützung ergänzt.

DANE-Plugin

Eine DANE-Validierung findet statt, wenn ein TLSA Record im DNS vorhanden ist. In diesem Fall wird zuerst das TLS-Zertifikat aus einem Handshake extrahiert und mit dem TLSA Record validiert. Ist diese Validierung erfolgreich, so folgt eine komplette DNSSEC-Validierung bis hin zum Trust Anchor. Die dadurch nachgewiesene Trust Chain ist in Abb. 2.6 zu sehen.

Dieses Plugin wurde nach den Spezifikationen in den DANE- und DNSSEC-RFCs implementiert, um eine korrekte Verifikation durchzuführen.

Datenbankanbindung

SSLyze wurde im Rahmen dieser Arbeit so angepasst, dass Scan-Ergebnisse auch in eine Datenbank geschrieben werden können. Das heißt, dass in jedem Plugin alle Scan-Ergebnisse als Python-Dictionary gesammelt werden. Schließt das Plugin seine Arbeit für ein Zielsystem ab, werden alle Ergebnisse in einer Datenbank persistiert. Um die Datenbankverbindung komfortabel angeben zu können, wurde SSLyze um ein Kommandozeilenparameter ergänzt. Das Kommando mit dem Parameter zum Speichern in einer Datenbank sieht wie folgt aus:

```
python sslyze.py --db=mongodb://localhost:27017/ --sslv2 8.8.8.8:443
```

Das Datenbanksystem sollte ein einfaches Einfügen und Suchen der gesammelten Daten ermöglichen. Zuerst wurde der Suchserver *Elasticsearch* [68] als Speichersystem in Erwägung gezogen, allerdings konnte schnell festgestellt werden, dass Suchkriterien auf eingebetteten Dokumenten nur möglich sind, wenn die Datenhierarchie vorher festgelegt wurde [69]. Da eine Strukturierung sehr aufwendig wäre, sind für diesen Einsatzzweck auch keine SQL-Datenbanken geeignet. Die Entscheidung fiel daher auf die NoSQL-Datenbank

MongoDB [70], weil diese eine einfache Installation bietet, sowie die Anforderungen erfüllt.

4.5 Organisatorisches und Infrastruktur

Bei der Organisation eines Scans, der große Bereiche des Internets betrifft, sind besondere Vorbereitungen nötig [49, 63]. Die Entwickler von ZMap haben ihre Erfahrungen mit solchen Scans in einer Veröffentlichung dokumentiert [49]. Deren Empfehlungen für die Vorbereitung und Durchführung von globalen Scans wurden wie nachfolgend beschrieben befolgt.

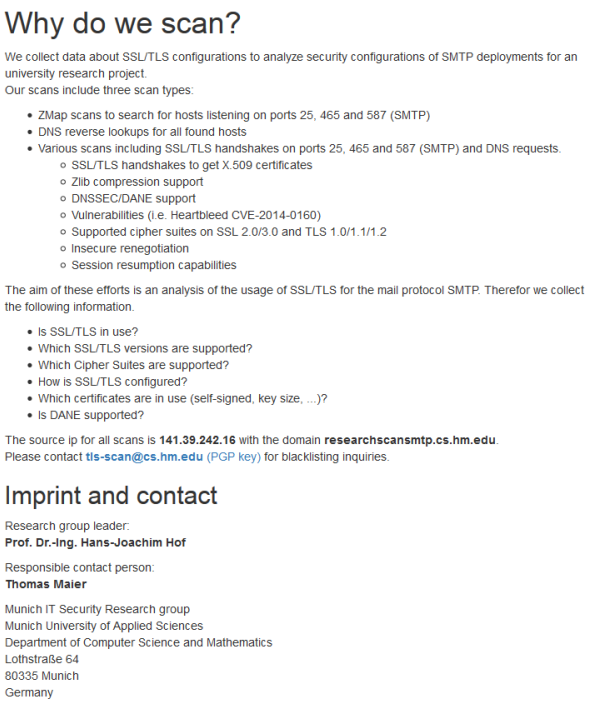
Für die Scans wurde im Rahmen der *Munich IT Security Research Group* (MuSe) ein Server zur Verfügung gestellt. Der Server befindet sich an der Fakultät für Informatik der Hochschule München und damit innerhalb der Infrastruktur, für die die zentrale IT-Abteilung der Hochschule verantwortlich ist. Diese ist über das *Leibniz-Rechenzentrum* (LRZ), das Teil des *Deutschen Forschungsnetzes* (DFN) ist, an das Internet angebunden.

Administratoren von Zielsystemen werden laut dem ZMap-Paper während der Durchführung auf Scans aufmerksam und wenden sich häufig an den im *Whois*-System gelisteten *Abuse contact*. Daher wurden Kontakte zur IT-Abteilung, zum LRZ und zum DFN bereits vor Beginn der Scans geknüpft, um Absprachen zu treffen. Involviert waren dabei sowohl der Verantwortliche der fakultätsinternen Infrastruktur, als auch die Verantwortlichen für Netzplanung der zentralen IT-Abteilung und die Abuse-Teams des LRZ und DFN. Mit diesen Kontakten wurden der Beginn und Ablauf der Scans genauer abgestimmt. Dabei wurden im Rahmen der Arbeit Standardantworten in deutscher und englischer Fassung an das LRZ weitergegeben, die genutzt wurden, um auf Beschwerden zu reagieren. Sollte eine Standard-Antwort auf eine Beschwerde nicht ausreichen, wird diese an eine Adresse weitergeleitet, die für diese Arbeit erstellt wurde. Genauso wurde verfahren, falls ein System von zukünftigen Scans ausgeschlossen werden sollte (Blacklisting). Um eine komplette Sperrung der IP des Scan-Systems durch das LRZ oder DFN vorzubeugen, wurde außerdem das *Computer Emergency Response Team* (CERT) des DFN über die Scanning-Aktivitäten informiert.

Nach den Empfehlungen des ZMap-Papers wurde ein Reverse-DNS-Eintrag angelegt, aus dem sofort ersichtlich ist, dass es sich um akademische Forschung im Bereich SMTP handelt:

`researchscansmtp.cs.hm.edu`

Da davon auszugehen ist, dass Administratoren von Zielsystemen nach einer Webseite suchen, um den Sinn der Scans in Erfahrung zu bringen, wurde auf dem Scanning-System eine Informationsseite online gestellt, die eine offizielle Kontakt-Mail-Adresse incl. PGP-Key für Signaturen und zur Verschlüsselung bereitstellt [49]. Die fertige Webseite war über die Scanning-IP und die genannte Domain erreichbar und enthält eine Beschreibung der Scans (siehe Abb. 4.4).



Why do we scan?

We collect data about SSL/TLS configurations to analyze security configurations of SMTP deployments for an university research project.
Our scans include three scan types:

- ZMap scans to search for hosts listening on ports 25, 465 and 587 (SMTP)
- DNS reverse lookups for all found hosts
- Various scans including SSL/TLS handshakes on ports 25, 465 and 587 (SMTP) and DNS requests.
  - SSL/TLS handshakes to get X.509 certificates
  - Zlib compression support
  - DNSSEC/DANE support
  - Vulnerabilities (i.e. Heartbleed CVE-2014-0160)
  - Supported cipher suites on SSL 2.0/3.0 and TLS 1.0/1.1/1.2
  - Insecure renegotiation
  - Session resumption capabilities

The aim of these efforts is an analysis of the usage of SSL/TLS for the mail protocol SMTP. Therefor we collect the following information.

- Is SSL/TLS in use?
- Which SSL/TLS versions are supported?
- Which Cipher Suites are supported?
- How is SSL/TLS configured?
- Which certificates are in use (self-signed, key size, ...)?
- Is DANE supported?

The source ip for all scans is **141.39.242.16** with the domain **researchscansmtp.cs.hm.edu**.
Please contact **tls-scan@cs.hm.edu** (PGP key) for blacklisting inquiries.

Imprint and contact

Research group leader:
Prof. Dr.-Ing. Hans-Joachim Hof

Responsible contact person:
Thomas Maier

Munich IT Security Research group
Munich University of Applied Sciences
Department of Computer Science and Mathematics
Lothstraße 64
80335 Munich
Germany

Abbildung 4.4: Webseite auf dem Scanning-System

Um die Scans durchzuführen, wurde dem System von der Fakultät für Informatik eine öffentliche IP zugewiesen.

```
1  iptables -F
2  iptables -X
3  iptables -A INPUT -p tcp --dport 3369 -j ACCEPT
4  iptables -A INPUT -p tcp --dport 80 -j ACCEPT
5  iptables -A INPUT -p icmp -j ACCEPT
6  iptables -A INPUT -s 127.0.0.1 -j ACCEPT
7  iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
8  iptables -A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
9  iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
10 iptables -A INPUT -j REJECT
11 iptables -A FORWARD -j REJECT
```

Listing 4.3: iptables-Regeln

Im Firewall-Listing 4.3 sind die `iptables`-Regeln [71] zu sehen, mit denen das System abgesichert wurde. Dabei ist zu beachten, dass sowohl der alternative Port `3369` (für SSH) zur Verwaltung des Systems und der Well-Known-Port `80` (für HTTP) zum Zugriff auf die Info-Webseite freigegeben wurden.

Des Weiteren wurden die Empfehlungen der *National Security Agency* (NSA) zur sicheren Konfiguration von SSH und Webservern umgesetzt [72].

4.6 Durchführung

Als Scanning-Umgebung wurde folgende Verzeichnishierarchie eingeführt:

```
/home/
    user/                   # Userverzeichnis
        workspace/          # Projektverzeichnis
            hostscanner/    # Hostscanner-Projekt
            sslyze/         # SSLyze-Projekt
        results/            # Ergebnis-Verzeichnis
```

Listing 4.4: Verzeichnishierarchie der Scan-Umgebung

Um Scans in der Umgebung aus Listing 4.4 im Hintergrund zu starten, hat sich das Kommandozeilen-Werkzeug `screen` [73] als sehr wertvoll erwiesen. Damit ist es möglich, mit einer einzigen SSH-Verbindungen beliebig viele Kommandozeilen gleichzeitig zu bedienen. So kann beispielsweise eine Log-Datei beobachtet werden, während auf einer anderen Kommandozeile der zugehörige SSLyze-Scan läuft.

Weitere kleine Werkzeuge, die Entwicklung, Tests und Durchführung der Datenerhebung enorm erleichtert haben, werden im Folgenden beschrieben.

4.6.1 Synchronisation der Projektverzeichnisse

Um den aktuellen Entwicklungsstand der Projektverzeichnisse mit minimalem Aufwand zum Scan-Server zu übertragen, wurde folgendes Shellscript geschrieben.

```
#!/bin/bash
rsync -avze "ssh -p 3369" workspace/sslyze user@researchscansmtp.cs.hm.edu:/home/
    user/workspace/
rm -f /workspace/hostscanner/results/*
rsync -avze "ssh -p 3369" workspace/hostscanner user@researchscansmtp.cs.hm.edu:/
    home/user/workspace/
```

Listing 4.5: Shellscript zur Synchronisation der Projektverzeichnisse

Beim Aufruf des Shellscripts in Listing 4.5 werden die Projektverzeichnisse `hostscanner` und `sslyze` im aktuellen Entwicklungsstand auf den Server kopiert.

4.6.2 Laden der Scan-Ergebnisse aus der Datenbank

Der Scan-Server speichert alle Ergebnisse von SSLyze in einer Datenbank. Um diese Daten zu analysieren, wurde ein Shellscript geschrieben, das einen aktuellen Dump der Datenbank erstellt.

```
1 #!/bin/bash
2 ssh user@researchscansmtp.cs.hm.edu -p 3369 "mongodump -d sslyze -o mongodb_export"
3 rsync --delete -avze "ssh -p 3369" user@researchscansmtp.cs.hm.edu:/home/user/
      mongodb_export ./
4 ssh user@researchscansmtp.cs.hm.edu -p 3369 "rm -r mongodb_export"
5 cd mongodb_export/
6 mongorestore sslyze
```

Listing 4.6: Shellscript zum Laden der Ergebnis-Datenbank

In Listing 4.6 wird der Datenbank-Dump per SSH erstellt, auf das lokale System heruntergeladen und anschließend in die lokale Datenbank geschrieben. Die Analyse der Daten ist also unabhängig vom Server und hat somit den Vorteil, dass der Server permanent für Scans verwendet werden kann. Daher ist es auch nicht nötig, einen weiteren öffentlichen Port für den Datenbankzugriff auf dem Serversystem zu öffnen, was die Angriffsfläche klein hält.

4.6.3 Blacklisting

Nachdem die Entwicklung abgeschlossen war und erste Tests auf kleinere IP Ranges durchgeführt wurden, konnten auch größere IP Ranges mit bis zu 30.000 Zielsystemen überprüft werden, um Probleme bei einer hohen Anzahl von Zielsystemen feststellen zu können. Aufgrund der großen Anzahl an gescannten Zielsystemen kamen auch viele Beschwerden bei LRZ, DFN-CERT und der auf der Webseite angegebenen Mail-Adresse, wurden vor jedem Testlauf neue Blacklisting-Anfragen eingepflegt.

4.6.4 Laufzeit-Optimierung

Da große IP Ranges gescannt werden sollen, sollte die Laufzeit von SSLyze linear zur Anzahl der überprüften Zielsysteme steigen. Mit der Netzmaske `/24` konnten noch keine Probleme festgestellt werden, allerdings kam es bei `/16` zu Fehlern bei der parallelen Verarbeitung von SSLyze. Im Rahmen dieser Arbeit konnte die Ursache aufgrund der begrenzten Zeit nicht genauer analysiert werden, weshalb ein Workaround geschrieben wurde. Mit folgendem Shellscript wird SSLyze mehrfach, dafür aber mit weniger Zielsystemen gestartet.

```
1  #!/bin/sh
2
3  hosts_to_scan_full=/home/user/results/hosts_to_scan
4  hosts_to_scan=/home/user/results/hosts_to_scan_reduced
5  hosts_per_file=200
6  hosts_amount=30000
7
8  sort --random-sort $hosts_to_scan_full | head -n $hosts_amount > $hosts_to_scan
9  start=$(date +'%s')
10 # Split $hosts_to_scan
11 rm -f ${hosts_to_scan}_*
12 counter=0
13 current_file=${hosts_to_scan}_0
14 file_counter=1
15 while read line; do
16      echo $line >> $current_file
17      counter=`expr $counter + 1`
18      if [ $counter -eq $hosts_per_file ]; then
19           current_file=${hosts_to_scan}_$file_counter
20           touch $current_file
21           file_counter=`expr $file_counter + 1`
22           counter=0
23      fi
24 done < $hosts_to_scan
25
26 # Start SSLyze for every file
27 for current_file in ${hosts_to_scan}_*
28 do
29      python sslyze.py --db=mongodb://localhost:27017/ --tlsv1_2 --chrome_sha1 --
           resum_rate --tlsv1 --sslv3 --compression --reneg --tlsv1_1 --certinfo=full
           --sslv2 --heartbleed --resum --dane --starttls=auto --targets_in=
           $current_file
30 done
31 echo --------------------
32 echo "It took $(($(date +'%s') - $start)) seconds"
```

Listing 4.7: Shellscript als Workaround zum Start von SSLyze

Die Anzahl der Zielsysteme, mit denen SSLyze pro Start versorgt wird, kann in den ersten Zeilen des Listings 4.7 mit der Variable `hosts_per_file` gesetzt werden. Die Variable `hosts_amount` enthält die insgesamt zu überprüfenden Zielsysteme. Bei Tests wurde festgestellt, dass sich die Laufzeit von SSLyze mit 200 Zielsystemen am stabilsten verhält.

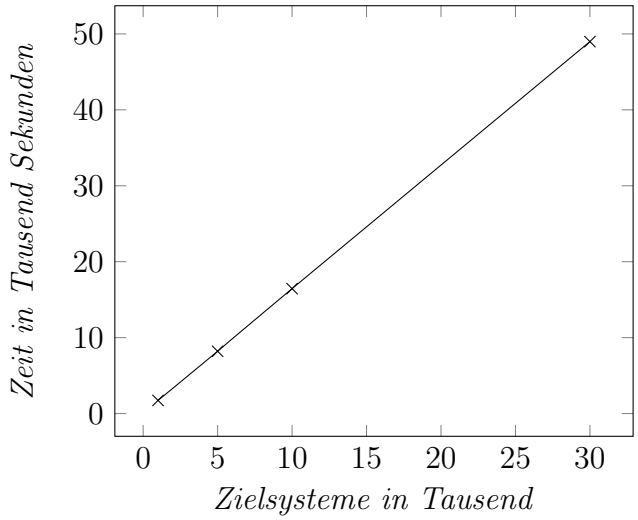


Abbildung 4.5: Laufzeit-Übersicht von SSLyze-Scans

Die Messungen, die in Abb. 4.5 visualisiert wurden, lassen vermuten, dass die Laufzeit in der genannten Konfiguration annähernd linear ansteigt. In den Zeilen 10 bis 24 von Listing 4.7 werden die zu scannenden Zielsysteme in einzelne Dateien mit je 200 Einträgen aufgeteilt. SSLyze wird anschließend mit je einer Datei iterativ gestartet.

Die Dauer eines Scans auf 30.000 Zielsysteme beträgt laut Abb. 4.5 ca. 14 Stunden. Bei einem Testlauf des Hostscanners, mit dem das komplette Internet nach SMTP-Ports durchsucht wurde, konnten 18,7 Millionen Zielsysteme gefunden werden. Hochgerechnet würde dies für den TLS-Scan eine Laufzeit von ca. 23 Jahren bedeuten. Daher wurde mit Zeile 8 in Listing 4.7 eine Konsequenz gezogen: Von allen Systemen werden mithilfe einer Zufallsstichprobe 50.000 zu überprüfende Zielsysteme ausgewählt, was hochgerechnet eine Laufzeit von ca. 23 Stunden bedeutet.

Um eine möglichst repräsentative Aussage über die Sicherheit des Mail-Transports zu treffen, werden außerdem nur die IP Ranges eines Landes durchsucht. Da diese

Bachelorarbeit an einer deutschen Hochschule verfasst wird, fällt die Entscheidung auf Deutschland. Beim Durchsuchen der auf *Country IP Blocks*^{TM} [74] gelisteten deutschen IP Ranges, konnten 900.000 Zielsysteme gefunden werden, aus denen 50.000 zufällig ausgewählt und überprüft wurden.

5 Evaluierung

Um eine möglichst repräsentative Aussage über die Transportsicherheit des SMTP-Verkehrs zu treffen, wurden 50.000 Zielsysteme zufällig aus allen verfügbaren deutschen Servern ausgewählt, die einen Service auf einem der in 2.2.2 genannten Ports gestartet haben. 29% dieser Zielsysteme können aus verschiedenen Gründen, auf die später eingegangen wird, nicht genauer untersucht werden.

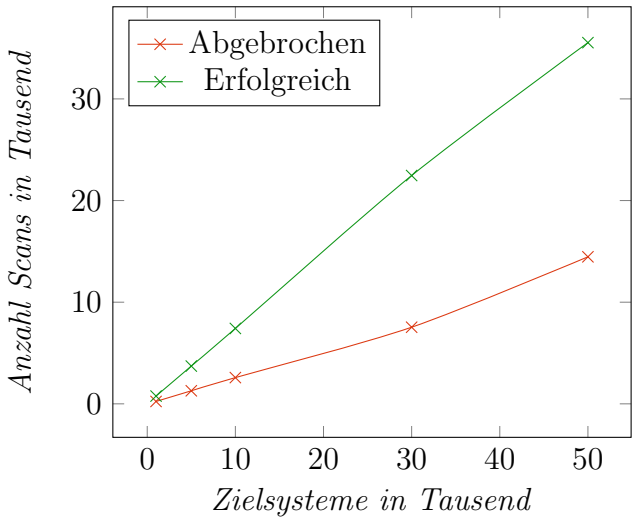


Abbildung 5.1: Erfolgreiche und abgebrochene SSLyze-Scans

Abb. 5.1 lässt vermuten, dass die Anzahl der erfolgreichen Scans abhängig von der Anzahl der ausgewählten Zielsysteme ist. Es konnten verschiedene Gründe für den Abbruch eines Scans festgestellt werden.

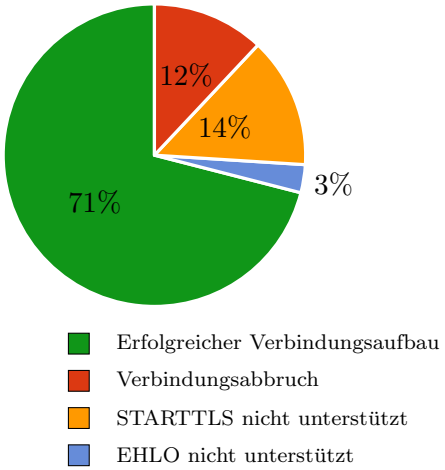


Abbildung 5.2: Gründe für das Abbrechen von Scans

Bei einem Scan von 50.000 Zielsystemen verlaufen 71% erfolgreich. 12% können aufgrund verschiedener Verbindungsabbrüche nicht untersucht werden. Aufsummiert wurde herausgefunden, dass 17% der Zielsysteme entweder generell keine SMTP-Erweiterungen (`EHLO`-Kommando) oder keine `STARTTLS`-Erweiterung unterstützen. Für diese Systeme ist es also prinzipiell nicht möglich, den Transportverkehr mit TLS zu verschlüsseln.

Bei Stichproben konnte festgestellt werden, dass verschiedene Gründe zu Verbindungsabbrüchen führen können. Auffällig dabei ist, dass Zielsysteme teilweise auf Port 25 und 587 nur native TLS-Verbindungen annehmen, was nicht den Standards entspricht [31]. Im Gegensatz dazu erwarteten manche Server auf Port 465 statt nativen TLS-Verbindungen eine Kommunikation per SMTP. Des Weiteren wird vermutet, dass es sich bei Verbindungsabbrüchen um SMTP-Honeypots handelt oder um SMTP-fremde Dienste, die auf diesen Ports laufen. Vor allem bei dem Port 465 ist dies naheliegend, da der Port auch für andere Dienste bekannt ist [75] und von der IANA für einen anderen Service standardisiert wurde [76]. Vor dem TLS-Scan wurden mit dem Hostscanner bereits offene Ports des Zielsystems gesucht, daher ist es auch möglich, dass die IP des Scan-Servers bereits gesperrt wurde. In diesem Fall sind vermutlich keine weiteren Verbindungen und damit auch keine weiteren Scans möglich.

In den folgenden Analysen stellen die Zielsysteme, zu denen erfolgreiche TLS-Verbindungen aufgebaut werden konnten, die Absolutzahl dar. Alle anderen Zielsysteme werden nicht weiter betrachtet.

Die TLS-fähigen Zielsysteme werden im Folgenden genauer auf deren Konfiguration untersucht. Da lediglich 3,72% der Zielsysteme anfällig für die Heartbleed-Schwachstelle sind, wird darauf nicht genauer eingegangen.

5.1 Zertifikate

Nach dem Aufbau einer TLS-Verbindung ist laut Spezifikation die Feststellung der Identität des Gegenübers nötig. Nachdem die Verifikation über die X.509-PKI und DANE im Folgenden untersucht wird, folgt eine Überprüfung von Schlüssellänge und Gültigkeitszeitraum der verwendeten Zertifikate.

5.1.1 Verifikation

Beim Versuch alle gefundenen Zertifikate über die X.509-PKI zu verifizieren, konnten verschiedene Ergebnisse ausgemacht werden. Ein Zertifikat wird nachfolgend nur als erfolgreich verifizierbar gewertet, wenn alle getesteten Trust Stores [77] das gleiche Ergebnis liefern. Verwendet wurden dazu die in SSLyze enthaltenen Trust Stores von Mozilla, Microsoft, Apple und Java.

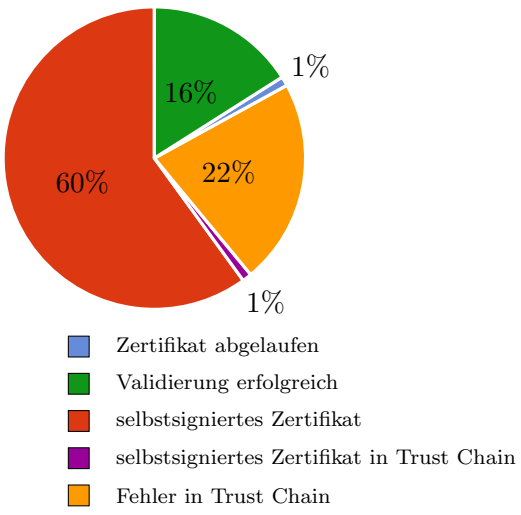


Abbildung 5.3: Verifikation von Zertifikaten über die X.509-PKI

Wie in Abb. 5.3 zu sehen, wurde die Echtheit und Gültigkeit bei insgesamt 16% aller geprüften Zertifikate von allen verwendeten Trust Stores bestätigt. 1% der Zielsysteme scheiterten dabei bereits bei der Überprüfung des Gültigkeitszeitraums. Es ist also nicht mehr möglich, die Vertrauenswürdigkeit dieser Zertifikate festzustellen. Bei insgesamt 60% handelt es sich um selbstsignierte Zertifikate, 1% beinhaltet ein selbstsigniertes Zertifikat in der empfangenen Trust Chain. Bei 22% aller Zertifikate kommt es zu Fehlern bei der Validierung der Trust Chain. In den letzten drei Fällen kann also ebenfalls keine Vertrauensbasis aufgebaut werden.

Bei nur 5,2% der beim *SSL Pulse* geprüften HTTP-Server kann die Trust Chain nicht nachvollzogen werden [60]. Bei Webservern wird also mehr Wert auf die eindeutige Identifikation gelegt als bei den in dieser Arbeit geprüften SMTP-Servern, deren komplette Trust Chain bei 84% der Zielsysteme nicht verifizierbar ist.

Beim Verifikationsprotokoll DANE fallen die positiven Ergebnisse weit niedriger aus. Nur fünf von 35.535 gescannten Zielsystemen verfügen über einen TLSA Record, der bei allen fünf Systemen mit dem über SMTP erhaltenen Zertifikat übereinstimmt. Nur drei dieser Zertifikate können anschließend über DNSSEC validiert werden.

5.1.2 Schlüssellänge

Alle aufgefundenen Zertifikate verwenden für den enthaltenen öffentlichen Schlüssel den RSA-Algorithmus, deren Schlüssellängen in folgendem Diagramm analysiert werden. Nach einer Empfehlung des *National Institute of Standards and Technology* (NIST) sollten RSA-Schlüssel eine Länge von mindestens 2048 Bit haben [78].

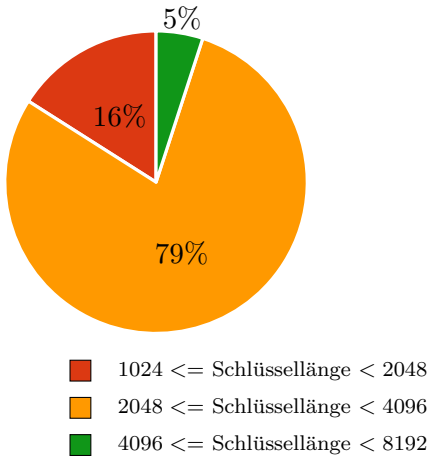


Abbildung 5.4: Verwendung von Schlüssellängen

Diese Empfehlung wird also von einem Großteil der SMTP-Server in Abb. 5.4 eingehalten oder sogar übertroffen. Ein kleinerer Teil von 16% orientiert sich nicht an dieser Empfehlung. Das arithmetische Mittel aller verwendeten Schlüssellängen liegt bei 1985 Bits und befindet sich damit knapp unter der Empfehlung der NIST.

Im Gegensatz zu Mailservern zeigt der *SSL Pulse*, dass Webserver fast immer Schlüssel verwenden, die größer als 1024 Bit sind [60]. Dafür tauchen bei Webservern auch nur sehr selten Schlüssellängen auf, die höher als 4096 Bit sind. Bei Mailservern beinhaltet jedes 20. Zertifikat eine Schlüssellänge, die höher als 4096 Bit ist.

5.1.3 Gültigkeitszeitraum

In den Scans wurden sowohl selbstsignierte Zertifikate als auch von CAs ausgestellte Zertifikate betrachtet.

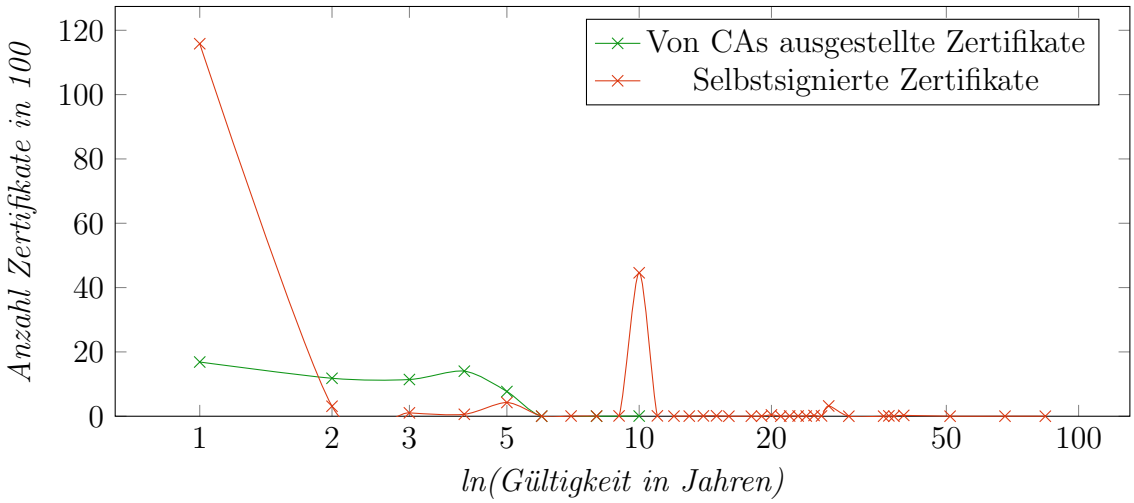


Abbildung 5.5: Überblick über die Gültigkeit von Zertifikaten

In Abb. 5.5 ist deutlich zu sehen, dass selbstsignierte und von CAs ausgestellte Zertifikate starke Unterschiede bei der Gültigkeit aufweisen. Von CAs ausgestellte Zertifikate weisen hauptsächlich Gültigkeitszeiträume von ein bis fünf Jahren auf. Im Gegensatz dazu entscheiden sich Betreiber beim Einsatz von selbstsignierten Zertifikaten am häufigsten für eine Laufzeit von einem Jahr, was vermutlich auf schnell auffindbare Online-Tutorials zurückzuführen ist [79]. Selbstsignierte Zertifikate weisen zum einen ein hohes Aufkommen bei zehn Jahren auf, zum anderen sind auch sehr viel längere Laufzeiten als bei von CAs ausgestellten Zertifikaten zu sehen. Auch das arithmetische Mittel der Gültigkeit von Zertifikaten, die von CAs ausgestellt wurden, ist mit 2,73 Jahren weitaus niedriger als bei selbstsignierten Zertifikaten, die durchschnittlich 4,84 Jahre gültig sind.

Wie in Abschnitt 2.1 beschrieben, sollte der Gültigkeitszeitraum von Zertifikaten eingeschränkt werden, um die Sicherheit zu erhöhen. Im Diagramm ist zu sehen, dass sich CAs größtenteils an die genannten Empfehlungen halten. Kritisch zu betrachten ist die vergleichsweise hohe Menge an selbstsignierten Zertifikaten, da bei deren Verwendung die Gefahr für Man-in-the-Middle-Angriffe steigt.

5.2 TLS-Versionen

Da vom *Bundesamt für Sicherheit in der Informationstechnik* (BSI) empfohlen wird [80], dass weder SSL 2 oder 3 noch TLS 1.0 verwendet werden soll, wird der Einsatz der TLS-Versionen im Folgenden überprüft.

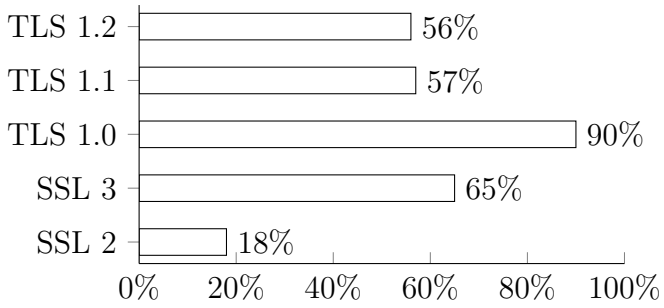


Abbildung 5.6: Unterstützte TLS-Versionen

Obwohl seit langem bekannt ist, dass SSL 2 Schwachstellen enthält, ist es bei SMTP-Servern mit 18% immer noch weit verbreitet [81]. Seit 2014 der POODLE-Angriff bekannt wurde, gilt auch SSL 3 als unsichere Protokollversion [82]. Dass die meisten Mailserver TLS in der Version 1.0 unterstützen, ist laut dem BSI nicht weiter problematisch, falls existierende Anwendungen mit aktuellen Sicherheitsupdates versorgt werden [83]. Dennoch empfiehlt das BSI, dass diese Installationen baldmöglichst durch TLS 1.2 ersetzt werden. [80].

Im Vergleich zu den Webservern bei *SSL Pulse* verhält sich diese Statistik ähnlich bei den Versionen TLS 1.1 und 1.2 [60]. TLS 1.0 wird bei Webservern zu fast 100% unterstützt, wobei bei Mailservern in dieser Arbeit nur eine Unterstützung von 90% festgestellt wurde. Die Werte der Versionen SSL 2 und 3 sind bei HTTP weitaus niedriger als bei SMTP.

Weiterhin wird überprüft, ob Cipher Suites im Allgemeinen, sowie die darin verwendeten Algorithmen, den Empfehlungen des BSI und des *Open Web Application Security Project* (OWASP) [84] entsprechen.

Empfehlungen des BSI

Das BSI empfiehlt eine Liste von Cipher Suites, welche eine erhöhte Transportsicherheit gewährleisten sollen.

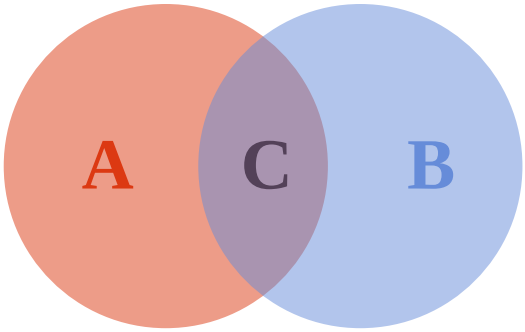


Abbildung 5.7: Überblick der Unterstützung von Cipher Suites

Jedes Zielsystem kann verschiedene Cipher Suites unterstützen, deren Verteilung in Abb. 5.7 dargestellt wird.

$\boldsymbol{A}$ Mindestens eine vom BSI empfohlene Cipher Suite wird unterstützt

$\boldsymbol{B}$ Mindestens eine vom BSI nicht empfohlene Cipher Suite wird unterstützt

Werden A und B im Zusammenhang betrachtet, können drei weitere Mengen unterschieden werden.

$\boldsymbol{A} \setminus \boldsymbol{B}$ Nur vom BSI empfohlene Cipher Suites werden unterstützt $\Rightarrow 0,34\%$

$\boldsymbol{B} \setminus \boldsymbol{A}$ Nur vom BSI nicht empfohlene Cipher Suites werden unterstützt $\Rightarrow 90,73\%$

$\boldsymbol{C} = \boldsymbol{A} \cap \boldsymbol{B}$ Sowohl empfohlene als auch nicht empfohlene Cipher Suites werden unterstützt $\Rightarrow 8,93\%$

Hervorzuheben ist, dass nur sehr wenige Zielsysteme nur vom BSI empfohlene Cipher Suites verwenden. 90,73% der Zielsysteme unterstützen ausschließlich Cipher Suites, die nicht empfohlen werden. Insgesamt halten sich also 99,66% der Zielsysteme nicht vollständig an die Empfehlungen des BSI.

In folgender Tabelle sind die sechs vom BSI empfohlenen Cipher Suites gelistet, die bei der Datenerhebung aufgefunden wurden.

#	Cipher Suite	Unterstützung
1	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	78,42%
2	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	78,39%
3	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	78,09%
4	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	77,82%
5	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	50,35%
6	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	48,91%

Tabelle 5.1: Unterstützung der vom BSI empfohlenen Cipher Suites

Auffällig dabei ist, dass alle gefundenen Cipher Suites in Tabelle 5.1 bei der Schlüsselaushandlung mit PFS arbeiten. Das BSI empfiehlt auch Cipher Suites ohne PFS, allerdings wird dazu geraten, PFS-Cipher-Suites zu bevorzugen.

Empfehlungen zu Algorithmen

Weitere Empfehlungen wurden vom OWASP veröffentlicht [85]. Diese konzentrieren sich darauf, bestimmte Cipher Suites aufgrund der dabei verwendeten schwachen Algorithmen zu vermeiden. Im Folgenden werden die erhobenen Daten im Vergleich zu den Empfehlungen der OWASP ausgewertet. Die nachfolgenden Kategorien von Cipher Suites sollten laut OWASP deaktiviert werden:

NULL Cipher Suites: Cipher Suites, die keine Verschlüsselung enthalten, konnten bei nur 0,03% der gescannten Systeme gefunden werden. Sehr viel häufiger werden Cipher Suites unterstützt, die Schlüssel ohne Authentifizierung, also anonym, aushandeln. Auffällig waren hierbei die Verfahren `AECDH` mit 35% und `ADH` mit sogar 63%.

Export Cipher Suites: Export Ciphers werden für SSL 2 immer noch von mehr als 90% der Zielsysteme verwendet. Bei SSL 3 sinkt die Verwendung auf 45%. Bei TLS 1.0, 1.1 und 1.2 liegt die Unterstützung nur noch bei 34%. Bei neueren TLS-Versionen werden also durchschnittlich weniger Export Ciphers unterstützt.

RC4-/IDEA-Verschlüsselung: Der Verschlüsselungsalgorithmus IDEA wird mit 16 bis 25% nicht so häufig wie RC4 verwendet. RC4 wird vor allem bei SSL 2 mit bis zu 95% unterstützt, jedoch sinkt dieser Wert mit fortlaufenden TLS-Versionen bis auf 68% für TLS 1.2.

MD5-Hashes: Ähnliche Muster wie bei Export Ciphers sind auch bei Cipher Suites erkennbar, die MD5 als Hash-Funktion verwenden. Bei SSL 2 unterstützen bis zu 95% der Systeme MD5, wobei die nachfolgenden Versionen bis TLS 1.1 Werte von bis zu 67% aufweisen. Für TLS 1.2 liegen keine Werte vor, da die Hash-Funktion hier nicht mehr durch die gewählte Cipher Suite festgelegt wird.

6 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden TLS-Konfigurationen von SMTP-Installationen überprüft. Dazu wurde zuerst eine Recherche zu bereits vorhandenen Arbeiten durchgeführt. Es konnte festgestellt werden, dass nur wenige verwandte Projekte sich mit diesem Thema beschäftigen. SSLyze ist die einzige Software, mit der es möglich ist, ganze IP Ranges zu scannen und die dabei auch `STARTTLS` unterstützt. Dieses Werkzeug wurde erweitert, um DANE zu verifizieren und die gesammelten Daten sinnvoll in einer Datenbank abzuspeichern. Da es nur wenige verwandte Arbeiten gibt, musste die DANE-Verifizierung von Grund auf nach DANE- und DNSSEC-RFCs implementiert werden. Mit den gesammelten Daten konnte eine repräsentative Aussage über den Stand der Sicherheit im Mail-Verkehr getroffen werden. Bei der Analyse der gesammelten Daten war auffällig, dass der SMTP-Transport teilweise deutlich schlechter als der HTTP-Verkehr abgesichert ist.

Bei Scans müssen viele TLS-Verbindungen zu Zielsystemen aufgebaut werden, wodurch bei Administratoren von Zielsystemen ein hoher Netzwerkverkehr auffällt. Daher darf der Zeitaufwand beim Bearbeiten der ankommenden Beschwerden nicht unterschätzt werden. Verringern könnte man diese Beschwerden, indem die Scans randomisiert werden, wodurch nur wenige Verbindungen gleichzeitig zu einem Zielsystem aufgebaut werden. Durch die geringere Auslastung stört die Datenerhebung seitens der Zielsysteme weniger beim Tagesgeschäft.

Da eine Aussage zum aktuellen Stand der Transportsicherheit des Mailsystems im Internet getroffen werden konnte, kann diese Arbeit als Erfolg gesehen werden. Aufgrund der begrenzten Zeit einer Bachelorarbeit musste die Analyse auf einen Teil der deutschen IP-Bereiche im Internet eingeschränkt werden. Trotz dieser Begrenzung geben die Ergebnisse einen ersten Überblick über den Zustand des Mailsystems.

Für die Zukunft sind, statt den in dieser Arbeit durchgeführten Scans auf deutsche IP Ranges, globale Datenerhebungen des kompletten Internets geplant. Diese Scans sollen, wie beim Projekt *SSL Pulse*, regelmäßig durchgeführt werden, um Veränderungen der Sicherheit beim Mail-Verkehr festzustellen.

Die verschiedenen TLS-Versionen sollten dabei stärker auf deren Häufigkeiten überprüft werden. Unter Anderem wäre interessant, wie oft nur die TLS-Versionen SSL 2 und 3 in Kombination vertreten sind. Auch verwendete Cipher Suites und Zertifikate bieten noch viele weitere Möglichkeiten für Analysen. Beispielsweise wäre es möglich, den Einsatz von bestimmten Algorithmen genauer zu analysieren. Anbieten würde sich hier z.B. eine Clusteranalyse, um herauszufinden welche Kombinationen von Algorithmen häufig verwendet werden.

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

Literaturverzeichnis

[1] A. Beutelspacher, "Kryptologie - Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen, 10. aktualisierte Auflage," *Braunschweig/Wiesbaden: Vieweg*, 2015.

[2] "Globaler Abhörwahn," *heise online*, http://www.heise.de/ct/artikel/Globaler-Abhoerwahn-1913829.html (abgerufen: 29.04.2015).

[3] C. Sorge, N. Gruschka, and L. L. Iacono, *Sicherheit in Kommunikationsnetzen*. Oldenbourg Verlag, 2013.

[4] "RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," *Internet Engineering Task Force*, https://tools.ietf.org/html/rfc5280 (abgerufen: 01.02.2015).

[5] "The EFF SSL Observatory," *Electronic Frontier Foundation*, https://www.eff.org/de/observatory (abgerufen: 26.03.2015).

[6] "RFC 2986 - PKCS 10: Certification Request Syntax Specification - Version 1.7," *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc2986 (abgerufen: 07.02.2015).

[7] *CA/Browser Forum*, https://cabforum.org/ (abgerufen: 12.04.2015).

[8] "Google and Mozilla to Reject SSL Certificates Issued for Longer than 60 Months," *Softpedia*, http://news.softpedia.com/news/Google-and-Mozilla-to-Reject-SSL-Certificates-Issued-for-Longer-than-60-Months-378001.shtml (abgerufen: 12.04.2015).

[9] "Gone in 60 Months or Less," *Venafi*, https://www.venafi.com/blog/post/gone-in-60-months-or-less/ (abgerufen: 12.04.2015).

[10] “Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates,” *CA/Browser Forum*, https://cabforum.org/wp-content/uploads/BRv1.2.3.pdf (abgerufen: 12.04.2015).

[11] “Der schnelle, kostenlose Browser,” *Google*, https://www.google.de/chrome/browser/desktop/ (abgerufen: 01.05.2015).

[12] “Deprecating support for long-lived certificates,” *CA/Browser Forum*, https://cabforum.org/pipermail/public/2013-August/002135.html (abgerufen: 12.04.2015).

[13] “RFC 6960 - X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol - OCSP,” *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc6960 (abgerufen: 06.02.2015).

[14] *Netscape Communications Corp.*, http://netscape.aol.com/ (abgerufen: 19.04.2015).

[15] Kipp E.B. Hickman, “SSL 0.2 Protocol Specification,” *Netscape Communications Corp.*, Februar 1995, http://www-archive.mozilla.org/projects/security/pki/nss/ssl/draft02.html (abgerufen: 01.02.2015).

[16] Alan O. Freier, Philip Karlton, Paul C. Kocher, “The SSL Protocol - Version 3.0,” *Netscape Communications Corp.*, November 1996, http://web.archive.org/web/20080206214535/http://wp.netscape.com/eng/ssl3/draft302.txt (abgerufen: 08.02.2015).

[17] “RFC 2246 - The TLS Protocol - Version 1.0,” *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc2246 (abgerufen: 08.02.2015).

[18] Ristić, Ivan, “Bulletproof SSL and TLS,” *Feisty Duck*, 2014.

[19] “RFC 5246 - The Transport Layer Security (TLS) Protocol, Version 1.2,” *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc5246 (abgerufen: 01.02.2015).

[20] C. Eckert, *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. München: Oldenbourg Wissenschaftsverlag GmbH, 2014.

[21] “Independent Iranian hacker claims responsibility for Comodo hack,” *Ars Technica*, http://arstechnica.com/security/2011/03/independent-iranian-hacker-claims-responsibility-for-comodo-hack/ (abgerufen: 03.03.2015).

[22] "Another fraudulent certificate raises the same old questions about certificate authorities," *Ars Technica*, http://arstechnica.com/security/2011/08/earlier-this-year-an-iranian/ (abgerufen: 03.03.2015).

[23] M. Gielesberger, "Alternatives to x.509," *Innovative Internet Technologies and Mobile Communications (IITM)*, vol. 51, 2013.

[24] "French agency caught minting SSL certificates impersonating Google," *Ars Technica*, http://arstechnica.com/security/2013/12/french-agency-caught-minting-ssl-certificates-impersonating-google/ (abgerufen: 03.03.2015).

[25] "How secure is HTTPS today? How often is it attacked?" *Ars Technica*, https://www.eff.org/deeplinks/2011/10/how-secure-https-today (abgerufen: 04.03.2015).

[26] "ciphers - SSL cipher display and cipher list tool." *OpenSSL*, https://www.openssl.org/docs/apps/ciphers.html (abgerufen: 20.04.2015).

[27] "Attack of the week: FREAK," *A Few Thoughts on Cryptographic Engineering*, http://blog.cryptographyengineering.com/2015/03/attack-of-week-freak-or-factoring-nsa.html (abgerufen: 20.04.2015).

[28] "Counterpane Labs Releases Windows 95-compatible S/MIME 40-bit RC2 Cracking ScreenSaver," *Schneier on Security*, https://www.schneier.com/smime.html (abgerufen: 20.04.2015).

[29] "RFC 821 - Simple Mail Transfer Protocol," *Internet Engineering Task Force*, https://tools.ietf.org/html/rfc821 (abgerufen: 07.03.2015).

[30] K. Strauser, "The history and the future of smtp," *Free Software Magazine*, no. 2, 2005.

[31] "25, 465, 587... What port should I use?" *Mailgun*, http://blog.mailgun.com/25-465-587-what-port-should-i-use/ (abgerufen: 07.03.2015).

[32] "RFC 2476 - Message Submission," *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc2476 (abgerufen: 07.03.2015).

[33] "RFC 3207 - SMTP Service Extension for Secure SMTP over TLS," *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc2487 (abgerufen: 07.03.2015).

[34] J. Dietrich and J. Keller-Herder, "De-mail—verschlüsselt, authentisch, nachweisbar," *Datenschutz und Datensicherheit-DuD*, vol. 34, no. 5, pp. 299–301, 2010.

[35] "Postfix TLS Support," *Postfix*, http://www.postfix.org/TLS_README.html (abgerufen: 07.03.2015).

[36] "E-Mail Made in Germany: SSL-Verschlüsselung für (fast) alle," *heise online*, http://www.heise.de/newsticker/meldung/E-Mail-Made-in-Germany-SSL-Verschluesselung-fuer-fast-alle-1932962.html (abgerufen: 13.03.2015).

[37] "E-Mail Made in Germany: SSL-Verschlüsselung für (fast) alle," *E-Mail made in Germany*, http://www.e-mail-made-in-germany.de/Teilnehmer.html (abgerufen: 13.03.2015).

[38] Gerling, Rainer W., "De-Mail und E-Mail made in Germany sind ein konsequenter Schritt," *Datenschutz und Datensicherheit - DuD*, vol. 38, no. 2, pp. 109–111, 2014.

[39] "So funktioniert E-Mail made in Germany," *heise online*, http://www.heise.de/netze/meldung/So-funktioniert-E-Mail-made-in-Germany-2188248.html (abgerufen: 13.03.2015).

[40] "RFC 6698 - The DNS-Based Authentication of Named Entities (DANE) - Transport Layer Security (TLS) Protocol: TLSA," *Internet Engineering Task Force*, https://tools.ietf.org/html/rfc6698 (abgerufen: 01.02.2015).

[41] "RFC 1035 - Domain names - implementation and specification," *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc1035 (abgerufen: 01.02.2015).

[42] "RFC 4033 - DNS Security Introduction and Requirements," *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc4033 (abgerufen: 01.02.2015).

[43] "DNSSEC: The Antidote to DNS Cache Poisoning and Other DNS Attacks," *F5 Networks, Inc.*, https://f5.com/resources/white-papers/dnssec-the-antidote-to-dns-cache-poisoning-and-other-dns-attacks (abgerufen: 21.04.2015).

[44] "RFC 3833 - Threat Analysis of the Domain Name System (DNS)," *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc3833 (abgerufen: 01.02.2015).

[45] "RFC 4034 - Resource Records for the DNS Security Extensions," *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc4034 (abgerufen: 01.02.2015).

[46] "DNSSEC und DANE bei E-Mail-Anbietern," http://www.email-vergleich.com/2014/07/dnssec-und-dane-bei-e-mail-anbietern/ (abgerufen: 02.04.2015).

[47] "DANE TLS authentication," *Postfix TLS Support*, http://www.postfix.org/TLS_README.html#client_tls_dane (abgerufen: 02.04.2015).

[48] *NMAP.ORG*, http://nmap.org/ (abgerufen: 26.03.2015).

[49] Zakir Durumeric and Eric Wustrow and J. Alex Halderman, "ZMap: Fast Internet-Wide Scanning and its Security Applications," in *Proceedings of the 22nd USENIX Security Symposium*, aug 2013.

[50] "masscan," *Github*, https://github.com/robertdavidgraham/masscan (abgerufen: 26.03.2015).

[51] David Adrian and Zakir Durumeric and Gulshan Singh and J. Alex Halderman, "Zippier ZMap: Internet-Wide Scanning at 10 Gbps," in *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. San Diego, CA: USENIX Association, Aug 2014.

[52] Lee, Homin K and Malkin, Tal and Nahum, Erich, "Cryptographic Strength of SSL/TLS Servers: Current and Recent Practices," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 83–92.

[53] Holz, Ralph and Braun, Lothar and Kammenhuber, Nils and Carle, Georg, "The SSL Landscape – A Thorough Analysis of the X.509 PKI Using Active and Passive Measurements," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 427–444.

[54] Zakir Durumeric and James Kasten and Michael Bailey and J. Alex Halderman, "Analysis of the HTTPS Certificate Ecosystem," in *Proceedings of the 13th Internet Measurement Conference*, oct 2013.

[55] "The Internet-Wide Scan Data Repository," *ZMap Team at the University of Michigan*, https://scans.io/ (abgerufen: 27.03.2015).

[56] "SSL Labs," *Qualys, Inc.*, https://www.ssllabs.com/ (abgerufen: 24.03.2015).

[57] “SSL Labs - SSL Server Test,” *Qualys, Inc.*, https://www.ssllabs.com/ssltest/ (abgerufen: 24.03.2015).

[58] “SSL Labs - SSL Server Rating Guide,” *Qualys, Inc.*, https://www.ssllabs.com/projects/rating-guide/ (abgerufen: 24.03.2015).

[59] “SSL Labs: Dank neuer API Hunderte Webserver komfortabel scannen,” *heise online*, http://www.heise.de/security/meldung/SSL-Labs-Dank-neuer-API-Hunderte-Webserver-komfortabel-scannen-2582760.html (abgerufen: 24.03.2015).

[60] “SSL Pulse - Survey of the SSL Implementation of the Most Popular Web Sites,” *Trustworthy Internet Movement*, https://www.trustworthyinternet.org/ssl-pulse/ (abgerufen: 22.04.2015).

[61] “SSLyze - Fast and full-featured SSL scanner,” *Github*, https://github.com/nabla-c0d3/sslyze (abgerufen: 27.03.2015).

[62] “ZMap - multiple ports support? - issue 44,” *Github*, https://github.com/zmap/zmap/issues/44 (abgerufen: 30.03.2015).

[63] “ZMap Documentation,” *zmap.io*, https://zmap.io/documentation.html#blacklisting (abgerufen: 30.03.2015).

[64] “RFC 5321 - Simple Mail Transfer Protocol,” *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc5321 (abgerufen: 01.02.2015).

[65] “multiprocessing — Process-based threading interface,” *Python Software Foundation*, https://docs.python.org/2/library/multiprocessing.html (abgerufen: 27.03.2015).

[66] “ssl — TLS/SSL wrapper for socket objects,” *Python Software Foundation*, https://docs.python.org/2/library/ssl.html (abgerufen: 30.03.2015).

[67] “nassl - Experimental OpenSSL wrapper for Python,” *Github*, https://github.com/nabla-c0d3/nassl (abgerufen: 30.03.2015).

[68] “Elasticsearch - Search and Analyze Data in Real Time,” *elastic*, https://www.elastic.co/products/elasticsearch (abgerufen: 05.04.2015).

[69] “Elasticsearch reference - Nested Query,” *elastic*, http://www.elastic.co/guide/en/elasticsearch/reference/1.5/query-dsl-nested-query.html (abgerufen: 05.04.2015).

[70] *MongoDB*, https://www.mongodb.org/ (abgerufen: 05.04.2015).

[71] "The netfilter.org ïptables"project," *netfilter/iptables*, http://www.iptables.org/projects/iptables/index.html (abgerufen: 03.04.2015).

[72] "Guide to the Secure Configuration of Red Hat Enterprise Linux 5," *National Security Agency - Operating Systems Division Unix Team of the Systems and Network Analysis Center*, https://www.nsa.gov/ia/_files/os/redhat/NSA_RHEL_5_GUIDE_v4.2.pdf (abgerufen: 03.04.2015).

[73] "GNU Screen," *gnu.org*, http://www.gnu.org/software/screen/ (abgerufen: 03.04.2015).

[74] "Create Country ACL," *Country IP Blocks*, https://www.countryipblocks.net/country_selection.php (abgerufen: 05.04.2015).

[75] "Port 465 Details," *speedguide.net*, http://www.speedguide.net/port.php?port=465 (abgerufen: 06.04.2015).

[76] "Service Name and Transport Protocol Port Number Registry," *Internet Assigned Numbers Authority*, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt (abgerufen: 06.04.2015).

[77] "SSLyze - Instructions on how to generate trust stores that can be used in SSLyze," *Github*, https://github.com/nabla-c0d3/catt/blob/master/sslyze.md (abgerufen: 01.05.2015).

[78] "Recommendation for Key Management - Part 1," *Computer Security Division Information Technology Laboratory - National Institute of Standards and Technology*, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf (abgerufen: 12.04.2015).

[79] "self-signed certificate," *Google-Suche*, https://www.google.de/?gws_rd=ssl#q=self-signed+certificate (abgerufen: 12.04.2015).

[80] "Kryptographische Verfahren - Empfehlungen und Schlüssellängen - Teil 2 – Verwendung von Transport Layer Security (TLS)," *Bundesamt für Sicherheit in der Informationstechnik*, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/

Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2_pdf.pdf?__blob=publicationFile (abgerufen: 12.04.2015).

[81] "RFC 6176 - Prohibiting Secure Sockets Layer (SSL) Version 2.0," *Internet Engineering Task Force*, http://tools.ietf.org/html/rfc6176 (abgerufen: 12.04.2015).

[82] "Poodle Bites TLS," *Qualys*, https://community.qualys.com/blogs/securitylabs/2014/12/08/poodle-bites-tls (abgerufen: 12.04.2015).

[83] "BSI veröffentlicht Mindeststandard für verschlüsselte Internetverbindungen," *Bundesamt für Sicherheit in der Informationstechnik*, https://www.bsi.bund.de/DE/Presse/Pressemitteilungen/Presse2013/BSI_veroeffentlicht_Mindeststandard_fuer_verschluesselte_Internetverbindungen_08102013.html (abgerufen: 12.04.2015).

[84] *Open Web Application Security Project*, https://www.owasp.org/ (abgerufen: 19.04.2015).

[85] "Transport Layer Security Cheat Sheat," *OWASP*, https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet (abgerufen: 17.04.2015).