Sicherer E-Mail-Dienste-Anbieter

basierend auf
Domain Name System Security Extension (DNSSec)
&
DNS-based Authentication of Named Entities (DANE)

Eine technische Einführung in die Materie
mit Fokus auf
praktisch orientierter Umsetzung im „How-To“ Stil

Zielgruppe: Mailserver / DNS-Server Administratoren

Autor: Gunnar Haslinger
https://www.haslinger.biz/

Kurzfassung

Diese Arbeit beschäftigt sich mit der praktischen Implementierung eines sicheren E-Mail-Dienste-Anbieters, basierend auf DANE (DNS-based Authentication of Named Entities) und DNSSec (Domain Name System Security Extension). Die Umsetzung erfolgt mittels Debian 8 „Jessie“ unter Verwendung des DNS-Servers BIND, des Mailservers Postfix, des WebServers Apache mit WebMailer RoundCube. Die IMAP und POP3 Dienste werden mittels Dovecot realisiert.

Kryptographische Grundlagen werden hierbei vorausgesetzt, spezifische Funktionalitäten jedoch stets zuvor erläutert und anschließend implementiert, die Implementierung nachvollziehbar dokumentiert und ausführlich getestet.

Dabei wurde auf sichere Konfiguration der verwendeten TLS-Protokolle und CipherSuites sowie auf zeitgemäße Verwendung und Konfiguration von DNSSec, DANE, Certificate-Pinning, Public-Key-Pinning, HTTP-Strict-Transport-Security u.v.m. geachtet.

Die Empfehlungen von BetterCrypto.org[1] wurden hierbei berücksichtigt, neue Erkenntnisse werden an das BetterCrypto-Team rückgemeldet und fließen in zukünftige Versionen des Applied-Crypto-Hardening-Guides ein.

Bedeutung der Formatierung

Dieses Dokument hebt Code-Schnippsel bzw. Konfigurationen oder Befehle gemäß folgender Formatierungen hervor:

- Konfigurationen, allgemeine Konsolen-Ausgaben, …
- Bash-Prompt, Kommentare ohne besonderer Relevanz, …
- Konfigurationen, Ausgaben mit besonderer Relevanz, …
- Anweisungen, Befehle

Beispiel:

```
[root@Sec-NS1 ~]# nslookup it-sec.ovh
Server:         100.114.178.6
Address:        100.114.178.6#53

Non-authoritative answer:
Name:   it-sec.ovh
Address: 104.46.42.66
```

[1] Applied Crypto Hardening Guide von https://bettercrypto.org/

Abkürzungsverzeichnis

DANE	DNS-based Authentication of Named Entities
DH	Diffie-Hellman
DNS	Domain Name System
DNSKey	Resource Record zum Propagieren öffentlicher Schlüssel im DNS
DNSSec	Domain Name System Security Extension
DS	Delegation Signer Resource Record
ECDH	Elliptic Curve Diffie-Hellman
HPKP	HTTP Public Key Pinning
HSTS	HTTP Strict Transport Security
MTA	Mail Transfer Agent (z.B. Postfix)
MDA	Mail Delivery Agent (z.B. Dovecot)
MUA	Mail User Agent (z.B. Mozilla Thunderbird)
NSEC	Next Secure Resource Record
NSEC3	Next Secure v3 (oder NSEC Hash) Resource Record
KSK	Key Signing Key, signiert das DNSKEY Resource Record Set
LDA	Local Delivery Agent (z.B. Dovecot)
OCSP	Online Certificate Status Protocol
PFS	Perfect Forward Secrecy
RRSIG	Signature Resource-Record
SASL	Simple Authentication and Security Layer
STARTTLS	Verfahren zur Umschaltung von Klartext- auf TLS-Kommunikation
TLD	Top Level Domain
TLSA	Transport Layer Security Resource-Record
TOFU	Trust on First Use
ZSK	Zone Signing Key, signiert die ganze Zone (jeden einzelnen Eintrag)

Anmerkung: Eine Erläuterung sämtlicher DNSSec Resource-Records findet sich in [RFC-4034].

Inhaltsverzeichnis

1. Einführung – Sicherer E-Mail Dienste Anbieter

1.1. Aufgabenstellung

In Anlehnung an die Technischen Richtlinie TR-03108 des BSI (Bundesamt für Sicherheit in der Informationstechnik, [BSI-TR03108]) soll ein sicherer E-Mail-Dienste-Anbieter implementiert werden. Der Fokus liegt hierbei auf der Umsetzung der geforderten kryptographischen und technischen Anforderungen. Organisatorische Anforderungen sowie Spezifika die eine mögliche Zertifizierung beziehungsweise ein Informationssicherheitsmanagementsystem der Lösung auferlegen sind nicht Gegenstand der Betrachtung im Rahmen dieser Ausarbeitung. Die zu lösenden Teilaufgaben sind somit in den senkrechten Säulen von Abbildung 1 erkennbar:

- Vertrauenswürdige Zertifikate
- Sichere Kryptographie
- Gesicherte DNS-Abfragen (DNSSec)
- Obligatorische Verschlüsselung (DANE/TLSA)

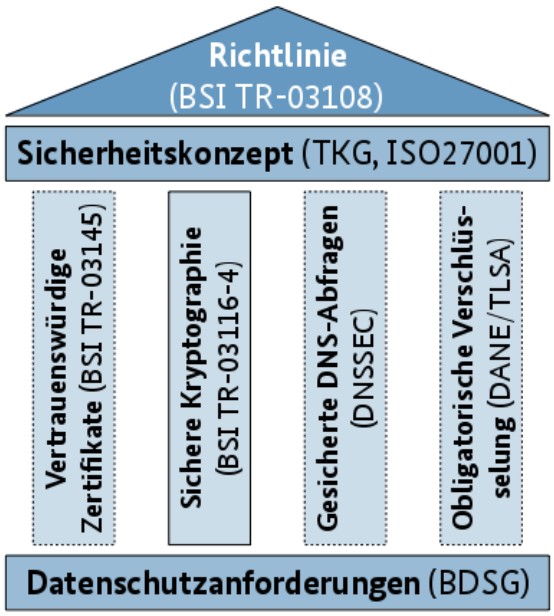


Abbildung 1: BSI TR-03108 Sicherer E-Mail Transport - Quelle: [BSI-TR03108]

1.2. DNSSec

Die *Domain Name System Security Extension* (DNSSec) dient der integritätsgesicherten Auslieferung von DNS Records. Dafür werden Einträge mit Zonenschlüsseln signiert, Zonenschlüssel wiederum als DNS Einträge in übergeordneten Zonen hinterlegt. So entsteht, ähnlich einer PKI, eine Vertrauenshierarchie. Dieses System bietet die Grundlage für weitere Technologien wie *DNS based Authentication of Named Entities* (DANE).

Ziel ist die bereits für diesen Zweck registrierte Domain IT-Sec.ovh mit DNSSec auszustatten, also zwei autoritative Nameserver in Betrieb zu nehmen und derart zu konfigurieren, dass die

Domain sowie deren Records, insbesondere der MX-Record von IT-Sec.ovh integritätsgeschützt über DNSSec aufgelöst werden können.

Die zwei Server sollen, wenn möglich, mit zwei unterschiedlichen Linux-Serverbetriebssystemen betrieben werden, wobei der Fokus auf Enterprise/Long-Term-Distributionen liegen sollte, zum Beispiel RedHat-Enterprise-Linux 7 bzw. das kostenfreie Pendant CentOS 7 und Debian 8.

1.3. DANE

DNS-based Authentication of Named Entities (DANE) ist ein Verfahren zur Absicherung der weit verbreiteten Transportwegverschlüsselung SSL/TLS. Es stellt sicher, dass die verwendeten Zertifikate nicht unbemerkt ausgewechselt werden können, und erhöht so die Sicherheit beim verschlüsselten Transport von E-Mails.

Zielsetzung ist, einen mit DANE abgesicherten Mailserver (MTA) inklusive den zum Betrieb nötigen Nameserver bzw. DNS-Resolver einzurichten, die nötigen Schritte zu beschreiben und die verwendete Konfiguration zu optimieren sowie den praktischen Betrieb zu testen.

Die Umsetzung erfolgt vorzugsweise mit den hierfür gängigsten OpenSource-Softwarelösungen (Bind, Postfix) auf den weit verbreiteten Long-Term/Enterprise-Linux-Distributionen RHEL/CentOS 7 und Debian 8.

1.4. Mailserver

Konkret sollen folgende fachlichen Anforderungen aus [BSI-TR03108, Kapitel 3.2] umgesetzt werden:

- Kommunikation Mail-User-Agent (MUA) des Anwenders zum Mail-Transfer-Agent (MTA) erfolgt mittels SMTP mit STARTTLS unter Verwendung hochwertiger Algorithmen.
- Eine dem Benutzer bereitgestellte WebMail-Oberfläche wird ausschließlich über einen sicher konfigurierten (d.h. mit solider HTTPS Konfiguration betriebenen) WebServer verfügbar gemacht.
- Verbindungen zwischen MTAs müssen mit STARTTLS unter Verwendung hochwertiger Algorithmen und vorzugsweise mit Perfect Forward Secrecy erfolgen, sofern der Kommunikationspartner dies unterstützt.
- Die verwendeten Zertifikate stammen aus einer vertrauenswürdigen Certificate Authority deren Stammzertifikat in gängigen Distributionen bereits getrustet hinterlegt ist.
- Verbindungen zwischen MTAs erfolgen unter Verwendung von DANE, sofern der Kommunikationspartner dies unterstützt.

1.5. Test-Equipment (Server, Zertifikate, …)

- Die Server werden in der Microsoft Azure-Cloud als virtuelle Server gehostet.
- Zertifikate werden von StartCom Ltd. StartSSL (oder alternativ Let's Encrypt) bezogen.

2. DNS mit DNSSec unter Verwendung von BIND 9.9

Das Domain Name System (DNS) ist ein verteiltes System, Requests und Responses werden über ein ungeschütztes UDP-basiertes Klartextprotokoll abgewickelt, dass sich daher mit bewältigbarem Aufwand stören bzw. täuschen lässt (Stichworte: fehlender Integritätsschutz, einfache Absenderadressfälschung, DNS-Spoofing / DNS-Cache-Poisoning, behördliche Eingriffe „Stoppschild“, …).

Security war ursprünglich kein Bestandteil von DNS. Lediglich für den Zonen-Transfer zwischen Master und Slave-Servern wurden Transaction Signatures (TSIG) auf Basis von shared Secrets eingesetzt (siehe Kapitel 2.4.3). Eine Einführung in DNS und die Absicherung von DNS-Servern (ohne hierbei jedoch im Detail auf DNSSec einzugehen) bietet [Bauer-LinSrv].

Abbildung 2 stellt eine Übersicht über mögliche Angriffspunkte dar.

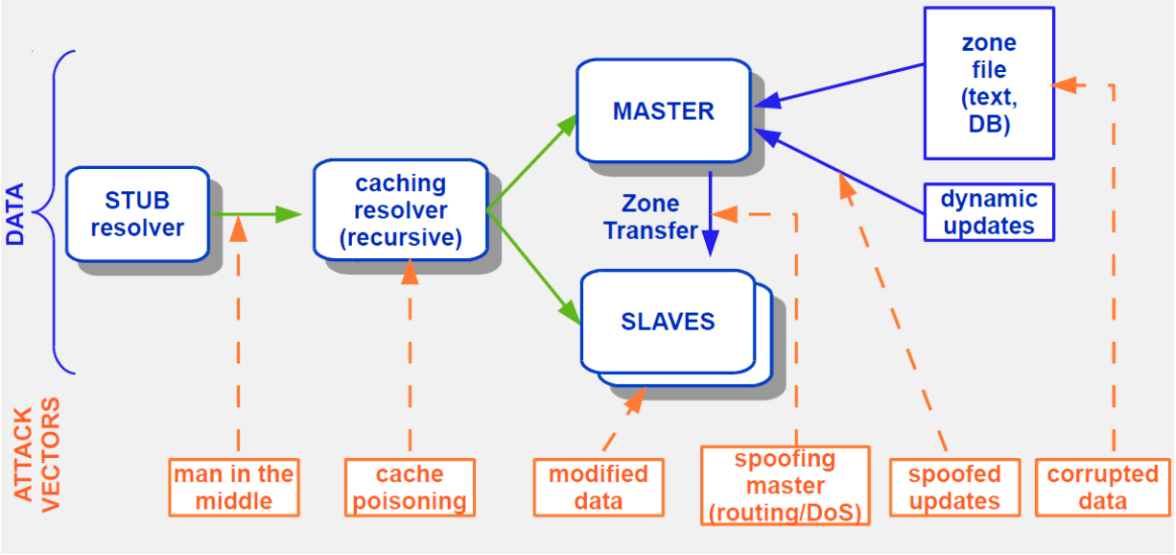


Abbildung 2: DNS Daten-Fluss, mögliche Angriffspunkte - Quelle: [SNE-DNSSec1b]

2.1. Einführung DNS Security Extension (DNSSec)

Die DNS Security Extension (DNSSec) ergänzt das Domain Name System um einen kryptographisch gesicherten Integritätsschutz, basierend auf asymmetrischer Kryptographie und ermöglicht Data-Origin-Authentication. Es schützt jedoch weiterhin nicht vor Denial of Service und bietet auch keine Vertraulichkeit [RFC-4033].

Die Funktionsweise des heute im Einsatz befindlichen DNSSec wird in [RFC-4033] beschrieben, mit [RFC-4034] werden die hierfür nötigen zusätzlichen Resource-Records eingeführt und mit [RFC-4035] das Protokoll entsprechend angepasst. Mit [RFC-5155] wurde schließlich noch etwas später eine Erweiterung des authenticated Denial of Existance Verfahrens (NSEC3) eingeführt und hiermit die Privacy-Situation erheblich verbessert (siehe Kapitel 2nd6th).

Die zuvor angeführten RFC’s zum Thema DNSSec stammen aus dem Frühjahr 2005, sind somit mehr als zehn Jahre alt. Bis DNSSec überhaupt breitflächig im Internet anwendbar war vergingen einige Jahre, erst im Jahr 2010 wurde die DNS-Root-Zone signiert[2]. Heute im Jahr 2015 sind die meisten relevanten globalen und nationalen Top-Level-Domain-Zonen[3] signiert. Dennoch ist die Verbreitung von DNSSec bis heute überschaubar. Per Oktober 2015 sind zum

[2] http://www.root-dnssec.org/2010/07/16/status-update-2010-07-16/

[3] http://stats.research.icann.org/dns/tld_report/

Beispiel in der `.com` Zone lediglich ½ Million der ca. 119 Millionen Domains signiert, was nur 0,4% entspricht [StatDNS]. Interessanter und praxisnäher ist jedoch eine andere Statistik: Die Verbreitung DNSSec fähiger Resolver und Server ist nämlich sehr gut. So hat ein Master Research Project der Universität Amsterdam im Frühjahr 2014 erhoben, dass 88% der DNS-Queries im Internet mit gesetztem DO-Bit erfolgen (also von Clients erfolgen, die auf eine DNSSec Query Wert legen), und von den verwendeten Resolvern auch tatsächlich 28% der Antworten mit gesetztem AD-Bit (Authenticated Data, somit DNSSec verifiziert) geliefert wurden [SNE-DNSSec2]. Die Diskrepanz ist einerseits darauf zurückzuführen, dass einige populäre Domains DNSSec nutzen, welche viel häufiger abgefragt werden, andererseits vermutlich auch darauf, dass die Zonen der TLDs selbst ja mehrheitlich DNSSec aktiviert sind, und ein beträchtlicher Anteil der Queries auch auf TLDs selbst entfällt.

2.2. Überblick über die Funktionsweise von DNSSec

DNSSec verwendet Signaturen auf Basis asymmetrischer Kryptographie (hauptsächlich RSA, jedoch auch die Verwendung von DSA und ECC ist vorgesehen [IANA-DNSSec]). Hierzu werden neue Resource-Record-Typen eingeführt, um Signaturen und Public-Keys im DNS zu hinterlegen und zu transportieren [RFC-4034]:

`RRSIG` Signature Resource Record
enthält die kryptographische Signatur zu einem Record

`DNSKEY` DNS Key Resource Record
enthält den Public Key, wird von Resolver zur Signatur-Verifikation genutzt

`DS` Delegation Signer Resource Record (in der darüber liegenden Parent-Zone)
enthält den Hash eines DNSKEY (typischerweise des Key Signing Keys)

Weiters wurden noch drei Resource-Records eingeführt, um eine authentifizierte denial of existence zu ermöglichen (Details siehe hierzu Kapitel 2nd6th):

`NSEC` Next Secure Resource Record

`NSEC3` Next Secure v3 (oder NSEC Hash) Resource Record

`NSEC3PARAM` NSEC3 Parameter

Jede abzusichernde Zone verfügt über zumindest ein einzigartiges Schlüsselpaar. Signiert wird mittels nur beim Administrator der DNS-Zone verfügbarer private-Keys. Die Public-Keys werden in der Zone als DNSKEY Resource-Record hinterlegt und deren Hash in der darüber liegenden Zone als Delegation-Signer (DS) verankert. So entsteht ausgehend von der ebenfalls signierten Root-Zone und den signierten Top-Level-Domains eine Chain of Trust. Das System stellt es den Clients bzw. Resolvern frei, ob sie von der Möglichkeit DNSSec zu verwenden Gebrauch machen. Tun sie das, können diese sämtliche rekursiv ermittelten Antworten kryptografisch prüfen, indem zu jedem Eintrag der zugehörige Signature-Resource-Record (RRSIG) geprüft und die darin enthaltene Signatur mittels der ebenfalls abrufbaren Public-Keys verifiziert werden. Durch die über die Delegation-Signer gewährleistete Chain of Trust kann sich so ein Client bzw. ein Resolver von der Authentizität und Integrität der aufgelösten Anfragen überzeugen, lediglich der IANA-Root-Key muss hierfür (statisch) im Resolver hinterlegt sein.

Abbildung 3 illustriert das Zusammenspiel der DNSSec Resource-Records. In der Zone von `.ovh` finden sich neben den NameServer-Einträgen nun auch zumindest ein Delegation-Signer (DS) Eintrag für die Zone `it-sec.ovh`. In der Zone `it-sec.ovh` selbst befindet sich der langlebige Key-Signing-Key sowie der kurzlebigere Zone-Signing-Key. Mittels des ZSK werden alle Einträge in der Zone `it-sec.ovh` einzeln signiert. Der ZSK selbst wird mittels KSK signiert. Und

der Hash des KSK findet sich als DS-Eintrag in der darüber liegenden `.ovh` Zone. Die mit DNSSec ausgestatteten Einträge der `.ovh` Zone wiederum sind mit dem ZSK der `.ovh` Zone signiert, welche vom KSK der `.ovh` Zone signiert ist (vgl. [heise-DNSSec1]).

In Abbildung 3 nicht mehr dargestellt: Der Hash des KSK der `.ovh` Zone wiederum findet sich in Form eines DS-Eintrages in der Root-Zone. Sämtliche Schlüssel und Signaturen sind mit Zeitstempeln und Ablauf-Datum versehen. Zwecks Key-Rollover ist es daher nicht unüblich, dass zur gleichen Zeit auch zwei oder mehr KSK/ZSK-Keys gültig sind, und somit auch mehrere DS-Einträge vorhanden sein können. Im einfachen Fall jedoch existiert für die Domain nur ein KSK, dessen Hash in der darüber liegenden TLD als DS-Eintrag verankert ist. Der ZSK kann dann am Nameserver selbst (ohne Änderungen an der Registrierung der Domain über den Domainregistrar vornehmen lassen zu müssen) in der Zone jederzeit gewechselt werden.

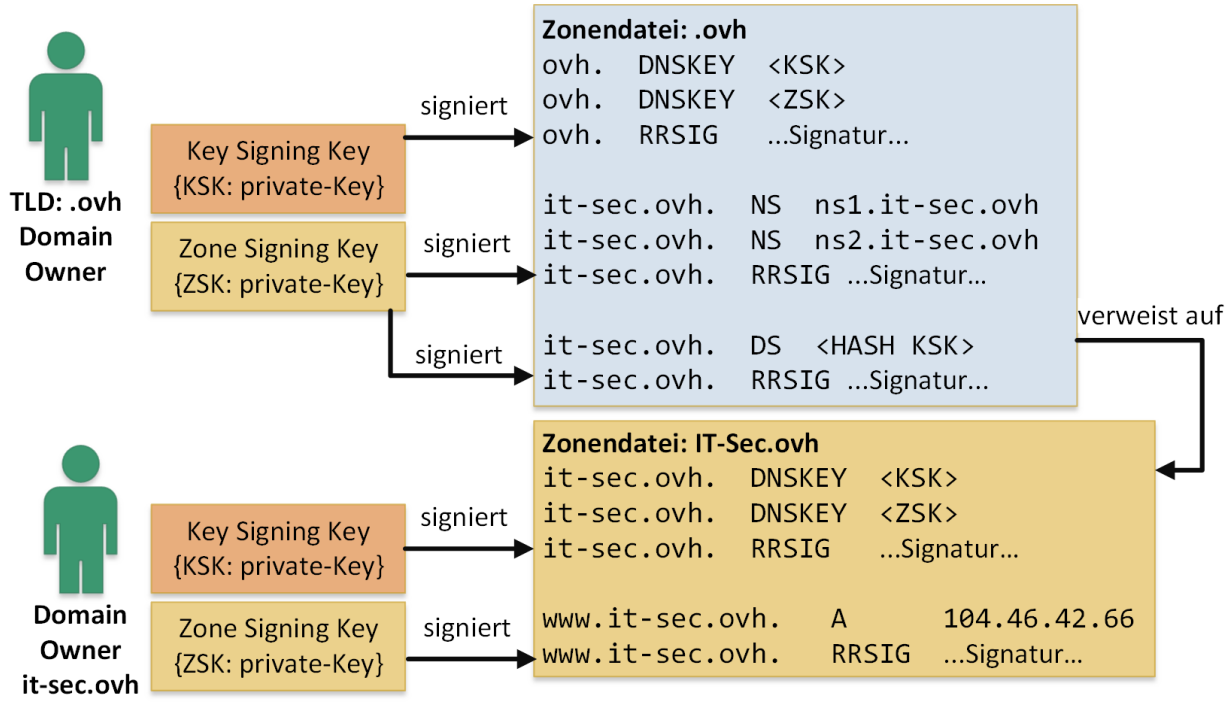


Abbildung 3: DNSSec Einträge und deren Zusammenwirken

Anhand der im Zuge der Ausarbeitung dieses Dokuments in Betrieb genommenen Demonstrations-Domain „it-sec.ovh“ lässt sich der Ablauf aber gut nachvollziehen, eine grafische Darstellung der Hierarchie und Vertrauensstellungen der Schlüssel ist mittels des grafischen Tools DNSViz möglich, siehe hierzu Kapitel 2.10.5.

2.3. Installation und Basis-Konfiguration von BIND 9.9

Nameserver-Software steht für zahlreiche Betriebssysteme zur Verfügung, teils ist diese Bestandteil von Serversystemen (z.B. Windows Server 2008 / 2012), in der Regel wird ein Nameserver-Daemon jedoch als zusätzlicher Dienst auf einem Serversystem ergänzt. Prominente Vertreter sind z.B. BIND[4], djbdns[5], KNOT DNS[6], NSD[7] und PowerDNS[8], hinzu kommen noch zahlreiche Server wie Dnsmasq[9] oder Unbound[10] die lediglich die Funktionalitäten eines Resolvers, nicht jedoch eines autoritativen Nameservers bereitstellen. Eine sehr umfassende Übersicht findet sich auf Wikipedia[11].

Die ursprünglich von der University of California at Berkeley entwickelte und nun vom Internet Systems Consortium als OpenSource-Software bereitgestellte Software BIND (Berkeley Internet Name Domain) implementiert einen Internet-Domain-Nameserver. Gemäß ISC-Website ist BIND die am weitesten verbreitete Nameserver-Software im Internet (gemäß eines Blog-Eintrages[12] auf der ISC-Website wird ein Marktanteil von 85% angenommen).

DNSSec ist bereits seit einigen Jahren fixer Bestandteil von BIND, grundsätzlich eignen sich für nachfolgende Schritte alle BIND9-Versionen, die konkrete Konfiguration kann sich bei BIND9-Versionen < 9.9 jedoch geringfügig unterscheiden `[ISC-DNSSec, Kapitel 2.1.1]`. Als aktuelle Linux-Long-Term-Distributionen werden CentOS 7 und Debian 8 herangezogen, diese bringen BIND in den Versionen 9.9.4 bzw. 9.9.5 mit.

2.3.1. BIND 9.9.4 auf CentOS 7

CentOS 7 wird mit BIND 9.9.4 ausgeliefert. Die Installation erfolgt über die Paketverwaltung:

```
[root@Sec-NS1 named]# yum install bind bind-utils
[root@Sec-NS1 named]# systemctl enable named
[root@Sec-NS1 named]# systemctl start named
```

Prüfen der installierten Version:

```
[root@Sec-NS1 named]# named -V
BIND 9.9.4-RedHat-9.9.4-18.el7_1.3 …
using OpenSSL version: OpenSSL 1.0.1e 11 Feb 2013
using libxml2 version: 2.9.1
```

Die Konfiguration liegt in `/etc` und die Hauptkonfigurationsdatei lautet `/etc/named.conf` und sollte nicht an einen anderen Ort verlegt werden. Eigene autoritative Zonen sollen unterhalb von `/etc/bind` angelegt und in die `named.conf` eingetragen werden. Sämtliche Konfigurationen um einen DNSSec-tauglichen rekursiven Resolver zu betreiben sind bereits vorbereitet.

4 http://www.isc.org/downloads/bind/

5 http://cr.yp.to/djbdns.html

6 https://www.knot-dns.cz/

7 http://www.nlnetlabs.nl/projects/nsd/

8 https://www.powerdns.com/

9 http://thekelleys.org.uk/dnsmasq/doc.html

10 https://www.unbound.net/

11 https://en.wikipedia.org/wiki/Comparison_of_DNS_server_software

12 https://www.isc.org/blogs/dnsbind-canards-redux/

Die mitgelieferten Haupt-Konfigurationsdatei `/etc/named.conf` bindet weitere Konfigurationsdateien ein:

DNSSec Root-Zonen-Key:	`/etc/named.root.key`
Default-Zonen (localhost, …):	`/etc/ named.rfc1912.zones`
BIND options:	sind unter CentOS direkt in der `/etc/named.conf`
Eigene lokale Konfigurationen:	sind unter CentOS nicht als dediziertes File vorbereitet

BIND läuft als Prozess `named` unter CentOS nicht mit root-Rechten sondern als User `named` mit Gruppe `named`. Die Konfigurationsdateien müssen von diesem nicht privilegierten User gelesen werden können. Die Verwendung von SELinux wird in dieser Konfiguration nicht berücksichtigt, SELinux wurde im Modus `permissive` (zu konfigurieren in `/etc/selinux/config`) betrieben.

Wird die CentOS-Firewall verwendet, so ist das Service DNS wie folgt zu aktivieren:

```
[root@Sec-NS1 ~]# firewall-cmd --add-service=dns
success
```

BIND ist auf CentOS 7 standardmäßig nur auf localhost gebunden, dies ist wie folgt in der `/etc/named.conf` mit „`any;`“ oder der Server-IP-Adresse zu ergänzen, weiters ist `allow-query` und `allow-recursion` zu konfigurieren. Sollte der Server nicht über IPv6 verfügen ist der Konfigurationseintrag `listen-on-v6` auszukommentieren:

```
options {
        listen-on port 53 { 127.0.0.1; any; };
        listen-on-v6 port 53 { ::1; any; };
…
        allow-query     { localhost; any; };
        allow-recursion { localhost; };
```

2.3.2. BIND 9.9.5 auf Debian 8

Debian 8 wird mit BIND 9.9.5 ausgeliefert: https://packages.debian.org/jessie/bind9

Installation:

```
root@Sec-NS2:~# aptitude install bind9 bind9-doc dnsutils
```

Prüfen der installierten Version:

```
root@Sec-NS2:~# named -V
BIND 9.9.5-9+deb8u3-Debian (Extended Support Version) …
using OpenSSL version: OpenSSL 1.0.1k 8 Jan 2015
using libxml2 version: 2.9.1
```

Die Spezifika des mit der Debian-Distribution gelieferten BIND-Paketes werden in `/usr/share/doc/bind9/README.Debian.gz` erläutert.

Die Konfiguration liegt in `/etc/bind` und die Hauptkonfigurationsdatei lautet `/etc/bind/named.conf` und sollte nicht an einen anderen Ort verlegt werden. Eigene autoritative Zonen können unterhalb von `/etc/bind` angelegt und in die `named.conf.local` eingetragen werden. Sämtliche Konfigurationen um einen DNSSec-tauglichen rekursiven Resolver zu betreiben sind bereits vorbereitet, diese sind insbesondere:

DNSSec Root-Zonen-Key:	`/etc/bind/bind.keys`
BIND options:	`/etc/bind/named.conf.options`
Eigene lokale Konfigurationen:	`/etc/bind/named.conf.local`
Default-Zonen (localhost, …):	`/etc/bind/named.conf.default-zones`

BIND läuft als Prozess `named` unter Debian nicht mit root-Rechten sondern als User `bind` mit Gruppe `bind`. Die Konfigurationsdateien müssen von diesem nicht privilegierten User gelesen werden können.

Auf der verwendeten virtuellen Hardware (Azure-Cloud) ist kein IPv6 verfügbar. Unter Debian wird IPv6 für BIND durch Modifikation der `/etc/default/bind9` deaktiviert. Die `OPTIONS` sind um ein `-4` zu ergänzen:

```
root@Sec-NS2:~# vi /etc/default/bind9
…
# startup options for the server
OPTIONS="-4 -u bind"
```

Das `bind9.service` Script von Debian Jessie ist jedoch fehlerbehaftet, der Package-Maintainer hat diesen Bug (obwohl bereits seit einem Jahr bekannt) noch immer nicht behoben[13]. Die OPTIONS-Variable wird nicht eingelesen, das Script muss manuell wie folgt ergänzt werden:

```
root@Sec-NS2:~# vi /lib/systemd/system/bind9.service
…
[Service]
EnvironmentFile=/etc/default/bind9
ExecStart=/usr/sbin/named -f $OPTIONS
…
```

[13] https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=767798

2.3.3. Überlegungen zur BIND 9.9 Konfiguration

Von einem Betrieb in einer ChangeRoot-Konfiguration wird an dieser Stelle abgesehen, in Zeiten von Cloud-Computing und fortgeschrittener Virtualisierung erscheint es dem Autor nicht mehr zweckmäßig, Hosts trotz derartiger Sicherheitsbedenken mit mehreren Aufgaben zu versehen, stattdessen sollte eine Trennung – wenn nötig – gleich auf Host-Ebene durch Verwendung separater virtueller Maschinen erfolgen.

Im Default-Zustand gibt Bind bereitwillig Informationen zu Hostname und Version Preis:

```
root@Sec-NS2:/etc/named/domains# dig +short chaos txt hostname.bind @localhost
"Sec-NS2"
root@Sec-NS2:/etc/named/domains# dig +short chaos txt version.bind @localhost
"9.9.5-9+deb8u3-Debian"
```

Es empfiehlt sich dies durch setzen entsprechender Optionen zu unterbinden. Im Falle, dass der Server (auch) als Resolver genutzt wird, sollte die Aktivierung von DNSSec an dieser Stelle geprüft und nötigenfalls aktiviert werden (`dnssec-enable`). Des Weiteren sollte geprüft werden, dass die `allow-query` Konfiguration eine Nutzung des Nameservers aus allen Netzen zulässt (wir haben vor einen autoritativen Nameserver zu betreiben), und die `allow-recursion` Option nur `localhost` oder eventuell dedizierte lokale Netze erlaubt (der Nameserver soll keinesfalls von Unautorisierten als Resolver genutzt werden können!):

```
CentOS: vi /etc/named.conf
Debian: vi /etc/bind/named.conf.options

…
options {
        …
        version none;
        hostname none;
        server-id none;

        allow-query     { localhost; any; };
        allow-recursion { localhost; };

        dnssec-enable yes;
        dnssec-validation auto;
        dnssec-lookaside auto;
        …
}
```

Die Option `dnssec-validation` ist mit `auto` zu konfigurieren [ISC-DNSSec, Kapitel 3.3] um den Built-In Default-Trust-Anchor der Root-Zone zu nutzen. Alternativ müsste bei Konfiguration mit `yes` der Key der Root-Zone z.B. mittels des unter Debian mitgelieferten Files `/etc/bind/bind.keys` manuell konfiguriert werden.

Neu-Laden der Nameserver-Konfiguration:

```
CentOS: systemctl reload named
Debian: systemctl reload bind9
```

2.3.4. Prüfung des DNSSec Resolvers

Die Funktionsweise des DNSSec fähigen Resolvers kann mittels dig geprüft werden. Das „do“ Flag bedeutet „DNSSec OK“ und signalisiert, dass der verwendete Nameserver DNSSec unterstützt. Das „ad“ Flag bedeutet „Authenticated Data“ und zeigt somit an, dass die Antwort den DNSSec Validierungsprozess bestanden hat [ISC-DNSSec, Kapitel 3.2.2].

```
root@Sec-NS2:~# dig @localhost www.isc.org. A +dnssec +multiline

; <<>> DiG 9.9.5-9+deb8u3-Debian <<>> @localhost www.isc.org. A +dnssec +multiline
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52840
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.isc.org.           IN A

;; ANSWER SECTION:
www.isc.org.            60 IN A 149.20.64.69
www.isc.org.            60 IN RRSIG A 5 3 60 (
                                20151107032012 20151008032012 6003 isc.org.
                                CVlfW5gDT9owDi+kdUZrYChhtK28aRxAdYeAWZ5ObtbI
                                HOdH4HQl++JD3N4wizi7vpSvyGTLDmrPaN8K7KmTF7B2
                                IG1/FGg00mGkgk1NzeT3V1O0RlUq9Gv9HX438om52D+O
                                1lWXK6FLHXOZVdTUrQ2LlY0bMaI41zp3RfEgD+s= )

;; Query time: 5127 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Oct 11 15:33:23 CEST 2015
;; MSG SIZE  rcvd: 223
```

Der RRSIG Record beinhaltet zum einen die prüfbare Signatur zum angefragten A-Record, andererseits auch Attribute um diese validieren zu können [RFC-4034, Kapitel 3.3], z.B.:

60 … TTL des RRSIG Records in Sekunden

A … der RRSIG Eintrag ist für den zugehörigen A-Record bestimmt

5 … es wird RSA/SHA1 verwendet

3 … Anzahl der Labels im Original-Namen (Stichwort: Nutzung von Wildcards)

60 … TTL des zugehörigen (A-)Records

20151107032012 … 2015-11-07 03:20:12 (UTC) = Ablaufdatum der Signatur

20151008032012 … 2015-10-08 03:20:12 (UTC) … Gültigkeitsdatum (Start) der Signatur

6003 isc.org … es wurde der DNSKEY mit dem Key-Tag 6003 der Zone isc.org für die Signatur verwendet, die Prüfung hat daher mit diesem Key zu erfolgen.

Wäre die Signatur-Prüfung fehlgeschlagen, wäre die Antwort nicht NOERROR sondern SERVFAIL gewesen [ISC-DNSSec, Kapitel 3.3.3].

2.4. Zonen-Konfiguration (ohne DNSSec)

Aus Gründen der Übersichtlichkeit und zur klaren Trennung bzw. Kapselung der eigenen BIND-Konfigurationen und Zonen wird ein separates (Distributions-unabhängiges) Konfigurationsverzeichnis erstellt und dieses über die `named.conf` bzw. `named.conf.local` der jeweiligen Distribution eingebunden. Dies hat auch den Vorteil, dass sämtliche Konfigurationen einfach gesichert und von einer Maschine zur anderen transferiert werden können. Der Betrieb mehrerer autoritativer Nameserver mit gleicher Konfiguration ist so einfach möglich (z.B. mittels eines rsync-Scripts).

2.4.1. Erstellung des Konfigurationsverzeichnisses

```
mkdir /etc/named/domains
Debian: chown -R root:bind /etc/named
CentOS: chown -R root:named /etc/named
chmod -R 2750 /etc/named ;#(SGID-Bit gesetzt um die Owner-Group auf Files zu vererben)
```

2.4.2. Erstellung und Einbindung der Zonen-Konfigurationsdatei

Erstellung der Datei: `/etc/named/domains/domains.conf`

```
// zone "example.com"  IN { type master; file "/etc/named/domains/zone_example.com"; };
zone "it-sec.ovh"      IN { type master; file "/etc/named/domains/zone_it-sec.ovh"; };
```

In dieser Datei domains.conf wird für jede autoritativ aufzulösende Zone (Domain) eine Zeile nach obigem Muster angelegt.

Die Datei `domains.conf` wird in die Konfiguration des Nameservers eingebunden. Unter Debian ist nachfolgender Eintrag in der `/etc/bind/named.conf.local` zu ergänzen. Unter CentOS in Ermangelung einer lokalen Konfigurationsdatei direkt in `/etc/named.conf` (am Ende der Datei):

```
include "/etc/named/domains/domains.conf";
```

Anschließend wird für jede Zone eine Konfigurationsdatei nach folgendem Muster erstellt: `/etc/named.domains/zone_it-sec.ovh`

```
$TTL 20M              ; Time To Live (wie lange sollen Resolver die Antworten cachen)
                      ; Zone-Origin (FQDN des prim. NameSrv)
                      ; Zone-Contact (Mail-Adresse des DNS-Admin, @ = Punkt!)
@ IN SOA ns1.it-sec.ovh. dns-admin.hitco.at. (
     2015091301       ; Serial, z.B. YYYYMMDDrr
             20M      ; Refresh (Wartezeit fuer Sec-NameSrv)
             2M       ; Retry (Wartezeit fuer Sec-NameSrv bei Zone-Transfer Fehlschlag)
             25D      ; Expire (Wie lang soll Sec-NameSrv Zone behalten)
             20M )    ; TTL Negativ Cache (wie lange soll NXDomain gecached werden)

@                       IN      NS      ns1.it-sec.ovh.
@                       IN      NS      ns2.it-sec.ovh.

ns1             IN      A       40.74.62.0
srv1            IN      A       40.74.62.0
ns2             IN      A       104.46.42.66
srv2            IN      A       104.46.42.66

@               IN      MX 10   mail.it-sec.ovh.
@               IN      A       81.89.102.244
www             IN      A       81.89.102.244
```

Checklist für die Zonen-Konfiguration (SOA) [NIST-81.2, Kapitel 10.1]:

- Refresh: sollte für signierte Zonen geringer sein, als die RRSIG Gültigkeitsdauer (siehe Kapitel 2.11). Refresh gibt an, wie lange die sekundären Nameserver warten sollen, bevor sie wieder beim Master-Server betreffend aktualisierter Zonen-Daten nachfragen (Polling). Diese Zeit sollte typischerweise zwischen 20 Minuten und 2 Stunden liegen, bei Verwendung von `notify` (Konfiguration siehe Abschnitt 2.4.3) informiert der Master Server seine Slave-Server zusätzlich aktiv bei Änderung (Push).
- Retry: sollte 1/10 des Refresh-Wertes betragen, gibt an wie oft ein sekundärer Nameserver bei Fehlschlagen eines Zone-Transfers bis zur Wiederholung warten soll.
- Expire: sollte 2-4 Wochen betragen. Es handelt sich um jene Zeit die ein sekundärer Nameserver die Zonen-Informationen auch bei Ausfall des primären Nameservers als gültig ansieht.
- TTL: sollte mindestens 5 Minuten, typischerweise 30min bis 5 Tage betragen. Dies hängt davon ab, wie häufig Updates zu erwarten sind und wie schnell diese auf allen cachenden Resolvern gültig werden müssen.

Syntax-Prüfung der Zonen-Datei:

```
root@Sec-NS2:/etc/named/domains# named-checkzone it-sec.ovh zone_it-sec.ovh
zone it-sec.ovh/IN: loaded serial 2015091301
OK
```

Syntax-Prüfung der Nameserver-Konfiguration:

```
root@Sec-NS2:/etc/named/domains# named-checkconf -z
zone it-sec.ovh/IN: loaded serial 2015091301
zone localhost/IN: loaded serial 2
zone 127.in-addr.arpa/IN: loaded serial 1
zone 0.in-addr.arpa/IN: loaded serial 1
zone 255.in-addr.arpa/IN: loaded serial 1 zone 0.in-addr.arpa/IN: loaded serial 0
```

Neu-Laden der Nameserver-Konfiguration:

```
CentOS: systemctl reload named
Debian: systemctl reload bind9
```

2.4.3. Master-Slave Betrieb

Zwecks Ausfallsicherheit werden Nameservices in der Regel redundant betrieben. Mehrere Server manuell oder gescriptet mit identischen Zonen-Konfigurationen zu versehen wäre zwar denkbar, für gewöhnlich soll eine Konfigurationsänderung jedoch nur auf einem Master-Server durchgeführt und von allen anderen Slave-Servern unmittelbar übernommen werden.

Die Server zu Server Kommunikation zum Zwecke von Zone-Transfers wurde ursprünglich gar nicht abgesichert, später in der Regel auf IP-Adressen beschränkt. Beides bietet heutzutage jedoch keinen akzeptablen Schutz, weshalb schon seit einigen Jahren die Nutzung von TSIG (Transaction Signatures) gebräuchlich ist. Transaction Signatures (TSIG) gewährleisten einen kryptographischen Schutz. Die Vorgangsweise ist im BIND-Manual [ISC-BIND9, Kapitel 4.5] beschrieben. Hierbei sichert ein auf beiden Servern bekanntes (symmetrisches) pre-shared Secret zusammen mit Zeitstempel sowohl den Zone-Transfer-Request (AXFR) als auch den eigentlichen Zonen-Transfer kryptographisch ab.

Ein `keys` Verzeichnis wird angelegt und darin ein Key für das Serverpaar `ns1.it-sec.ovh/ns2.it-sec.ovh` erstellt. Dabei wird ein sicherer 256bit Schlüssel gewählt.

```
root@Sec-NS2:/etc/named/domains# mkdir /etc/named/domains/keys
root@Sec-NS2:/etc/named/domains# chmod 2770 /etc/named/domains/keys

root@Sec-NS2:/etc/named/domains/keys#
dnssec-keygen -a hmac-sha256 -b 256 -n HOST ns1.it-sec.ovh_ns2.it-sec.ovh
Kns1.it-sec.ovh_ns2.it-sec.ovh.+163+47076

root@Sec-NS2:/etc/named/domains/keys# ls -la *ns1*
-rw-r--r-- 1 root bind  93 Sep 27 17:34 Kns1.it-sec.ovh_ns2.it-sec.ovh.+163+47076.key
-rw------- 1 root bind 188 Sep 27 17:34 Kns1.it-sec.ovh_ns2.it-sec.ovh.+163+47076.private

root@Sec-NS2:/etc/named/domains/keys# grep -h Key *ns1.it-sec.ovh_ns2.it-sec.ovh*
Key: B8BJ/Kgf4g3xDHXbT3x6lWfbdwXXMuXetqX6J1vz/Kk=
```

Das so erzeugte Shared-Secret wird danach in den Konfigurationsdateien der beiden Server wie folgt hinterlegt. Aus Sicherheitsgründen sollte für jedes Host-Pärchen ein separater, sicherer TSIG Key mit einer Länge von mindestens 128 Bit verwendet werden, anstelle die Keys direkt im Konfigurationsfile `/etc/named.conf` zu hinterlegen sollten diese in separaten (mit Filesystem-Rechten vor Zugriff durch andere User geschützten) Konfigurationsfiles die mittels include eingebunden werden hinterlegt sein [NIST-81.2, Kapitel 8.2.5].

Am Master-Server in der `/etc/named/domains/domains.conf`:

```
key ns1.it-sec.ovh_ns2.it-sec.ovh
                      { algorithm hmac-sha256;
                        secret "B8BJ/Kgf4g3xDHXbT3x6lWfbdwXXMuXetqX6J1vz/Kk=";
                      };

server 40.74.62.0     { keys { ns1.it-sec.ovh_ns2.it-sec.ovh; }; };

zone "it-sec.ovh"      IN { type master; file "/etc/named/domains/zone_it-sec.ovh";
                            allow-transfer { key ns1.it-sec.ovh_ns2.it-sec.ovh; };
                            also-notify { 40.74.62.0; };
                          };
```

- Mittels `key` wird ein neuer Schüssel angelegt.
- Mittels `server` wird definiert, welcher Key für die Kommunikation mit dem Peer-Server (also dem Slave) verwendet werden soll.
- In der Zone wird definiert, dass ein Zonen-Transfer nur unter Verwendung des definierten Schlüssels erfolgen darf. Außerdem wird definiert, dass der zweite Server aktiv über Änderungen an den Einträgen der Zone informiert werden soll.

Am Slave-Server wird eine `/etc/named/domains/domains.slave.conf` angelegt und über die `named.conf` bzw. `named.conf.local` der jeweiligen Distribution eingebunden. Auch hier wird der gleiche `key` definiert, als `server` wird hier der Master-Server angegeben, und in der Zonen-Konfiguration wird als Typ `slave` gewählt und ein Master definiert.

```
key ns1.it-sec.ovh_ns2.it-sec.ovh
                      { algorithm hmac-sha256;
                        secret "B8BJ/Kgf4g3xDHXbT3x6lWfbdwXXMuXetqX6J1vz/Kk=";
                      };

server 104.46.42.66   { keys { ns1.it-sec.ovh_ns2.it-sec.ovh; }; };

zone "it-sec.ovh"      IN { type slave; file "/etc/named/domains/slavezone_it-sec.ovh";
                            masters { 104.46.42.66; };
                            allow-transfer { key ns1.it-sec.ovh_ns2.it-sec.ovh; };
                          };
```

2.4.4. Hidden-Master

In Umgebungen mit besonderem Schutzbedarf sollte die Verwendung eines „Hidden-Masters“ angedacht werden. Der eigentliche Master-Server (der wie das nachfolgende Kapitel zeigen wird auch die DNSSec-Keys hält) wird hierbei von einer Firewall geschützt (nicht öffentlich erreichbar) betrieben. Mindestens zwei Slave-Server werden geografisch dislozíert in getrennten Netzen vorgehalten, nur diese können den Master-Server über eine Firewall erreichen und mittels Transaction-Signatures einen Zone-Transfer auslösen. Die IP-Adresse des Hidden-Masters sollte hierbei nicht als Nameserver im Zone-File geführt werden. [NIST-81.2, Kapitel 7.2.7].

2.4.5. Test und Aktivierung beim Domain-Registrar

Bevor die neu eingerichteten Domain-Name-Server nun als autoritative Server beim Domain-Registrar aktiviert werden, sollten zumindest folgende Tests durchgeführt werden:

Von einem separaten Gerät aus wird geprüft, ob NS1 und NS2 autoritativ antworten:

```
root@test:~# dig @40.74.62.0 www.it-sec.ovh
…

root@test:~# dig @104.46.42.66 www.it-sec.ovh

; <<>> DiG 9.9.5-3ubuntu0.5-Ubuntu <<>> @104.46.42.66 www.it-sec.ovh
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41502
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.it-sec.ovh.                        IN      A

;; ANSWER SECTION:
www.it-sec.ovh.         1200    IN      A       81.89.102.244

;; AUTHORITY SECTION:
it-sec.ovh.             1200    IN      NS      ns1.it-sec.ovh.
it-sec.ovh.             1200    IN      NS      ns2.it-sec.ovh.

;; ADDITIONAL SECTION:
ns1.it-sec.ovh.         1200    IN      A       40.74.62.0
ns2.it-sec.ovh.         1200    IN      A       104.46.42.66

;; Query time: 23 msec
;; SERVER: 104.46.42.66#53(104.46.42.66)
;; WHEN: Sun Sep 13 16:04:55 EDT 2015
;; MSG SIZE  rcvd: 127
```

Es wird geprüft, ob die Server nach außen keine rekursiven Resolver-Dienste anbieten:

```
root@test:~# nslookup nic.at 104.46.42.66
Server:         104.46.42.66
Address:        104.46.42.66#53

** server can't find nic.at: REFUSED

root@test:~# nslookup nic.at 40.74.62.0
Server:         40.74.62.0
Address:        40.74.62.0#53

** server can't find nic.at: REFUSED
```

Und auf der Maschine selbst wird geprüft, ob die rekursive Auflösung funktioniert:

```
[root@Sec-NS1 ~]# nslookup nic.at localhost
Server:         localhost
Address:        ::1#53

Non-authoritative answer:
Name:   nic.at
Address: 131.130.249.238

root@Sec-NS2:~# nslookup nic.at localhost
Server:         localhost
Address:        ::1#53

Non-authoritative answer:
Name:   nic.at
Address: 131.130.249.238
```

Fallen diese Basis-Tests positiv aus, so kann die Aktivierung der beiden Nameserver beim Domain-Registrar erfolgen. Da die Nameserver der Domain `it-sec.ovh` auf die Domain selbst referenzieren, ist in diesem Fall wichtig, dass sogenannte GLUE-Records (Einträge welche die Auflösung von `ns1.it-sec.ovh` und `ns2.it-sec.ovh` in der darüber liegenden Zone, also der Top-Level-Domain ermöglichen) zu setzen sind. Hierzu sind beim Domain-Registrar nicht nur die Namen sondern auch die IP-Adressen zu hinterlegen:

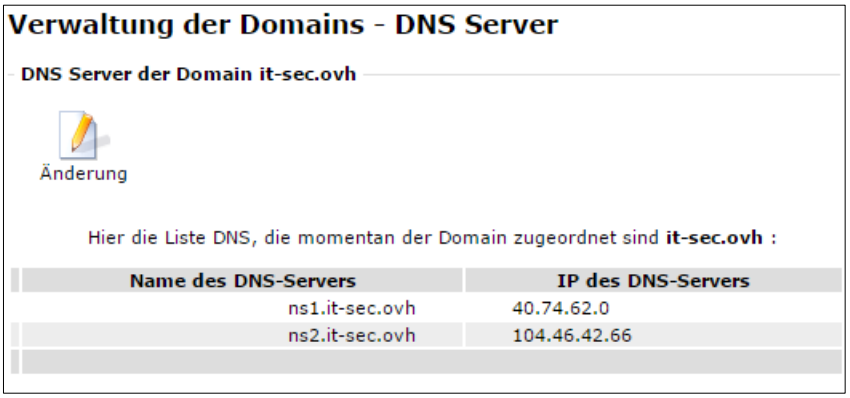


Abbildung 4: OVH.de Web-Interface: DNS-Server Konfiguration beim Domain-Registrar

2.5. DNSSec Zonen-Konfiguration

Eine Modifikation ist im Master-Slave Betrieb nur am Master-Server nötig.

Zuerst werden ein ZSK (Zone Signing Key) und ein KSK (Key Signing Key) erzeugt. Es handelt sich hierbei um RSA-Schlüsselpaare. DSA und ECDSA wären zwar unterstützt, jedoch sind die Root-Zonen und TLDs mit RSA signiert, und RSA ist auch der aktuell einzig verpflichtend zu unterstützende Algorithmus, weshalb aktuell noch RSA der Vorzug gegeben werden sollte, als Hash-Funktion sollte bereits auf SHA-256 gesetzt werden [ISC-DNSSec, Kapitel 6.1.3] und [RFC-6781, Kapitel 3.4]:

```
root@Sec-NS2:/etc/named/domains/keys#
dnssec-keygen -a RSASHA256 -b 1024 it-sec.ovh
Generating key pair..++++++ ............++++++
Kit-sec.ovh.+008+60261

root@Sec-NS2:/etc/named/domains/keys#
dnssec-keygen -a RSASHA256 -b 2048 -f KSK it-sec.ovh
Generating key pair.........................+++ ........................+++
Kit-sec.ovh.+008+50137

root@Sec-NS2:/etc/named/domains/keys# cat Kit-sec.ovh.+008+50137.key
; This is a key-signing key, keyid 50137, for it-sec.ovh.
; Created: 20150913210741 (Sun Sep 13 23:07:41 2015)
; Publish: 20150913210741 (Sun Sep 13 23:07:41 2015)
; Activate: 20150913210741 (Sun Sep 13 23:07:41 2015)
it-sec.ovh. IN DNSKEY 257 3 8 AwEAAd8F8FNYQQFl0pjZTclbUzSzWpgPt3kgHDXWrU1OXSk9++ykoWRW
WX0qHbPCiFklDU1C8U94/wxhiYzafyGMYm/Tf4D/tU9WoWo3q+bRnHeB
P2Jg9AgzPN8E7Q0ZZztbGnpKajlBvWjjSU2Qfd/wWvmHS/3lzb1fQPwN
L5HX0SsDB4wOEqC0Q18RNGHjmxLrOoIYi0z7QZNtp/EOMJ2ANHmF4SjJ
ZV2eQgMroN8KGdxaAlGivdknZFiT99RUj+PzXm7vINEpABxrBv5V8qG5
aUWQuwHIhXW7EvRp1O3iHCm/xMFflmMXNNMJtVMe4PkqyTbBPlezxDys u6mQcg1pzpU=

root@Sec-NS2:/etc/named/domains/keys# cat Kit-sec.ovh.+008+60261.key
; This is a zone-signing key, keyid 60261, for it-sec.ovh.
; Created: 20150913210726 (Sun Sep 13 23:07:26 2015)
; Publish: 20150913210726 (Sun Sep 13 23:07:26 2015)
; Activate: 20150913210726 (Sun Sep 13 23:07:26 2015)
it-sec.ovh. IN DNSKEY 256 3 8 AwEAAcScX9FiKgA+S0Q8Cwmz5WnPPJK0qWPMsbG2AJtibCiWh9nQzR60
lQ11PGVaFpPSOsMWuFnu5tPtwLlx1v2Ij/eIhAD2t+BCx+MVoWpbgaIf
9OiMPah4HskTxJ1SgsNdqSWi6XUdXd/TtJYwFvox9Hf2t8V/cvO8BmIj Ee57UWk1
```

Der Zone Signing Key (Type 256 = ZSK) signiert die ganze Zone, also jeden einzelnen Eintrag in der Zone. Um das Datenvolumen der Signaturen und somit der DNS-Antwort-Paket-Größen, sowie den mit der Prüfung verbundenen Rechenaufwand auf vertretbarem Niveau zu halten, werden auch heute noch Zone Signing Keys mit einer Schlüssellänge von nur 1024 Bit (RSA) verwendet [ISC-DNSSec, Kapitel 6.1.4]. Da die darüber liegenden Zonen aktuell ebenfalls nach dieser Praxis verfahren, erscheint eine Erhöhung des Niveaus in der eigenen Zone nicht zielführend und wird auch von [RFC-6781, Kapitel 3.4.2] nicht als nötig erachtet.

Der 2048bit Key-Signing-Key (Type 257 = KSK) hingegen signiert nur den DNSKEY-Eintrag des Zone-Signing-Keys und wird in der darüber liegenden Zone (Top-Level-Domain, im vorliegenden Beispiel ist das „.ovh") verankert.

Der Zone-Signing-Key ist daher einfacher austauschbar, ein Tausch des Key-Signing-Keys hingegen erfordert einen Eingriff beim Registrar (siehe hierzu auch Kapitel 2.11).

Anmerkung: Das Generieren von Schlüsseln erfordert Entropie. Speziell auf virtuellen Maschinen steht oft nicht genügend Entropie bereit, um effizient Schlüssel in kurzer Zeit zu generieren. Abhilfe kann hier das Paket `haveged` (Hardware Volatile Entropy Gathering and

Expansion Daemon) schaffen, welches sowohl unter CentOS als auch unter Debian bei Bedarf über die Paketverwaltung installiert werden kann [ISC-DNSSec, Kapitel 2.1.3].

Im `domains` Verzeichnis benötigt der als bind bzw. named laufende Prozess nun Schreibrechte, um die signierten Zonen-Files anzulegen. Die Keys (auch Private-Keys) müssen lesbar sein.

```
root@Sec-NS2:/etc/named/domains/keys# chown -R root:bind /etc/named/domains
root@Sec-NS2:/etc/named/domains/keys# chmod –R 0660 /etc/named/domains
root@Sec-NS2:/etc/named/domains/keys# chmod 2770 /etc/named/domains
root@Sec-NS2:/etc/named/domains/keys# chmod 2770 /etc/named/domains/keys

root@Sec-NS2:/etc/named/domains/keys# ls -la
drwxrws--- 2 root bind 4096 Sep 18 09:04 .
drwxrws--- 3 root bind 4096 Sep 18 11:33 ..
-rw-rw---- 1 root bind  603 Sep 13 23:07 Kit-sec.ovh.+008+50137.key
-rw-rw---- 1 root bind 1776 Sep 13 23:07 Kit-sec.ovh.+008+50137.private
-rw-rw---- 1 root bind  429 Sep 13 23:07 Kit-sec.ovh.+008+60261.key
-rw-rw---- 1 root bind 1012 Sep 13 23:07 Kit-sec.ovh.+008+60261.private
```

Der Eintrag in der Datei `/etc/named/domains/domains.conf` wird wie folgt ergänzt:

```
key ns1.it-sec.ovh_ns2.it-sec.ovh
                       { algorithm hmac-sha256;
                         secret "B8BJ/Kgf4g3xDHXbT3x6lWfbdwXXMuXetqX6J1vz/Kk=";
                       };

server 40.74.62.0      { keys { ns1.it-sec.ovh_ns2.it-sec.ovh; }; };

zone "it-sec.ovh"       IN { type master; file "/etc/named/domains/zone_it-sec.ovh";
                             key-directory "/etc/named/domains/keys";
                             inline-signing yes; auto-dnssec maintain;
                             allow-transfer { key ns1.it-sec.ovh_ns2.it-sec.ovh; };
                             also-notify { 40.74.62.0; };
                           };
```

Das Setting `inline-signing` automatisiert den DNSSec-Betrieb. Der BIND-Administrator muss sich nicht um die Signatur der Zone kümmern, das Zonen-File `zone_it-sec.ovh` bleibt unverändert, die nötigen Signaturen für jeden Eintrag werden automatisch beim Reload des Dienstes bzw. der Zone erzeugt und periodisch automatisch neu signiert `[ISC-DNSSec, Kapitel 6.5.2]`. Die signierte Version der Zone wird in einem gleichnamigen File mit dem Suffix `.signed` abgelegt. Das Setting `auto-dnssec maintain` sorgt für eine automatische Überwachung des `key-directory` und lädt neue Schlüssel für eine Zone automatisch [ISC-DNSSec, 6.4.2.1].

Auf dem als Master betriebenen `ns2.it-sec.ovh` unter Debian 8 erfolgt der Reload mittels:

```
root@Sec-NS2:~# systemctl reload bind9
```

Die Prüfung des Status erfolgt mittels:

```
root@Sec-NS2:~# systemctl status bind9
```

Die korrekte Funktion des Inline-Signings ist unter Debian im `/var/log/syslog` zu erkennen:

```
… named[10710]: zone it-sec.ovh/IN (unsigned): loaded serial 2015092702
… named[10710]: zone it-sec.ovh/IN (signed): serial 2015092702 (unsigned 2015092702)
… named[10710]: zone it-sec.ovh/IN (signed): sending notifies (serial 2015092702)
… named[10710]: zone it-sec.ovh/IN (signed): reconfiguring zone keys
… named[10710]: zone it-sec.ovh/IN (signed): next key event: 27-Sep-2015 21:17:27.166
```

Im Dateisystem existiert nun zu jedem Zonen-File noch ein signiertes Zonenfile sowie ein Journal-File mit der Datei-Endung `.jnl` und dessen signierte Version.

```
root@Sec-NS2:/etc/named/domains# ls -lat | grep it-sec
-rw-r--r-- 1 bind bind 5193 Sep 27 20:22 zone_it-sec.ovh.signed
-rw-r--r-- 1 bind bind 1814 Sep 27 20:17 zone_it-sec.ovh.jnl
-rw-r--r-- 1 bind bind 6062 Sep 27 20:17 zone_it-sec.ovh.signed.jnl
-rw-r----- 1 root bind 1025 Sep 27 20:17 zone_it-sec.ovh
-rw-r--r-- 1 bind bind  512 Sep 27 18:48 zone_it-sec.ovh.jbk
```

Der Inhalt des signierten Zonenfiles ist binär kodiert und kann wie folgt in ein lesbares Textfile gedumpt werden:

```
root@Sec-NS2:/etc/named/domains#
named-checkzone -D -i full -f raw it-sec.ovh zone_it-sec.ovh.signed >/tmp/zone.txt
zone it-sec.ovh/IN: loaded serial 2015110510 (DNSSEC signed)
OK

vi /tmp/zone.txt
…
```

Das Journalfile kann mittels folgenden Befehls betrachtet werden:

```
root@Sec-NS2:/etc/named/domains# named-journalprint zone_it-sec.ovh.jnl
…

root@Sec-NS2:/etc/named/domains# named-journalprint zone_it-sec.ovh.signed.jnl
…
```

Am mit CentOS 7 betriebenen Slave-Server `ns1.it-sec.ovh` erkennt man umgehend, dass die Slave-Zone aktualisiert wird, Voraussetzung hierfür ist, dass die Zonen-SerialNumber erhöht wurde:

```
[root@Sec-NS1 domains]# tail -f /var/log/messages
…
… named[2810]: client 104.46.42.66#1296/key ns1.it-sec.ovh_ns2.it-sec.ovh: received notify
for zone 'it-sec.ovh': TSIG 'ns1.it-sec.ovh_ns2.it-sec.ovh'
… named[2810]: zone it-sec.ovh/IN: Transfer started.
… named[2810]: transfer of 'it-sec.ovh/IN' from 104.46.42.66#53: connected using
100.114.178.187#38457
… named[2810]: zone it-sec.ovh/IN: transferred serial 2015092702: TSIG 'ns1.it-
sec.ovh_ns2.it-sec.ovh'
… named[2810]: transfer of 'it-sec.ovh/IN' from 104.46.42.66#53: Transfer completed: 1
messages, 10 records, 1013 bytes, 0.089 secs (11382 bytes/sec)
… named[2810]: zone it-sec.ovh/IN: sending notifies (serial 2015092702)
```

Am Slave-Server ns2.it-sec.ovh liegen Zonen-Dateien nur im RAW-Format vor, sind daher mittels eines Text-Editors nicht unmittelbar lesbar.

Mittels `named-checkzone` können diese jedoch geprüft werden. Als Argumente werden der Name der Zone sowie der Dateiname benötigt:

```
[root@Sec-NS1 domains]# named-checkzone -D -f raw it-sec.ovh slavezone_it-sec.ovh
…
```

Auch das Dumpen des binären Inhaltes in ein Textfile ist wiederum möglich:

```
[root@Sec-NS1 domains]#
named-checkzone -D -i full -f raw it-sec.ovh slavezone_it-sec.ovh >/tmp/zone.txt
zone it-sec.ovh/IN: loaded serial 2015110510 (DNSSEC signed)
```

2nd6th Authenticated Denial of Existence (NSEC & NSEC3)

Mittels DNSSec werden DNS-Einträge signiert, sodass deren Integrität kryptographisch gesichert ist. Versucht man einen nicht existenten Namen aufzulösen, so antwortet der Nameserver mit dem Status NXDOMAIN. Ein Angreifer könnte nun existierende Domains unterdrücken und stattdessen mit einem unsignierten NXDOMAIN antworten. Um einem mit DNSSec-Fähigkeiten ausgestatteten Resolver die Möglichkeit zu geben, das Nicht-Vorhandensein eines angefragten Hostnamens auch kryptographisch zu verifizieren, wurde der Resource-Record vom Typ NSEC (=Next Secure) eingeführt [RFC-4034, Kapitel 4].

NSEC verkettet alle vorhandenen Einträge einer Zone alphabetisch sortiert ringförmig miteinander, sodass zu jedem Eintrag der alphabetisch nächste Eintrag abgefragt werden kann, z.B. werden nachfolgende alphabetisch sortierten Zonen-Einträge wie folgt verkettet:

```
demo        IN  A    10.11.12.13          demo NSEC ns1
ns1         IN  A    40.74.62.0           ns1  NSEC ns2
ns2         IN  A   104.46.42.66          ns2  NSEC srv1
srv1        IN  A    40.74.62.0           srv1 NSEC srv2
srv2        IN  A   104.46.42.66          srv2 NSEC www
www         IN  A   104.46.42.66          www  NSEC demo
```

Ruft man nun den nicht vorhandenen Namen `test.it-sec.ovh` ab, so wird zur Antwort NXDOMAIN auch der NSEC Record `srv2 -> www` geliefert (da `test` zwischen `srv2` und `www` liegen würde).

```
dig @localhost test.it-sec.ovh. A +dnssec +multiline
…
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 10100
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1
…
srv2.it-sec.ovh.        1200 IN NSEC www.it-sec.ovh. A RRSIG NSEC
srv2.it-sec.ovh.        1200 IN RRSIG NSEC 8 3 1200 (
                                20151106174146 20151007164146 60261 it-sec.ovh.
                                gkj1G5Kb4m3VrUOEdFvMNTr/yr/3kKqInowg76MeG1y4
                                1PoWkVzUlqwkigusu/o+8UBU0tXelVEd/zEdhYnoLLSL
                                HI+BY1n4W8PAJNBSkmpmO/RBHvLo22NBTRalbu9tPJCy
                                m4fal290QArggg5+Cc0R7a+FCmgjOh3bmqW/srs= )
```

Ruft man einen nicht existenten Eintrag `m` ab, wird der `demo -> ns1` Eintrag geliefert:

```
dig @localhost m.it-sec.ovh. A +dnssec +multiline
demo.it-sec.ovh.        1200 IN NSEC ns1.it-sec.ovh. A RRSIG NSEC
demo.it-sec.ovh.        1200 IN RRSIG NSEC 8 3 1200 (
                                20151106174146 20151007164146 60261 it-sec.ovh.
                                Q0pQIEi7dX8ERdJvgZlO3Dmf98Eqmq8xH9rtZwHK6Zh7
                                HgPSJIVm4qMi8kVLTMtImekVH70rykgByrS7hYZ1Qs7Q
                                pQZtyijrv+dIRiwPKh9vIngG9OUHLFzd0ldMvlFO8qjz
                                cuJ98rxHhxH25QjFp4vLPiSkor3VQR15PYkfm+E= )
```

Nun kann ein Resolver damit zwar zweifelsfrei prüfen, dass ein Eintrag tatsächlich nicht vorhanden ist (der NSEC Eintrag beweist, dass sich zwischen den beiden Einträgen der angefragte Eintrag nicht befindet), kann damit aber auch relativ einfach die komplette Zone durch wiederkehrende Abfrage vollständig auslesen (Zone Walking). Dieser Umstand kann jedoch ein Privacy-Problem darstellen. Besser wäre ein Verfahren, welches nicht sämtliche Einträge einer Zone zwingend offenlegt.

Das Zone-Walking kann mittels des Tools `ldns-walk` von NLnet Labs[14] (im Debian-Paket `ldnsutils` enthalten) durchgeführt und somit einfach demonstriert werden:

```
root@Sec-NS2:~# aptitude install ldnsutils

root@Sec-NS2:~# ldns-walk @localhost it-sec.ovh
it-sec.ovh.     it-sec.ovh. A NS SOA MX RRSIG NSEC DNSKEY
demo.it-sec.ovh. A RRSIG NSEC
mail.it-sec.ovh. A RRSIG NSEC
ns1.it-sec.ovh. A RRSIG NSEC
ns2.it-sec.ovh. A RRSIG NSEC
srv1.it-sec.ovh. A RRSIG NSEC
srv2.it-sec.ovh. A RRSIG NSEC
www.it-sec.ovh. A RRSIG NSEC
```

Das Zone-Walking Verfahren kann auch in der DNSSec signierten Root-Zone angewendet werden, es werden sämtliche Top-Level-Domains geliefert:

```
root@Sec-NS2:~# ldns-walk @localhost .
.       . NS SOA RRSIG NSEC DNSKEY
aaa. NS DS RRSIG NSEC
aarp. NS DS RRSIG NSEC
abb. NS DS RRSIG NSEC
abbott. NS DS RRSIG NSEC
abogado. NS DS RRSIG NSEC
ac. NS DS RRSIG NSEC
academy. NS DS RRSIG NSEC
accenture. NS DS RRSIG NSEC
…
```

Auch auf einige Registries von Länder-Domains ist Zone-Walking so anwendbar:

```
root@Sec-NS2:~# ldns-walk @localhost br.
br.     br. NS SOA RRSIG NSEC DNSKEY
0800.br. NS DS RRSIG NSEC
acai.br. NS RRSIG NSEC
ace.br. NS RRSIG NSEC
ad1.br. NS RRSIG NSEC
adm.br. NS DS RRSIG NSEC
administracao.br. NS RRSIG NSEC
adolec.br. NS RRSIG NSEC
adv.br. NS DS RRSIG NSEC
…
```

Nicht anwendbar ist das Verfahren z.B. bei `.de` oder `.at` Domains, da diese NSEC3 einsetzen:

```
root@Sec-NS2:~# ldns-walk @localhost de.
de.     Zone does not seem to be DNSSEC secured,or it uses NSEC3.
root@Sec-NS2:~# ldns-walk @localhost at.
at.     Zone does not seem to be DNSSEC secured,or it uses NSEC3.
```

Das Nachfolge-Verfahren NSEC3 (=Next Secure v3 oder NSEC Hashed Authenticated Denial of Existence [RFC-5155]) ermöglicht es, einem Resolver ähnlich dem NSEC-Verfahren die Nicht-Existenz eines Eintrages zu belegen und dennoch die vorhandenen Zonen-Einträge nicht unmittelbar offenzulegen. Möglich wird dies durch Verwendung von Hashes anstelle von Klartext-Zonen-Einträgen. Aus Privacy-Gründen ist in der Regel daher ein Umstieg auf NSEC3 ratsam. Unzweckmäßig ist ein Einsatz von NSEC3 lediglich dann, wenn eine Zone

[14] http://nlnetlabs.nl/projects/ldns/

ausschließlich einige wenige allgemein bekannte Einträge enthält, oder diese derart strukturiert vorgehalten werden, dass ein Erraten oder sequenzielles Abfragen der Einträge mittels Schleifen trivial möglich ist, z.B. PTR-Einträge von `in-addr.arpa` Zonen [RFC-6781, Kapitel 5.3]. Auch bei der Root-Zone selbst (siehe Zone-Walking Demonstration zuvor) ist NSEC3 nicht nötig, da alle existierenden TLDs ohnehin allgemein bekannt sind.

Nach Installation und Konfiguration von Bind 9.9.5 in der zuvor beschriebenen DNSSec fähigen Konfiguration wird das zuvor beschriebene NSEC genutzt. Eine Migration auf NSEC3 ist wie folgt mittels des NameServer Control-Utility `rndc` möglich [ISC-DNSSec, Kapitel 6.2 und Kapitel 7.3.1]:

```
root@Sec-NS2:~# RANDOM=$(openssl rand -hex 16)
root@Sec-NS2:~# echo $RANDOM
578c30cc6333adf7ea61a5da07142d20

rndc signing -nsec3param 1 0 100 $RANDOM it-sec.ovh
request queued
```

Die Argumente für `-nsec3param` in angegebener Reihenfolge bedeuten:

- `1`: hash algorithm, 1 = SHA1 (derzeit kein anderer Algorithmus definiert)
- `0`: flags, 0 = Opt-Out Flag nicht gesetzt
- `100`: iterations, 100 = Bei der Hash-Generierung Hundert Iterationen durchführen.
- `$RANDOM`: salt = Salt in HEX-Schreibweise
- `it-sec.ovh`: Name der zu konfigurierenden Zone

Ein Salt sorgt dafür, dass Angreifer nicht mittels vorberechneter Tabellen vom Hash-Wert auf den Ausgangswert (den FQDN) rückschließen können. Iteratives Hashen soll zusätzlich den Aufwand für das Anfertigen von Hash-Tables oder das Brute-Forcen erhöhen und somit die Geschwindigkeit des Angriffs bremsen [RFC-6781, Kapitel 5.3.2].

Eine höhere Anzahl an Iterationen bremst Angreifer dabei, mittels Brute-Force-Angriff aus NSEC3-Antworten auf die ursprünglichen Zonen-Einträge zu schließen. Eine höhere Zahl an Iterationen bremst jedoch auch legitime Clients bei der Prüfung. Der Wert kann gemäß Empfehlung des NIST auf 200 Iterationen erhöht werden, als Entscheidungsgrundlage dient einerseits die zur Verfügung stehende Rechenleistung, andererseits muss auch beurteilt werden ob und wenn ja welche Privacy-Relevanz die in der Zone hinterlegten Einträge überhaupt aufweisen [NIST-81.2, Kapitel 10.6]. Die Anzahl der Iterationen ist somit ein Trade-off zwischen Sicherheit und Performance, es empfiehlt sich für durchschnittlichen Schutzbedarf den Wert 100 zu verwenden [RFC-6781, Kapitel 5.3].

Gemäß Empfehlung des NIST [NIST-81.2, Kapitel 10.4 und 10.6] sollte das Salt regelmäßig gewechselt werden. Ab Bind-Version 9.10 stünde hierfür ein Automatismus zur Verfügung, welcher mittels des Salts „`auto`“ aktiviert werden könnte [ISC-DNSSec, Kapitel 7.3.4], die unter Debian verfügbare Bind 9.9.5 Version akzeptiert dieses Argument jedoch noch nicht. Stattdessen werden im obigen Konfigurationsbeispiel 32 zufällige Hex-Ziffern (128 Bit entsprechen 16 Byte, generiert mittels OpenSSL) verwendet. Um bei Verwendung von Bind Versionen <9.10 das Salt regelmäßig zu wechseln, kann obige Neu-Initialisierung des Salt (welche implizit eine Neu-Generierung sämtlicher NSEC3-Einträge zur Folge hat) jederzeit wiederholt werden. Eine Rollover-Prozedur ist hierbei nicht erforderlich.

Im Log-File `/var/log/daemon.log` kann die Verarbeitung des Befehls verfolgt werden:

```
… zone it-sec.ovh/IN (signed): zone_addnsec3chain(1,CREATE,100,578c30cc6333adf7ea6…)
… zone it-sec.ovh/IN (signed): sending notifies (serial 2015100903)
```

Ein Abruf eines nicht existenten Eintrages führt nun u.a. zu folgendem Resultat:

```
dig @localhost test.it-sec.ovh. A +dnssec +multiline
…
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 22518
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 8, ADDITIONAL: 1
…
IF6JDFQA8OILAS7OEKML1RPDCV2GMHN5.it-sec.ovh. 1200 IN NSEC3 1 0 100
                                              578C30CC6333ADF7EA61A5DA07142D20 (
                                 V6OLG2RU8PQ5FU45JNU02J5HRUTIRPM9
                                 A RRSIG )
IF6JDFQA8OILAS7OEKML1RPDCV2GMHN5.it-sec.ovh. 1200 IN RRSIG NSEC3 8 3 1200 (
                                 20151109145641 20151010145608 60261 it-sec.ovh.
                                 nXELoGANXMzLnY4n2sLzEs2n9MkCD1mS/M7SV1G+6yPP
                                 Q/h0b4AZwAERa7znbEVkuCCw0aAKr/w9GeArhxD/wALF
                                 kYILbWpm9XSc1D4NXhdC9UA44+kAKRo90jimutUSDoAz
                                 zIKP/q0MTToSsAJzqcJ9LMjfr9dJz2vsv/9qw84= )
…
```

Ein Resolver kann die Prüfung folgendermaßen durchführen: Zuerst bringt der Resolver die NSEC3-Parameter in Erfahrung und verifiziert dessen Signatur:

```
dig @localhost it-sec.ovh nsec3param +dnssec +multiline
…
;; ANSWER SECTION:
it-sec.ovh.             0 IN NSEC3PARAM 1 0 100 578C30CC6333ADF7EA61A5DA07142D20
it-sec.ovh.             0 IN RRSIG NSEC3PARAM 8 2 0 (
                                 20151109155512 20151010145608 60261 it-sec.ovh.
                                 DLa2Z//PMommENkTZNVIFOGU1Vo4QVnpHP9hutWUw+st
                                 jwGxfdQcraLEmHumPiS9OieLjbjIyYv0EleFvE8ysDzr
                                 zhYV9AcI6MSx6XQqWNcmxoid+dJNYgyvoToCCF/VY34X
                                 dxY61PVTmJktzKvPWVOGWSScnvzEvlRJRXqWE04= )
…
```

Anschließend ermittelt der Resolver den Hash des angefragten FQDN, im obigen Beispiel also `test.it-sec.ovh.` Hierbei werden die NSEC3-Parameter (Salt und Anzahl der Iterationen) benötigt. Um dies auf der Konsole manuell zu prüfen, können abermals die bereits zuvor im Zuge des Zone-Walkings erwähnten LDNS-Utils installiert und herangezogen werden:

```
root@Sec-NS2:~#
ldns-nsec3-hash -a 1 -t 100 -s 578C30CC6333ADF7EA61A5DA07142D20 test.it-sec.ovh.
oeapkduouj10ukpio42bv4qst7goj5sn.
```

Der Resolver kann nun also verifizieren, dass der Hash `oeapkduouj10ukpio42bv4qst7goj5sn.` zwischen den beiden Hashes beginnend mit `I` und `V` des NSEC3 Records liegt - siehe zuvor:

```
IF6JDFQA8OILAS7OEKML1RPDCV2GMHN5.it-sec.ovh. 1200 IN NSEC3 1 0 100
                                              578C30CC6333ADF7EA61A5DA07142D20 (
                                 V6OLG2RU8PQ5FU45JNU02J5HRUTIRPM9
                                 A RRSIG )
```

Nicht nur der Resolver/Client muss hierbei eine aufwändige iterative Hash-Berechnung durchführen, auch der Nameserver selbst musst – um den zum angefragten Eintrag `test.it-sec.ovh` gehörigen Hash zu ermitteln den gleichen Aufwand erledigen.

Das hier dargestellte Beispiel spiegelt nur einen Bruchteil der Komplexität von NSEC3 wieder. Als Stichwort sei hier noch der Nachweis der Nicht-Existenz von DNS-Wildcards genannt, mehr Details sind im Bedarfsfall z.B. dem Whitepaper [SIDN-NSEC3] der niederländischen Domain-Registry zu entnehmen.

Der Vollständigkeit halber sei erwähnt, dass auch eine Rückkehr vom NSEC3 zum herkömmlichen NSEC Verfahren mittels NameServer Control-Utility möglich ist:

```
root@Sec-NS2:/etc/named/domains# rndc signing -nsec3param none it-sec.ovh
request queued
```

Ein Master Research Project der Universität Amsterdam hat im Frühjahr 2014 ergeben, dass etwa 58% der DNSSec nutzenden untersuchten Domains NSEC3 verwenden, wohingegen 42% weiterhin NSEC einsetzen [SNE-DNSSec1, Kapitel 5.4.2].

2.7. DNSSec Konfiguration beim Domain-Registrar

Um nun die Hinterlegung des RSA 2048bit Key-Signing-Key beim Registrar durchführen zu können, muss dieser aus den erzeugten Key-Files ermittelt werden.

In den Key-Files sind alle nötigen Daten unmittelbar erkennbar.

Das Key-Signing-Key File beinhaltet den String „`key-signing key`“, gefolgt von der Key-ID und dem Base64-kodierten Public-Key.

```
root@Sec-NS2:/etc/named/domains/keys# cat Kit-sec.ovh.+008+50137.key
; This is a key-signing key, keyid 50137, for it-sec.ovh.
; Created: 20150913210741 (Sun Sep 13 23:07:41 2015)
; Publish: 20150913210741 (Sun Sep 13 23:07:41 2015)
; Activate: 20150913210741 (Sun Sep 13 23:07:41 2015)
it-sec.ovh. IN DNSKEY 257 3 8
AwEAAd8F8FNYQQFl0pjZTclbUzSzWpgPt3kgHDXWrU1OXSk9++ykoWRW
WX0qHbPCiFklDU1C8U94/wxhiYzafyGMYm/Tf4D/tU9WoWo3q+bRnHeB
P2Jg9AgzPN8E7Q0ZZztbGnpKajlBvWjjSU2Qfd/wWvmHS/3lzb1fQPwN
L5HX0SsDB4wOEqC0Q18RNGHjmxLrOoIYi0z7QZNtp/EOMJ2ANHmF4SjJ
ZV2eQgMroN8KGdxaAlGivdknZFiT99RUj+PzXm7vINEpABxrBv5V8qG5
aUWQuwHIhXW7EvRp1O3iHCm/xMFflmMXNNMJtVMe4PkqyTbBPlezxDys
u6mQcg1pzpU=
```

Sollte der Registrar nicht den Base64-kodierten Public-Key sondern die Angabe des Delegation Signer (DS) resource record (RR) verlangen, so kann dieser wie folgt ermittelt werden:

```
root@Sec-NS2:/etc/named/domains/keys#
dnssec-dsfromkey Kit-sec.ovh.+008+50137.key
it-sec.ovh. IN DS 50137 8 1 A755CAFC6B341759BB369B7C36B3661AF2D4F221
it-sec.ovh. IN DS 50137 8 2 04554A4A3E73FEFD8676B35BC0497639B739B2E3814D2EF6362A6EA300BDB0F9
```

Die Ziffer `8` steht hierbei für den Algorithmus RSA und `2` steht für `SHA256`, `1` steht für `SHA1`.

Die oben blau hervorgehobenen Angaben werden wie Abbildung 6 und Abbildung 7 zeigen zur sicheren Delegation in Form eines Resource-Record vom Domain-Registrar in der übergeordneten Zone (im Beispiel it-sec.ovh somit in der Zone der Top-Level-Domain ovh.) eingetragen. Als Kennung ist hierbei die KeyID aus dem Key-File zu verwenden. Der Algorithmus RSA/SHA-256 entspricht der Ziffer 8 (auch nachzuschlagen auf der WebSite der IANA[15] bzw unter [IANA-DNSSec]) und 257 bedeutet, dass es sich hierbei um einen Resource-Record vom Typ „Key Signing Key“ handelt [RFC-4034] [ISC-DNSSec, Kapitel 4.3.1].

[15] http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml

2.7.1. Praxisbeispiel: Eintragung des KSK beim Registrar OVH

Nachfolgend sind Screenshots vom Kunden-Interface des weltweit agierenden französischen Domain-Registrars OVH.de abgebildet. OVH hat im Jahr 2014 ein „neues Kundeninterface“ mit modernerer Optik in Betrieb genommen, mit diesem besteht jedoch mit Stand 04.10.2015 immer noch keine Möglichkeit DNSSec Einträge vorzunehmen. Hierzu muss in der Menüleiste (links oben) der Link Altes Kunden-Interface angewählt werden, welches sich dann wie folgt präsentiert:

Abbildung 5: OVH.de Domain-Registrar: "altes Kunden-Interface"

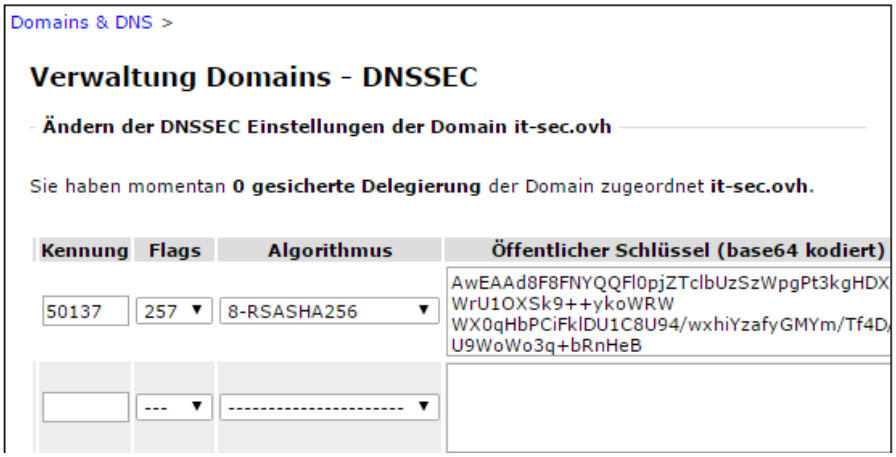


Abbildung 6: OVH.de Web-Interface: Hinterlegung der Schlüssel für DNSSec

Abbildung 7: OVH.de Web-Interface: Hinterlegte DNSSec Schlüssel

2.7.2. Praxisbeispiel: Eintragung des KSK beim Registrar GoDaddy

Ein weiterer sehr populärer, weltweit agierender Domain-Registrar ist GoDaddy.com, welcher ebenfalls DNSSec Funktionalität ermöglicht. Im Kundenmenü findet sich unter Meine Domains -> Domain auswählen -> Verwalten die Möglichkeit „DS-Datensätze“ (Delegation of Signing) zu verwalten. Das GoDaddy-WebInterface verlangt hier die Eingabe jener Daten, die zuvor mittels `dnssec-dsfromkey` ermittelt wurden.

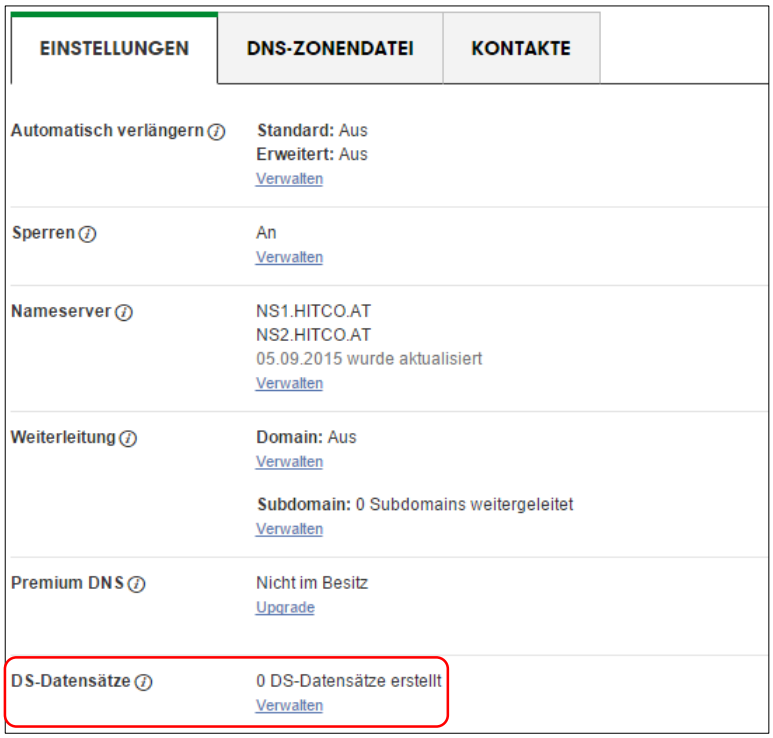


Abbildung 8. GoDaddy.com Web-Interface, Domain-Konfiguration

```
root@Sec-NS2:/etc/named/domains/keys#
dnssec-dsfromkey -a SHA256 Kit-sec.ovh.+008+50137.key
it-sec.ovh. IN DS 50137 8 2
04554A4A3E73FEFD8676B35BC0497639B739B2E3814D2EF6362A6EA300BDB0F9
```

Die Ziffer `8` steht hierbei für den Algorithmus RSA und `2` steht für `SHA256`, `1` würde für `SHA1` stehen.

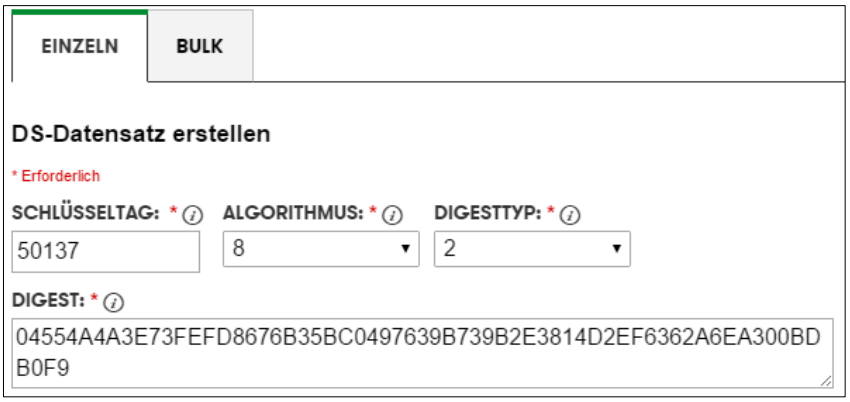


Abbildung 9: GoDaddy.com Web-Interface, Einrichtung DNSSec

2.8. Update der Zonen-Einträge

Ein Update der Zonen Einträge erfolgt immer am Master-Server, entweder mittels des hierfür bereitgestellten Commandline-Tools `nsupdate` oder durch Modifikation der Zonen-Datei. Bei Verwendung von `nsupdate` sollten Updates ebenso wie die Zone-Transfers zum Slave Server mittels Transaction-Signatures (TSIG) abgesichert werden, die Vorgangsweise ist sehr ähnlich wie bereits in Abschnitt 2.4.3 erläutert (siehe [ISC-BIND9, Kapitel 4.2, 7.3 und B.16]).

Manuelle Updates der Zonen-Einträge erfordert stets eine Erhöhung der Versionsnummer:

```
root@Sec-NS2:/etc/named/domains# vi zone_it-sec.ovh
…
@ IN SOA ns1.it-sec.ovh. dns-admin.hitco.at. (
          2015100702 ; Serial, z.B. YYYYMMDDrr
…
demo            IN      A       10.11.12.13
```

Die Änderung wird durch einen Reload des Bind-Servers unmittelbar wirksam:

```
root@Sec-NS2:/etc/named/domains# systemctl reload bind9
```

Sichtung des Logfiles `/var/log/syslog`

```
… systemd[1]: Reloading BIND Domain Name Server.
…
… named[22656]: zone it-sec.ovh/IN (unsigned): loaded serial 2015100702
… named[22656]: zone it-sec.ovh/IN (signed): serial 2015100702 (unsigned 2015100702)
… named[22656]: zone it-sec.ovh/IN (signed): sending notifies (serial 2015100702)
… named[22656]: zone it-sec.ovh/IN (signed): reconfiguring zone keys
… named[22656]: zone it-sec.ovh/IN (signed): next key event: 07-Oct-2015 20:41:46.589
…
… named[22656]: all zones loaded
… named[22656]: running
```

Am sekundären Nameserver NS1 ist hierbei im `/var/log/messages` das Update ebenfalls protokolliert: `tail -f /var/log/messages`

```
… named[11707]: client 104.46.42.66#1041/key ns1.it-sec.ovh_ns2.it-sec.ovh: received notify
for zone 'it-sec.ovh': TSIG 'ns1.it-sec.ovh_ns2.it-sec.ovh'
… named[11707]: zone it-sec.ovh/IN: Transfer started.
… named[11707]: transfer of 'it-sec.ovh/IN' from 104.46.42.66#53: connected using
100.114.178.187#36976
… named[11707]: zone it-sec.ovh/IN: transferred serial 2015100702: TSIG 'ns1.it-
sec.ovh_ns2.it-sec.ovh'
… named[11707]: transfer of 'it-sec.ovh/IN' from 104.46.42.66#53: Transfer completed: 1
messages, 22 records, 2263 bytes, 0.062 secs (36500 bytes/sec)
… named[11707]: zone it-sec.ovh/IN: sending notifies (serial 2015100702)
```

Durch Signatur-Operationen oder Verwendung des NameServer Control-Utility `rndc` kann die aktuell in Betrieb befindliche Seriennummer von jener im manuell verwalteten Zonen-File abweichen.

Es empfiehlt sich daher vor Änderung der Seriennummer im manuell verwalteten Zone-File diese im aktuell verwendeten, signierten Zone-File zu prüfen:

```
root@Sec-NS2:/etc/named/domains# grep Serial zone_it-sec.ovh
          2015101045 ; Serial, z.B. YYYYMMDDrr

root@Sec-NS2:/etc/named/domains#
named-checkzone -f raw it-sec.ovh zone_it-sec.ovh.signed
zone it-sec.ovh/IN: loaded serial 2015101055 (DNSSEC signed)
OK
```

2.9. Verwendung des DNSSec fähigen Resolvers

In weiterer Folge wird am Debian-Server auch der Mailserver installiert werden. Damit DANE und DNSSec genutzt werden können, ist ein vertrauenswürdiger DNSSec fähiger Resolver nötig. Im Idealfall läuft dieser auf der gleichen Maschine oder zumindest in gleichen (vertrauenswürdigen) Netz.

Die hier beschriebene Bind 9 Konfiguration stellt einen DNSSec fähigen Resolver bereit, dieser soll nun lokal (`127.0.0.1`) genutzt werden.

```
root@Sec-NS2:~# vi /etc/resolv.conf
…
nameserver 127.0.0.1
nameserver 40.74.62.0
nameserver 8.8.8.8
nameserver 8.8.4.4
…
```

Falls die Nameserver (wie z.B. in der Windows-Azure-Cloud üblich) am Server mittels DHCP gesetzt werden, ist folgende Vorgangsweise möglich (beachte die umgekehrte Reihenfolge der Einträge):

```
root@Sec-NS2: ~ # vi /etc/dhcp/dhclient.conf
…
prepend domain-name-servers 40.74.62.0;
prepend domain-name-servers 127.0.0.1;
…
```

Die in der Azure-Infrastruktur angebotenen Nameserver unterstützen mit Stand Oktober 2015 immer noch kein DNSSec und es gibt auch keine konkreten Aussagen wann dies soweit sein wird.[16] Es sollten daher – falls der lokale DNSSec fähige Resolver ausfällt – eventuell noch weitere Ersatz-Resolver mit DNSSec-Funktionalität konfiguriert werden – dafür bietet sich der vorbereitete Secondary-Nameserver, welcher für diesen Host als weiterer Resolver dienen kann. Das Setting „`allow-recursion { localhost; 104.46.42.66; };`“ ist hierfür am anderen Server um die IP-Adresse des zugreifenden Hosts zu erweitern.).

Die frei nutzbaren Google-Public-DNS-Resolver[17] `8.8.8.8` und `8.8.4.4` sind grundsätzlich auch geeignet (DNSSec-fähig), allerdings sollte bedacht werden, dass die Verwendung der Google-Public-DNS-Resolver eine Abhängigkeit zu einem externen Unternehmen schafft, welchem man auch kritisch gegenüber stehen kann. Außerdem sollte bedacht werden, dass die Google-Resolver von DNS-basierten Blocklists wie sie bei der SPAM-Bekämpfung gerne zum Einsatz kommen blockiert werden (könnten)[18].

---

16 Siehe auch https://azure.microsoft.com/en-us/documentation/articles/dns-getstarted-create-dnszone/

17 Siehe https://developers.google.com/speed/public-dns/

18 https://www.dnswl.org/?p=152

2.10. Prüfung der Nameserver

2.10.1. Prüfung der Zonen-Konfiguration

Durch Verwendung des Inline-Signing Modus ist eine Sichtkontrolle der vollständigen Zonen-Files mittels Text-Editor nicht so einfach möglich. Die manuell angelegte Zonen-Datei `/etc/named/domains/zone_it-sec.ovh` enthält keine DNSSec Einträge. Das vollständige Zonen-File wird von Bind selbständig verwaltet und erzeugt und liegt binär kodiert unter `/etc/named/domains/zone_it-sec.ovh.signed` vor.

Mittels des Tools `named-checkzone` können jedoch auch die Inhalte dieser Binärdatei (Raw-Format) geprüft werden:

```
root@Sec-NS2:/etc/named/domains#
named-checkzone -D -i full -f raw it-sec.ovh zone_it-sec.ovh.signed >/tmp/zone.txt
zone it-sec.ovh/IN: loaded serial 2015101049 (DNSSEC signed)
OK
```

Ein Blick in das mit `named-checkzone` erzeugte File `/tmp/zone.txt` zeigt nun auch sämtlich vorhandenen `RRSIG` und `NSEC3` Einträge (siehe Abbildung 10).

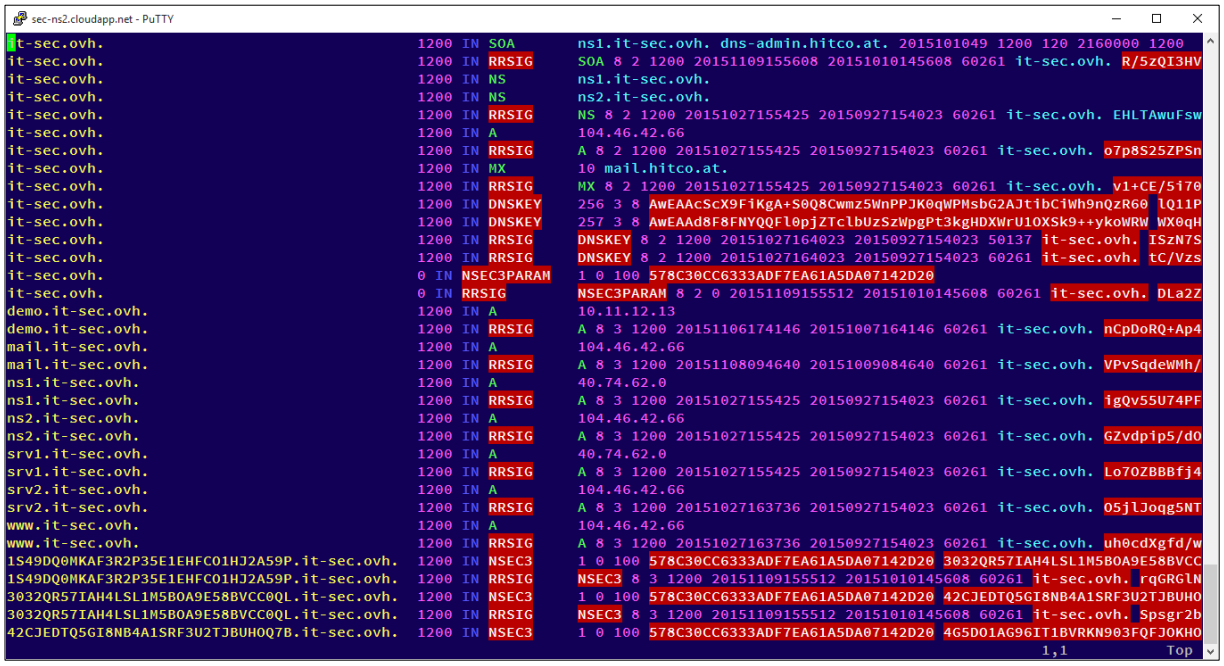


Abbildung 10: Signiertes Zone-File, in ein lesbares Textfile konvertiert

Das im Kapitel 2nd6th bereits angesprochene Paket LDNS-Utils enthält zahlreiche nützliche Tools der niederländischen NLnet Labs[19], unter anderem das Tool `ldns-verify-zone`. Mit diesem lässt sich das zuvor ins Textformat konvertierte Zonen-File einer intensiven (DNSSec-) Prüfung unterziehen, hierzu muss auch der ZSK und KSK angegeben werden:

```
root@Sec-NS2:~# aptitude install ldnsutils

root@Sec-NS2:/etc/named/domains# ldns-verify-zone /tmp/zone.txt -S –V 4 ⮐
                -k keys/Kit-sec.ovh.+008+50137.key -k keys/Kit-sec.ovh.+008+60261.key
Zone is verified and complete
```

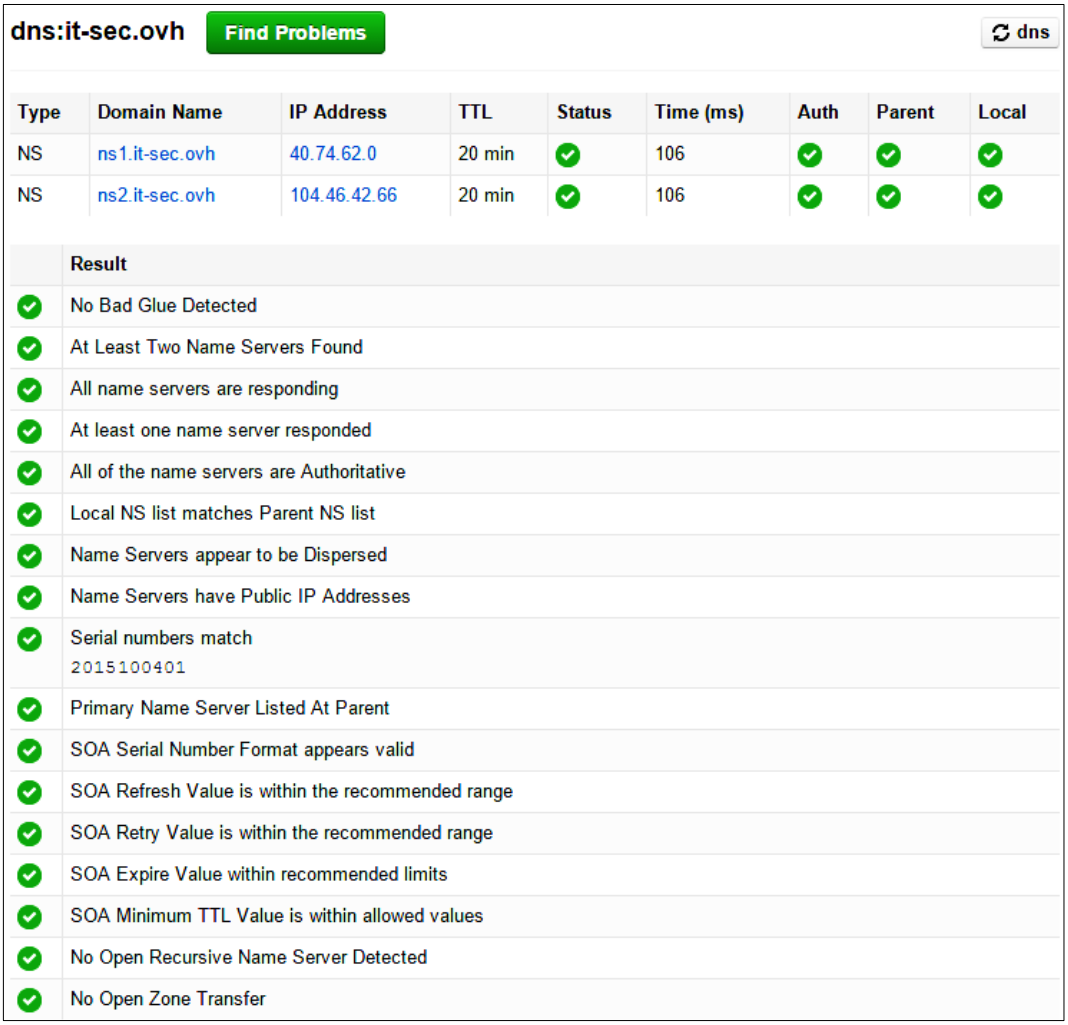
19 http://nlnetlabs.nl/projects/ldns/

Zum Vergleich: Ein (manuell zu Demonstrationszwecken) um einige Einträge gekürztes Zone-File wird wie folgt beanstandet:

```
root@Sec-NS2:/etc/named/domains# ldns-verify-zone /tmp/zone.txt -S -V 4 ↵
                -k keys/Kit-sec.ovh.+008+50137.key -k keys/Kit-sec.ovh.+008+60261.key
Error: there is no NSEC(3) for it-sec.ovh.
Error: no signatures for demo.it-sec.ovh.  A
Error: The NSEC3 record for points to the wrong next hashed owner name
should point to mail.it-sec.ovh., whose hashed name is 4g5do1ag96it1bvrkn903fqfjokhob83.
There were errors in the zone
```

2.10.2. Allgemeine DNS-Tests

Für allgemeine (nicht DNSSec spezifische) Tests bieten sich die populären Online-Dienste von http://mxtoolbox.com/NetworkTools.aspx an. Einer der dort angebotenen Tests widmet sich dem Thema DNS-Check: http://mxtoolbox.com/DNSCheck.aspx



Type	Domain Name	IP Address	TTL	Status	Time (ms)	Auth	Parent	Local
NS	ns1.it-sec.ovh	40.74.62.0	20 min	✓	106	✓	✓	✓
NS	ns2.it-sec.ovh	104.46.42.66	20 min	✓	106	✓	✓	✓

	Result
✓	No Bad Glue Detected
✓	At Least Two Name Servers Found
✓	All name servers are responding
✓	At least one name server responded
✓	All of the name servers are Authoritative
✓	Local NS list matches Parent NS list
✓	Name Servers appear to be Dispersed
✓	Name Servers have Public IP Addresses
✓	Serial numbers match 2015100401
✓	Primary Name Server Listed At Parent
✓	SOA Serial Number Format appears valid
✓	SOA Refresh Value is within the recommended range
✓	SOA Retry Value is within the recommended range
✓	SOA Expire Value within recommended limits
✓	SOA Minimum TTL Value is within allowed values
✓	No Open Recursive Name Server Detected
✓	No Open Zone Transfer

Abbildung 11: DNSCheck von mxtoolbox.com

Ähnliche Dienste bietet auch http://viewdns.info/ an. Mittels des Tools http://viewdns.info/dnsreport/ kann ein Domain-Zonen-Check inklusive „best practice“ Verbesserungsvorschlägen online abgerufen werden. Außerdem steht mittels http://viewdns.info/dnssec/ auch ein einfacher DNSSec-Domain-Check zur Verfügung.

2.10.3.DNS-Check inklusive DNSSec Prüfung

Eine sehr ausführliche Prüfung bietet http://dnscheck.iis.se/ an.

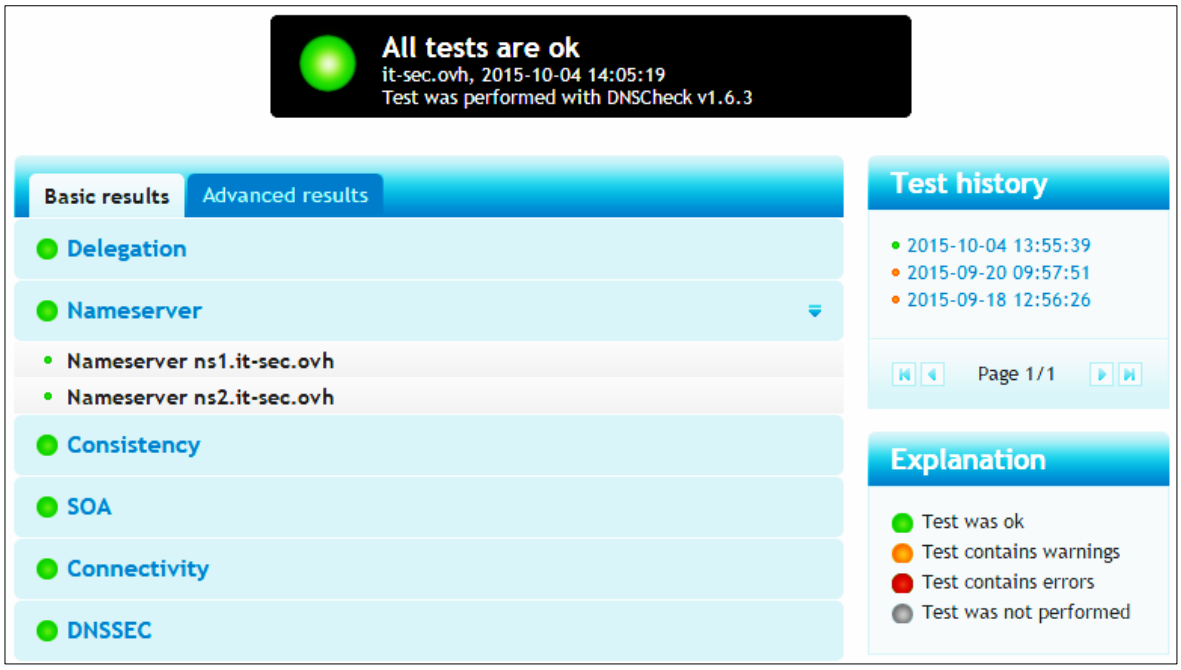


Abbildung 12: DNSCheck und DNSSec Check von dnscheck.iis.se

Es empfiehlt sich zusätzlich zu den Basic-Results die „Advanced results“ zu kontrollieren, diese fallen sehr ausführlich aus, sind daher aus Platzgründen hier nicht vollständig abgebildet:

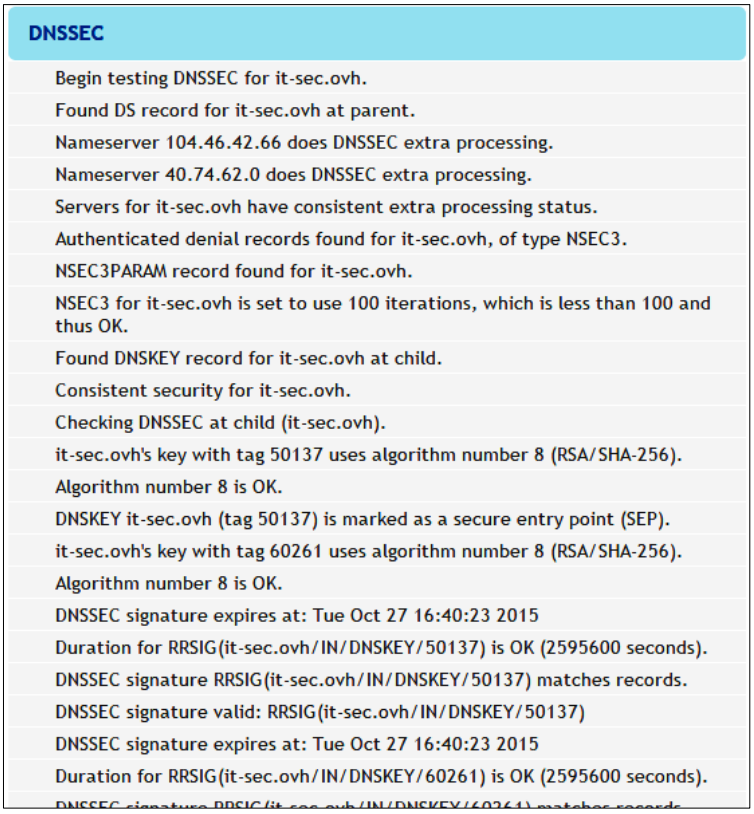


Abbildung 13: Ausschnitt aus: Advanced Results von dnscheck.iis.se

2.10.4.Test mittels des Verisign Labs DNSSec-Debuggers

http://dnssec-debugger.verisignlabs.com

.	✓ Found 3 DNSKEY records for . ✓ DS-19036/SHA-1 verifies DNSKEY-19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG-19036 and DNSKEY-19036/SEP verifies the DNSKEY RRset
ovh	✓ Found 2 DS records for ovh in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG-62530 and DNSKEY-62530 verifies the DS RRset ✓ Found 3 DNSKEY records for ovh ✓ DS-2660/SHA-256 verifies DNSKEY-2660/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG-2660 and DNSKEY-2660/SEP verifies the DNSKEY RRset
it-sec.ovh	✓ Found 1 DS records for it-sec.ovh in the ovh zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG-54175 and DNSKEY-54175 verifies the DS RRset ✓ Found 2 DNSKEY records for it-sec.ovh ✓ DS-50137/SHA-1 verifies DNSKEY-50137/SEP ✓ Found 2 RRSIGs over DNSKEY RRset ✓ RRSIG-50137 and DNSKEY-50137/SEP verifies the DNSKEY RRset ✓ it-sec.ovh A RR has value 104.46.42.66 ✓ Found 1 RRSIGs over A RRset ✓ RRSIG-60261 and DNSKEY-60261 verifies the A RRset

Detailansicht

Checking DS between ovh and it-sec.ovh

✓ Found 1 DS records for it-sec.ovh in the ovh zone

✓ Found 1 RRSIGs over DS RRset

```
it-sec.ovh. 172800 IN RRSIG DS 8 2 172800 ( 20151124193614 20150925193614 54175 ovh.
Qy9h5BmrbjlRnre6xHPcyCUne9LCU2X+JaBNXI/c+s2t/VYTex0EUX4ELf1Y84iJucpa3l7Tj0A
iTFEg4zyztEWbks/Q69QM/TUKmGdd0Fv5Ao9kVmF44RFIarmgBU/RaETYB77FfcaBO8E9wb2K8nd DuYWYXjucY9EERKBKwk= )
```

✓ RRSIG-54175 and DNSKEY-54175 verifies the DS RRset

DS-50137/SHA-1 is now in the chain-of-trust

```
it-sec.ovh. 172800 IN DS 50137 8 1 a755cafc6b341759bb369b7c36b3661af2d4f221
```

✓ Found 2 DNSKEY records for it-sec.ovh

```
it-sec.ovh. 1200 IN DNSKEY 256 3 8 ( AwEAAcScX9FiKgA+S0Q8Cwmz5WnPPJK0qWPMsbG2AJtibCiWh9nQzR60lQ11PGVaFpPSOsMWuFnu
5tPtwLlx1v2Ij/eIhAD2t+BCx+MVoWpbgaIf9OiMPah4HskTxJ1SgsNdqSWi6XUdXd/TtJYwFvox 9Hf2t8V/cvO8BmIjEe57UWk1 ) ; Key ID = 60261
it-sec.ovh. 1200 IN DNSKEY 257 3 8 ( AwEAAd8F8FNYQQFl0pjZTclbUzSzWpgPt3kgHDXWrU1OXSk9++ykoWRWWX0qHbPCiFklDU1C8U94
/wxhiYzafyGMYm/Tf4D/tU9WoWo3q+bRnHeBP2Jg9AgzPN8E7Q0ZZztbGnpKajlBvWjjSU2Qfd/w
WvmHS/3lzb1fQPwNL5HX0SsDB4wOEqC0Q18RNGHjmxLrOoIYi0z7QZNtp/EOMJ2ANHmF4SjJZV2e
QgMroN8KGdxaAlGivdknZFiT99RUj+PzXm7vINEpABxrBv5V8qG5aUWQuwHIhXW7EvRp1O3iHCm/ xMFflmMXNNMJtVMe4PkqyTbBPlezxDysu6mQcg1pzpU= ) ;
Key ID = 50137
```

DNSKEY-50137/SEP is now in the chain-of-trust

✓ DS-50137/SHA-1 verifies DNSKEY-50137/SEP

✓ Found 2 RRSIGs over DNSKEY RRset

```
it-sec.ovh. 1200 IN RRSIG DNSKEY 8 2 1200 ( 20151027164023 20150927154023 50137 it-sec.ovh.
ISzN7SfAE1r2fvioUrTEmpJ3Ip9N7gjZ0TFM/jZ3joEy1uhyMAjBVXseYZwr21py1AFUUqsFmGY9
21Gu6vHBaUNFAP2s8IDA+i368h6SBBOY67r4znUSNuVEUejhB/aHBc1HlZCPF3lPzAKFWK3byWz8
ixaxJdGYLQdYROviUWNSh8HWzsfCp++0feSQCHaBR2nI6IefnsoHdAj0TMk4fYydcrfuSoEpuPuK
b5R6VXvZ/+6xRHwjXXKDBDo6LO6RE3jAqkxiRIk4PICsv43rAEmHIiq3kwQ+KchCpD6fHPdF47ma dKsKHMhdYFyYkljdGX6txxaHLzoaH2+ss8/lIA== )
```

```
it-sec.ovh. 1200 IN RRSIG DNSKEY 8 2 1200 ( 20151027164023 20150927154023 60261 it-sec.ovh.
tC/Vzs8pJAiYOC88ALyN2cA0hiyGy3eiIzO05pPz+8aRXQV9RqnLpNF2bJUrlEmWRtGXkinGkrc9
TDGwlzu8y4vyNiOsT6gNnALLCfL3iq1PDCe7cfKOAbHVSXW5dQSq8k6yM0H8/ghIk9eYfuJKTQsh F42JsmMy4Xq48YQ6C5Y= )
```

✓ RRSIG-50137 and DNSKEY-50137/SEP verifies the DNSKEY RRset

RRSIG-60261 and DNSKEY-60261 verifies the DNSKEY RRset

DNSKEY-60261 is now in the chain-of-trust

it-sec.ovh is authoritative for it-sec.ovh

✓ it-sec.ovh A RR has value 104.46.42.66

✓ Found 1 RRSIGs over A RRset

```
it-sec.ovh. 1200 IN RRSIG A 8 2 1200 ( 20151027155425 20150927154023 60261 it-sec.ovh.
o7p8S25ZPSnFGFCbsjY/HMGBWQhCn3LRggmgxHghl8E318EoHlvbIKbGFUrSAxYrQAy8+OzMFU0A
OkbcYZItNZgVi9+hsDT8SrMTdxWBh4pvFgQ2wpmd+QRkWBqlg4qvHyqa6E1ScItWBXGb5VvjMsn+ RCWRo7WAgKiDh71feKc= )
```

✓ RRSIG-60261 and DNSKEY-60261 verifies the A RRset

Abbildung 14: Verisign Labs DNSSec-Debugger

2.10.5.Test und grafische Darstellung mittels DNSViz

http://dnsviz.net/

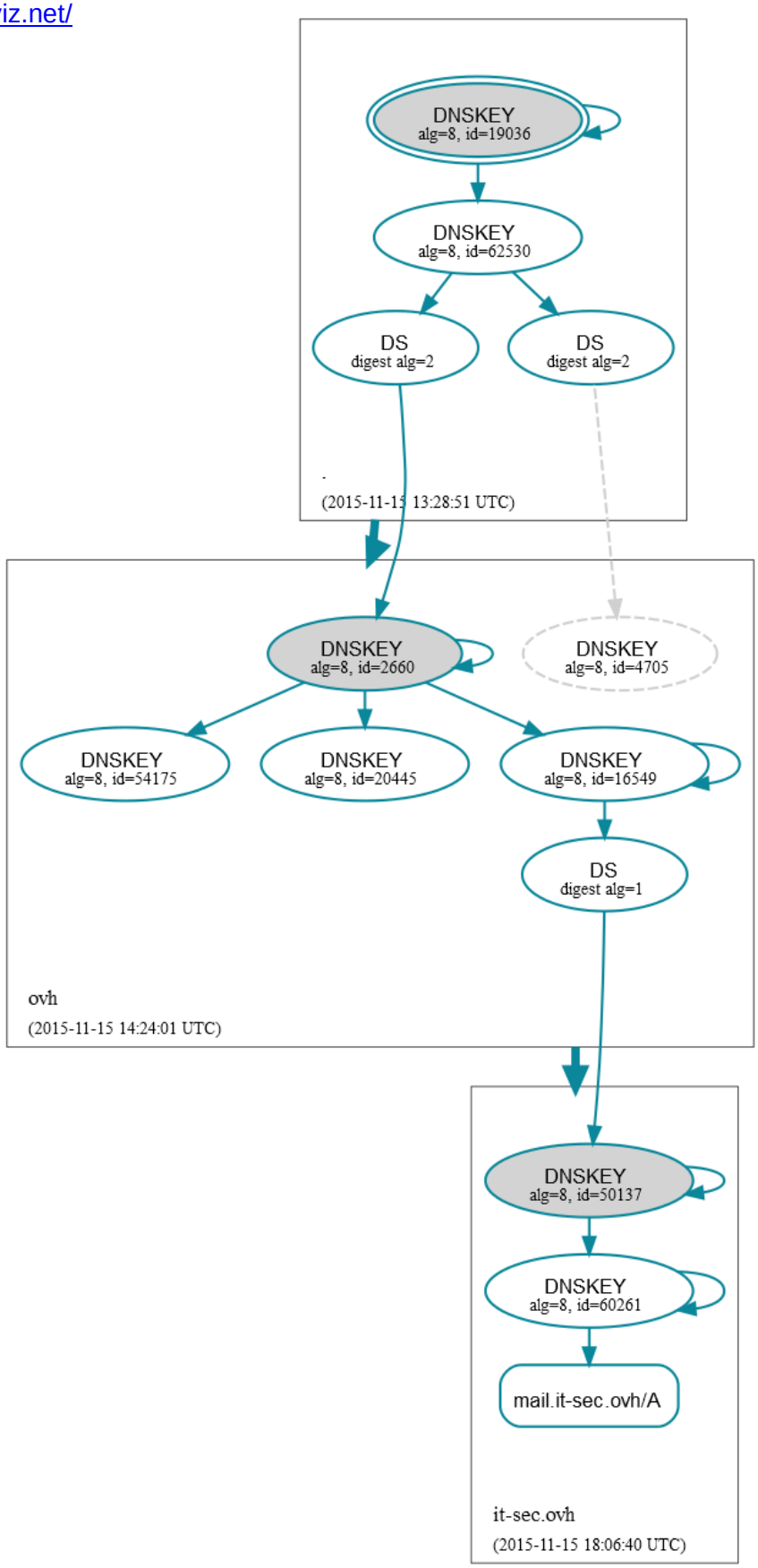


Abbildung 15: Grafische Darstellung der Delegation mittels dnsviz.net

2.11. Zone-ReSigning und KSK/ZSK-Schlüsselwechsel

Dem Thema Key-Rollover widmet sich [RFC-6781] und die praktische Umsetzung mittels BIND ist in [ISC-DNSSec, Kapitel 4.6 und 6.4] erläutert.

DNSSec erfordert nicht zwingend die Verwendung von zwei Schlüsseln (Key Signing Key – KSK und Zone Signing Key – ZSK) jedoch hat sich diese Vorgangsweise als „best practise“ etabliert.

Der Public-Teil des längeren (2048bit) KSK wird hierbei (siehe Kapitel 2.7) über den Domain-Registrar in der darüber liegenden Zone hinterlegt. Dieser signiert den DNSKEY Record des in der Zone angesiedelten Zone-Signing-Key. Der Zone-Signing-Key wiederum ist kürzer (lediglich 1024bit, das ergibt kürzere Signaturen und weniger Prüfungs-Aufwand) und signiert sämtliche Einträge der Zone.

Die Gültigkeitsdauer aller in der Zone vorhandenen RRSIG Einträge selbst (diese sind ZSK-signiert) sollte 2 Tage bis 1 Woche betragen, um im Falle einer Kompromittierung die Gültigkeitsdauer in einem überschaubaren Zeitrahmen zu halten [NIST-81.2, Kapitel 10.1]. Dies hat jedoch nichts mit der Gültigkeitsdauer des ZSK zu tun, sondern beschränkt lediglich die Gültigkeit der RRSIG-Einträge.

BIND führt die nötige Neu-Signatur der einzelnen Einträge im gewählten `inline-signing` Modus vollständig selbstständig und rechtzeitig vor Ablauf durch. Mittels der Zonen-Option `sig-validity-interval` könnte konfiguriert werden, wie lange Signaturen gültig sein sollen, mit einem optionalen zweiten Parameter kann außerdem angegeben werden, wie lange vor Ablauf die Signaturen neu generiert werden soll. Der verwendete Default ist 30 Tage Signatur-Gültigkeit und 7½ Tage Re-Signing-Intervall [ISC-BIND9, Kapitel 6.2.16.15].

Bei der Erneuerung des ZSK und KSK sind unbedingt die Empfehlungen in [RFC-6781] und [ISC-DNSSec, Kapitel 6.4.1] zu beachten, da sonst – und prominente Beispiele wie die `.se` Zone haben dies vorgemacht – sehr leicht ein Missgeschick zu einer temporären Nicht-Erreichbarkeit einer Zone führen kann.

Die aktuelle Empfehlung aus [ISC-DNSSec, Kapitel 4.6] lautet den ZSK jährlich zu wechseln und den KSK alle 5 Jahre zu tauschen. Eine harte Notwendigkeit hierfür besteht jedoch nicht, das heißt die Schlüssel laufen nicht aus.

Ein Austausch des ZSK findet in der eigenen Zone statt, ein KSK-Tausch hingegen erfordert auch eine Modifikation an der darüber liegenden Zone. Grundsätzlich ist hierbei zu beachten, dass entweder die rechtzeitige Vor-Verteilung der Schlüssel vor einer Nutzung nötig ist, oder die Zone eine Zeit lang parallel mit mehreren Keys signiert werden muss. Das gebräuchlichere Szenario scheint die Vor-Verteilung zu sein, der neue Public-Key muss mindestens eine TTL-Periode vor einem Key-Rollover als zweiter Key publiziert werden, eine solche Empfehlung ist auch [NIST-81.2, Kapitel 11.2] zu entnehmen. Das Entfernen alter Public-Keys darf frühestens eine TTL nach dem Wechsel des Schlüssels erfolgen.

Das in der Zonen-Definition in der `/etc/named/domains/ domains.conf` verwendete Setting `auto-dnssec maintain` sorgt für eine automatische Überwachung des `key-directory` und lädt neue Schlüssel für eine Zone automatisch [ISC-DNSSec, 6.4.2.1].

In den Key-Files findet sich hierzu Timing-Information, die Bind auswertet bzw. führt. Die Meta-Daten können mittels `dnssec-keygen` und `dnssec-settime` angepasst werden.

Nachfolgendes Beispiel demonstriert der Tausch des Zone-Signing-Keys:

Mittels `dnssec-keygen -S` wird ein „Successor“ (Nachfolger) für einen Key generiert, dies schlägt fehl, solange der Vorfahre nicht mit einem Ende-Datum versehen wurde:

```
root@Sec-NS2:/etc/named/domains/keys# dnssec-keygen -S Kit-sec.ovh.+008+60261.key
dnssec-keygen: fatal: Key it-sec.ovh/RSASHA256/60261 has no inactivation date.
        You must use dnssec-settime -I to set one before generating a successor.
```

Es wird daher der aktuelle ZSK zuerst mit einem Inaktivitätsdatum sowie einem Löschdatum ab dem er aus der Zone gelöscht werden soll versehen:

```
root@Sec-NS2:/etc/named/domains/keys#
dnssec-settime -I 20160101 -D 20160201 Kit-sec.ovh.+008+60261.key
./Kit-sec.ovh.+008+60261.key
./Kit-sec.ovh.+008+60261.private
```

Nun wird der Successor erstellt:

```
root@Sec-NS2:/etc/named/domains/keys# dnssec-keygen -S Kit-sec.ovh.+008+60261.key
Generating key pair.......................++++++
.............................................................++++++
Kit-sec.ovh.+008+23040
```

Der alte Schlüssel wurde somit mit einem Ablauf- und Lösch-Datum versehen, der neue Schlüssel erstellt und mit einem Zeitplan für die Publizierung und die Inbetriebnahme versehen:

```
root@Sec-NS2:/etc/named/domains/keys# cat Kit-sec.ovh.+008+60261.key
; This is a zone-signing key, keyid 60261, for it-sec.ovh.
; Created: 20150913210726 (Sun Sep 13 23:07:26 2015)
; Publish: 20150913210726 (Sun Sep 13 23:07:26 2015)
; Activate: 20150913210726 (Sun Sep 13 23:07:26 2015)
; Inactive: 20160101000000 (Fri Jan  1 01:00:00 2016)
; Delete: 20160201000000 (Mon Feb  1 01:00:00 2016)
it-sec.ovh. IN DNSKEY 256 3 8 AwEAAcScX9FiKgA+S0Q8Cwmz5WnPPJK0qWPMsbG2AJtibCiWh9nQzR60
lQ11PGVaFpPSOsMWuFnu5tPtwLlx1v2Ij/eIhAD2t+BCx+MVoWpbgaIf
9OiMPah4HskTxJ1SgsNdqSWi6XUdXd/TtJYwFvox9Hf2t8V/cvO8BmIj Ee57UWk1

root@Sec-NS2:/etc/named/domains/keys# cat Kit-sec.ovh.+008+23040.key
; This is a zone-signing key, keyid 23040, for it-sec.ovh.
; Created: 20151106161238 (Fri Nov  6 17:12:38 2015)
; Publish: 20151202000000 (Wed Dec  2 01:00:00 2015)
; Activate: 20160101000000 (Fri Jan  1 01:00:00 2016)
it-sec.ovh. IN DNSKEY 256 3 8 AwEAAcn4kzhhjVxHgZi4nFZuIuq4plb2IzJPQNuLnDgzcqu03ULkNdId
Sg8EaiFGEJgoQ2yJu8XcsqnhBntNsPMHsia3RpMKVWGF9ruSPZ1+pG/7
tp8CVMt3xhmBnri9jdm+1CGrBKP4zg8dwHbNmdPyIOkLwnLrQfJrTpLW Wrfl1zd1
```

Eine Erläuterung dieser Meta-Daten findet sich in [ISC-DNSSec, Kapitel 6.4.2]:

Metadata	Im Zone-File enthalten	Für Signatur verwendet	Zweck
Publish	Ja	Nein	Einführen eines neuen Schlüssels der bald aktiv wird
Activate	Ja	Ja	Aktivierungs-Datum für neuen Schlüssel
Revoke	Ja	Ja	Notifizierung, dass ein Schlüssel bald ungültig wird
Inactive	Ja	Nein	Nicht mehr aktiver Schlüssel
Delete	No	No	Löschung aus dem Zonen-File

3. DANE: Zertifikate + DNSSec + TLSA-Records

DNS-based Authentication of Named Entities (DANE) ist ein in [RFC-6698] allgemein und in [RFC-7672] nochmals speziell für SMTP spezifiziertes Verfahren zur Absicherung der SSL/TLS-Transportwegverschlüsselung. Es baut auf DNSSec auf und sorgt mittels TLSA-Records dafür, dass die verwendeten Zertifikate nicht unbemerkt ausgewechselt und/oder mittels Man-in-the-Middle oder Umlenkung des Datenverkehrs z.B. mittels DNS-Cache Poisoning auf unverschlüsseltes Klartext-SMTP downgegraded werden kann.

Da SMTP ursprünglich als Klartext-Protokoll spezifiziert wurde, und Transportsicherheit erst später mittels STARTTLS Funktionalität ergänzt wurde, ist ein Downgrade-Angriff relativ simpel durch eine Man-in-the-Middle Attacke zu bewerkstelligen. Hierbei muss lediglich die Aufforderung TLS zu nutzen (wird announced mittels `250-STARTTLS`) aus dem Klartext-Response des Datenstromes entfernt werden, der Sender wird daraufhin seine Mail unverschlüsselt anliefern (Abbildung 16).

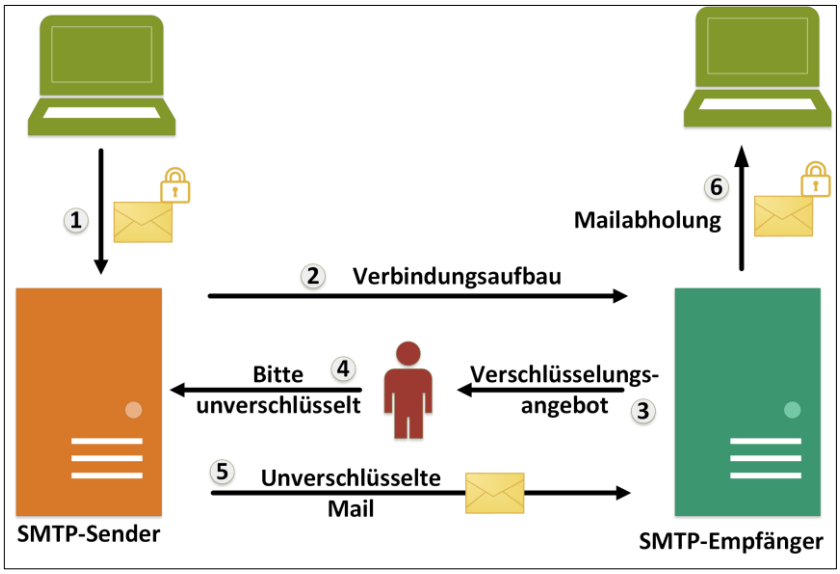


Abbildung 16: Man-in-the-Middle, Mailabgriff per Downgrade

Auch die Umlenkung des Datenstroms auf einen fremden Server ist denkbar (siehe hierzu Abbildung 17), bewerkstelligt kann dies z.B. mittels eines Angriffes auf das DNS-System werden. DNSSec verhindert solche Angriffe, jedoch nutzen zahlreiche Server und Resolver kein DNSSec und es ist nur die Minderheit der Domains bereits mit DNSSec abgesichert (siehe hierzu die Statistik in Kapitel 2.1).

Der Server auf den die Umlenkung durchgeführt wird, kann hierbei durchaus TLS unterstützen, für gewöhnlich authentifizieren Mailserver ihren Kommunikationspartner nämlich nicht (das Trusten von Zertifikaten erfordert im Falle eines nicht bereits bekannten Stammzertifikats eine Benutzer-Interaktion, auf einem Mailserver ist keine Zustimmung durch einen Benutzer praktikabel, weshalb Mailserver in der Regel allen Zertifikaten vertrauen – Motto: schlechte Verschlüsselung ist immer noch besser als Klartextkommunikation). Selbst wenn eine Prüfung des Zertifikates stattfindet, so ist ein Zertifikat aus einer beliebigen hinterlegten CA hierfür geeignet, die in Abschnitt 5.1.7 angeführten Beispiele zeigen jedoch, dass CAs leider nicht so vertrauenswürdig wie eigentlich nötig und erwartet agieren).

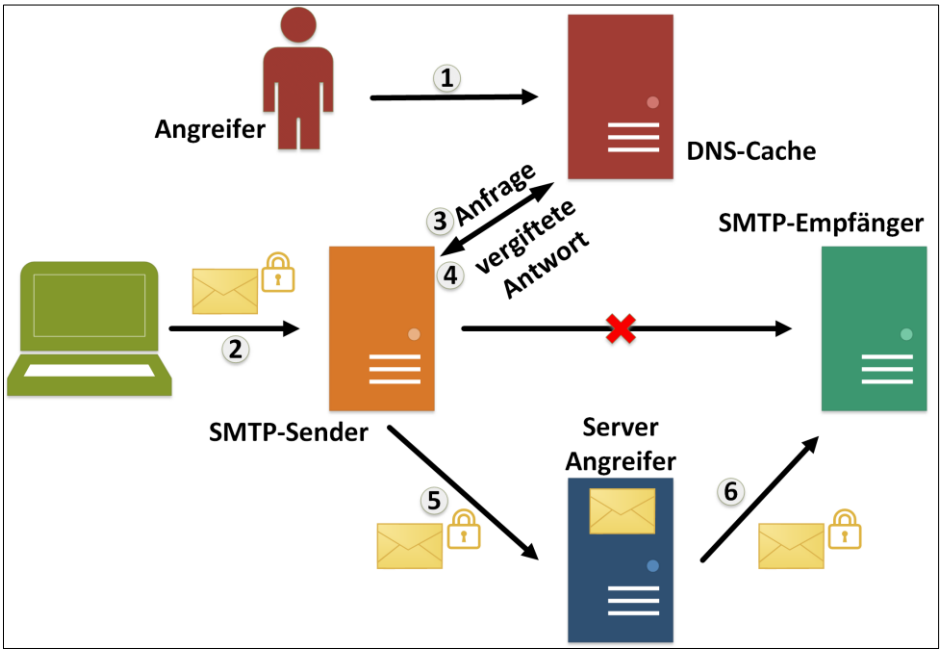


Abbildung 17: Umleitungsangriff mittels DNS-Cache-Poisoning - Quelle: [heise-DANE]

DANE löst dieses Problem der mangelnden Transportsicherheit und das Problem nicht vertrauenswürdiger CA's derart, dass mittels DNSSec die DNS-Antworten abgesichert (signiert) werden, und außerdem mittels eines TLSA-Records dem sendenden SMTP-Server mitgeteilt wird, dass ausschließlich verschlüsselt gesendet werden soll und hierbei nur einem Server welcher über ein Zertifikat mit definiertem Fingerprint verfügt vertraut werden darf (Abbildung 18).

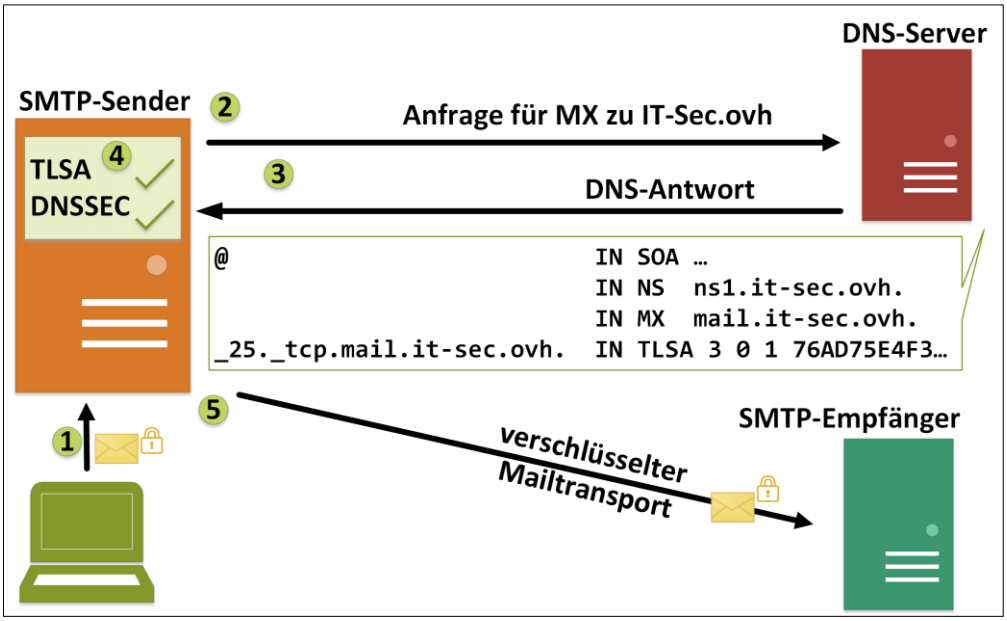


Abbildung 18: Ablauf - SMTP mit DANE/TLSA/DNSSEC

Auch das Problem des Klartext-Downgrades ist mit DANE gelöst, denn die Existenz des TLSA-Records zeigt an, dass dieser Server – selbst wenn ein Man-in-the-Middle das STARTTLS-Announcement aus dem Datenstrom entfernt – nicht mehr im Klartext angesprochen werden darf.

DANE ist keine Technologie, die auf E-Mail beschränkt ist. Sämtliche TLS-Verbindungen lassen sich mittels DANE und TLSA-Records absichern. Es lassen sich aber auch GPG-Keys oder SMIME-Zertifikate mittels DANE im DNS hinterlegen. Tatsächlich ist jedoch E-Mail eine der wenigen aktuell praxistauglichen Anwendungsmöglichkeiten, denn mangels Built-In-Browser-Unterstützung für DANE sowie aufgrund fehlender rekursiver DNSSec Resolver auf den Endgeräte-Plattformen verbreitet sich DANE abseits von E-Mail nur schleppend.

Gemäß [sys4-DANE] ist DANE auf etwa 1200 E-Mail Domains bereits im Einsatz, darunter findet man z.B.: posteo.de, mailbox.org, bund.de, Unitymedia (UPC-Deutschland), bayernd.de, SWITCH, IETF, …

Nachfolgend soll nicht nur die E-Mail Kommunikation, sondern auch der HTTPS-Zugriff auf das WebMail Interface mittels DANE abgesichert werden.

3.1. Zertifikate

Zertifikate werden in weiterer Folge benötigt werden für:

- Mailserver Postfix (Absicherung SMTP mit STARTTLS sowie des Submission-Ports)
- Mailserver Dovecot (Absicherung POP3S und IMAPS)
- Webserver Apache (Absicherung des HTTPS WebMail Interface)

Nachfolgend werden daher vorbereitend die Zertifikate für die nachfolgend zu konfigurierenden Services erzeugt und auch unmittelbar mit TLSA-Records versehen.

Grundsätzlich benötigt DANE nicht zwingend Zertifikate aus allgemein gebräuchlichen (d.h. in den Betriebssystemen und Browsern bereits vom Hersteller hinterlegte) Certificate Authorities, auch self-signed Zertifikate sind grundsätzlich einsetzbar. Da zum einen der Verbreitungsgrad von DANE noch gering ist, und andererseits auch für den Zugriff mittels Browser (WebMail) Zertifikate aus möglichst bereits getrusteten CA's benötigt werden, sei an dieser Stelle jedoch der Hinweis auf kostenfreie, in allen gängigen Betriebssystemen und Browsern getrustete CA-Anbieter gestattet:

- StartCom Ltd.: https://www.startssl.com/
- ISRG & Mozilla: https://letsencrypt.org/
- WoSign: https://www.wosign.com/english/freeSSL.htm

Die genannten Anbieter offerieren kostenfreie SSL/TLS Server-Zertifikate. Konkret genutzt wurde für die Umsetzung der Aufgabenstellung der israelischen Anbieters StartCom Ltd.

3.1.1. Schlüsselpaar generieren

Erstellung eines RSA-Schlüsselpaares mit einer Schlüssellänge von 4096 Bit:

```
root@Sec-NS2:/etc/ssl/private# openssl genrsa -out it-sec.ovh.key 4096
Generating RSA private key, 4096 bit long modulus
.....................................................................................
.............++
..............................++
e is 65537 (0x10001)

root@Sec-NS2:/etc/ssl/private# chmod 640 it-sec.ovh.key
```

Sichtkontrolle:

```
root@Sec-NS2:/etc/ssl/private# openssl rsa -in it-sec.ovh.key -text -noout
Private-Key: (4096 bit)
modulus: … [Daten entfernt] …
publicExponent: 65537 (0x10001)
privateExponent: … [Daten entfernt] …
prime1: … [Daten entfernt] …
prime2: … [Daten entfernt] …
…
```

Die Generierung des RSA-Keys ist nur einmalig erforderlich, solange dieser nicht kompromittiert oder die Schlüssellänge als zu gering erachtet wird, ist für die weitere aufzeit des Servers keine neue Schlüsselerstellung nötig.

3.1.2. Certificate Signing Request (CSR) erstellen

Zur Anforderung eines Zertifikates bei einer Certificate Authority wird ein Certificate Signing Request (CSR) benötigt.

Einen Certificate Signing Request unter Verwendung von SHA-256 erstellen:

```
root@Sec-NS2:/etc/ssl/private#
openssl req -out it-sec.ovh.csr -key it-sec.ovh.key -sha256 -new
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AT
State or Province Name (full name) [Some-State]:Vienna
Locality Name (eg, city) []:Vienna
Organization Name (eg, company) [Internet Widgits Pty Ltd]:it-sec.ovh
Organizational Unit Name (eg, section) []:it-sec.ovh
Common Name (e.g. server FQDN or YOUR name) []:it-sec.ovh
Email Address []:hostmaster@it-sec.ovh

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Anmerkung: Die Befüllung der Subject-Attribute ist im konkreten Fall nebensächlich, da die verwendete Certificate Authority (StartSSL.com) ohnehin alle Angaben entfernt bzw. ersetzt. Auch der Common Name wird von StartSSL überschrieben.

Sichtkontrolle:

```
root@Sec-NS2:/etc/ssl/private# openssl req -verify -in it-sec.ovh.csr -text -noout
verify OK
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=AT, ST=Vienna, L=Vienna, O=it-sec.ovh, OU=it-sec.ovh,
                 CN=it-sec.ovh/emailAddress=hostmaster@it-sec.ovh
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus: … [Daten entfernt] …
                Exponent: 65537 (0x10001)
        Attributes:
            challengePassword        :unable to print attribute
            unstructuredName         :unable to print attribute
    Signature Algorithm: sha256WithRSAEncryption
```

Auch die Generierung des CSR ist nur einmalig erforderlich, solange der Private-Key sich nicht ändert, ist keine neue CSR-Generierung nötig. Eine Verlängerung oder Neuausstellung des Zertifikates kann unter Verwendung des gleichen CSR erneut erfolgen.

3.1.3. Anforderung des Zertifikates bei der CA

Die Anforderung eines Zertifikates erfolgt bei den meisten CAs nach folgendem Prozess:

1: Erstellung eines Benutzerkontos, eventuell Nachweis/Validierung der Identität

2: Validierung des Domain-Namens, es wird geprüft ob administrative Kontrolle über die Domain besteht, z.B. durch Zusendung eines Autorisierungscodes per E-Mail an den im Whois hinterlegten Admin-C (siehe Abbildung 19).

3: Anforderung des Zertifikates mittels Certificate Request (siehe Abbildung 20).

4: Abruf des erstellten Zertifikates (Abbildung 21).

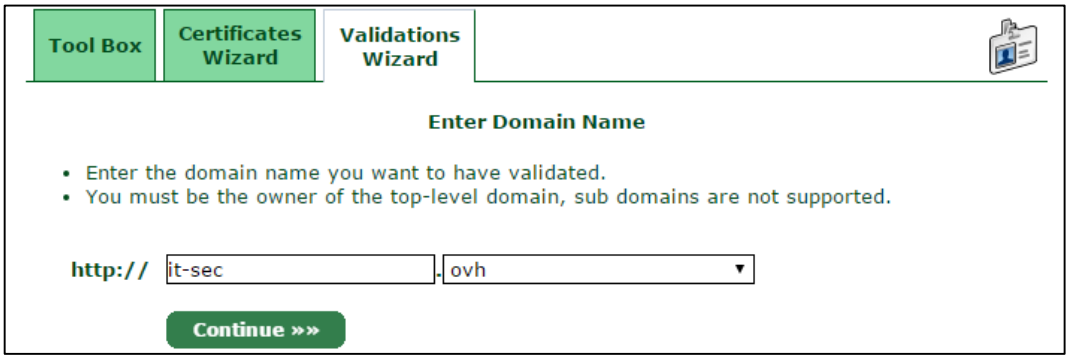


Abbildung 19: Domain-Validierung beim Anbieter https://www.startssl.com/

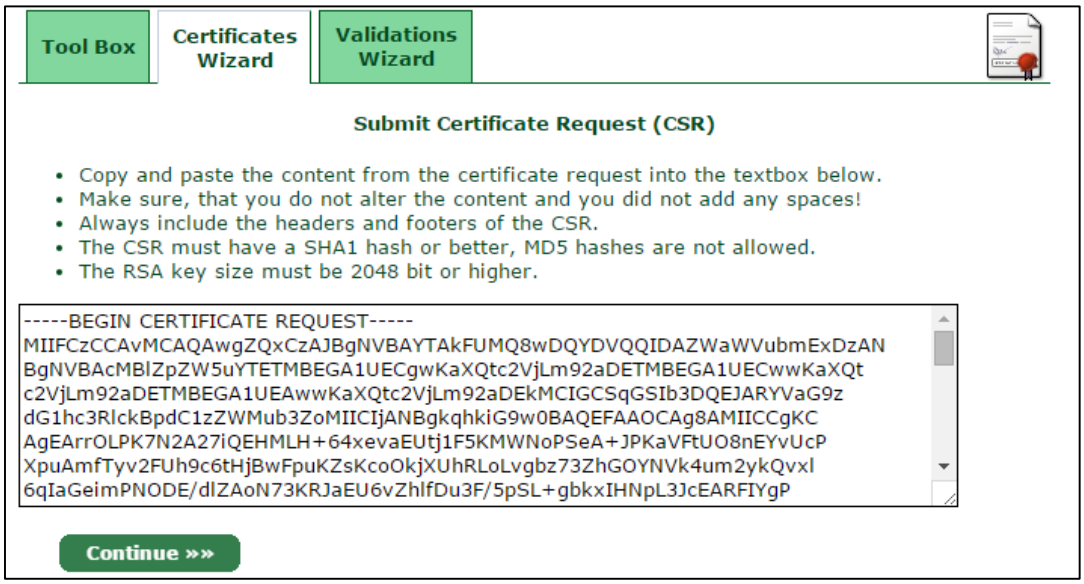


Abbildung 20: Zertifikats-Anforderung mittels CSR von https://www.startssl.com/

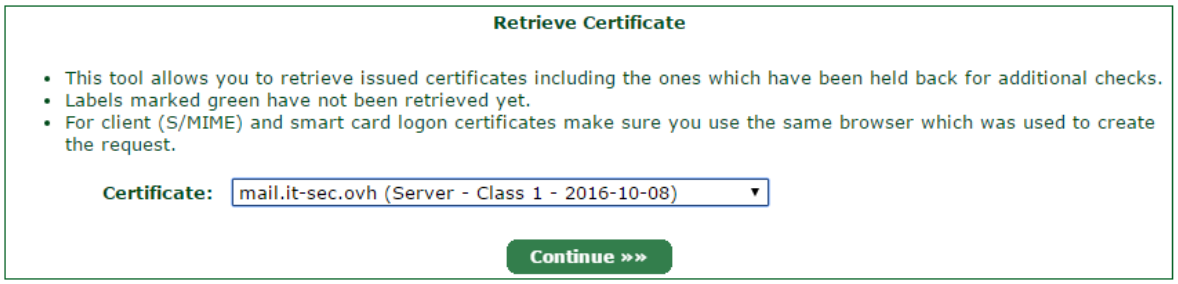


Abbildung 21: Abruf des Zertifikates vom Anbieter https://www.startssl.com/

Das abgerufene Zertifikat wird unter `/etc/ssl/certs/mail.it-sec.ovh.crt` abgelegt und einer Sichtkontrolle unterzogen:

```
root@Sec-NS2:/etc/ssl/certs# openssl x509 -in mail.it-sec.ovh.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1803383374255614 (0x6682ae96a95fe)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=IL, O=StartCom Ltd., OU=Secure Digital Certificate Signing,
                CN=StartCom Class 1 Primary Intermediate Server CA
        Validity
            Not Before: Oct  8 13:25:42 2015 GMT
            Not After : Oct  8 11:48:46 2016 GMT
        Subject: C=AT, CN=mail.it-sec.ovh/emailAddress=domain.admin@hitco.at
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus: …
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Key Identifier:
```

```
                68:0C:CB:39:43:D9:63:39:E5:9F:68:C3:6E:37:B8:1F:28:02:8B:8B
            X509v3 Authority Key Identifier:
                keyid:EB:42:34:D0:98:B0:AB:9F:F4:1B:6B:08:F7:CC:64:2E:EF:0E:2C:45

            X509v3 Subject Alternative Name:
                DNS:mail.it-sec.ovh, DNS:it-sec.ovh
            X509v3 Certificate Policies:
                Policy: 2.23.140.1.2.1
                Policy: 1.3.6.1.4.1.23223.1.2.3
                  CPS: http://www.startssl.com/policy.pdf
                  User Notice:
                    Organization: StartCom Certification Authority
                    Number: 1
                    Explicit Text: This certificate was issued according to the Class 1
Validation requirements of the StartCom CA policy, reliance only for the intended purpose
in compliance of the relying party obligations.

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.startssl.com/crt1-crl.crl

            Authority Information Access:
                OCSP - URI:http://ocsp.startssl.com/sub/class1/server/ca
                CA Issuers - URI:http://aia.startssl.com/certs/sub.class1.server.ca.crt

            X509v3 Issuer Alternative Name:
                URI:http://www.startssl.com/
    Signature Algorithm: sha256WithRSAEncryption
     …
```

3.1.4. Certificate-Chain (Intermediate-Zertifikate)

Um die erhaltenen SSL/TLS-Server-Zertifikate zu validieren ist ein Root-of-Trust gegen ein vertrauenswürdiges Stammzertifikat herzustellen (siehe Abbildung 22).

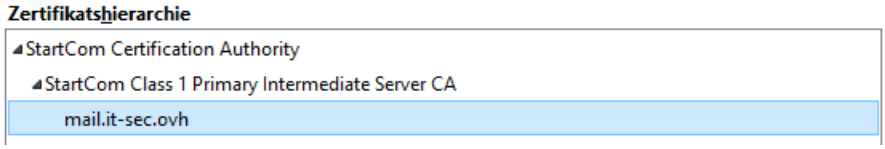


Abbildung 22: Zertifikatshierarchie der StartCom Class 1 Server-Zertifikate

Abruf des Intermediat-Zertifikates und Hinterlegung im vorgesehenen Verzeichnis:

```
root@Sec-NS2:/tmp#
wget https://www.startssl.com/certs/class1/sha2/pem/sub.class1.server.sha2.ca.pem
cp sub.class1.server.sha2.ca.pem /etc/ssl/certs/startssl.chain.class1.server.crt

root@Sec-NS2:/tmp# cd /etc/ssl/private
root@Sec-NS2:/etc/ssl/private# ln -s ../certs/startssl.chain.class1.server.crt .
```

Sichtkontrolle des Intermediate-Zertifikates:

```
root@Sec-NS2:/tmp# openssl x509 -in sub.class1.server.sha2.ca.pem -text -noout
…
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=IL, O=StartCom Ltd., OU=Secure Digital Certificate Signing,
                CN=StartCom Certification Authority
…
        Subject: C=IL, O=StartCom Ltd., OU=Secure Digital Certificate Signing,
                 CN=StartCom Class 1 Primary Intermediate Server CA
…
```

3.1.5. Diffie-Hellman Parameter

Für den sicheren Betrieb von TLS mit Diffie-Hellman wird die Erstellung eigener Diffie-Hellman-Parameter empfohlen [BetterCrypto, Kapitel 2.3.4]. Nachfolgend werden daher schwache und stärkere DH-Parameter generiert und in zwei Dateien abgelegt, diese werden in weiterer Folge für Postfix in Abschnitt 4.2.4 zur Anwendung kommen:

```
root@Sec-NS2:/etc/ssl/private# openssl dhparam -out dh_params_2048.pem 2048
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
…

root@Sec-NS2:/etc/ssl/private# openssl dhparam -out dh_params_1024.pem 1024
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
…
```

3.1.6. Zertifikatsdateien – Übersicht

Für die Konfiguration von Postfix wird ein Zertifikats-File nötig sein, welches das TLS-Server-Zertifikat sowie die Intermediate-Zertifikate in einem File beinhaltet:

```
root@Sec-NS2:/etc/ssl/certs#
cp mail.it-sec.ovh.crt mail.it-sec.ovh+chain.crt
cat startssl.chain.class1.server.crt >>mail.it-sec.ovh+chain.crt

root@Sec-NS2:/etc/ssl/certs# ls -la | egrep "it-sec.ovh|startssl"
-rw-r--r-- 1 root root   4660 Oct 25 16:07 mail.it-sec.ovh+chain.crt
-rw-r--r-- 1 root root   2569 Oct  9 11:43 mail.it-sec.ovh.crt
-rw-r--r-- 1 root root 201830 Oct 17  2014 startssl.ca-bundle.pem
-rw-r--r-- 1 root root   2091 Oct 25 11:26 startssl.chain.class1.server.crt
-rw-r--r-- 1 root root   2569 Oct  9 11:42 www.it-sec.ovh.crt
```

Um die Übersicht einfacher zu bewahren, werden die erstellten Zertifikate auch im Private-Verzeichnis verlinkt:

```
root@Sec-NS2:/etc/ssl/private# ln -s ../certs/mail.it-sec.ovh .
root@Sec-NS2:/etc/ssl/private# ln -s ../certs/mail.it-sec.ovh+chain.crt .
root@Sec-NS2:/etc/ssl/private# ln -s ../certs/www.it-sec.ovh.crt .
```

Die erhaltenen Zertifikate nochmals abschließend im Überblick – der Zugriff auf den Private-Key ist nur durch den Root-User gestattet. Die Zertifikate selbst sind – um die Übersicht zu bewahren – ebenfalls ins private-Verzeichnis verlinkt:

```
root@Sec-NS2:~# ls -la /etc/ssl/private/
drwx--x--- 2 root ssl-cert 4096 Oct 25 11:26 .
drwxr-xr-x 4 root root     4096 Jun 29 11:19 ..
-rw-r--r-- 1 root root      245 Oct 25 17:48 dh_params_1024.pem
-rw-r--r-- 1 root root      424 Oct 25 17:47 dh_params_2048.pem
-rw-r--r-- 1 root root     1825 Oct 14  2014 it-sec.ovh.csr
-rw-r----- 1 root root     3272 Oct 14  2014 it-sec.ovh.key
lrwxrwxrwx 1 root root       34 Oct 25 16:10 mail.it-sec.ovh+chain.crt ↩
                                          -> ../certs/mail.it-sec.ovh+chain.crt
lrwxrwxrwx 1 root root       28 Oct  9 11:43 mail.it-sec.ovh.crt ↩
                                          -> ../certs/mail.it-sec.ovh.crt
-rw-r----- 1 root ssl-cert 1708 Sep 15 21:11 ssl-cert-snakeoil.key
lrwxrwxrwx 1 root root       41 Oct 25 09:15 startssl.chain.class1.server.crt ↩
                                          -> ../certs/startssl.chain.class1.server.crt
lrwxrwxrwx 1 root root       27 Oct  9 11:43 www.it-sec.ovh.crt ↩
                                          -> ../certs/www.it-sec.ovh.crt
```

3.1.7. Alternative: Nutzung von Let's Encrypt

Das Projekt Lets's Encrypt[20] der Internet Security Research Group (ISRG) wird von Mozilla, Akamai, Cisco, IdenTrust und anderen bekannten Branchengrößen gesponsert. Es hat sich zum Ziel gesetzt kostenfreie Zertifikate für den Betrieb von (Web-)Servern für jedermann zur Verfügung zu stellen. Mit Stand Oktober 2015 ist das Service noch nicht allgemein verfügbar, ein geschlossener Beta-Test wird jedoch bereits durchgeführt.

Let's Encrypt stellt nur kurzzeitig (3 Monate) gültige automatisiert Domain-validierte Zertifikate aus. Der Validierungs- Zertifikats-Erstellungs- und Renew-Prozess sollte gemäß den zur Verfügung gestellten Anleitungen mittels eines bereitgestellten Python-Clients automatisiert werden.

Dieser Automatismus würde dafür sorgen, dass alle ca. 60 Tage ein neues Schlüsselpaar generiert und neue Zertifikate angefordert werden. Eine solche Vorgangsweise ist im Zusammenhang mit DANE / TLSA-Records oder dem noch später in Abschnitt 5.1.7 vorgestellten http Public Key Pinning (HPKP) jedoch eher wenig geeignet, denn ein neuer Private-Key bedingt auch einen neuen Public-Key, ein Pinning müsste daher mühsam erneuert werden, was alle 60 Tage einen zu hohen Aufwand darstellt, der auch in einer Automatisierung mit einigem Aufwand und Tests verbunden wäre.

Die Alternative besteht darin, nicht den Automatismus von Let's Encrypt zu benutzen, sondern das Zertifikat basierend auf einem selbst erstellten Certificate Signing Request zu erneuern. So bleibt – solange der gleiche Private-Key verwendet wird – auch der PublicKey im Zertifikat sowie der CSR identisch.

Let's Encrypt verlangt für deren Python-Client jedoch die Verwendung von DER-kodierten CSR's und benötigt zwingend, dass die Domain-Namen als Subject-Alternative-Names im CSR enthalten sind.

Die CSR-Erstellung funktioniert daher – ausgehend vom in Abschnitt 3.1.1 erstellen Schlüsselpaar – wie folgt:

Installation von git, herunterladen der Sourcen und Nutzung dieser gemäß der Anleitung unter https://letsencrypt.readthedocs.org/en/latest/using.html#installation-and-usage

```
root@Sec-NS2:~# aptitude install git
root@Sec-NS2:~# git clone https://github.com/letsencrypt/letsencrypt
root@Sec-NS2:~# cd letsencrypt
root@Sec-NS2:~/letsencrypt# ./letsencrypt-auto --help
```

Erstellung des DER-kodierten CSR auf Basis des vorhandenen Private-Keys `it-sec.ovh.key` und eines zuvor erstellen Konfigurationsfiles `it-sec.ovh.cnf`:

```
root@Sec-NS2:/etc/letsencrypt/my# openssl req -new -out www.it-sec.ovh.csr -outform der ↵
-config ./it-sec.ovh.cnf -key /etc/ssl/private/it-sec.ovh.key -batch
```

Das Konfigurationsfile `it-sec.ovh.cnf` enthält den Distinguished-Name, die Verwendungszwecke des gewünschten Zertifikates sowie die Subject-Alternative-Names.

[20] https://letsencrypt.org/

```
root@Sec-NS2:/etc/letsencrypt/my# vi it-sec.ovh.cnf
[ req ]
default_bits        = 4096
default_md          = sha256
distinguished_name  = req_distinguished_name
req_extensions      = req_ext

[ req_ext ]
basicConstraints      = CA:FALSE
keyUsage              = nonRepudiation, digitalSignature, keyEncipherment, dataEncipherment
extendedKeyUsage      = serverAuth, clientAuth
subjectAltName        = @alt_names

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = AT
stateOrProvinceName             = State or Province
stateOrProvinceName_default     = Vienna
localityName                    = Locality Name (eg, city)
localityName_default            = Wien
organizationName                = Organization Name
organizationName_default        = HITCo

emailAddress                    = E-Mail Address
emailAddress_default            = domain.admin@hitco.at

commonName                      = Common Name
commonName_default              = it-sec.ovh

[alt_names]
DNS.1   = it-sec.ovh
DNS.2   = www.it-sec.ovh
```

Der erhaltene Certificate Signing Request (CSR) sieht wie folgt aus:

```
root@Sec-NS2:/etc/letsencrypt/my# openssl req -in www.it-sec.ovh.csr -inform DER -text
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=AT, ST=Vienna, L=Wien, O=HITCo/emailAddress=domain.admin@hitco.at,
                 CN=it-sec.ovh
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus: …
                Exponent: 65537 (0x10001)
        Attributes:
        Requested Extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Subject Alternative Name:
                DNS:it-sec.ovh, DNS:www.it-sec.ovh
    Signature Algorithm: sha256WithRSAEncryption
…
```

Abruf des Zertifikats mittels Let's Encrypt Client:

```
root@Sec-NS2:~/letsencrypt#
./letsencrypt-auto certonly --csr /etc/letsencrypt/my/www.it-sec.ovh.csr
```

3.2. DANE

Basierend auf den im vorherigen Abschnitt erzeugten Zertifikaten sollen nun TLSA-Records für DANE erzeugt und im DNS hinterlegt und mit DNSSec abgesichert werden.

3.2.1. TLSA-Record erzeugen

DANE beruht auf der Hinterlegung eines SHA256-Fingerprints des verwendeten Zertifikates (oder des verwendeten Public-Keys) im DNSSec gesicherten DNS mittels eines TLSA-Records[21]. Im Gegensatz zum in Abschnitt 5.1.7 später noch vorgestellten Public-Key-Pinning wird hierbei nicht zwingend ein Hash des Public-Keys, sondern wahlweise entweder ein Hash des kompletten X509-Zertifikats (ein sogenannter Fingerprint) oder ein Hash des Public-Keys benötigt.

Die Variante mittels Public-Key ist insofern vorteilhaft, als sich der Hash bei einer Neuausstellung bzw. Verlängerung des Zertifikates nicht ändern, sofern der gleiche Certificate Signing Request (gleiches RSA-Schlüsselpaar) verwendet wird. Nachfolgend sind beide Varianten beschrieben, zur Anwendung wird die Public-Key-Variante gelangen.

Variante mit Selector: 0 = Hash des Zertifikates

```
root@Sec-NS2:/etc/ssl/private#
openssl x509 -in mail.it-sec.ovh.crt -outform DER | openssl sha256
(stdin)= 91ba252c1e6bd40767d646070c015d0b9edad8957f04880b3a6b1c63abfc4d43

openssl x509 -in www.it-sec.ovh.crt -outform DER | openssl sha256
(stdin)= 1597dc6d11cf983dee0d0c7094daff635dfc5ffa3fe9390d021a121a36e8e92e
```

Erläuterung: Es handelt sich hierbei um den SHA256-Fingerprint des Zertifikates, welcher auch direkt wie nachfolgend gezeigt mittels OpenSSL ermittelbar wäre, jedoch wird in weiterer Folge die oben dargestellte Schreibweise ohne Trennzeichen benötigt:

```
root@Sec-NS2:/etc/ssl/private#
openssl x509 -noout -fingerprint -sha256 -in www.it-sec.ovh.crt
SHA256 Fingerprint=15:97:DC:6D:11:CF:98:3D:EE:0D:0C:70:94:DA:FF:63:5D:FC: ↵
                   5F:FA:3F:E9:39:0D:02:1A:12:1A:36:E8:E9:2E
```

Variante mit Selector: 1 = Hash des PublicKeys (gleicher PubKey => gleicher Hash)

```
root@Sec-NS2:/etc/ssl/private# openssl x509 -in mail.it-sec.ovh.crt -pubkey -noout |  ↵
 openssl rsa -pubin -outform der | openssl dgst -sha256 -hex
writing RSA key
(stdin)= a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a6c6f0c16b5
```

Erläuterung: Public Key aus X509-Zertifikat extrahieren, den Base64-kodierten RSA-Public-Key in binäre DER-Kodierung umwandeln, davon nun einen HEX-kodierten SHA256-Hash erzeugen und ausgeben.

[21] TLSA ist der Name des hierfür verwendeten Resource-Records, dieser stellt gemäß [RFC-6698, Kapitel 1.2] kein Akronym dar, er ermöglicht die Nutzung des DANE Transport Layer Security Protokolls

Alternativ zur manuellen Erstellung des TLSA-Records kann hierfür auch ein WebService genutzt werden: https://www.huque.com/bin/gen_tlsa (siehe Abbildung 23).

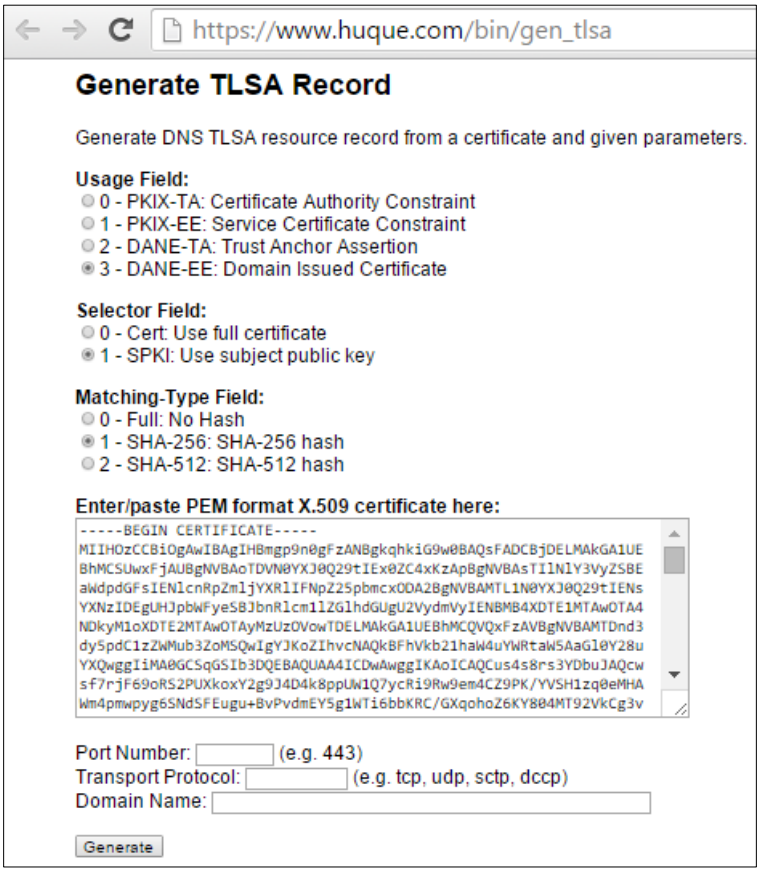


Abbildung 23: TLSA-Records erstellen mit https://www.huque.com/bin/gen_tlsa

Generate TLSA Record

Generate DNS TLSA resource record from a certificate and given parameters.

Certificate Information:

Serial : 66829f67d2017
Issuer : C=IL, O=StartCom Ltd., OU=Secure Digital Certificate Signing, CN=StartCom Class 1 Primary Intermediate Server CA
Subject: C=AT, CN=www.it-sec.ovh/emailAddress=domain.admin@hitco.at
Subject Alternative Name(s): DNS:www.it-sec.ovh, DNS:it-sec.ovh
Certificate Inception: 2015-10-09 08:49:23+00:00 UTC
Certificate Expiration: 2016-10-09 02:35:39+00:00 UTC

TLSA Parameters:

Usage: 3 - DANE-EE: Domain Issued Certificate
Selector: 1 - SPKI: Subject Public Key
Matching Type: 1 - SHA-256: SHA-256 Hash

Service Parameters:

Port: 25
Transport: tcp
Domain name: mail.it-sec.ovh.

Generated DNS TLSA Record:

```
_25._tcp.mail.it-sec.ovh. IN TLSA 3 1 1 a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a6c6f0c16b5
```

Der TLSA-Record ist wie folgt aufgebaut [RFC-6698, Kapitel 7]:

```
_25._tcp.mail   IN TLSA 3 1 1 a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a…
```

`_25._tcp.mail`	Port 25, Protokoll TCP, Host mail.it-sec.ovh
`IN TLSA`	Ein Record vom Typ TLSA
`3`	Usage = Domain-issued certificate, das heißt der Fingerprint wurde anhand des verwendeten Zertifikats ermittelt, das Zertifikat muss hierzu nicht zwingend von einer getrusteten CA ausgestellt sein. Alternativ dazu könnte auch eine CA oder ein Trust-Anchor verwendet werden.
`0 | 1`	Selector: 0 = Hash des ganzen Zertifikates / 1 = Hash des PublicKey
`1`	Matching Type – entspricht SHA-256
`a46dd6b2…`	der zuvor ermittelte Zertifikats-Fingerprint

3.2.2. TLSA-Record in DNSSec gesichertem DNS hinterlegen

Die ermittelten SHA256-Hashes der Zertifikate werden in der DNS-Zonen-Konfiguration wie folgt hinterlegt:

Variante mit Selector: 0 = Hash des Zertifikates

```
root@Sec-NS2:/etc/named/domains# vi zone_it-sec.ovh
…
; DANE / TLSA fuer SMTP und HTTPS: mail.it-sec.ovh (Selector 0 = ganzes Zertifikat)
_25._tcp.mail   IN TLSA 3 0 1 91ba252c1e6bd40767d646070c015d0b9edad8957f04880b3a6b1c…
_443._tcp.mail  IN TLSA 3 0 1 91ba252c1e6bd40767d646070c015d0b9edad8957f04880b3a6b1c…

; DANE / TLSA fuer HTTPS: it-sec.ovh und www.it-sec.ovh (Selector 0 = ganzes Zertifikat)
_443._tcp.www   IN TLSA 3 0 1 1597dc6d11cf983dee0d0c7094daff635dfc5ffa3fe9390d021a12…
_443._tcp       IN TLSA 3 0 1 1597dc6d11cf983dee0d0c7094daff635dfc5ffa3fe9390d021a12…
…
```

Variante mit Selector: 1 = Hash des PublicKeys

```
root@Sec-NS2:/etc/named/domains# vi zone_it-sec.ovh
…
; DANE / TLSA fuer SMTP und HTTPS: mail.it-sec.ovh (Selector 1 = PubKey)
_25._tcp.mail   IN TLSA 3 1 1 a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a…
_443._tcp.mail  IN TLSA 3 1 1 a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a…

; DANE / TLSA fuer HTTPS: it-sec.ovh und www.it-sec.ovh (Selector 1 = PubKey)
_443._tcp.www   IN TLSA 3 1 1 a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a…
_443._tcp       IN TLSA 3 1 1 a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a…

; DANE / TLSA fuer POP3S und IMAPS: mail.it-sec.ovh (Selector 1 = PubKey)
_995._tcp.mail  IN TLSA 3 1 1 a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a…
_993._tcp.mail  IN TLSA 3 1 1 a46dd6b2cb43b1f32445d1778410a55d1298c712942483aef03c1a…
…
```

Die Variante mit Selector 1 bietet wie bereits erwähnt den Vorteil, dass bei gleichbleibendem Public-Key sich der TLSA-Record nicht ändert. Wird also das Zertifikat unter Beibehaltung des Schlüsselpaares erneuert (d.h. der CSR beibehalten) so müssen keine Anpassungen im DNS vorgenommen werden. Wird – wie im Beispiel demonstriert – das gleiche Schlüsselpaar für zwei Zertifikate verwendet, so sind die TLSA-Records identisch.

Abschließender Reload des Nameservers:

```
root@Sec-NS2:/etc/named/domains# systemctl reload bind9
```

3.3. Zertifikatswechsel – Erneuerung der TLSA-Records

Bei einem Zertifikatswechsel ist zu bedenken, dass eventuell auch die TLSA-Records neu zu erzeugen sind. Da DNS als cachendes, verteiltes System arbeitet, ist hierfür gegebenenfalls rechtzeitig die TimeToLive der betreffenden Zone zu reduzieren, um den möglichen Mismatch zwischen tatsächlich verwendetem Zertifikat und gemäß TLSA-Record vorgesehenem Zertifikat möglichst kurz zu halten. Wird wie hier gezeigt mit dem Selector vom Typ 1 (Public-Key) gearbeitet, und wird ein neues Zertifikat unter Beibehaltung des Certificate Signing Requests (CSR, siehe Abschnitt 3.1.2) angefordert, ist (da der Public-Key beibehalten wird) auch keine Änderung der TLSA-Records nötig. Sollen zwei Zertifikate überlappend Gültigkeit haben, so sind mehrere TLSA-Records lautend auf das gleiche Service aber mit unterschiedlichen Fingerprints anzulegen [RFC-7672, Kapitel 4].

3.4. Prüfung der TLSA-Records

Das Vorhandensein eines TLSA Records inklusive der DNSSec Signatur kann folgendermaßen mittels DIG geprüft werden:

```
root@Sec-NS2:~# dig +dnssec +noall +answer +multi _25._tcp.mail.it-sec.ovh TLSA
_25._tcp.mail.it-sec.ovh. 1200 IN TLSA 3 1 1 (
                               A46DD6B2CB43B1F32445D1778410A55D1298C7129424
                               83AEF03C1A6C6F0C16B5 )
_25._tcp.mail.it-sec.ovh. 1200 IN RRSIG TLSA 8 5 1200 (
                               20151123165240 20151024155240 60261 it-sec.ovh.
                               LmdSnSHnZKvPiu93SDawfSihB+ye1txY30etpKBhAj5K … = )
```

Die WebSites https://dane.sys4.de/ sowie https://www.tlsa.info/ stellen einen Online-Validator bereit – hiermit kann auch bereits ohne installiertem MailServer der Zustand hinsichtlich des TLSA-Records geprüft werden (siehe Abbildung 24):

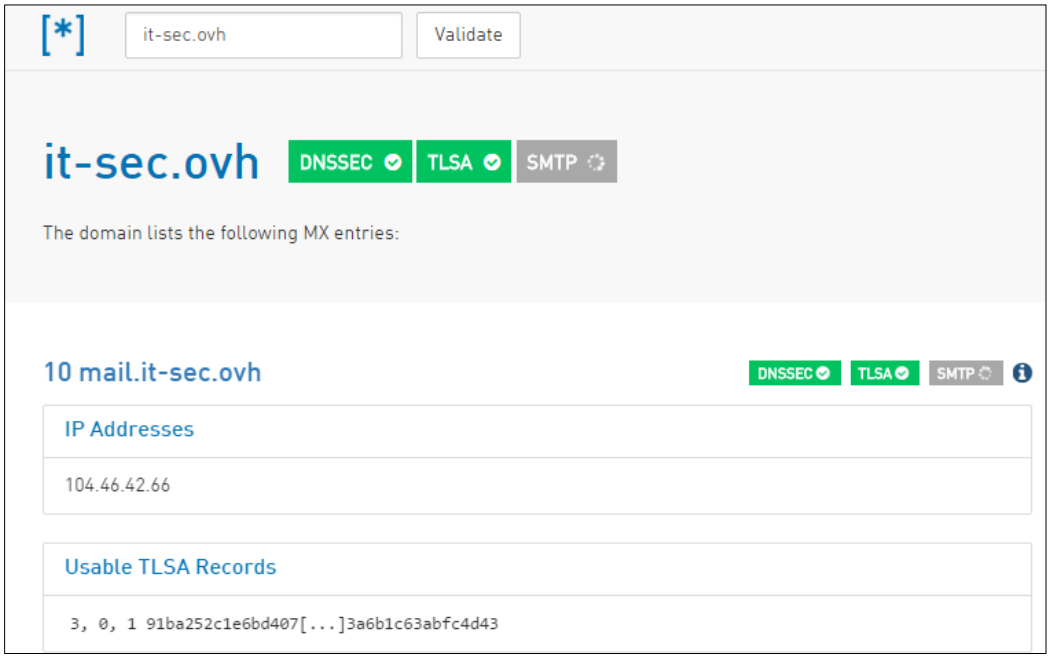


Abbildung 24: DANE / DNSSec / TLSA Check auf https://dane.sys3.de

DANE / TLSA gesicherte WebSites können mittels des Browser-PlugIns von https://www.dnssec-validator.cz/ validiert werden, hierzu muss an dieser Stelle jedoch erst ein Webserver installiert und konfiguriert werden – dieser Test wurde nach Einrichtung des RoundCube-WebMailers durchgeführt, siehe Abschnitt 5.3.3 und 5.3.4.

4. E-Mail

Eine umfassende Setup-Empfehlung hinsichtlich eines Mailserversystems würde den Rahmen dieser Dokumentation sprengen und den hier betrachteten Fokus verfehlen. Zu unterschiedlich sind die Anforderungen und daraus resultierenden Varianten.

Abbildung 25 zeigt exemplarisch den Aufbau und das Zusammenwirken der Komponenten eines typischen E-Mail Systems basierend auf Postfix, Dovecot und Roundcube.

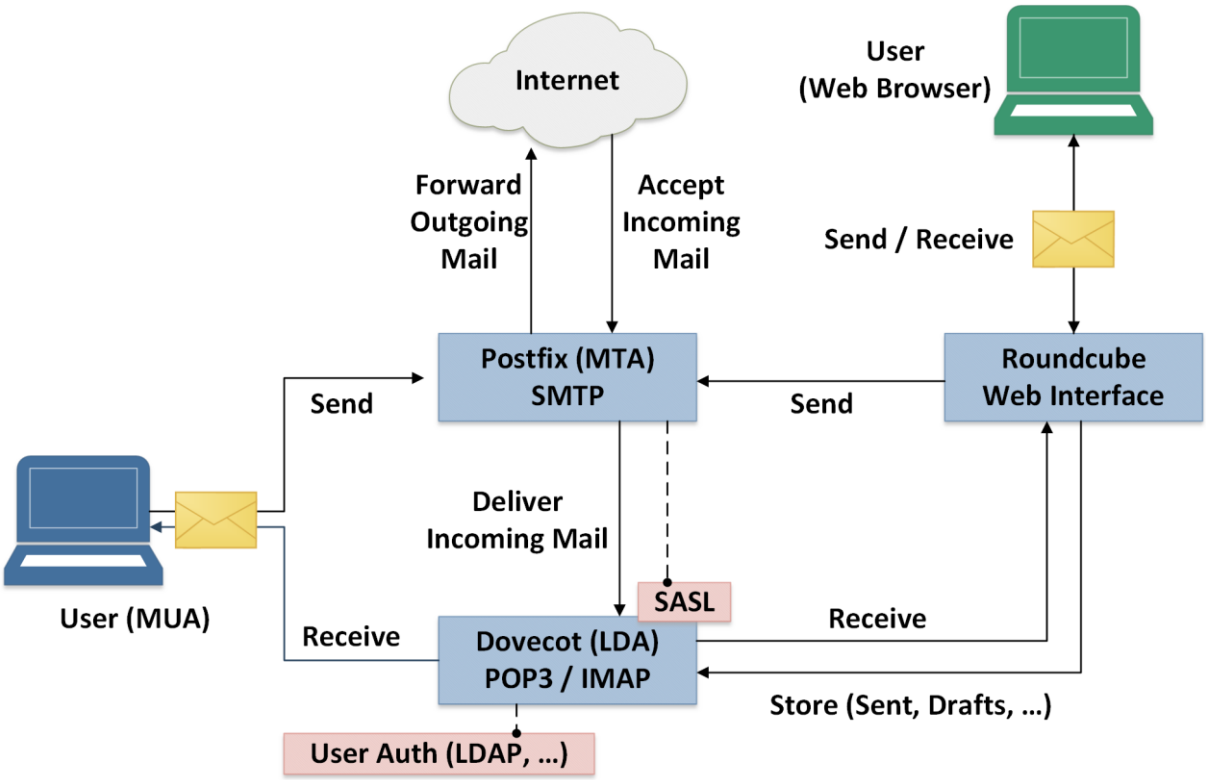


Abbildung 25: Zusammenwirken eines typischen E-Mail Systems

Das nachfolgend beschriebene Mail-System wird grundsätzlich aus den in Abbildung 25 dargestellten Komponenten bestehen, lediglich die Benutzerauthentifizierung wird nicht über LDAP sondern mittels lokaler Linux-Konten vorgenommen werden.

- Die Mail-Übermittlung von und ins Internet wird hierbei vom Mail Transfer Agent (MTA) Postfix realisiert, mittels STARTTLS (siehe Abbildung 26) auf dem Standard-SMTP-Port 25/tcp erfolgen, und mittels Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) nach [RFC-7672] abgesichert werden.
- Die Mail-Einlieferung seitens Benutzer (MUA – Mail User Agent) erfolgt ebenfalls über STARTTLS, allerdings über den Submission-Port 587/tcp und nach Authentifizierung.
- Der Mail-Abruf seitens Benutzer erfolgt über POP3s (995/tcp) sowie IMAPs (993/tcp) welches vom Local Delivery Agent (LDA) Dovecot bereitgestellt wird.
- Alternativ kann das Mailsystem mittels Web-Interface (RoundCube) genutzt werden.
- Um mit einem einzelnen Zertifikat das Auslangen zu finden, werden die Dienste allesamt unter dem gleichen Full-Qualified-Domain-Name (FQDN) mail.it-sec.ovh zur Verfügung gestellt.

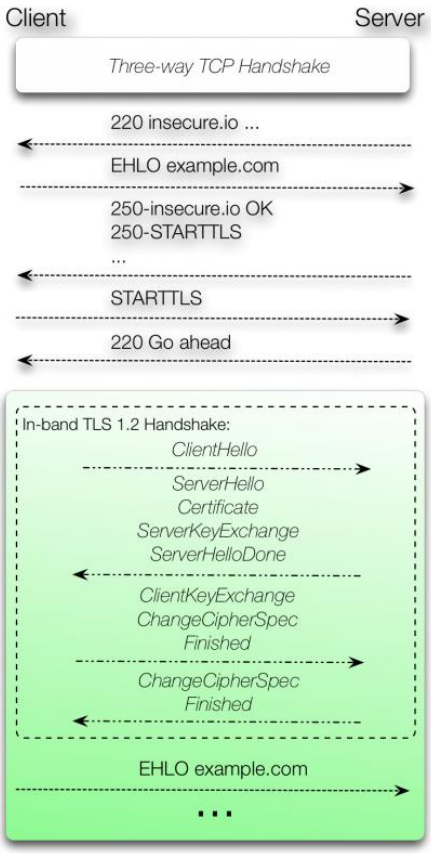


Abbildung 26: SMTP-Connection mit STARTTLS - Quelle: [SBA-TLSmail]

Abbildung 27 zeigt exemplarisch den Weg einer E-Mail von der Anlieferung des Benutzers über den Mail-Submission-Agent, den Transport über zwei Mail-Transport-Agents bis zum Mail-Delivery-Agent der die E-Mail für den Anwender per POP3(S) und IMAP(S) verfügbar macht.

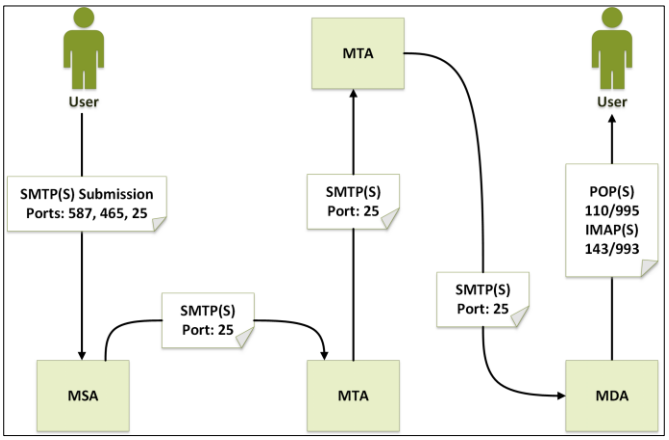


Abbildung 27: Transport-Weg einer E-Mail

Die Verwendung von „nur“ opportunistischem StartTLS resultiert aus der Tatsache, dass mangels breitflächiger Server-Unterstützung immer noch ein signifikanter Teil an E-Mails ausschließlich im Klartext transportiert werden kann. Google als einer der größten Mail-Anbieter weltweit veröffentlichte, dass deren GMail-Dienst per Oktober 2015 lediglich 56% der erhaltenen E-Mails von externen Mail-Servern verschlüsselt angeliefert bekam, und auch ausgehend nur 82% der E-Mails mittels TLS gesichert zugestellt werden konnten (siehe Abbildung 28) [Google-Mail].

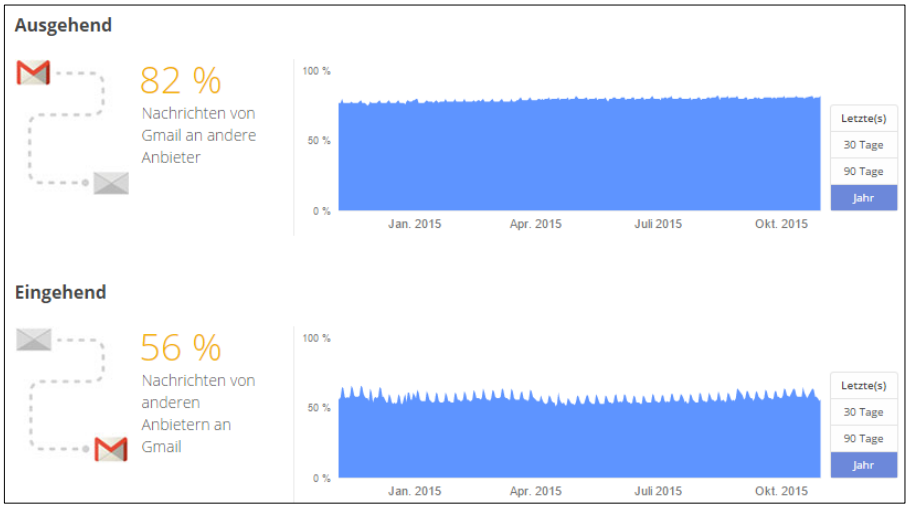


Abbildung 28: Google Transparenzbericht vom Oktober 2015 - Quelle: [Google-Mail]

Aktuelle Studien von SBA-Research stellen zudem fest, dass es um die Konfiguration hinsichtlich Verwendung aktueller TLS-Protokolle und zeitgemäßer Cipher-Suiten nicht zum Besten bestellt ist, und die eingesetzten Zertifikate oftmals lediglich Self-Signed, auf falsche Hostnamen ausgestellt oder abgelaufen sind. Abbildung 29 zeigt die angetroffenen Zertifikate, auf der Mehrzahl an Mail-Servern sind diese nicht von vertrauenswürdigen CA's ausgestellt (geprüft wurde gegen die CA's aus dem Trust-Store von Mozilla) sondern Self-Signed oder aus unbekannten CAs („local") beziehungsweise mit ungültiger Certificate-Chains („ssc" = SelfSigned in Chain).
Abbildung 30 zeigt, welche SSL/TLS Protokolle auf Mailservern im gesamten Internet IPv4 Address-Space auf den unterschiedlichen Ports und somit Diensten (SMTP/POP3/IMAP) angetroffen wurden [SBA-TLSmail]. Abbildung 31 illustriert den immer noch vorhandenen hohen Verbreitungsgrad nicht mehr zeitgemäßer Cipher auf Mail-Servern.

Hier gilt es die Devise „**besser schlecht verschlüsselter E-Mail-Transport, als Klartext E-Mail-Transport**" anzuwenden.

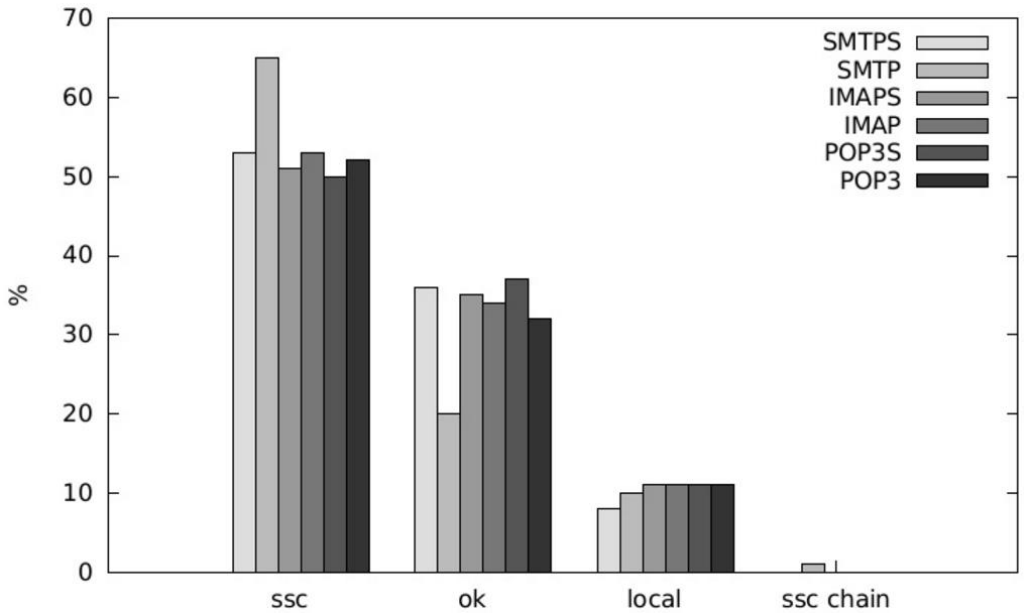


Abbildung 29: genutzte Zertifikate - SelfSigned / OK- Quelle: [SBA-TLSmail]

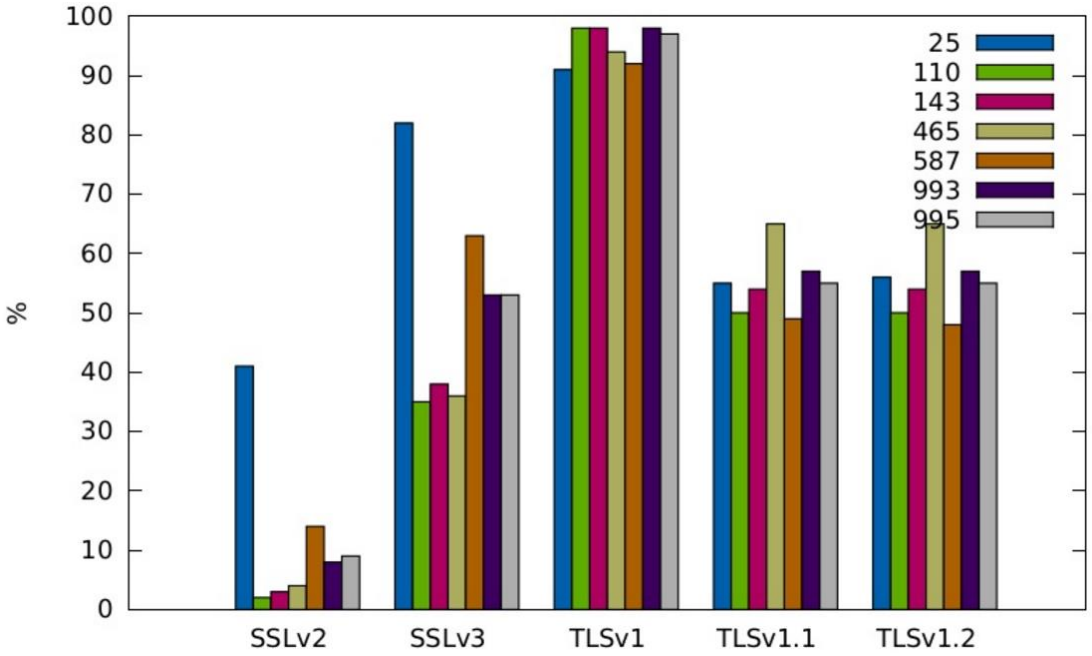


Abbildung 30: SSL/TLS Protokoll-Unterstützung auf Mail-Servern - Quelle: [SBA-TLSmail]

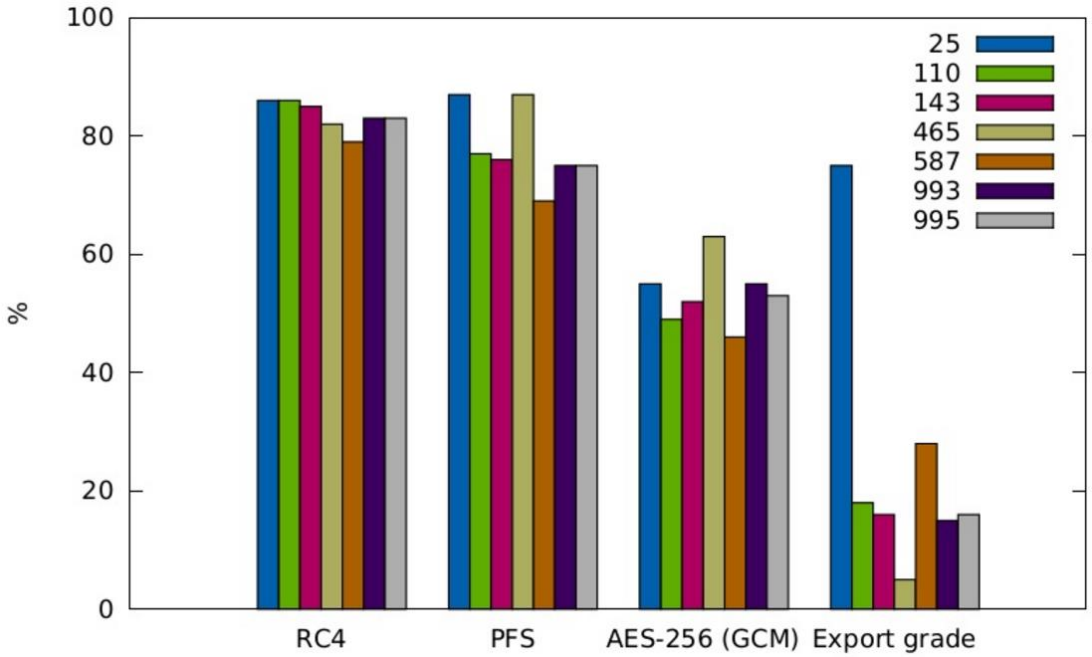


Abbildung 31: Cipher-Suite Unterstützung auf Mail-Servern - Quelle: [SBA-TLSmail]

Ein Mail-Server sollte es sich daher (solange die Gegenstelle nicht DANE einsetzt) nicht leisten, seinen Kommunikationspartner aufgrund von veralteten Cipher-Suiten oder Zertifikatsproblemen abzulehnen, die Folge wäre ein Fallback auf Klartext-Übertragung. In der Rolle als MTA ist daher auch für die nächste Zeit weiterhin „jede Verschlüsselung recht“. Anders sieht es bei den MSA und MDA Diensten aus, hier kann durchaus dazu übergegangen werden die Anwender zu aktueller Software zu drängen, welche sich problemlos mit aktueller Verschlüsselungstechnologie (TLS, DHE/ECDHE, AES, SHA2) verbinden kann.

4.1. Auswahl des Mailservers und der Distribution

Die Auswahl an Mailservern mit DANE-Unterstützung ist derzeit noch relativ überschaubar, eine kurze Recherche betreffend der vier populärsten[22] Mailserver (gemeinsamer Verbreitungsgrad mehr als 90%) ergibt:

- Postfix[23] ab Version 2.11 [Postfix-TLS, Abschnitt: DANE TLS authentication]
  - Verfügbar ab Debian 8 (Jessie): https://packages.debian.org/jessie/postfix
  - Verfügbar unter Ubuntu 14.04 LTS: http://packages.ubuntu.com/trusty/postfix
  - Noch nicht verfügbar unter CentOS 7 (beinhaltet 2.10.1), um Postfix mit DANE unter CentOS 7 bzw. RHEL 7 zu betreiben müsste auf Drittanbieter-Pakete zurückgegriffen werden, wobei unklar ist wie lange hierfür mit zuverlässigem Support (Updates) gerechnet werden kann. Das in Redhat/CentOS Kreisen als sehr zuverlässig geltende EPEL-Repository bietet leider kein Postfix Paket an: https://dl.fedoraproject.org/pub/epel/7/x86_64/repoview/mail-server.group.html
    Mögliche Quellen wären :
    - https://www.oostergo.net/ - bietet Postfix 2.11 und 3.0 Pakete an
    - http://ghettoforge.org/index.php/Packages - bietet Postfix 3.0 Pakete an
    - http://repo.mailserver.guru/7/os/x86_64/repoview/postfix.html - Postfix 2.11
- EXIM[24] ab Version 4.85[25]
  - Debian 8 (Jessie) beinhaltet nur 4.84 https://packages.debian.org/stable/mail/exim4
  - Verfügbar ab Ubuntu 15.10 LTS: http://packages.ubuntu.com/wily/exim4
  - CentOS 7 beinhaltet ebenfalls nur 4.84
- Sendmail[26] bietet bislang bis zur aktuellen Version 8.15.2 keine DANE-Unterstützung
- Microsoft Exchange[27] bietet aktuell keine DANE-Unterstützung, es dürften jedoch Drittanbieter-Erweiterungen wie z.B. DataEnter CryptoFilter[28] existieren, mit denen eine Nachrüstung von DANA möglich sein dürfte.

Die Auswahl der verwendeten Software fällt somit relativ alternativlos (sofern man keine Experimente eingehen möchte) auf Postfix 2.11 unter Debian 8 (Jessie).

22 http://www.securityspace.com/s_survey/data/man.201403/mxsurvey.html

23 http://www.postfix.org/

24 http://exim.org/

25 https://github.com/Exim/exim/blob/master/doc/doc-txt/ChangeLog

26 https://www.sendmail.com/sm/open_source/docs/

27 https://www.microsoft.com/de-de/office/exchange/default.aspx

28 http://www.dataenter.com/doc/cryptofilter.htm

4.2. Setup: Postfix 2.11

Wie eingangs bereits erwähnt würde eine umfassende Dokumentation der Installations-Möglichkeiten den Rahmen dieser Arbeit sprengen. Nachfolgend wird daher ohne jeden Schritt ausführlich zu kommentieren eine einfach gehaltene Postfix / Dovecot Installation, durchgeführt. Im Anschluss erfolgt fokussiert die SSL/TLS und DANE Konfiguration.

Eine sehr ausführliche Installationsanleitung mit umfangreichen Erläuterungen findet man im DokuWiki von Michael Nausch[29] sowie im Debian-Wiki[30] und auf der Postfix Website[31].

4.2.1. Postfix 2.11 Basis-Installation unter Debian 8

Postfix 2.11 wird von der Paketverwaltung angeboten und wie folgt installiert:

```
root@Sec-NS2:~# aptitude show postfix
Package: postfix
State: not installed
Version: 2.11.3-1
…

root@Sec-NS2:~# aptitude install postfix mailutils
```

Die Installation fragt einige Optionen ab:

```
General type of mail configuration: Internet Site
System mail name: mail.it-sec.ovh
```

Nach der Installation sind zahlreiche Konfigurationen in der `/etc/postfix/main.cf` gesetzt, welche mittels des Commandline-Tools `postconf` sowohl abgefragt, als auch editiert werden können.

`postconf -n` zeigt nur die gesetzten Optionen aus der `/etc/postfix/main.cf`

`postconf -d` zeigt alle Default-Optionen an

`postconf -e` ermöglicht Konfigurationsänderungen

`postconf -p` zeigt die aktuell wirksamen Settings an

[29] http://dokuwiki.nausch.org/doku.php/centos:mail_c7:start

[30] https://wiki.debian.org/Postfix

[31] http://www.postfix.org/docs.html

4.2.2. Konfiguration von Postfix

Folgende Konfiguration ist nach Installation in `/etc/postfix/main.cf` der vorzufinden:

```
root@Sec-NS2:/etc/postfix# postconf -n
alias_database = hash:/etc/aliases
alias_maps = hash:/etc/aliases
append_dot_mydomain = no
biff = no
config_directory = /etc/postfix
inet_interfaces = all
mailbox_size_limit = 0
mydestination = mail.it-sec.ovh, Sec-NS2.Sec-NS2.a6.internal.cloudapp.net, localhost.Sec-NS2.a6.internal.cloudapp.net, localhost
myhostname = Sec-NS2.Sec-NS2.a6.internal.cloudapp.net
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
myorigin = /etc/mailname
readme_directory = no
recipient_delimiter = +
relayhost =
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
smtpd_tls_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file = /etc/ssl/private/ssl-cert-snakeoil.key
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtpd_use_tls = yes
```

Zu korrigieren ist: `myhostname` wird im SMTP-Banner verwendet, sollte dem MX-Record der Domain und möglichst auch dem PTR-Eintrag der IP-Adresse des Servers entsprechen:

```
root@Sec-NS2:/etc/postfix# postconf -e "myhostname=mail.it-sec.ovh"

root@Sec-NS2:/etc/postfix# dig it-sec.ovh MX +short
10 mail.it-sec.ovh.

root@Sec-NS2:/etc/postfix# dig mail.it-sec.ovh +short
104.46.42.66

root@Sec-NS2:/etc/postfix# dig -x 104.46.42.66 +short
mail.it-sec.ovh.
```

Der Reverse-Eintrag zur IP-Adresse ist durch den Administrator des IP-Adress-Blocks zu setzen, Anbieter von virtuellen Servern oder Hosting-Lösungen ermöglichen dies in der Regel über eine API und/oder ein WebInterface.

`mydomain` entspricht der Internet-Domain des Mailsystems, `myorigin` legt fest unter welcher Domain die lokalen Benutzer geführt werden sollen:

```
root@Sec-NS2:/etc/postfix# postconf -e "mydomain=mail.it-sec.ovh"
root@Sec-NS2:/etc/postfix# postconf -p myorigin
myorigin = /etc/mailname
root@Sec-NS2:/etc/postfix# cat /etc/mailname
mail.it-sec.ovh
```

Statt des MBox-Formates soll das Maildir-Format verwendet werden:

```
root@Sec-NS2:/etc/postfix# postconf -e "home_mailbox=Maildir/"
```

Es sollen Virtual-Alias Domains und User Mapping auf LINUX-System Accounts verwendet werden [Postfix-Virtual]:

```
root@Sec-NS2:~# postconf -e "virtual_alias_domains=/etc/postfix/local-host-names"
root@Sec-NS2:~# postconf -e "virtual_alias_maps=hash:/etc/postfix/virtusertable"
```

In die Konfigurationsdatei `local-host-names` werden alle Mail-Domains eingetragen:

```
root@Sec-NS2:/etc/postfix# vi /etc/postfix/local-host-names
it-sec.ovh
test.it-sec.ovh
```

In der Konfigurationsdatei `virtusertable` werden die Zuordnungen zwischen Mail-Adressen und Linux-Usernamen getätigt:

```
root@Sec-NS2:/etc/postfix# vi /etc/postfix/virtusertable
gunnar@it-sec.ovh               gunnar
gunnar.haslinger@it-sec.ovh     gunnar

max@it-sec.ovh                  max
max.mustermann@it-sec.ovh       max

webmaster@it-sec.ovh            gunnar
hostmaster@it-sec.ovh           gunnar

# Catch-All
@it-sec.ovh                     gunnar
@test.it-sec.ovh                gunnar
```

Die Hash-Map der Virtuser-Table ist bei jeder Änderung neu zu generieren:

```
root@Sec-NS2:/etc/postfix# postmap hash:/etc/postfix/virtusertable
```

Damit die Änderungen wirksam werden, ist ein Reload durchzuführen:

```
root@Sec-NS2:/etc/postfix# postfix reload
```

4.2.3. SASL Authentifizierung für Mail-Relaying

Benutzer die den Mailserver zum Relaying von E-Mails benutzen müssen sich authentifizieren. Postfix selbst beinhaltet jedoch kein Backend-System um die Authentifizierung durchzuführen, hierfür wird SASL (Simple Authentication and Security Layer) verwendet.

Im nächsten Schritt (Kapitel 4.3) wird der IMAP/POP3-Server Dovecot installiert werden, welcher ein SASL-Interface[32] für Postfix bereitstellt.

Die Konfiguration wird für Postfix wie folgt vorgenommen:

```
root@Sec-NS2:/etc/postfix# postconf -e "smtpd_sasl_auth_enable=yes"
root@Sec-NS2:/etc/postfix# postconf -e "smtpd_sasl_type=dovecot"
root@Sec-NS2:/etc/postfix# postconf -e "smtpd_sasl_path=private/auth"
root@Sec-NS2:/etc/postfix# postconf -e "broken_sasl_auth_clients=yes"
```

Der Pfad `private/auth` zeigt hierbei nach `/var/spool/postfix/private/auth`, Postfix wird mit einer ChangeRoot-Umgebung betrieben, weshalb der Pfad relativ anzugeben ist.

[32] http://wiki2.dovecot.org/HowTo/PostfixAndDovecotSASL

4.2.4. Zertifikate und TLS/SSL-Konfiguration

Die in Kapitel 3.1 vorbereiteten Zertifikate kommen nun zur Anwendung, diese müssen im PEM-Format vorliegen, das TLS-Server-Zertifikat und die Chain (das Intermediate-Zertifikat) müssen zusammengefasst in einer Datei vorliegen. Das Private-Key-File darf nicht mittels Passphrase geschützt sein:

```
root@Sec-NS2:/etc/postfix#
postconf -e "smtpd_tls_cert_file=/etc/ssl/certs/mail.it-sec.ovh+chain.crt"
postconf -e "smtpd_tls_key_file=/etc/ssl/private/it-sec.ovh.key"
```

Anmerkung: Die hier nicht verwendeten Parameter `smtpd_tls_CAfile` und `smtpd_tls_CApath` sind nicht für die Aufnahme der Certificate-Chain zuständig, sondern kämen zur Anwendung, wenn SSL-Client-Authentifizierung zum Einsatz kommen bzw. definieren, welchen Stammzertifikaten bei ein bzw. ausgehenden Verbindungen vertraut werden sollen. Diese werden mit dem Certificate-Store des Betriebssystems belegt:

```
root@Sec-NS2:/etc/postfix#
postconf -e "smtpd_tls_CApath=/etc/ssl/certs"
postconf -e "smtp_tls_CApath=/etc/ssl/certs"
```

Die Konfiguration wird gemäß [BetterCrypto, Kapitel 2.3.4] und [Postfix-TLS] sowie [Postfix-PFS] vorgenommen.

```
root@Sec-NS2:/etc/postfix# postconf -e "smtpd_tls_security_level=may"
root@Sec-NS2:/etc/postfix# postconf -e "smtpd_tls_loglevel=1"
root@Sec-NS2:/etc/postfix# postconf -e "smtpd_tls_auth_only=yes"

root@Sec-NS2:/etc/postfix# postconf -e "smtp_tls_security_level=may"
root@Sec-NS2:/etc/postfix# postconf -e "smtp_tls_loglevel=1"

root@Sec-NS2:/etc/postfix# postconf -e "tls_ssl_options=NO_COMPRESSION"
```

Anmerkung: Die Konfiguration wird für ausgehende (`smtp`) und eingehende (`smtpd`) Verbindungen getrennt vorgenommen. Authentifizierung der User soll nur mit TLS und nicht im Klartext gestattet sein.

Hinsichtlich der verwendeten Cipher-Suite wird auch seitens [BetterCrypto, Kapitel 2.3.4] empfohlen für den MTA die Default-Konfiguration beizubehalten. Der Grund hierfür liegt im Argument, dass eine schlechte Verschlüsselung immer noch besser ist als Klartext-Kommunikation. Die Vielzahl an schlecht gewarteten Mailservern macht es nötig, auch „schlechte“ Protokolle und Chiffren zuzulassen.

```
root@Sec-NS2:/etc/ssl/private# postconf -p smtpd_tls_ciphers
smtpd_tls_ciphers = export
root@Sec-NS2:/etc/ssl/private# postconf -p smtp_tls_ciphers
smtp_tls_ciphers = export
```

Für die Anlieferung von Mails über den Submission-Port sollen die Clients der Anwender jedoch ausschließlich hochgradige Chiffren nutzen. Als Cipher-String wurde eine angepasste BetterCrypto-Configuration verwendet, die Überlegungen hierzu sind im Zuge der Apache-Konfiguration in Abschnitt 5.1.2 erläutert:

```
root@Sec-NS2:/etc/ssl/private# postconf -p | grep smtpd_tls_mandatory
smtpd_tls_mandatory_ciphers = medium
smtpd_tls_mandatory_exclude_ciphers =
smtpd_tls_mandatory_protocols = !SSLv2

root@Sec-NS2:/etc/postfix# postconf -e 'smtpd_tls_mandatory_protocols=!SSLv2, !SSLv3'
root@Sec-NS2:/etc/postfix# postconf -e "smtpd_tls_mandatory_ciphers=high"
root@Sec-NS2:/etc/postfix# postconf -e 'tls_high_cipherlist=-ALL:ECDH+aRSA+AES: ⮐
DH+aRSA+AES:aRSA+kRSA+AES:+AES256'
```

Es empfiehlt sich auch, die in Kapitel 3.1.5 erstellen Diffie-Hellman-Parameter zu setzen. Die Option `smtpd_tls_dh512_param_file` stellt die Diffie-Hellman-Parameter für Export-Cipher-Suites zur Verfügung, die Option `smtpd_tls_dh1024_param_file` wird für alle anderen Diffie-Hellman Cipher-Suites genutzt.

```
root@Sec-NS2:/etc/postfix#
postconf -e "smtpd_tls_dh512_param_file=/etc/ssl/private/dh_params_1024.pem"
postconf -e "smtpd_tls_dh1024_param_file=/etc/ssl/private/dh_params_2048.pem"
```

Bei Verwendung von Elliptic-Curve soll die `secp384r1` Kurve verwendet werden:

```
root@Sec-NS2:/etc/postfix# postconf -p | grep tls_eecdh_
smtpd_tls_eecdh_grade = strong
tls_eecdh_strong_curve = prime256v1
tls_eecdh_ultra_curve = secp384r1

root@Sec-NS2:/etc/postfix# postconf -e "smtpd_tls_eecdh_grade=ultra"
```

Bei eingehenden Mails soll im Mail-Header protokolliert werden, ob diese mittels TLS angeliefert wurden:

```
root@Sec-NS2:/etc/postfix# postconf -e "smtpd_tls_received_header=yes"
```

In der `master.cf` wird nun der Submission-Port aktiviert und die Verschlüsselung auf Mandatory konfiguriert. Außerdem wird dafür gesorgt, dass nicht der Client, sondern der Server auf Basis seiner Prioritäts-Reihenfolge definiert, welcher Cipher konkret zur Anwendung kommt:

```
root@Sec-NS2:/etc/postfix# vi master.cf
…
submission inet n       -       -       -       -       smtpd
  -o smtpd_tls_security_level=encrypt
  -o tls_preempt_cipherlist=yes
  -o syslog_name=postfix/submission
  -o smtpd_sasl_auth_enable=yes
…
```

Mittels `smtpd_sasl_auth_enable=yes` wird am Submission-Port die Verwendung von `AUTH PLAIN` erlaubt und announced, während diese Option am Standard-MTA-Port 25 deaktiviert bleibt.

Damit die Änderungen wirksam werden, ist ein Reload durchzuführen:

```
root@Sec-NS2:/etc/postfix# postfix reload
```

4.2.5. Konfiguration von DANE in Postfix

Die Konfiguration von DANE ist in [Postfix-TLS, Abschnitt DANE TLS authentication] beschrieben und im Auslieferungszustand bereits wie folgt konfiguriert:

```
root@Sec-NS2:/etc/postfix#
postconf -p | egrep "dane|smtpd_use_tls|smtp_dns_support|smtp_tls_security"
smtp_dns_support_level =
smtp_tls_security_level = may
smtpd_use_tls = yes
tls_dane_digest_agility = on
tls_dane_digests = sha512 sha256
tls_dane_trust_anchor_digest_enable = yes
```

- Mittels smtpd_use_tls wird zugreifenden Gegenstellen bereits ein STARTTLS Header offeriert, um diesen opportunistisches TLS zu offerieren.
- `tls_dane_digest_agility` und `tls_dane_digests` ermöglichen bereits eine flexible Verwendung von SHA256 und SHA512 Hashes in TLSA-Records.
- `tls_dane_trust_anchor_digest_enable` ermöglicht auch die Verwendung von Stammzertifikats-Pinning (Usage 2 – „Trust Anchor assertion“) mittels DANE TLSA Records.

Zusätzlich müssen noch folgende Konfigurationen vorgenommen werden:

```
root@Sec-NS2:/etc/postfix# postconf -e "smtp_dns_support_level=dnssec"
root@Sec-NS2:/etc/postfix# postconf -e "smtp_tls_security_level=dane"
```

- Aktivieren von DNSSec Lookups, dabei wird stets versucht DNSSec zu nutzen, steht DNSSec zur Verfügung werden die Antworten auch validiert. Dies stellt die Voraussetzung zur Nutzung von DANE dar.
- Konfiguration des `smtp_tls_security_level`, im Auslieferungszustand „`may`“ wird ein opportunistischer Ansatz gewählt, unterstützt eine Gegenstellt TLS wird dies verwendet, ansonsten werden Mails im Klartext zugestellt. Die mit Postfix 2.11 neu hinzugekommenen Optionen „`dane`“ und „`dane-only`“ ermöglichen nun zusätzlich die Validierung der Zertifikate mittels DANE. Während „`dane`“ einen Fallback auf „`may`“ erlaubt, wenn für eine Domain keine TLSA-Records hinterlegt sind, kommuniziert Postfix wie bisher TLS-Verschlüsselt oder im Klartext, ohne Authentifizierung der Gegenstelle. Mit „`dane-only`“ kann ein Fallback verhindert werden, allerdings ist diese Einstellung aktuell aufgrund des geringen Verbreitungsgrades von DANE für öffentliche Mailserver wenig zweckmäßig.

Damit die Änderungen wirksam werden, ist abermals ein Reload durchzuführen:

```
root@Sec-NS2:/etc/postfix# postfix reload
```

Wird eine Gegenstelle mittels DNSSEC als für DANE konfiguriert erkannt, so wird nicht nur das Zertifikat geprüft, sondern auch die Cipher-Suite aus `smtpd_tls_mandatory_ciphers` verwendet, welche im Abschnitt zuvor ausschließlich mit vertrauenswürdigen Cipher-Suites belegt wurde.

4.3. Setup: Dovecot als POP3 und IMAP Server

Wie auch bereits beim MTA Postfix geht dieses Dokument nicht im Detail auf die weitreichenden Überlegungen für eine produktive Dovecot POP3/IMAP-Mailserver Installation ein, sondern beschreibt lediglich die nötige Basis-Konfiguration und fokussiert auf die Absicherung mittels SSL/TLS. Eine ausführliche Installationsanleitung zu Dovecot findet sich ebenfalls im DokuWiki von Michael Nausch[33] sowie in der Dovecot-Dokumentation[34].

4.3.1. Dovecot 2.2 Basis-Installation unter Debian 8

```
root@Sec-NS2:/etc/bind# aptitude show dovecot-core
…
Version: 1:2.2.13-12~deb8u1
…
Suggests: ntp, dovecot-gssapi, dovecot-sieve, dovecot-pgsql, dovecot-mysql, dovecot-sqlite,
dovecot-ldap, dovecot-imapd, dovecot-pop3d, dovecot-lmtpd,
          dovecot-managesieved, dovecot-solr, dovecot-lucene
…
root@Sec-NS2:/etc/bind# aptitude install dovecot-imapd dovecot-pop3d
```

Die Konfiguration befindet sich in `/etc/dovecot/dovecot.conf` und bindet alle mit ausgelieferten Konfigurationsdateien in `/etc/dovecot/conf.d` sowie die manuell zu erstellende `local.conf` mit ein:

```
root@Sec-NS2:~# find /etc/dovecot -type f -iname "*.conf" | sort
/etc/dovecot/conf.d/10-auth.conf
/etc/dovecot/conf.d/10-director.conf
/etc/dovecot/conf.d/10-logging.conf
/etc/dovecot/conf.d/10-mail.conf
/etc/dovecot/conf.d/10-master.conf
/etc/dovecot/conf.d/10-ssl.conf
/etc/dovecot/conf.d/10-tcpwrapper.conf
/etc/dovecot/conf.d/15-lda.conf
/etc/dovecot/conf.d/15-mailboxes.conf
/etc/dovecot/conf.d/20-imap.conf
/etc/dovecot/conf.d/20-pop3.conf
/etc/dovecot/conf.d/90-acl.conf
/etc/dovecot/conf.d/90-plugin.conf
/etc/dovecot/conf.d/90-quota.conf
/etc/dovecot/dovecot.conf
/etc/dovecot/local.conf
```

Folgende Anpassungen sind (durch Übersteuern der Werte der bestehenden Konfigurationsdateien) mittels `local.conf` durchzuführen:

- Verwendung des Maildir-Formats
- Automatisches Anlegen der Spezial-Folder für Sent, Trash, Junk und Draft
- Die Klartext-Services imap und pop3 sollen nur über Localhost (127.0.0.1) und nicht aus dem Internet verfügbar gemacht werden, diese werden lediglich vom lokalen Webmail-Interface Roundcube benutzt werden.
- Für Postfix muss eine SASL-Schnittstelle bereitgestellt werden (siehe Abschnitt 4.2.3).

[33] http://dokuwiki.nausch.org/doku.php/centos:mail_c7:start

[34] http://wiki2.dovecot.org/

Das Konfigurationsfile `/etc/dovecot/local.conf` wird wie folgt konfiguriert:

```
root@Sec-NS2:/etc/dovecot# vi local.conf
```

```
mail_location = maildir:~/Maildir

namespace inbox {

  mailbox Drafts {
    auto=subscribe
    special_use = \Drafts
  }

  mailbox Junk {
    auto=subscribe
    special_use = \Junk
  }

  mailbox Trash {
    auto=subscribe
    special_use = \Trash
  }

  mailbox Sent {
    auto=subscribe
    special_use = \Sent
  }
}

# Klartext-Kommunikation nur ueber Localhost
service imap-login {
   inet_listener imap {
     address = 127.0.0.1
   }
}
service pop3-login {
   inet_listener pop3 {
     address = 127.0.0.1
    }
}

# SASL-Authentifizierung fuer Postfix zur Verfuegung stellen
service auth {
  unix_listener /var/spool/postfix/private/auth {
    mode = 0660
    user = postfix
    group = postfix
  }
}
```

4.3.2. Zertifikate und TLS/SSL-Konfiguration

Die Default-SSL-Konfiguration von Dovecot findet sich in `/etc/dovecot/conf.d/10-ssl.conf`, wie bereits bei der vorangegangenen Konfiguration werden die eigenen Anpassungen jedoch in die `/etc/dovecot/local.conf` ausgelagert. Dabei wird den Empfehlungen von [Bettercrypto, Kapitel 2.3.2] gefolgt, als zusätzliche Referenz dient das Dovecot Wiki[35]. Da POP3S und IMAPS das gleiche Zertifikat verwenden fällt die Konfiguration sehr kompakt aus, Konfigurationsmöglichkeiten getrennt nach POP3 und IMAP-Dämon werden im Wiki erläutert. Als Cipher-String wurde eine angepasste BetterCrypto-Configuration verwendet, die Überlegungen hierzu sind im Zuge der Apache-Konfiguration in Abschnitt 5.1.2 erläutert.

```
root@Sec-NS2:/etc/dovecot# vi local.conf
…

# SSL Settings, Doku: http://wiki2.dovecot.org/SSL/DovecotConfiguration
ssl = yes
ssl_cert = </etc/ssl/certs/mail.it-sec.ovh+chain.crt
ssl_key  = </etc/ssl/private/it-sec.ovh.key
ssl_client_ca_dir = /etc/ssl/certs

# SSL protocols to use, disable SSL, use TLS only
ssl_protocols = !SSLv3 !SSLv2

# SSL ciphers to use (modified bettercrypto.org Configuration):
ssl_cipher_list = -ALL:ECDH+aRSA+AES:DH+aRSA+AES:aRSA+kRSA+AES:+AES256

# Prefer the server's order of ciphers over client's.
ssl_prefer_server_ciphers = yes

# Diffie-Hellman parameters length to use (Default is only 1024!)
# ToDo: for ReGenerating DH-Parameters: manually delete /var/lib/dovecot/ssl-parameters.dat
#       and restart Dovecot to regenerate /var/lib/dovecot/ssl-parameters.dat
ssl_dh_parameters_length = 2048

# Turn SSL-Debug Logging on/off
verbose_ssl = no
```

Zum Abschluss wird das bestehende (mit schwachen Diffie-Hellman-Parametern bestückte) File `/var/lib/dovecot/ssl-parameters.dat` gelöscht und Dovecot neu gestartet.

```
root@Sec-NS2:/etc/dovecot# rm /var/lib/dovecot/ssl-parameters.dat
root@Sec-NS2:/etc/dovecot# service dovecot restart
…
root@Sec-NS2:/etc/dovecot# ls -la /var/lib/dovecot/ssl-*
-rw-r--r-- 1 root root 360 Oct 26 17:00 /var/lib/dovecot/ssl-parameters.dat
```

Nach dem Neustart stehen die Klartext-Services nur mehr auf Localhost und die verschlüsselten Services auf allen Interfaces zur Verfügung:

```
root@Sec-NS2:~# netstat -anp | grep "LISTEN " | grep dovecot
tcp     0      0 127.0.0.1:110           0.0.0.0:*               LISTEN      5182/dovecot
tcp     0      0 127.0.0.1:143           0.0.0.0:*               LISTEN      5182/dovecot
tcp     0      0 0.0.0.0:993             0.0.0.0:*               LISTEN      5182/dovecot
tcp     0      0 0.0.0.0:995             0.0.0.0:*               LISTEN      5182/dovecot
tcp6    0      0 :::993                  :::*                    LISTEN      5182/dovecot
tcp6    0      0 :::995                  :::*                    LISTEN      5182/dovecot
```

[35] http://wiki2.dovecot.org/SSL und http://wiki2.dovecot.org/SSL/DovecotConfiguration

4.4. Benutzeranlage

Die Benutzeranlage erfolgt im gewählten Szenario schlicht durch Erstellung eines Linux-Benutzers und Setzen eines Kennwortes:

```
root@Sec-NS2:~# useradd -c "Max Mustermann" --user-group --create-home mustermann
root@Sec-NS2:~# passwd mustermann
```

Das `Maildir` im Home-Verzeichnis wird durch Postfix oder Dovecot beim Einlangen der ersten E-Mail oder beim ersten Zugriff mittels Dovecot automatisch angelegt. Durch die getätigten Dovecot-Konfigurationen (siehe Abschnitt 4.3.1) werden auch die Spezial-Folders von Dovecot automatisch erstellt.

Um dem Benutzer zusätzlich zur primären Adresse Benutzername@mail.it-sec.ovh weitere Alias-Adressen zuzuweisen ist wie folgt vorzugehen:

```
root@Sec-NS2:/etc/postfix# vi /etc/postfix/virtusertable
…
max@it-sec.ovh                  max
max.mustermann@it-sec.ovh       max
…
```

Nach jeder Änderung der Virtusertable ist diese als Hashmap neu zu generieren:

```
root@Sec-NS2:/etc/postfix# postmap hash:/etc/postfix/virtusertable
```

4.5. Weitere zu berücksichtigende Themen

Um den Rahmen nicht zu sprengen wurden an dieser Stelle einige (für einen produktiven Einsatz relevante) Themen nicht näher betrachtet:

- Virtuelle User, virtuelle Mailboxen, Multi-Domain-Konfiguration (Local-Hostnames)
- SPF - Sender Policy Framework (RFC 6652), eine Möglichkeit mittels DNS-Einträgen einzugrenzen, von welchen Hosts aus Mails im Namen einer Domain verschickt werden dürfen.
- DKIM – DomainKeys Identified Mail (RFC 5585), stellt eine kryptographische Möglichkeit bereit festzustellen, ob eine Mail von einem legitimen Mailserver einer Domain versendet wurde. Der Empfänger kann also verifizieren, ob die E-Mail tatsächlich vom bekanntgegebenen Mail-Server des Absenders versendet wurde.
- DMARC - Domain-based Message Authentication, Reporting, and Conformance – aufbauend auf SPF und DKIM, ermöglicht eine Festlegung wie mit E-Mails die nicht den DKIM bzw. SPF-Regeln entsprechen verfahren werden soll.
- Anti-Spam z.B. mittels policyd-weight (http://www.policyd-weight.org/) bzw. SpamAssassin (http://spamassassin.apache.org/)
- Anti-Malware z.B. mittels Amavisd-new (http://www.ijs.si/software/amavisd/) bzw. AV-Lösungen – z.B. ClamAV (http://www.clamav.net/)

4.6. Postfix-Sicherheits-Checks

Eine sehr übersichtliche und praxisrelevante Checkliste wird vom deutschen BSI publiziert, eine Abarbeitung dieser wird empfohlen: [BSI-Postfix]

4.7. SMTP - Mailserver-Checks

Nachfolgend werden mehrere Möglichkeiten erläutert einen Mailserver hinsichtlich korrekter Konfiguration zu prüfen.

4.7.1. Basis-Check mittels MxToolbox

Die WebSite http://mxtoolbox.com/ bietet zahlreiche Tests hinsichtlich häufig anzutreffender Fehlkonfigurationen:

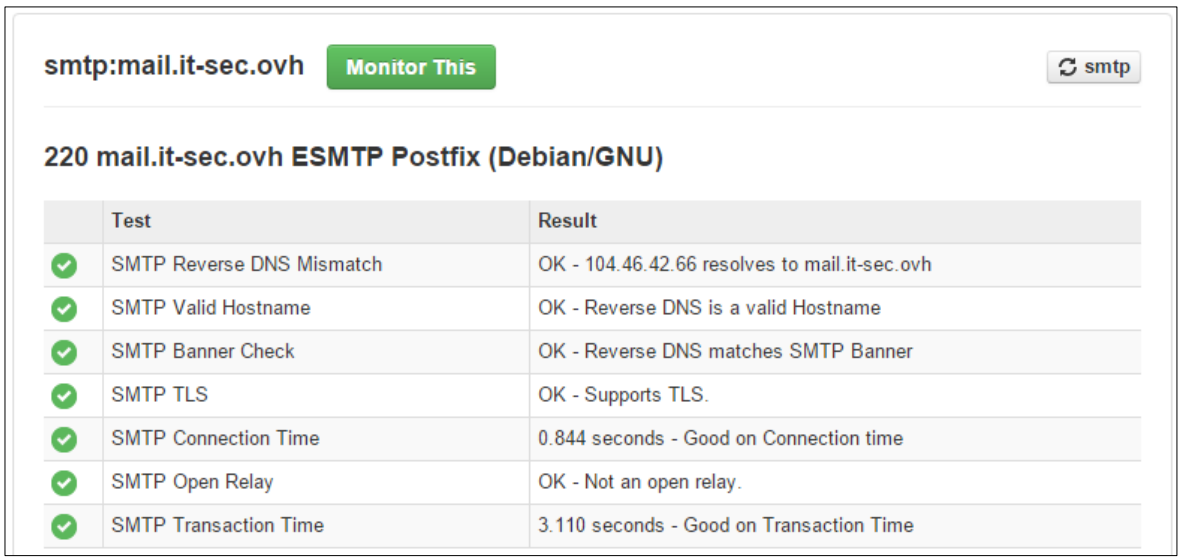


Abbildung 32: Mailserver-Check von http://mxtoolbox.com/

4.7.2. Authentifizierter SMTP-Relay-Versand

Im Basis-Check 4.7.1 wurde bereits geprüft, dass der Server keine „offenen Relay-Dienste“ anbietet. Nach erfolgter Authentifizierung muss jedoch der Mailversand auch an externe Adressen möglich sein. Der Test ist mittels eines Mailclients (z.B. Mozilla Thunderbird) mittels SMTP mit StartTLS am Submission-Port 587 an eine externe E-Mail-Adresse durchzuführen.

4.7.3. Versand und Empfangs-Check mittels CheckTLS.com

Die WebSite http://checktls.com/ bietet mehrere Mailserver-Tests, sowohl der Versand, als auch der Empfang können geprüft werden.

Unter der URL http://checktls.com/perl/TestSender.pl wird die Möglichkeit geboten eine E-Mail mit definiertem, einmaligen Passcode an die Adresse Test-Empfänger-Adresse test@TestSender.CheckTLS.com zu senden, das Ergebnis der Prüfung wird dem Absender per E-Mail zugestellt.

Hierzu ist auf der WebSite zuerst der Passcode zu generieren, anschließend vom Server eine Testmail mit diesem Passcode als Subject zu versenden:

```
root@Sec-NS2:~# echo Test |
mail --subject=ncgyrmpfbizs4 -r root@it-sec.ovh test@TestSender.CheckTLS.com
```

Der CheckTLS Service antwortet anschließend mit einer Bestätigungs-Mail, die den Status übermittelt und ein Transcript der bei CheckTLS.com eingehenden SMTP-Verbindung ist der Bestätigungs-E-Mail angefügt. Empfangene E-Mail als Antwort auf die versendete Test-Mail:

```
From testsender@CheckTLS.com  Mon Oct 26 09:33:39 2015
Return-Path: <testsender@CheckTLS.com>
X-Original-To: root@it-sec.ovh
Delivered-To: gunnar@mail.it-sec.ovh
Received: from www3.checktls.com (www3.checktls.com [69.61.187.232])
        (using TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits))
        (No client certificate requested)
        by mail.it-sec.ovh (Postfix) with ESMTPS id 555B9200E9
        for <root@it-sec.ovh>; Mon, 26 Oct 2015 09:33:39 +0100 (CET)
Date: Mon, 26 Oct 2015 04:33:38 -0400
To: root@it-sec.ovh
From: testsender@CheckTLS.com
Subject: CheckTLS TestSender SUCCESSFUL
X-Mailer: swaks v20111230.0 jetmore.org/john/code/swaks/

Below are the details from your CheckTLS TestSender test
from <root@it-sec.ovh> via [104.46.42.66]
run on 2015-10-26 04:33:38 EDT.
Original email Subject: ncgyrmpfbizs4

Your email was successfully sent securely using TLS.

…

<-- 220 ts3.checktls.com CheckTLS TestSender Mon, 26 Oct 2015 04:33:37 -0400
--> EHLO mail.it-sec.ovh
<-- 250-ts3.checktls.com Hello mail.it-sec.ovh [104.46.42.66], pleased to meet you
<-- 250-ENHANCEDSTATUSCODES
<-- 250-8BITMIME
<-- 250-STARTTLS
<-- 250 HELP
--> STARTTLS
<-- 220 Ready to start TLS
==== tls negotiation successful (cypher: AES256-GCM-SHA384,
     client cert: Subject Name: undefined;Issuer  Name: undefined;)
~~> EHLO mail.it-sec.ovh
<~~ 250-ts3.checktls.com Hello mail.it-sec.ovh [104.46.42.66], pleased to meet you
<~~ 250-ENHANCEDSTATUSCODES
<~~ 250-8BITMIME
<~~ 250 HELP
~~> MAIL FROM:<root@it-sec.ovh>
<~~ 250 Ok - mail from root@it-sec.ovh
~~> RCPT TO:<test@TestSender.CheckTLS.com>
<~~ 250 Ok - recipient test@TestSender.CheckTLS.com
~~> DATA
<~~ 354 Send data.  End with CRLF.CRLF
~~> Received: by mail.it-sec.ovh (Postfix, from userid 0)
~~>     id D8850214F9; Mon, 26 Oct 2015 09:33:36 +0100 (CET)
~~> Subject: ncgyrmpfbizs4
~~> To: <test@TestSender.CheckTLS.com>
~~> X-Mailer: mail (GNU Mailutils 2.99.98)
~~> Message-Id: <20151026083336.D8850214F9@mail.it-sec.ovh>
~~> Date: Mon, 26 Oct 2015 09:33:36 +0100 (CET)
~~> From: root@it-sec.ovh (root)
~~>
~~> Test
~~> .
<~~ 250 Ok
~~> QUIT
<~~ 221 ts3.checktls.com closing connection
```

Auch der Mail-Empfang kann geprüft werden: http://checktls.com/perl/TestReceiver.pl
Nach Angabe der E-Mail-Adresse root@it-sec.ovh wird eine Test-Mail versandt und das Transcript inklusive Erläuterung und farblicher Hervorhebung ausgegeben:

```
Trying TLS on mail.it-sec.ovh[104.46.42.66] (10):
 seconds    test stage and result
[000.106]   Connected to server
[000.212]<--220 mail.it-sec.ovh ESMTP Postfix (Debian/GNU)
[000.213]   We are allowed to connect
[000.213]-->EHLO checktls.com
[000.318]<--250-mail.it-sec.ovh
            250-PIPELINING
            250-SIZE 10240000
            250-VRFY
            250-ETRN
            250-STARTTLS
            250-ENHANCEDSTATUSCODES
            250-8BITMIME
            250 DSN
[000.319]   We can use this server
[000.319]   TLS is an option on this server
[000.319]-->STARTTLS
[000.424]<--220 2.0.0 Ready to start TLS
[000.425]   STARTTLS command works on this server
[000.687]   Cipher in use: ECDHE-RSA-AES256-GCM-SHA384
[000.687]   Connection converted to SSL
[000.707]   Certificate 1 of 3 in chain:
            subject= /C=AT/CN=mail.it-sec.ovh/emailAddress=domain.admin@hitco.at
            issuer= /C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Class 1 Primary Intermediate Server CA
[000.726]   Certificate 2 of 3 in chain:
            subject= /C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Class 1 Primary Intermediate Server CA
            issuer= /C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Certification Authority
[000.745]   Certificate 3 of 3 in chain:
            subject= /C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Certification Authority
            issuer= /C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Certification Authority
[000.745]   Cert VALIDATED: ok
[000.745]   Cert Hostname VERIFIED (mail.it-sec.ovh = mail.it-sec.ovh)
[000.745]~~>EHLO checktls.com
[000.852]<~~250-mail.it-sec.ovh
            250-PIPELINING
            250-SIZE 10240000
            250-VRFY
            250-ETRN
            250-ENHANCEDSTATUSCODES
            250-8BITMIME
            250 DSN
[000.853]   TLS successfully started on this server
[000.853]~~>MAIL FROM:<test@checktls.com>
[000.959]<~~250 2.1.0 Ok
[000.959]   Sender is OK
[000.959]~~>RCPT TO:<root@it-sec.ovh>
[001.066]<~~250 2.1.5 Ok
[001.066]   Recipient OK, E-mail address proofed
[001.066]~~>QUIT
[001.172]<~~221 2.0.0 Bye
```

Abbildung 33: Details des Test-Mail Versandes von http://checktls.com

4.7.4. Funktionscheck mit openssl s_client

Mittels `openssl s_client` lässt sich sowohl der SMTP-Port 25, als auch der Submission-Port 587 prüfen:

```
root@Sec-NS2:/etc/postfix# echo |
openssl s_client -starttls smtp -connect mail.it-sec.ovh:25 -CApath /etc/ssl/certs
CONNECTED(00000003)
depth=2 C = IL, O = StartCom Ltd., OU = Secure Digital Certificate Signing, CN = StartCom Certification Authority
verify return:1
depth=1 C = IL, O = StartCom Ltd., OU = Secure Digital Certificate Signing, CN = StartCom Class 1 Primary
Intermediate Server CA
verify return:1
depth=0 C = AT, CN = mail.it-sec.ovh, emailAddress = domain.admin@hitco.at
verify return:1
---
Certificate chain
 0 s:/C=AT/CN=mail.it-sec.ovh/emailAddress=domain.admin@hitco.at
   i:/C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Class 1 Primary Intermediate Server CA
 1 s:/C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Class 1 Primary Intermediate Server CA
   i:/C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Certification Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIHPTCCBiWgAwIBAgIHBmgq6WqV/jANBgkqhkiG9w0BAQsFADCBjDELMAkGA1UE
…
Q9T7Zmpt4uSbajHzQPpqH/WHOumRbxCXbAqxHel4YgwB
-----END CERTIFICATE-----
subject=/C=AT/CN=mail.it-sec.ovh/emailAddress=domain.admin@hitco.at
issuer=/C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom Class 1
Primary Intermediate Server CA
---
No client certificate CA names sent
---
SSL handshake has read 4492 bytes and written 456 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 4096 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: A36A301FF84C195F70497772A05DE8587530B05BC9AAE487B9F5A5D42D1E8764
    Session-ID-ctx:
    Master-Key: 5A34269B28D311A5566EED55BEC533DDB75DBD75EC84FDAED0B33FC32DFFDBB41884C36E4E5C97CDE8228CAE7349F50F
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 7200 (seconds)
    TLS session ticket:
       …
    Start Time: 1445792492
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
250 DSN
DONE
```

Führt man die Prüfung mit OpenSSL Version 1.0.2 oder höher durch, so wird zusätzlich zum „`Server public key`“ auch die DH-Schlüssellänge als „`Server Temp Key`“ angegeben.

Die in Debian enthaltene 1.0.1.k Version zeigt dies nicht an, ein Check von einer anderen Maschine aus (z.B. eine Prüfung unter Windows[36]) zeigt, dass die 2048bit DH-Parameter verwendet werden (standardmäßig würden hier nur ephemen 1024bit DH-Schlüssel verwendet werden).

DH-Schlüssellänge: Die konfigurierten 2048bit DH-Parameter werden verwendet:

```
C:\OpenSSL-Win64>openssl version
OpenSSL 1.0.2d 9 Jul 2015

C:\OpenSSL-Win64>echo | ⮐
openssl s_client -starttls smtp -cipher "EDH" -connect mail.it-sec.ovh:25
…
Peer signing digest: SHA512
Server Temp Key: DH, 2048 bits
…
Server public key is 4096 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : DHE-RSA-AES256-GCM-SHA384
…
```

Bei Verwendung von ECDH wird konfigurationsgemäß nun die P384-Kurve verwendet:

```
C:\OpenSSL-Win64>echo | ⮐
openssl s_client -starttls smtp -cipher "ECDH" -connect mail.it-sec.ovh:25
…
Peer signing digest: SHA512
Server Temp Key: ECDH, P-384, 384 bits
…
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
```

Auch das für den HTTPS-Check in Kapitel 5.3.5 vorgestellte TestSSL.sh Script lässt sich für SMTP mit STARTTLS nutzen:

```
root@Sec-NS2:/tmp# ./testssl.sh --starttls smtp mail.it-sec.ovh:25
…
 SSLv2      not offered (OK)
 SSLv3      not offered (OK)
 TLS 1      offered
 TLS 1.1    offered
 TLS 1.2    offered (OK)
…
```

[36] Binaries für OpenSSL unter Windows sind verfügbar über: https://www.openssl.org/community/binaries.html

4.7.5. Funktionscheck mit posttls-finger

Das mitgelieferte Werkzeug `posttls-finger` ermöglicht einen raschen Funktionalitäts- und Zertifikats-Check inklusive Prüfung des Zertifikates mittels DANE, ohne tatsächlich eine E-Mail zu versenden.

```
root@Sec-NS2:~# posttls-finger -f -l dane -L summary it-sec.ovh
posttls-finger: Connected to mail.it-sec.ovh[104.46.42.66]:25
posttls-finger: < 220 mail.it-sec.ovh ESMTP Postfix (Debian/GNU)
posttls-finger: > EHLO mail.it-sec.ovh
posttls-finger: < 250-mail.it-sec.ovh
posttls-finger: < 250-PIPELINING
posttls-finger: < 250-SIZE 10240000
posttls-finger: < 250-VRFY
posttls-finger: < 250-ETRN
posttls-finger: < 250-STARTTLS
posttls-finger: < 250-ENHANCEDSTATUSCODES
posttls-finger: < 250-8BITMIME
posttls-finger: < 250 DSN
posttls-finger: > STARTTLS
posttls-finger: < 220 2.0.0 Ready to start TLS
posttls-finger: Verified TLS connection established to mail.it-sec.ovh[104.46.42.66]:25:
TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits)
posttls-finger: > EHLO mail.it-sec.ovh
posttls-finger: < 250-mail.it-sec.ovh
posttls-finger: < 250-PIPELINING
posttls-finger: < 250-SIZE 10240000
posttls-finger: < 250-VRFY
posttls-finger: < 250-ETRN
posttls-finger: < 250-ENHANCEDSTATUSCODES
posttls-finger: < 250-8BITMIME
posttls-finger: < 250 DSN
posttls-finger: > QUIT
posttls-finger: < 221 2.0.0 Bye
```

Hinweis: Damit `posttls-finger` das Zertifikat mittels DANE verifiziert, benötigt es vom DNS-Stub-Resolver des Systems eine mit dem `ad`-Flag (Authenticated Data) versehene Antwort. Der verwendete lokale Nameserver antwortet jedoch für die Zone `it-sec.ovh` nicht mit dem ad-Flag, sondern setzt – da er für diese Zone autoritativ ist (siehe hierzu die Erläuterung in [RFC-3655, Kapitel 2.2]) - das `aa`-Flag (Authoritative Answer).

Nachfolgend der Vergleich zwischen lokalem Nameserver und dem frei verwendbaren Google-Resolver `8.8.8.8`:

```
root@Sec-NS2:~# dig @localhost it-sec.ovh MX +dnssec
…
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 7
…

root@Sec-NS2:~# dig @8.8.8.8 it-sec.ovh MX +dnssec
…
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
…
```

Um den `posttls-finger` Test von der lokalen Maschine aus durchzuführen, kann daher in der `/etc/resolv.conf` kurzfristig der DNSSec-fähige Google-Resolver 8.8.8.8 als erster Nameserver konfiguriert werden:

```
root@Sec-NS2:~# vi /etc/resolv.conf
…
nameserver 8.8.8.8
…
```

4.7.6. TLSA/DANE Konfigurations-Check

Die WebSite https://dane.sys4.de/ stellt (wie bereits in Kapitel 3.4 erwähnt) einen Online-Validator bereit, dieser prüft auch, ob der TLSA-Record mit dem Mailserver-Zertifikat übereinstimmt und simuliert einen Handshake (Details siehe [heise-Test]):

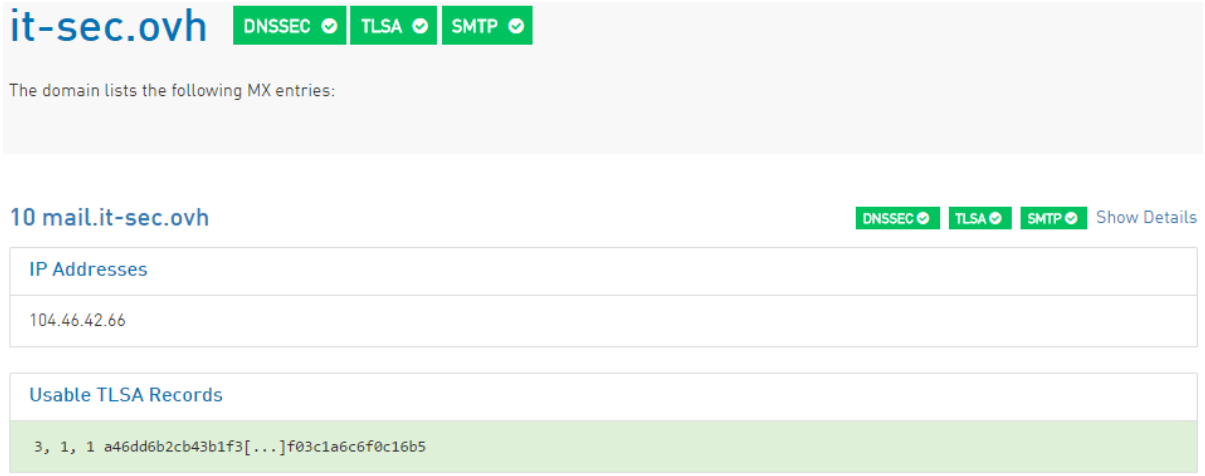


Abbildung 34: TLSA/DANE und Mailserver-Zertifikats-Check von https://dane.sys3.de

Weitere Testmöglichkeiten bietet https://ssl-tools.net/. Abbildung 35 zeigt das Testresultat, bemängelt wird, dass auch eine RC4 Cipher-Suite noch unterstützt wird. Angesichts dessen, dass zahlreiche schlecht gewartete Mailserver immer noch Mails mit RC4 verschlüsselt anliefern, wird dies aktuell absichtlich in Kauf genommen. Besser Mails werden schlecht verschlüsselt als im Klartext angeliefert.

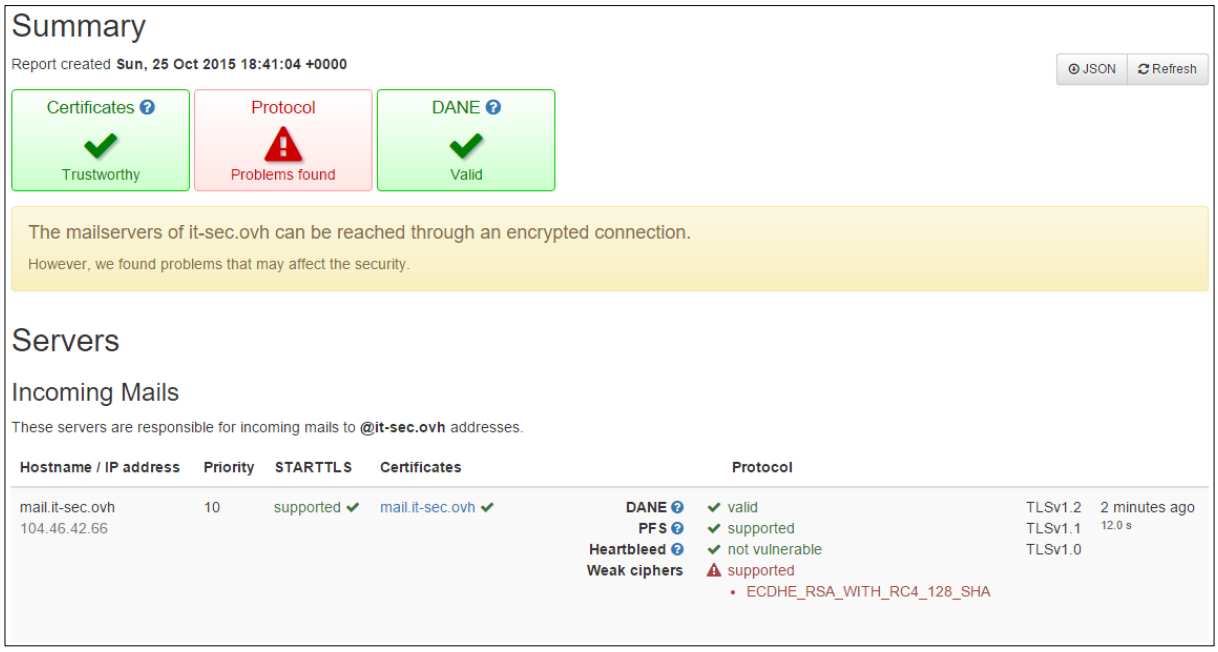


Abbildung 35: Mailserver-Test von https://ssl-tools.net

Abhilfe würde folgende Konfigurationsänderung schaffen:

```
root@Sec-NS2:~#
postconf -e "smtpd_tls_exclude_ciphers=aNULL, eNULL, LOW, EXPORT, RC4, DES, MD5, PSK"
```

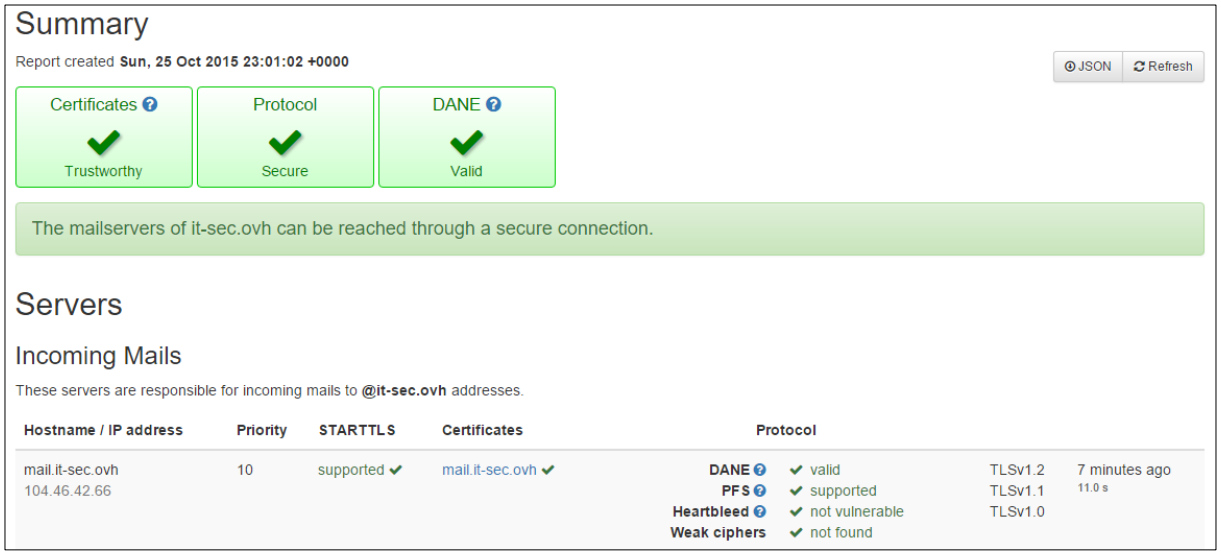


Abbildung 36: Mailserver-Test - Ergebnis nach Deaktivierung schwacher Cipher

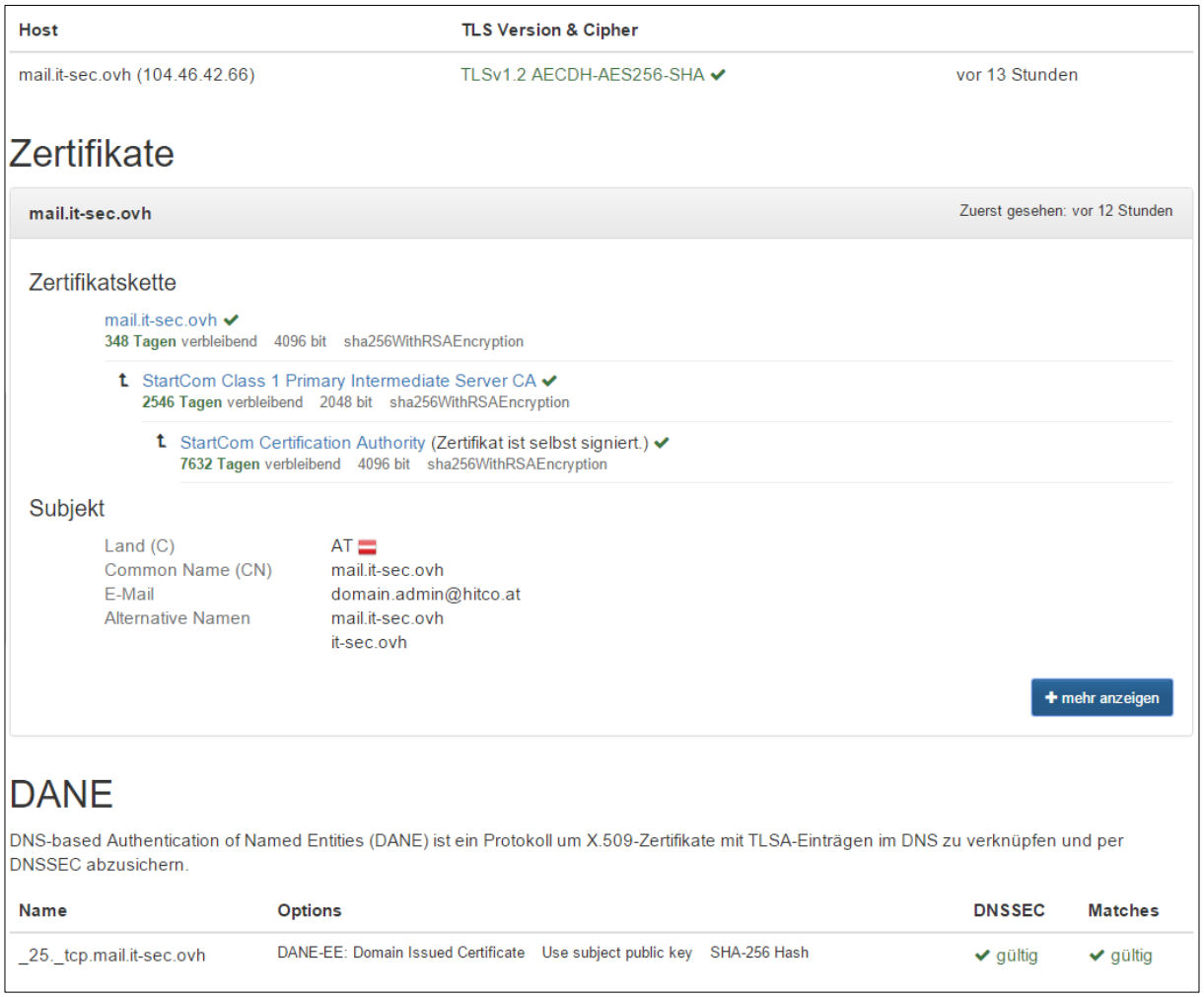


Abbildung 37: Zertifikats- und DANE-Check mittels Mailserver-Test https://ssl-tools.net

Mittels https://ssl-tools.net/ kann auch ein Mail-Delivery-Test durchgeführt werden. Eine leere Testmail an check@ssl-tools.net senden, Abbildung 38 zeigt das Testresultat.

```
root@Sec-NS2:~# echo Testmail | sendmail check@ssl-tools.net
```

Abbildung 38: Test des Mailversands mittels https://ssl-tools.net

4.7.7. Test der ausgehenden Mail-Zustellung mittels DANE

Um auch serverseitig zu prüfen, ob die Authentifizierung der Gegenstelle mittels DANE vorgenommen wurde, ist eine TLSA/DANE aktivierte Empfänger-Domain nötig. Ein Beispiel hierfür stellt posteo.de dar. Dass deren TLSA/DANE-Konfiguration korrekt ist wurde zuvor mittels des Validators https://dane.sys4.de/smtp/posteo.de geprüft.

```
root@Sec-NS2:~# echo Testmail | sendmail Testmail-test@posteo.de
root@Sec-NS2:~# tail -f /var/log/mail.log
Oct 26 12:30:10 Sec-NS2 postfix/smtp[2086]:
Verified TLS connection established to mx02.posteo.de[89.146.194.165]:25: TLSv1.2 with
cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits)
Oct 26 12:30:12 Sec-NS2 postfix/smtp[2086]:
1601920723: host mx02.posteo.de[89.146.194.165] said:
450 4.1.1 <Testmail-test@posteo.de>: Recipient address rejected: unverified address:
Recipient address lookup failed (in reply to RCPT TO command)
```

Die gewählte Empfängeradresse existiert zwar nicht, für einen Test des Verbindungsaufbaues mittels DANE reicht dieser Versuch jedoch vollkommen aus, der Log-Eintrag `Verified TLS connection` bedeutet, dass der Empfänger mittels DANE verifiziert werden konnte. Im Vergleich dazu würde die Zustellung an den GMX-Mailserver (der aktuell noch kein DANE unterstützt) lediglich eine `Trusted TLS connection` im Log protokollieren. Kann das von der Gegenstelle angebotene Zertifikat nicht einmal gegen die lokalen Stammzertifikate validiert werden, wird eine `Anonymous TLS connection` protokolliert:

```
Trusted TLS connection established to mx00.emig.gmx.net[212.227.15.9]:25: TLSv1.2 with
cipher DHE-RSA-AES256-GCM-SHA384 (256/256 bits)
```

```
Anonymous TLS connection established to mx1.bmlv.gv.at[193.171.152.59]:25: TLSv1.2 with
cipher ADH-AES256-GCM-SHA384 (256/256 bits)
```

Für den Fall, dass eine DANE-aktivierter Mail-Server sich mit dem falschen – also nicht dem im TLSA-Record vermerkten – Zertifikat ausweist, wird ein `Server certificate not verified` protokolliert und die Mail-Zustellung verweigert, das nachfolgende Beispiel demonstriert dies:

```
root@Sec-NS2:~# tail -f /var/log/mail.log
Nov 21 17:08:55 Sec-NS2 postfix/smtp[62081]: Trusted TLS connection established to
danebad.it-sec.ovh[84.200.20.238]:25: TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits)
Nov 21 17:08:55 Sec-NS2 postfix/smtp[62081]: 37BBA21507: to=<test@danebad.it-sec.ovh>,
relay=danebad.it-sec.ovh[84.200.20.238]:25, delay=963, delays=963/0.04/0.24/0, dsn=4.7.5,
status=deferred (Server certificate not verified)
```

4.8. IMAP/POP3 - Mailserver-Checks

Prüfung des Zugriffs:

- Klartext-Zugriff auf die Ports 143 und 110 darf vom Internet aus nicht möglich sein
- TLS-Zugriff auf die Ports 993 (IMAPS) und 995 (POP3S) muss möglich sein

Test von IMAP und POP3 z.B. mittels Thunderbird

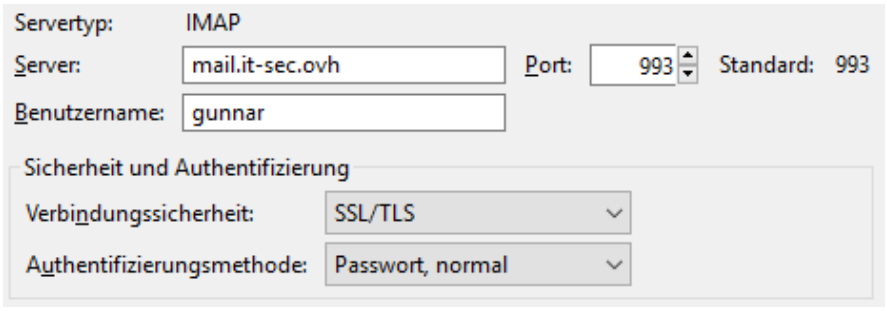


Abbildung 39: Konto-Einrichtung als IMAP Konto in Mozilla Thunderbird

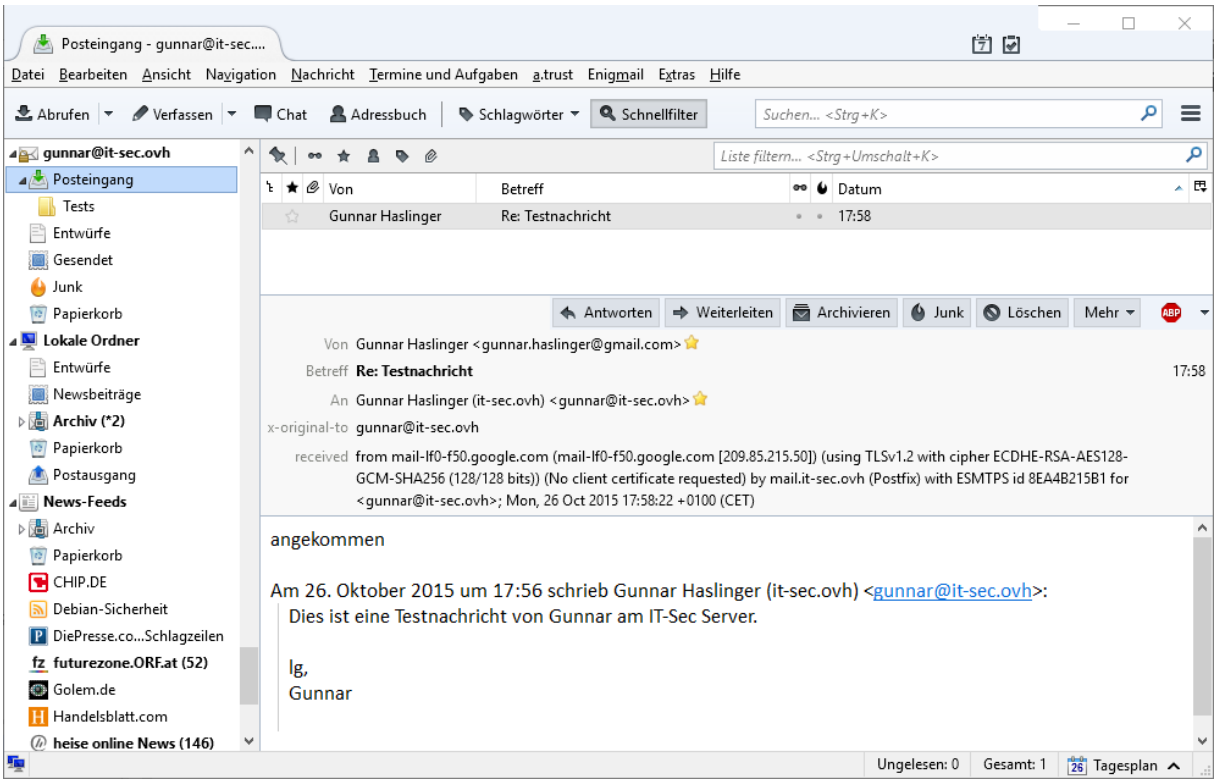


Abbildung 40: IMAP-Zugriff mittels Mozilla Thunderbird

4.8.1. TLS-Check mittels TestSSL (OpenSSL-Scripts)

Dirk Wetter stellt über GitHub[37] sowie auf https://testssl.sh/ ein Commandline-Tool zur Verfügung, um TLS/SSL Konfigurationen von Servern zu untersuchen. Das Tool besteht aus einem Bash-Script, welches OpenSSL zur Analyse nutzt. Download des Scripts:

```
root@Sec-NS2:/tmp# wget -q https://testssl.sh/testssl.sh
root@Sec-NS2:/tmp# wget -q https://testssl.sh/mapping-rfc.txt
root@Sec-NS2:/tmp# chmod 755 ./testssl.sh
```

Nutzung des Scripts, geprüft soll sowohl der POP3S als auch der IMAPS Dienst werden:

```
root@Sec-NS2:/tmp# ./testssl.sh mail.it-sec.ovh:993
root@Sec-NS2:/tmp# ./testssl.sh mail.it-sec.ovh:995
```

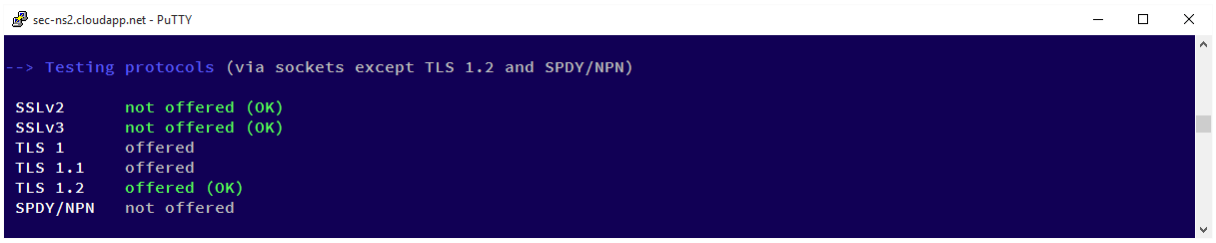


Abbildung 41: Test des IMAPS-Servers mit TestSSL.sh

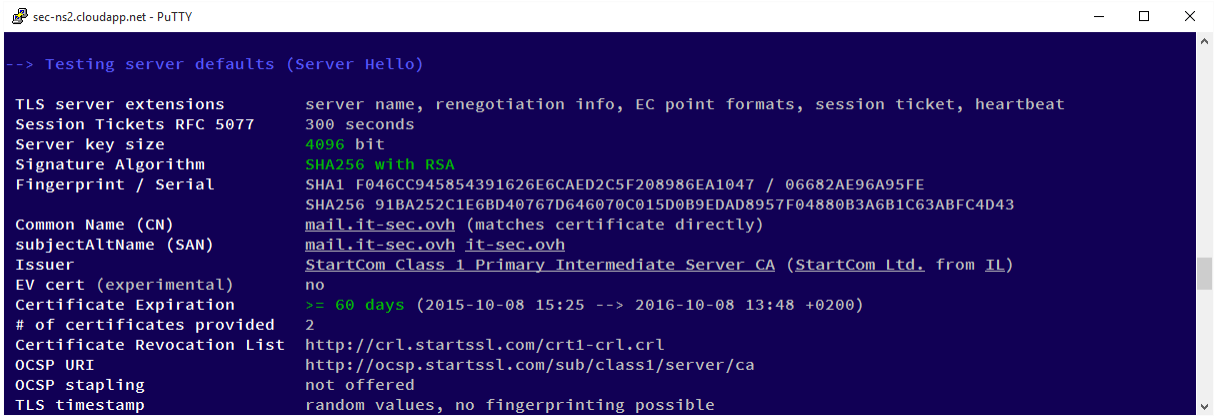


Abbildung 42: Zertifikat des IMAPS-Servers, mit TestSSL.sh ermittelt

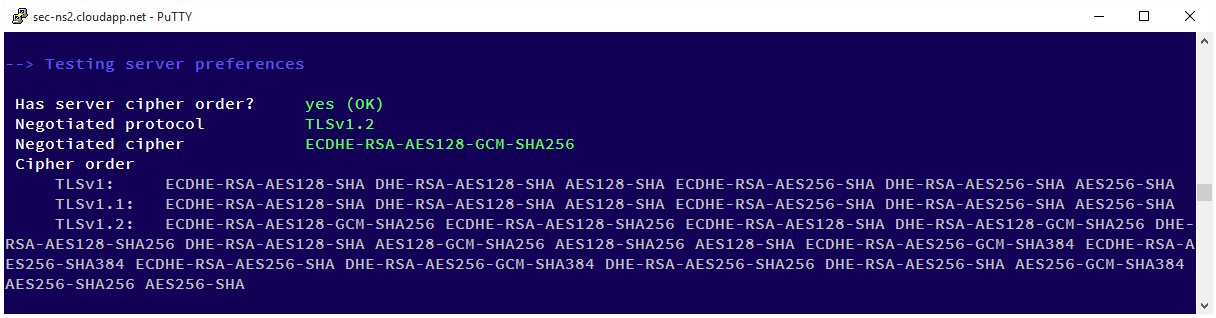


Abbildung 43: Cipher-Suiten des IMAPS-Servers, mit TestSSL.sh ermittelt

[37] https://github.com/drwetter/testssl.sh

4.8.2. TLS-Check mittels SSLyze

Mittels des kostenfreien Tools SSLyze[38] kann ebenfalls ein sehr ausführlicher Check vorgenommen werden. Die Accepted-Liste wird von SSLyze offenkundig nicht nach Server-Order sortiert ausgegeben:

```
root@Sec-NS2:~/sslyze# ./sslyze.py --regular mail.it-sec.ovh:993
root@Sec-NS2:~/sslyze# ./sslyze.py --regular mail.it-sec.ovh:995
root@Sec-NS2:~/sslyze# ./sslyze.py --regular --starttls=imap localhost:143
root@Sec-NS2:~/sslyze# ./sslyze.py --regular --starttls=pop3 localhost:110
…

  * TLSV1_2 Cipher Suites:
      Preferred:
                  ECDHE-RSA-AES128-GCM-SHA256   ECDH-384 bits  128 bits
      Accepted:
                  ECDHE-RSA-AES256-SHA384       ECDH-384 bits  256 bits
                  ECDHE-RSA-AES256-SHA          ECDH-384 bits  256 bits
                  ECDHE-RSA-AES256-GCM-SHA384   ECDH-384 bits  256 bits
                  DHE-RSA-AES256-SHA256         DH-2048 bits   256 bits
                  DHE-RSA-AES256-SHA            DH-2048 bits   256 bits
                  DHE-RSA-AES256-GCM-SHA384     DH-2048 bits   256 bits
                  AES256-SHA256                 -              256 bits
                  AES256-SHA                    -              256 bits
                  AES256-GCM-SHA384             -              256 bits
                  ECDHE-RSA-AES128-SHA256       ECDH-384 bits  128 bits
                  ECDHE-RSA-AES128-SHA          ECDH-384 bits  128 bits
                  ECDHE-RSA-AES128-GCM-SHA256   ECDH-384 bits  128 bits
                  DHE-RSA-AES128-SHA256         DH-2048 bits   128 bits
                  DHE-RSA-AES128-SHA            DH-2048 bits   128 bits
                  DHE-RSA-AES128-GCM-SHA256     DH-2048 bits   128 bits
                  AES128-SHA256                 -              128 bits
                  AES128-SHA                    -              128 bits
                  AES128-GCM-SHA256             -              128 bits

  * TLSV1_1 Cipher Suites:
      Preferred:
                  ECDHE-RSA-AES128-SHA          ECDH-384 bits  128 bits
      Accepted:
                  ECDHE-RSA-AES256-SHA          ECDH-384 bits  256 bits
                  DHE-RSA-AES256-SHA            DH-2048 bits   256 bits
                  AES256-SHA                    -              256 bits
                  ECDHE-RSA-AES128-SHA          ECDH-384 bits  128 bits
                  DHE-RSA-AES128-SHA            DH-2048 bits   128 bits
                  AES128-SHA                    -              128 bits

  * TLSV1 Cipher Suites:
      Preferred:
                  ECDHE-RSA-AES128-SHA          ECDH-384 bits  128 bits
      Accepted:
                  ECDHE-RSA-AES256-SHA          ECDH-384 bits  256 bits
                  DHE-RSA-AES256-SHA            DH-2048 bits   256 bits
                  AES256-SHA                    -              256 bits
                  ECDHE-RSA-AES128-SHA          ECDH-384 bits  128 bits
                  DHE-RSA-AES128-SHA            DH-2048 bits   128 bits
                  AES128-SHA                    -              128 bits

  * SSLV3 Cipher Suites:
      Server rejected all cipher suites.

  * SSLV2 Cipher Suites:
      Server rejected all cipher suites.
```

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

5. WebMail

Als WebMail-Lösung soll die sehr populäre und grafisch ansprechende RoundCube[39] Open-Source WebMail-Software zum Einsatz kommen. Diese setzt auf einem Webserver mit PHP und einem Datenbankserver (z.B. MySQL) auf.

5.1. Installation von Apache, PHP und MySQL

In Debian 8 sind über die Paketverwaltung die Apache-Version 2.4.10, PHP-Version 5.6.13 und MySQL-Version 5.5.46 jeweils mit backported Security-Updates verfügbar.

Installation von Apache 2.4.10:

```
root@Sec-NS2:~# aptitude install apache2
The following NEW packages will be installed:
  apache2 apache2-bin{a} apache2-data{a} apache2-utils{a} libapr1{a} libaprutil1{a}
libaprutil1-dbd-sqlite3{a} libaprutil1-ldap{a} libldap-2.4-2{a}
  liblua5.1-0{a} libsasl2-2{a} libsasl2-modules{a} libsasl2-modules-db{a} ssl-cert{a}
```

Installation von PHP 5.6.13:

```
root@Sec-NS2:~# aptitude install php5
The following NEW packages will be installed:
  libapache2-mod-php5{a} libmagic1{a} libonig2{a} libperl4-corelibs-perl{a} libqdbm14{a}
lsof{a} php5 php5-cli{a} php5-common{a} php5-json{a}
  php5-readline{a} psmisc{a}
```

Installation von MySQL 5.5.46:

```
root@Sec-NS2:~# aptitude install mysql-server php5-mysql
root@Sec-NS2:~# php5enmod mysql
```

Aktivierung der Apache-Module: SSL[40], Headers[41] (wird für HTTP Public Key Pinning nötig sein sowie für HTTP Strict Transport Security), Rewrite[42] (wird für den WebMailer RoundCube nötig sein) und der Default-SSL-Site (für einen ersten HTTPS-Test):

```
root@Sec-NS2:/etc/apache2/mods-available# a2enmod ssl
root@Sec-NS2:/etc/apache2/mods-available# a2enmod headers
root@Sec-NS2:/etc/apache2/mods-available# a2enmod rewrite
root@Sec-NS2:/etc/apache2/sites-available# a2ensite default-ssl.conf
root@Sec-NS2:/etc/apache2/sites-available# service apache2 restart
```

Die beiden WebSites https://www.it-sec.ovh/ sowie mail.it-sec.ovh sollen beide mit HTTPS verfügbar gemacht werden. Erstere URL wird lediglich eine statische Demo-Website beherbergen, unter mail.it-sec.ovh wird der RoundCube WebMailer zur Verfügung gestellt.

39 https://roundcube.net/

40 https://httpd.apache.org/docs/2.4/mod/mod_ssl.html

41 https://httpd.apache.org/docs/2.4/mod/mod_headers.html

42 https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html

5.1.1. Konfiguration der Apache-vHosts

Hierfür sind folgende Apache vHost-Konfigurationsdateien anzulegen:

Das Konfigurationsfile `/etc/apache2/sites-available/mail.it-sec.ovh.conf` stellt den vHost für mail.it-sec.ovh bei Zugriff mittels Klartext-HTTP dar. Es sorgt lediglich für einen Redirect auf die HTTPS-URL:

```
<VirtualHost *:80>
        ServerName mail.it-sec.ovh
        ServerAdmin hostmaster@it-sec.ovh
        DocumentRoot /var/www/mail.it-sec.ovh
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        Redirect permanent / https://mail.it-sec.ovh/
</VirtualHost>
```

Das Konfigurationsfile `/etc/apache2/sites-available/mail.it-sec.ovh_ssl.conf` konfiguriert den HTTPS-vHost wie folgt (Dokumentation siehe [IR-Apache], [IR-OpenSSL], [IR-TLS], Erläuterungen zur konkret gewählten Konfiguration finden sich in den nachfolgenden Abschnitten):

```
<IfModule mod_ssl.c>
  <VirtualHost _default_:443>
      ServerName mail.it-sec.ovh
      ServerAdmin hostmaster@it-sec.ovh

      DocumentRoot /var/www/mail.it-sec.ovh
      <Directory /var/www/mail.it-sec.ovh/>
                Options Indexes FollowSymLinks
                AllowOverride All
                Require all granted
      </Directory>

      ErrorLog ${APACHE_LOG_DIR}/error.log
      CustomLog ${APACHE_LOG_DIR}/access.log combined

      SSLEngine on
      SSLProtocol All -SSLv2 -SSLv3
      SSLHonorCipherOrder On
      SSLCompression off
      SSLUseStapling on

      # 1/2 Jahr HSTS header (60*60*24*182=15724800sek), inklusive Subdomains
      Header always set Strict-Transport-Security "max-age=15724800 ; includeSubDomains"

      # HTTP Public Key Pinning für Zertifikat (HPKP), 90 Tage (60*60*24*90=7776000)
      # mail.it-sec.ovh.crt PubKey = pG3WsstDsfMkRdF3hBClXRKYxxKUJIOu8DwabG8MFrU=
      # StartCom Certification Authority: "5C8kvU039KouVrl52D0eZSGf4Onjo4Khs8tmyTlV3nU="
      # zweiter Backup-Key: "i89r5g0BAe/9ubBIuTxrW5phVO8lv7cBvXxK/evFhXw="
      Header always set Public-Key-Pins
          "pin-sha256=\"pG3WsstDsfMkRdF3hBClXRKYxxKUJIOu8DwabG8MFrU=\";
           pin-sha256=\"5C8kvU039KouVrl52D0eZSGf4Onjo4Khs8tmyTlV3nU=\";
           pin-sha256=\"i89r5g0BAe/9ubBIuTxrW5phVO8lv7cBvXxK/evFhXw=\";
           max-age=7776000;
           report-uri=\"https://report-uri.io/report/866c4f253035d817119b9401f6116434\""

      SSLCipherSuite '-ALL:ECDH+aRSA+AES:DH+aRSA+AES:aRSA+kRSA+AES:+AES256'

      SSLCertificateFile      /etc/ssl/certs/mail.it-sec.ovh.crt
      SSLCertificateKeyFile   /etc/ssl/private/it-sec.ovh.key
      SSLCertificateChainFile /etc/ssl/certs/startssl.chain.class1.server.crt
  </VirtualHost>
</IfModule>
```

Für den www.it-sec.ovh vHost sowie den zugehörigen HTTPS-vHost wird analog dazu vorgegangen.

Diese Konfigurationsfiles werden anschließend aktiviert und Apache neu gestartet:

```
root@Sec-NS2:/etc/apache2/sites-available# a2ensite it-sec.ovh
root@Sec-NS2:/etc/apache2/sites-available# a2ensite it-sec.ovh_ssl
root@Sec-NS2:/etc/apache2/sites-available# a2ensite mail.it-sec.ovh
root@Sec-NS2:/etc/apache2/sites-available# a2ensite mail.it-sec.ovh_ssl

root@Sec-NS2:/etc/apache2/sites-available# apache2ctl graceful
```

5.1.2. SSL/TLS-Protokolle und Cipher-Suites – der CipherString

Bei der Auswahl der SSL/TLS-Protokolle wurde [BetterCrypo, Kapitel 2.1.1] folgend nur mehr TLS (aktuell v1.0/1.1/1.2) zugelassen und SSLv3 und SSLv2 deaktiviert.

Hinsichtlich des zu verwendenden CipherStrings wird für Apache in [BetterCrypo, Kapitel 2.1.1] ein anderer String als im Theorieteil [BetterCrypo, Kapitel 3.2.3] empfohlen. Eine Rückfrage auf der BetterCrypto-Mailingliste[43] ergibt, dass diese Diskrepanz nicht beabsichtigt ist, der im Theorieteil empfohlene CipherString sollte verwendet werden.

Der BetterCrypto.org CipherString-B aus dem Theorieteil [BetterCrypo, Kapitel 3.2.3] lautet:

```
root@Sec-NS2:~# openssl ciphers -v 'EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:
EECDH+aRSA+SHA256:EECDH:+CAMELLIA128:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:
!EXP:!PSK:!DSS:!RC4:!SEED:!IDEA:!ECDSA:kEDH:CAMELLIA128-SHA:AES128-SHA'
DHE-RSA-AES256-GCM-SHA384   TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(256)   Mac=AEAD
DHE-RSA-AES256-SHA256       TLSv1.2 Kx=DH       Au=RSA  Enc=AES(256)      Mac=SHA256
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(256)   Mac=AEAD
ECDHE-RSA-AES256-SHA384     TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(256)      Mac=SHA384
DHE-RSA-AES128-GCM-SHA256   TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(128)   Mac=AEAD
DHE-RSA-AES128-SHA256       TLSv1.2 Kx=DH       Au=RSA  Enc=AES(128)      Mac=SHA256
ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(128)   Mac=AEAD
ECDHE-RSA-AES128-SHA256     TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(128)      Mac=SHA256
DHE-RSA-CAMELLIA256-SHA     SSLv3   Kx=DH       Au=RSA  Enc=Camellia(256) Mac=SHA1
DHE-RSA-AES256-SHA          SSLv3   Kx=DH       Au=RSA  Enc=AES(256)      Mac=SHA1
ECDHE-RSA-AES256-SHA        SSLv3   Kx=ECDH     Au=RSA  Enc=AES(256)      Mac=SHA1
DHE-RSA-CAMELLIA128-SHA     SSLv3   Kx=DH       Au=RSA  Enc=Camellia(128) Mac=SHA1
DHE-RSA-AES128-SHA          SSLv3   Kx=DH       Au=RSA  Enc=AES(128)      Mac=SHA1
ECDHE-RSA-AES128-SHA        SSLv3   Kx=ECDH     Au=RSA  Enc=AES(128)      Mac=SHA1
CAMELLIA128-SHA             SSLv3   Kx=RSA      Au=RSA  Enc=Camellia(128) Mac=SHA1
AES128-SHA                  SSLv3   Kx=RSA      Au=RSA  Enc=AES(128)      Mac=SHA1

root@Sec-NS2:~# openssl version
OpenSSL 1.0.1k 8 Jan 2015
root@Sec-NS2:~# cat /etc/debian_version
8.2
```

Weitere Diskussionen[44] mit Teilnehmern der BetterCrypto-Mailingliste führen zur Erkenntnis, dass die Reihenfolge der Cipher bei Verwendung aktueller OpenSSL-Versionen nicht jener entspricht, die vor etwa einem Jahr bei Erstellung des CipherString-B angedacht war.

Eine CipherString-Diskussion die für zahlreichen Plattformen und OpenSSL-Versionen zu einem soliden Ergebnis führt könnte einige Zeit in Anspruch nehmen, auf eine solche kann daher aktuell für dieses Demo-Setup nicht gewartet werden.

[43] http://lists.cert.at/pipermail/ach/2015-November/001983.html

[44] http://lists.cert.at/pipermail/ach/2015-November/001987.html

Gegenüber der in [BetterCrypo, Kapitel 3.2.3] vorgeschlagenen CipherSuite wurden daher eigenständig einige Veränderungen vorgenommen, um im Vergleich zum BetterCrypto-Vorschlag noch zusätzlich folgendes zu erreichen:

- Performance: Die schnelleren Ephemeral-ECDH-Key-Agreement-Varianten sollen den langsameren Ephemeral-DH-Varianten gegenüber bevorzugt werden. ECDHE schneidet in der Handshake-Simulation auf Server-Seite um das 2,4-fache performanter ab, als DHE (angegebene Werte sind die benötigte Zeit in Sekunden für 1000 Handshakes).

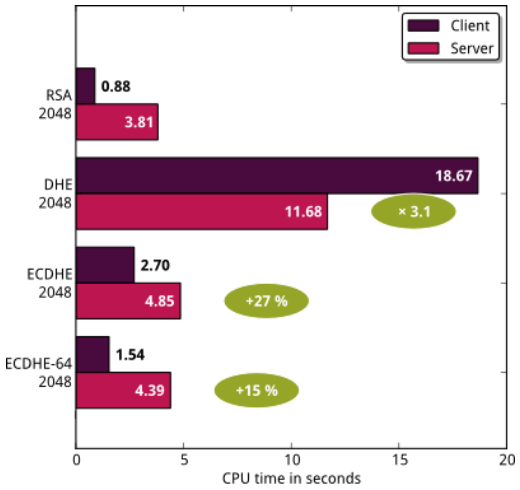


Abbildung 44: Performance – Zeitdauer für 1000 Handshakes - Quelle: [VB:TLS]

- Performance: AES-128 soll gegenüber AES-256 vorgereiht werden, hierorts wird die Meinung vertreten AES256 bringt keine praxisrelevante zusätzliche Sicherheit. AES128 ist auf der getesteten Zielplattform (Single-Core Azure-Cloud Maschine, OpenSSL 1.0.1k auf Debian 8.2) ca. 40% schneller als AES256:

```
root@Sec-NS2:~# openssl speed -evp aes-128-cbc aes-256-cbc
…
OpenSSL 1.0.1k 8 Jan 2015
…
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
aes-256 cbc      19209.65k    20422.18k    20565.70k    52596.02k    53019.07k
aes-128-cbc      25584.93k    28803.69k    29257.89k    74734.01k    76303.41k
```

Performance:	+33%	+41%	+42%	+42%	+44%

- Entfernung von Camellia – alle relevanten Clients ziehen AES ohnehin vor, es würde somit nicht zur Anwendung kommen, es existiert aufgrund der Popularität von AES und dem Nischen-Dasein von Camellia außerdem sehr wenig CryptoAnalyse zum Camellia-Cipher.
- Mit der Vorreihung von ECDH wird auch Java 1.7 unterstützt, die von BetterCrypto.org an den Beginn der CipherSuite gereihten DH-Varianten verwenden DH-Parameter >1024bit was Java 1.7 jedoch nicht unterstützt, der TLS-Handshake käme daher nicht zustande.
- Der Cipherstring sollte deutlich kürzer werden, dies erleichtert die Wartung und Übersicht.
- Es sollte zu keinen unerwünschten Effekten bei der Verwendung des Cipher-Strings auf älteren oder neueren OpenSSL-Versionen kommen.

Bei der Erstellung eines Cipherstrings sollten außerdem dringend die Hinweise in [IR-TLS] beachtet werden. Die Beurteilung des Resultats mittels Qualys SSL Labs[45] Online-Test (siehe Abschnitt 5.3.1) sollte mit Kenntnis des zugehörigen erläuternden Dokumentes [IR-SrvRating] erfolgen, welches beschreibt was konkret geprüft und warum mit welchem Scoring belohnt beziehungsweise bestraft wird. Die der OpenSSL-Dokumentation[46] entnommenen Syntax-Regeln sind nicht unbedingt intuitiv, wir rufen uns diese daher an dieser Stelle in Erinnerung:

- *If "!" Is used then the ciphers are permanently deleted from the list. The ciphers deleted can never reappear in the list even if they are explicitly stated.*
- *If "-" is used then the ciphers are deleted from the list, but some or all of the ciphers can be added again by later options.*
- *If "+" is used then the ciphers are moved to the end of the list. This option doesn't add any new ciphers it just moves matching existing ones.*

Folgende Überlegungen wurden angestellt um einen neuen Cipher-String zu formulieren:

Schritt 1: Alle Cipher deaktivieren, sodass die Cipher-Liste leer ist: `-ALL`

Schritt 2: Nur die Cipher hinzufügen, die aktuell empfohlen werden können. Es kommt ein RSA-Zertifikat zur Anwendung, die Authentifizierung hat daher jedenfalls mit RSA zu erfolgen. Da aktuell als Block-Chiffre nur AES128 und AES256 zur Anwendung kommt beschränken sich die geeigneten Chiffren auf:

- `ECDH+aRSA+AES` fügt auf einem ECDH Key-Agreement basierende Cipher hinzu:
  `ECDHE-RSA-AES256-GCM-SHA384`
  `ECDHE-RSA-AES256-SHA384`
  `ECDHE-RSA-AES256-SHA`
  `ECDHE-RSA-AES128-GCM-SHA256`
  `ECDHE-RSA-AES128-SHA256`
  `ECDHE-RSA-AES128-SHA`
- `DH+aRSA+AES` fügt auf einem DH Key-Agreement basierende Cipher hinzu:
  `DHE-RSA-AES256-GCM-SHA384`
  `DHE-RSA-AES256-SHA256`
  `DHE-RSA-AES256-SHA`
  `DHE-RSA-AES128-GCM-SHA256`
  `DHE-RSA-AES128-SHA256`
  `DHE-RSA-AES128-SHA`
- `aRSA+kRSA+AES` fügt zwecks Kompatibilität mit älteren Clients die kein Key-Agreement mittels PFS tauglichen ECDH und DH Varianten unterstützen noch Varianten mit RSA-Key-Agreement hinzu:
  `AES256-GCM-SHA384`
  `AES256-SHA256`
  `AES256-SHA`
  `AES128-GCM-SHA256`
  `AES128-SHA256`
  `AES128-SHA`

Schritt 3: Für passende Sortierung sorgen, `+AES256` rückt die AES256-Varianten ans Ende.

Das Resultat der zur Anwendung kommenden Cipher kann stark von der verwendeten OpenSSL-Version und Distribution abhängen! Ein Cipher-String darf daher nicht ungeprüft aus Empfehlungen übernommen werden, sondern muss individuell auf der zum Einsatz kommenden Distribution mittels des `openssl ciphers` Kommando geprüft und nötigenfalls modifiziert werden!

[45] https://www.ssllabs.com/ssltest/

[46] https://www.openssl.org/docs/manmaster/apps/ciphers.html

Resultat: `'-ALL:ECDH+aRSA+AES:DH+aRSA+AES:aRSA+kRSA+AES:+AES256'`

Der erstellte Cipher-String wird wie folgt auf der Zielplattform (OpenSSL 1.0.1k 8 Jan 2015 auf Debian 8.2) geprüft:

```
root@Sec-NS2:~# openssl ciphers -v '-ALL:ECDH+aRSA+AES:DH+aRSA+AES:aRSA+kRSA+AES:+AES256'
ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=RSA  Enc=AESGCM(128) Mac=AEAD
ECDHE-RSA-AES128-SHA256     TLSv1.2 Kx=ECDH Au=RSA  Enc=AES(128)    Mac=SHA256
ECDHE-RSA-AES128-SHA        SSLv3   Kx=ECDH Au=RSA  Enc=AES(128)    Mac=SHA1
DHE-RSA-AES128-GCM-SHA256   TLSv1.2 Kx=DH   Au=RSA  Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES128-SHA256       TLSv1.2 Kx=DH   Au=RSA  Enc=AES(128)    Mac=SHA256
DHE-RSA-AES128-SHA          SSLv3   Kx=DH   Au=RSA  Enc=AES(128)    Mac=SHA1
AES128-GCM-SHA256           TLSv1.2 Kx=RSA  Au=RSA  Enc=AESGCM(128) Mac=AEAD
AES128-SHA256               TLSv1.2 Kx=RSA  Au=RSA  Enc=AES(128)    Mac=SHA256
AES128-SHA                  SSLv3   Kx=RSA  Au=RSA  Enc=AES(128)    Mac=SHA1
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=RSA  Enc=AESGCM(256) Mac=AEAD
ECDHE-RSA-AES256-SHA384     TLSv1.2 Kx=ECDH Au=RSA  Enc=AES(256)    Mac=SHA384
ECDHE-RSA-AES256-SHA        SSLv3   Kx=ECDH Au=RSA  Enc=AES(256)    Mac=SHA1
DHE-RSA-AES256-GCM-SHA384   TLSv1.2 Kx=DH   Au=RSA  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-SHA256       TLSv1.2 Kx=DH   Au=RSA  Enc=AES(256)    Mac=SHA256
DHE-RSA-AES256-SHA          SSLv3   Kx=DH   Au=RSA  Enc=AES(256)    Mac=SHA1
AES256-GCM-SHA384           TLSv1.2 Kx=RSA  Au=RSA  Enc=AESGCM(256) Mac=AEAD
AES256-SHA256               TLSv1.2 Kx=RSA  Au=RSA  Enc=AES(256)    Mac=SHA256
AES256-SHA                  SSLv3   Kx=RSA  Au=RSA  Enc=AES(256)    Mac=SHA1

ECDH = Ephemeral Elliptic curve Diffie-Hellman
EDH  = Ephemeral Diffie-Hellman
GCM  = Galois-Counter-Mode realisiert AEAD = Authenticated Encryption with Associated Data
```

Anmerkung: „SSLv3“ bedeutet hier lediglich, dass diese Cipher-Suite ab SSLv3 unterstützt wurde, es bedeutet **nicht**, dass am Server noch SSLv3 aktiviert wäre!

Die aus diesem CipherString resultierenden Cipher-Suites wurden auch mittels Qualys SSL Labs Online-Test geprüft (Detailergebnisse siehe Abschnitt 5.3.1). Es ergibt sich damit folgende akzeptable Client-Kompatibilität:

Android 4.3	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA FS	128
Android 4.4.2	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 FS	128
Android 5.0.0	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 FS	128
Chrome 43 / OS X R	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 FS	128
Firefox 31.3.0 ESR / Win 7	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 FS	128
Firefox 39 / OS X	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 FS	128
IE 6 / XP		Protocol or cipher suite mismatch	Fail[3]
IE 7 / Vista	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA FS	128
IE 8 / XP		Incorrect certificate because this client doesn't support SNI	Fail[2]
IE 8-10 / Win 7 R	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA FS	128
IE 11 / Win 7 R	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128
IE 11 / Win 8.1	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128
IE 10 / Win Phone 8.0	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA FS	128
IE 11 / Win Phone 8.1	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128
IE 11 / Win Phone 8.1 Update	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128
Edge 12 / Win 10 (Build 10130)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 FS	128
Java 6u45		Incorrect certificate because this client doesn't support SNI	Fail[2]
Java 7u25	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA FS	128
Java 8u31	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 FS	128
Safari 5.1.9 / OS X 10.6.8	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA FS	128
Safari 6 / iOS 6.0.1	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128
Safari 6.0.4 / OS X 10.8.4	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA FS	128
Safari 7 / iOS 7.1	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128
Safari 7 / OS X 10.9	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128
Safari 8 / iOS 8.4	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128
Safari 8 / OS X 10.10	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 FS	128

FS … Forward Secrecy

5.1.3. Weitere mögliche Cipher-Strings

Diskussion des im vorherigen Abschnitt 5.1.3 verwendeten Cipher-Strings auf der BetterCrypto-Mailingliste führte zu weiteren Anregungen und Hinweisen. Diese wurden für die Umsetzung dieser Demo nachträglich (aus zeitlichen Gründen) jedoch nicht mehr berücksichtigt, sollten an dieser Stelle jedoch noch Erwähnung finden:

Der verwendete Cipher-String ging davon aus, dass ein Client AES128 unterstützt oder dies (bewusst) deaktiviert hat. Ist auf einem Client AES128 deaktiviert so sollen die langsameren AES256 Varianten zur Anwendung kommen.

Möglicherweise gibt es jedoch Situationen, in denen ein Client zwar kein `ECDHE-RSA-AES128-GCM-SHA256` aber `AES128-SHA` und diverse `AES256` Cipher unterstützen würde. Gemäß verwendeter Cipher-Suite würde dann jedoch dem nicht PFS fähigen `AES128-SHA` Chipher der Vorzug gegeben werden. Die Cipher mit RSA-Key-Agreement sollten daher zurückgereiht werden.

Außerdem sollten Cipher die noch mit SHA1 arbeiten zurückgereiht werden.

Es kann außerdem davon ausgegangen werden, dass Clients die SHA2 unterstützen auch ein PFS Key-Agreement mit DH oder ECDH beherrschen. Das Anbieten von SHA2-Chiffren mit RSA-Key-Agreement ist daher in der Praxis irrelevant und kann entfernt werden.

Dies resultiert in folgendem neuen Cipherstring:

Cipherstring: `'-ALL:kEECDH+aRSA+AES:kEDH+aRSA+AES:+AES256:+SHA1:aRSA+kRSA+AES+SHA1'`

Welcher folgende Cipher zur Verfügung stellt:

- 4 x AES128-Chiffren mit PFS die kein SHA1 mehr beinhalten
- 4 x AES256-Chiffren mit PFS die kein SHA1 mehr beinhalten
  (wobei jeweils dem schnelleren ECDH der Vorzug gegeben wird)
- 4 x Chiffren mit SHA1, für ältere Clients die kein SHA2 beherrschen.
- 2 x Chiffren mit RSA-Key-Agreement, somit ohne PFS – für Clients die weder SHA2 noch die Verwendung von DH mit akzeptablen DH-Parametern beherrschen.

```
root@Sec-NS2:~#
openssl ciphers -v '-ALL:kEECDH+aRSA+AES:kEDH+aRSA+AES:+AES256:+SHA1:aRSA+kRSA+AES+SHA1'
ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH    Au=RSA  Enc=AESGCM(128) Mac=AEAD
ECDHE-RSA-AES128-SHA256     TLSv1.2 Kx=ECDH    Au=RSA  Enc=AES(128)    Mac=SHA256
DHE-RSA-AES128-GCM-SHA256   TLSv1.2 Kx=DH      Au=RSA  Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES128-SHA256       TLSv1.2 Kx=DH      Au=RSA  Enc=AES(128)    Mac=SHA256

ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH    Au=RSA  Enc=AESGCM(256) Mac=AEAD
ECDHE-RSA-AES256-SHA384     TLSv1.2 Kx=ECDH    Au=RSA  Enc=AES(256)    Mac=SHA384
DHE-RSA-AES256-GCM-SHA384   TLSv1.2 Kx=DH      Au=RSA  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-SHA256       TLSv1.2 Kx=DH      Au=RSA  Enc=AES(256)    Mac=SHA256

ECDHE-RSA-AES128-SHA        SSLv3 Kx=ECDH      Au=RSA  Enc=AES(128)    Mac=SHA1
DHE-RSA-AES128-SHA          SSLv3 Kx=DH        Au=RSA  Enc=AES(128)    Mac=SHA1
ECDHE-RSA-AES256-SHA        SSLv3 Kx=ECDH      Au=RSA  Enc=AES(256)    Mac=SHA1
DHE-RSA-AES256-SHA          SSLv3 Kx=DH        Au=RSA  Enc=AES(256)    Mac=SHA1

AES256-SHA                  SSLv3 Kx=RSA       Au=RSA  Enc=AES(256)    Mac=SHA1
AES128-SHA                  SSLv3 Kx=RSA       Au=RSA  Enc=AES(128)    Mac=SHA1

ECDH = Ephemeral Elliptic curve Diffie-Hellman
EDH  = Ephemeral Diffie-Hellman
GCM  = Galois-Counter-Mode realisiert AEAD = Authenticated Encryption with Associated Data
```

5.1.4. Zertifikate

Die Verwendung der Zertifikate erfolgt unter Apache durch Angebe von:

- `SSLCertificateKeyFile` = Private-Key-File (als Benutzer „root“), siehe Abschnitt 3.1.1
- `SSLCertificateFile` = Zertifikat, siehe Abschnitt 3.1.3
- `SSLCertificateChainFile` = Intermediate-Zertifikate (Chain), siehe Abschnitt 3.1.4

Ein Pinning der Zertifikate mittels DNS-based Authentication of Named Entities (DANE) wurde bereits in Kapitel 3.2 durchgeführt und bedarf am WebServer keiner weiteren Anpassung. Ein Test von DANE erfolgt in Abschnitt 5.3.3 und 5.3.4.

5.1.5. OCSP-Stapeling

Mittels Online Certificate Status Protocol (OCSP) können WebBrowser den Status von X.509 Zertifikaten prüfen (Zurückziehen kompromittierter Zertifikate). Der Vorgang ist sehr zeitaufwändig und skaliert schlecht – Certificate Authorities halten hierfür Server bereit, wenn nun aber jeder WebBrowser bei jeder neuen TLS-Verbindung die zum Zertifikat gehörige CA kontaktiert, entsteht bei den CA’s ein Bottleneck, die Performance der Zugriffe leidet.

OCSP-Stapeling (siehe [RFC-6066, Kapitel 8]) ermöglicht es, dass im Zuge des TLS-Handshakes bereits signierte und mit kryptographischen Zeitstempeln versehene OCSP-Responses der CA an den Client mit ausgeliefert werden. Der Client muss daher die CA nicht separat kontaktieren, sondern kann sich bereits im Zuge des TLS-Handshakes (siehe Abbildung 45) vergewissern, dass das Zertifikat noch nicht als zurückgezogen von der CA markiert wurde [IR-TLS, Kapitel 3.5].

Die Aktivierung von OCSP-Stapeling erfordert zusätzlich zum Eintrag `SSLUseStapling on` in der vHost-Konfiguration noch eine Ergänzung in der `/etc/apache2/mods-enabled/ssl.conf`

```
…
# 128k high-performance cyclic buffer als OCSP-Stapeling Cache einrichten
SSLStaplingCache shmcb:${APACHE_RUN_DIR}/stapling_cache(128000)
…
```

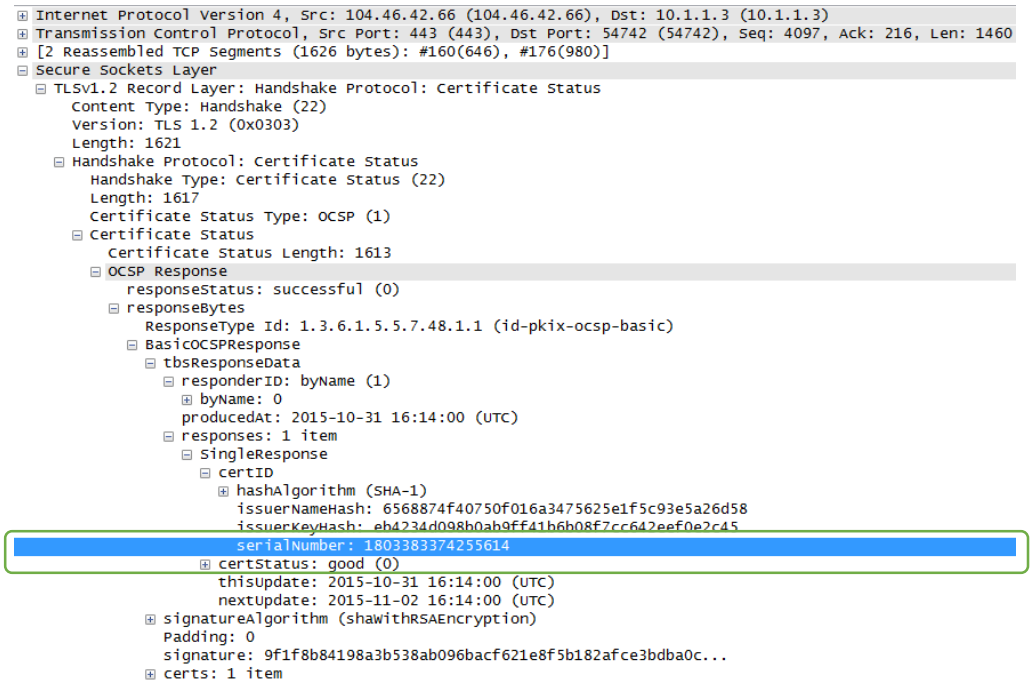


Abbildung 45: Wireshark-Dump: TLS-Handshake, OCSP-Stapling liefert Zertifikats-Status

5.1.6. HTTP Strict Transport Security (HSTS)

Beim Zugriff auf eine WebSite wird in der Regel zuerst eine HTTP-URL abgerufen, die einen Redirect auf HTTPS auslöst. Ein unaufmerksamer Benutzer wird kaum bemerken, wenn eine WebSite plötzlich über HTTP zugreifbar ist und/oder Teil-Inhalte der Site über Klartext-HTTP nachgeladen werden. Es existieren zahlreiche Angriffsmöglichkeiten durch Man-in-the-Middle Downgrade-Attacken, z.B. SSL-Stripping.

Mit [RFC-6797] wurde HTTP Strict Transport Security (HSTS) eingeführt, ein Verfahren das es ermöglicht dem Webbrowser ein sogenanntes Super-Cookie zu setzen welches definiert, dass sämtliche Zugriffe auf diese Domain (und allenfalls auch SubDomains) nur mehr über HTTPS erfolgen dürfen. Eine Zeitangabe definiert, wie lange (ab Zugriff) diese Policy im Browser gespeichert bleiben soll. Der Browser lässt – nachdem die Domain im Browser als mit HSTS geschützt markiert wurde – keine Klartext-HTTP Zugriffe mehr zu, alle Klartext-URLs werden automatisch mit HTTPS durchgeführt.

Zur Inbetriebnahme am Apache 2.4 Server muss das `mod_headers` Modul[47] in Betrieb genommen und nun ergänzend noch gemäß [RFC-6797, Kapitel 2.1] wie folgt konfiguriert und Apache neu gestartet werden:

```
root@Sec-NS2:/etc/apache2/mods-available# a2enmod headers

root@Sec-NS2:/etc/apache2/sites-available# vi mail.it-sec.ovh_ssl.conf
…
# 1/2 Jahr HSTS header (60*60*24*182=15724800sek), inklusive Subdomains
Header always set Strict-Transport-Security "max-age=15724800 ; includeSubDomains"
…

root@Sec-NS2:/etc/apache2/sites-available# apache2ctl graceful
```

Der gesetzte Header besagt, dass für die nächsten 182 Tage der Browser sich an die „Strict-Transport-Security Policy“ halten soll, also keine Klartext-HTTP-Zugriffe auf diese Domain inklusive Subdomains zulassen soll.

Unter Google Chrome kann der Status von Strict Transport Security für eine Domain durch Aufruf der Chrome-Internen URL chrome://net-internals/#hsts geprüft bzw. das gespeicherte HSTS-set auch bei Bedarf manuell gelöscht werden. HSTS wird von allen aktuellen Browser-Versionen (Internet Explorer 11, Microsoft Edge, Firefox, Chrome, Safari, Opera, …)[48] unterstützt.

Die Verwendung von HSTS wird auch vom SSLLabs.com SSL-WebServer-Test honoriert, dieses und weitere Test-Ergebnisse sind in Kapitel 5.3 zu finden.

[47] http://httpd.apache.org/docs/2.4/mod/mod_headers.html#header

[48] http://caniuse.com/#feat=stricttransportsecurity

5.1.7. HTTP Public Key Pinning (HPKP) als Ergänzung zu DANE

Die Sicherheit einer HTTPS Verbindung (TLS gesichertes HTTP) beruht zum einen auf sicherer Kryptographie, zum anderen aber auf der Annahme, dass man mit der richtigen Gegenstelle verbunden ist. Diese Annahme wird mittels Zertifikaten validiert, eine Zertifikatsprüfung resultiert in der Erkenntnis, dass die Gegenstelle nachweisen kann ein gültiges Zertifikat einer vertrauenswürdigen PKI (i.d.R. ein kommerzieller Zertifikatsanbieter) Ihr Eigen zu nennen. Das Problem ist jedoch, dass es um die Vertrauenswürdigkeit der hunderten im Webbrowser bzw. im Betriebssystem verankerten Stamm- und Intermediate-Zertifikate möglicherweise nicht ganz so gut bestellt ist, wie man dies gerne erwarten würde. Teils können sich die CA's jedoch staatlicher Einflussnahme nicht gänzlich entziehen (Stichwort: Lawful Interception in den USA[49], Iran[50], China[51], Indien[52], …) oder werden gehackt (z.B. DigiNotar[53], …).

Als Betreiber einer europäischen Website möchte man daher, dass die Browser der Benutzer beim Besuch der eigenen Website nicht jedem auf den passenden Domain-Namen ausgestellten Zertifikat aus einer beliebigen getrusteten CA aus China oder den USA vertrauen, sondern lediglich dem eigenen verwendeten Zertifikat bzw. den durch den Website-Betreiber verwendeten CA's.

Public Key Pinning Extension for HTTP ermöglicht es, beim Besuch einer Website dem Browser des Besuchers mitzuteilen, welche Zertifikate bei den nächsten Besuchen als erlaubt angesehen werden sollen. Das System beruht also auf einem „Trust On First Use" (TOFU) Konzept [RFC-7469]. Fällt man schon beim allerersten Besuch einer Website einem Man-in-the-Middle-Angriff zum Opfer, kann auch HPKP hierfür somit keinen Schutz gewährleisten. Weiterführende und sehr umfangreiche Informationen zum Thema Certificate and Public Key Pinning können [OWASP-PKP] entnommen werden. Alternativen zu HPKP wären z.B. DNSSEC mit TLSA-Records (wie bereits im Kapitel 3.2 zur Absicherung des SMTP-Servers verwendet), jedoch fehlt hierfür aktuell die Browser-Unterstützung, und generell ist die DNSSec Unterstützung auf Endgeräten und durch die Internet-Service-Provider nicht ausreichend gegeben, sodass dieser alternative Ansatz vermutlich auch nicht allzu bald zur Anwendung kommen wird.

Der WebServer-Administrator kann mittels des `Public-Key-Pins` HTTP-Headers den Browsern mitteilen, welche Zertifikate bzw. welche CA's zugelassen sind, wie lange (Zeitspanne) das Pinning wirken soll, und wer im Falle einer Unregelmäßigkeit benachrichtigt werden soll. Die Benachrichtigung ist optional, sie ermöglicht es dem Browser einen Fehlerbericht in einem definierten JSON-Format [RFC-7469, Kapitel 3] an eine konfigurierbare URL per HTTP-Post zu senden. Es ist auch möglich die Option in einem „Report-Only" Modus zu betreiben.

Aktuelle Versionen von Mozilla Firefox, Google Chrome und Opera unterstützen HPKP und bewahren somit den Benutzer vor eine ungewollten Interception oder einem ungewollten „Entführen" einer Website zu einem anderen Ziel-Server. Andere Browser[54] wie Apple Safari

[49] http://heise.de/-963857

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

[51] http://heise.de/-2595239, http://heise.de/-2583414

[52] http://heise.de/-2252544

[53] http://heise.de/-1741726, http://heise.de/-1340621,

[54] http://caniuse.com/#feat=publickeypinning

oder Microsoft Interned Explorer oder Edge hinken hier hinterher und bieten mit Stand Oktober 2015 noch keine Unterstützung für HPKP.

Anti-Malware Lösungen die lokal am Endgerät oder auf einem vorgelagerten Proxy-Server in der Lage sind SSL-Traffic zwecks Content-Inspection zu intercepten würden bei Verwendung mit HPKP fähigen Browsern schlagartig zu einem massiven Problem führen. Um weiterhin firmeninterne oder vom Benutzer gewollte SSL-Interception zu ermöglichen sieht [RFC-7469, Kapitel 2.6] vor, dass Zertifikats-Pinning für vom Benutzer definierte CAs nicht vorgenommen werden soll. Kommt SSL-Interception zum Zwecke des firmeninternen Content-Scannings bzw. für eine Anti-Malware-Lösung zum Einsatz, muss hierfür ein geeignetes CA-Stammzertifikat auf den Endgeräten bzw. im Browser installiert werden – sowohl Chrome, als auch Firefox haben HPKP gemäß dieser Empfehlung imlementiert, sind also weiterhin mit SSL-Interception-Lösungen kompatibel.

Eine Übersicht über diesbezügliche Fähigkeiten zahlreicher Browser und Plattformen ist auf https://projects.dm.id.lv/Public-Key-Pins_test zu finden, auch eine HPKP-Test-URL wird bereitgestellt.

Abbildung 47 demonstriert die Ablehnung eines eigentlich gültigen Comodo-Zertifikats aufgrund der HPKP-Restriktion durch Google Chrome, Abbildung 46 zeigt selbiges bei Verwendung von Mozilla Firefox.

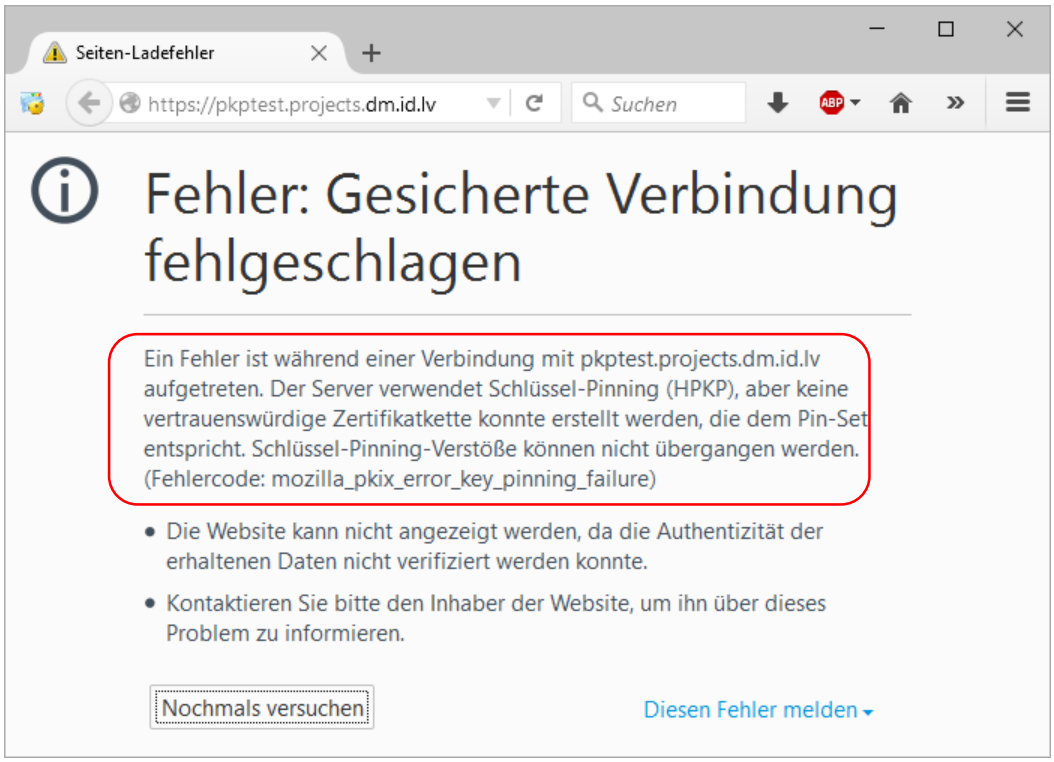


Abbildung 46: Mozilla Firefox 41: HTTP Public Key Pinning - abgelehntes Zertifikat

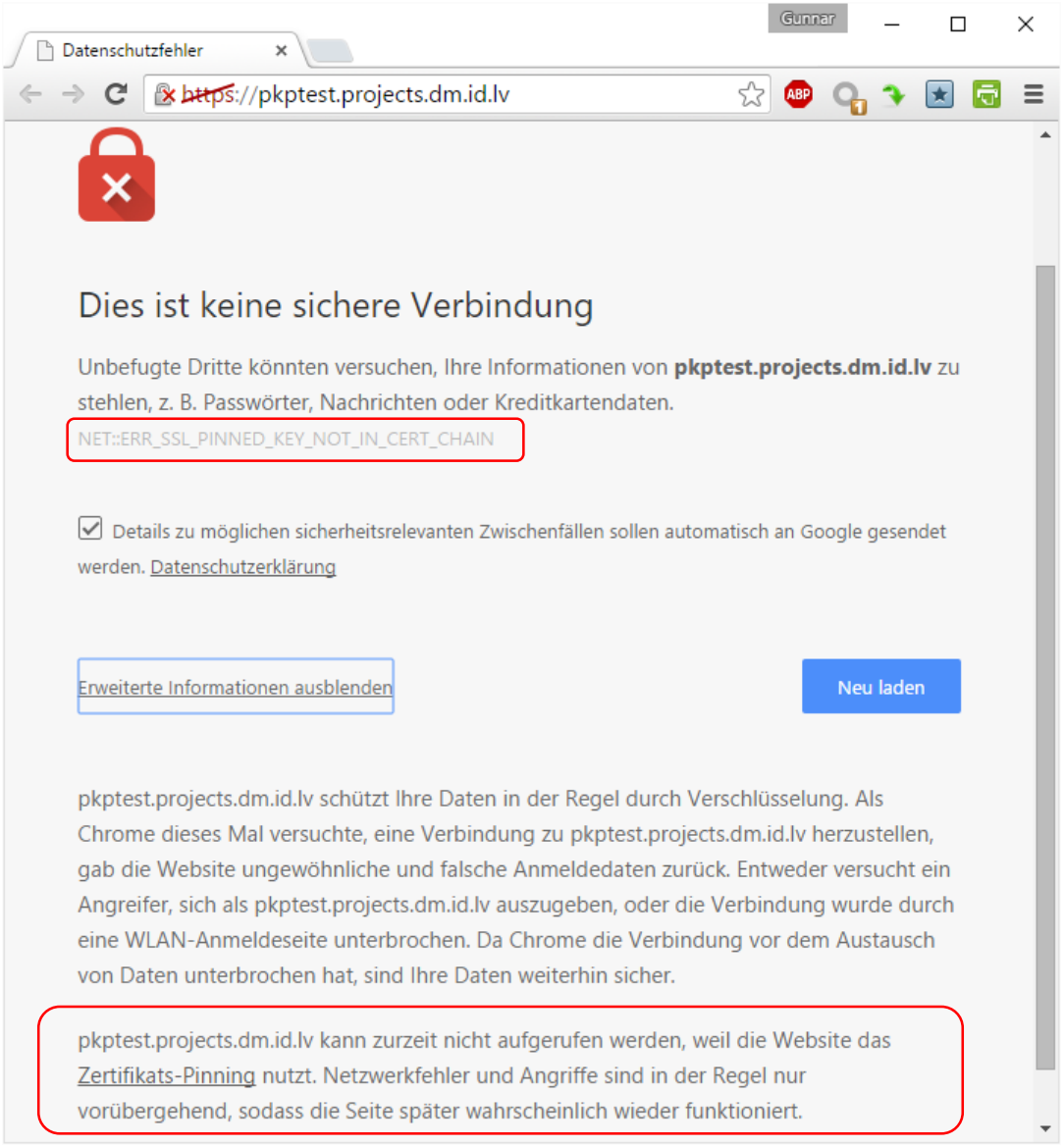


Abbildung 47: Google Chrome 46: HTTP Public Key Pinning - abgelehntes Zertifikat

Der Vollständigkeit halbe sei noch darauf hingewiesen, dass die Browser-Hersteller neben HPKP teils auch noch andere Verfahren zum Zertifikats-Pinning implementieren. Für zahlreiche populäre URLs wird auf Wunsch der Website-Betreiber ein Pinning welches im Browser fest verankert ist implementiert – in der Regel sind solche Services jedoch nur den Branchen-Größen wie Facebook, Google, Apple, Microsoft, u.a. vorbehalten, derartige proprietäre Lösungen skalieren nicht und sind daher nicht in der Lage eine breitflächig einsetzbare Lösung bereitzustellen.

HPKP ist als Ergänzung zum in Abschnitt 3.2.1 bereits durchgeführten DANE/TLSA-Record zu sehen. Die beiden Lösungen funktionieren völlig unabhängig voneinander, während DANE/TLSA bei Web-Browsern jedoch bislang nur mittels Plug-Ins[55] nachrüstbar ist, ist HPKP-Unterstützung bei zahlreichen Browsern heute bereits standardmäßig verfügbar.

[55] https://www.dnssec-validator.cz/

Zur Inbetriebnahme am Apache 2.4 Server muss (wie bereits in Abschnitt 5.1.6 im Zuge von HSTS erläutert) das `mod_headers` Modul[56] in Betrieb genommen und nun ergänzend noch gemäß [RFC-7469, Kapitel 2.1] wie folgt konfiguriert und Apache neu gestartet werden:

```
root@Sec-NS2:/etc/apache2/mods-available# a2enmod headers

root@Sec-NS2:/etc/apache2/sites-available# vi mail.it-sec.ovh_ssl.conf
…
# HTTP Public Key Pinning für Zertifikat (HPKP), 90 Tage (60*60*24*90=7776000)
# mail.it-sec.ovh.crt PubKey = pG3WsstDsfMkRdF3hBClXRKYxxKUJIOu8DwabG8MFrU=
# StartCom Certification Authority: "5C8kvU039KouVrl52D0eZSGf4Onjo4Khs8tmyTlV3nU="
# zweiter Backup-Key: "i89r5g0BAe/9ubBIuTxrW5phVO8lv7cBvXxK/evFhXw="
Header always set Public-Key-Pins    ↵
"pin-sha256=\"pG3WsstDsfMkRdF3hBClXRKYxxKUJIOu8DwabG8MFrU=\";     ↵
 pin-sha256=\"5C8kvU039KouVrl52D0eZSGf4Onjo4Khs8tmyTlV3nU=\";     ↵
 pin-sha256=\"i89r5g0BAe/9ubBIuTxrW5phVO8lv7cBvXxK/evFhXw=\";     ↵
 max-age=7776000;
 report-uri=\"https://report-uri.io/report/866c4f253035d817119b9401f6116434\";"
…
root@Sec-NS2:/etc/apache2/sites-available# apache2ctl graceful
```

Diese verwendete Konfiguration basiert auf folgenden Überlegungen:

Es müssen SHA256-Hashes der Public-Keys der möglichen Zertifikate oder CAs im `Public-Key-Pins` HTTP-Header hinterlegt werden. Der Hash eines Public-Keys ist wie folgt mittels OpenSSL ermittelbar:

```
root@Sec-NS2:/etc/ssl/private# openssl x509 -in mail.it-sec.ovh.crt -pubkey -noout |    ↵
 openssl rsa -pubin -outform der | openssl dgst -sha256 -binary | openssl enc -base64
writing RSA key
pG3WsstDsfMkRdF3hBClXRKYxxKUJIOu8DwabG8MFrU=

root@Sec-NS2:/etc/ssl/private# openssl rsa -in it-sec.ovh.backup.key -pubout -outform der↵
 | openssl dgst -sha256 -binary | openssl enc -base64
writing RSA key
i89r5g0BAe/9ubBIuTxrW5phVO8lv7cBvXxK/evFhXw=
```

Erläuterung: Public Key aus X509-Zertifikat extrahieren, den Base64-kodierten RSA-Public-Key in binäre DER-Kodierung umwandeln, davon nun einen binär kodierten SHA256-Hash erzeugen und diesen Base64-kodiert ausgeben.

Um die Schritte nicht manuell auf der Konsole durchzuführen, kann alternativ auch ein WebService (z.B. https://projects.dm.id.lv/s/pkp-online/calculator.html) für diesen Schritt herangezogen werden. Noch komfortabler ist die Verwendung des WebService https://report-uri.io/home/pkp_hash/, welches die aktuell verwendete Zertifikats-Kette live vom Webserver abruft und die nötigen Hashes liefert (siehe Abbildung 48).

[56] http://httpd.apache.org/docs/2.4/mod/mod_headers.html#header

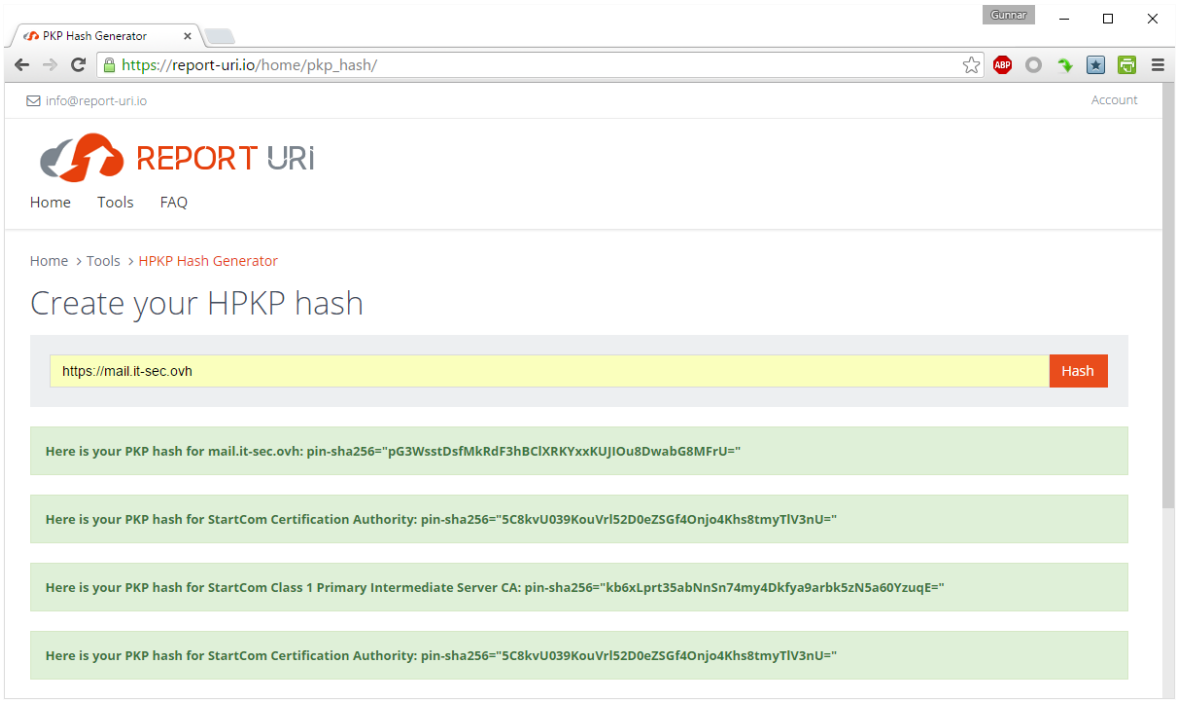


Abbildung 48: HPKP-Hash-Generierung mittels WebService https://report-uri.io

Um einen Zertifikatstausch zu ermöglichen, sind bereits bei der ersten Definition der HPKP-Settings zwingend die nötigen zukunftssicheren Überlegungen anzustellen. Die oben gewählte Konfiguration umfasst zwei Einträge, den Public-Key des SSL-Webserver-Zertifikats sowie das Stammzertifikat der verwendeten CA (StartCom startssl.com). Das Pinning des Zertifikats-Public-Keys ermöglicht die Verwendung eines beliebigen Zertifikats mit diesem Public-Key, also auch – sofern zukünftig nötig – die Verwendung eines Zertifikats aus einer anderen CA, solange das selbe RSA-Schlüsselpaar (d.h. gleicher CSR) erneut verwendet wird. Das Pinning auf die verwendete CA ermöglicht nun zusätzlich ein gänzlich neues Zertifikat (mit anderem RSA-Schlüsselpaar) zu verwenden, solange die Zertifikatsprüfung zu diesem CA-Zertifikat führt.

Es wäre außerdem sehr empfehlenswert, bereits jetzt einen weiteren CSR zu generieren und den daraus resultierenden PublicKey-Hash ebenfalls unmittelbar für zukünftige Notfälle in den HPKP-Header aufzunehmen. Die Verwendung eines aktuell nicht im Einsatz befindlichen Backup-PINs ist gemäß [RFC-7469, Kapitel 4.3] verpflichtend, so führt z.B. Google-Chrome das Pinning nicht durch, wenn nicht – wie im RFC vorgeschrieben – auch ein aktuell nicht verwendeter Hash eines weiteren Public-Keys im `Public-Key-Pins` Header aufscheint.

Um (optional) Fehlerberichte entgegenzunehmen wird ein WebService benötigt, welches die im JSON-Format als HTTP-Post übermittelten Berichte entgegennimmt. Ein solches Webservice ist mit einfachen Mitteln selbst betreibbar – ein paar Zeilen PHP-Script[57] reichen hierfür aus. Alternativ kann ein Dienst wie z.B. https://report-uri.io/ in Anspruch genommen werden, welcher eine grafische Übersicht über gemeldete Fehler bietet (Account kostenfrei).

[57] Demo-Script: https://blog.pregos.info/2015/02/23/http-public-key-pinning-hpkp-erklaerung-und-einrichtung/

Eine Analyse der HPKP-Header ist mittels https://report-uri.io/home/pkp_analyse/ möglich.

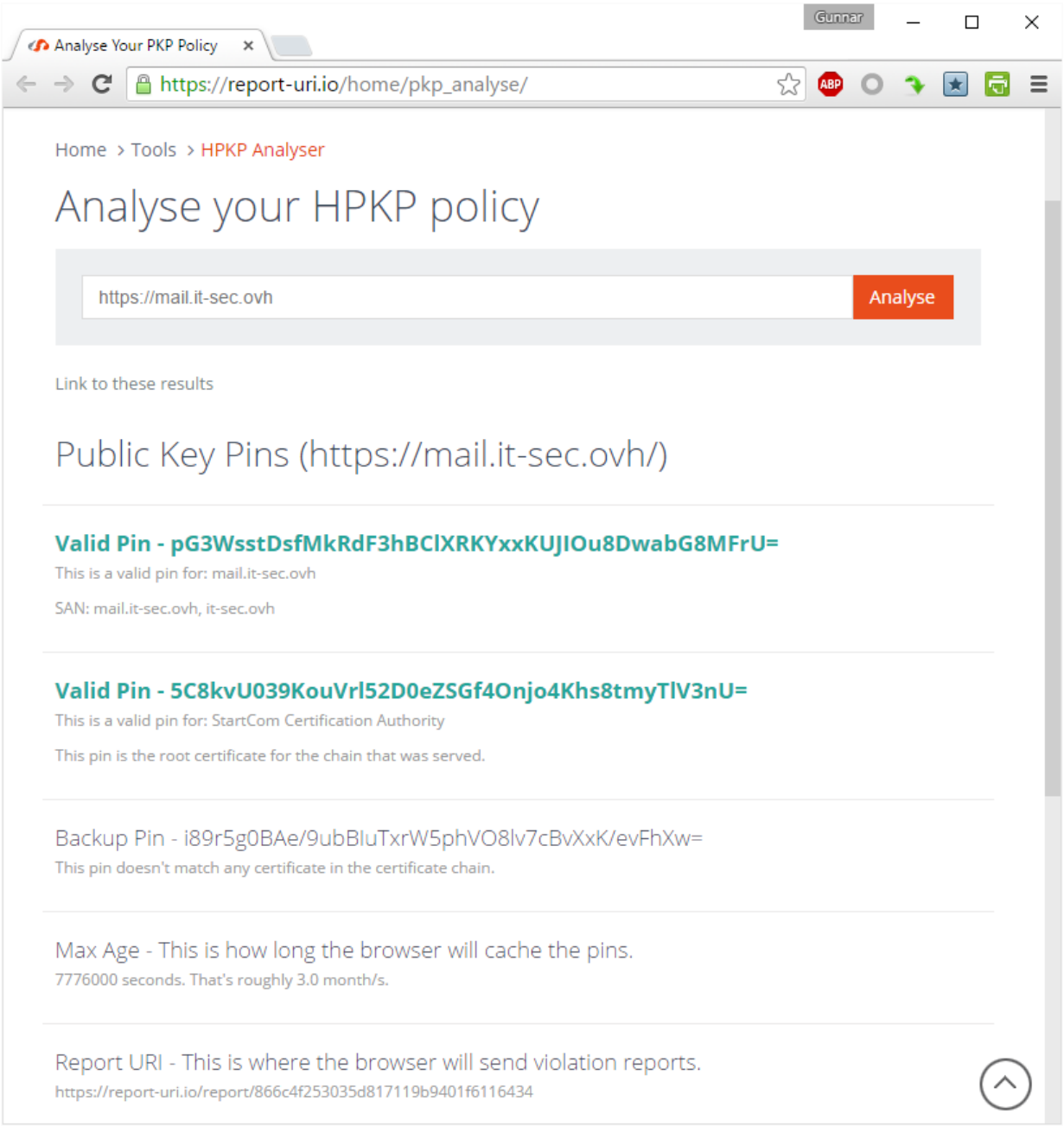


Abbildung 49: Analyse der HPKP Header mittels report-uri.io WebService

Die Verwendung von HPKP wird auch vom SSLLabs.com SSL-WebServer-Test honoriert, die Ergebnisse sind in Kapitel 5.3 zu finden.

Unter Google Chrome kann der Status von http Public Key Pinning (in Chrome abgekürzt mit PKP) für eine Domain durch Aufruf der Chrome-Internen URL chrome://net-internals/#hsts geprüft bzw. das gespeicherte „dynamic_pkp_...“-Set auch bei Bedarf manuell gelöscht werden.

5.2. RoundCube WebMail

Die Paketverwaltung von Debian 8 enthält kein RoundCube-WebMail System mehr, dieses muss daher aus den Sourcen von der WebSite selbst installiert werden.

Aktuell ist derzeit (Stand 09.10.2015) die Version 1.1.3: https://roundcube.net/download/

Die Installationsanleitung findet sich entweder in Form eines Textfiles im Paket sowie online: http://trac.roundcube.net/wiki/Howto_Install

5.2.1. Basis-Installation, Herstellung der Systemvoraussetzungen

Download und Installation des Complete-Packages:

```
root@Sec-NS2:/var/www# wget https://downloads.sourceforge.net/project/roundcubemail/ ↩
roundcubemail/1.1.3/roundcubemail-1.1.3-complete.tar.gz
root@Sec-NS2:/var/www# tar -xvpzf roundcubemail-1.1.3-complete.tar.gz
root@Sec-NS2:/var/www# rm roundcubemail-1.1.3-complete.tar.gz
root@Sec-NS2:/var/www# mv roundcubemail-1.1.3 mail.it-sec.ovh
root@Sec-NS2:/var/www# chown -R root:www-data mail.it-sec.ovh
```

Der Installer (Web-Oberfläche: https://mail.it-sec.ovh/installer/) prüft die Voraussetzungen, etwaige Mängel sind an dieser Stelle zu beheben, konkret sind das unter Debian 8 folgende:

Schaffen der nötigen Voraussetzungen – PHP und MySQL wurden bereits in Kapitel 5.1 installiert, folgende Module werden nun noch zusätzlich benötigt:

```
root@Sec-NS2:~# aptitude install php5-mcrypt php5-intl php5-ldap
root@Sec-NS2:~# php5enmod mcrypt intl ldap
```

Setzen der Zeitzone in der PHP-Konfiguration

```
root@Sec-NS2:~# vi /etc/php5/apache2/php.ini
…
[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone = Europe/Vienna
…
```

Setzen der Berechtigungen gemäß Doku:

```
root@Sec-NS2:/var/www/mail.it-sec.ovh# chmod g+w temp logs
```

5.2.2. Anlage MySQL-Datenbank

Anlage einer MySQL-Datenbank für RoundCube

```
root@Sec-NS2:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 48
Server version: 5.5.44-0+deb8u1 (Debian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
CREATE DATABASE roundcubemail /*!40101 CHARACTER SET utf8 COLLATE utf8_general_ci */;
Query OK, 1 row affected (0.00 sec)

mysql>
GRANT ALL PRIVILEGES ON roundcubemail.* TO roundcube@localhost IDENTIFIED BY 'myPwd';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

Anmerkung: Kennwort myPWD ist durch ein sicheres Kennwort zu ersetzen, dass im RoundCube-Installer für den Connect zur mySQL-DB anschließend zu verwenden ist. Das mySQL-Root-Kennwort für den Login wurde bei der mySQL-Installation im Schritt zuvor festgelegt.

Neu-Laden der Apache-Konfiguration

```
root@Sec-NS2:~# apache2ctl graceful
```

5.2.3. Installation von RoundCube mittels Web-Installer

Sind alle Vorbereitungen getroffen kann mit dem RoundCube-Installer (Web-Oberfläche: https://mail.it-sec.ovh/installer/) fortgesetzt werden (siehe nachfolgende Abbildungen).

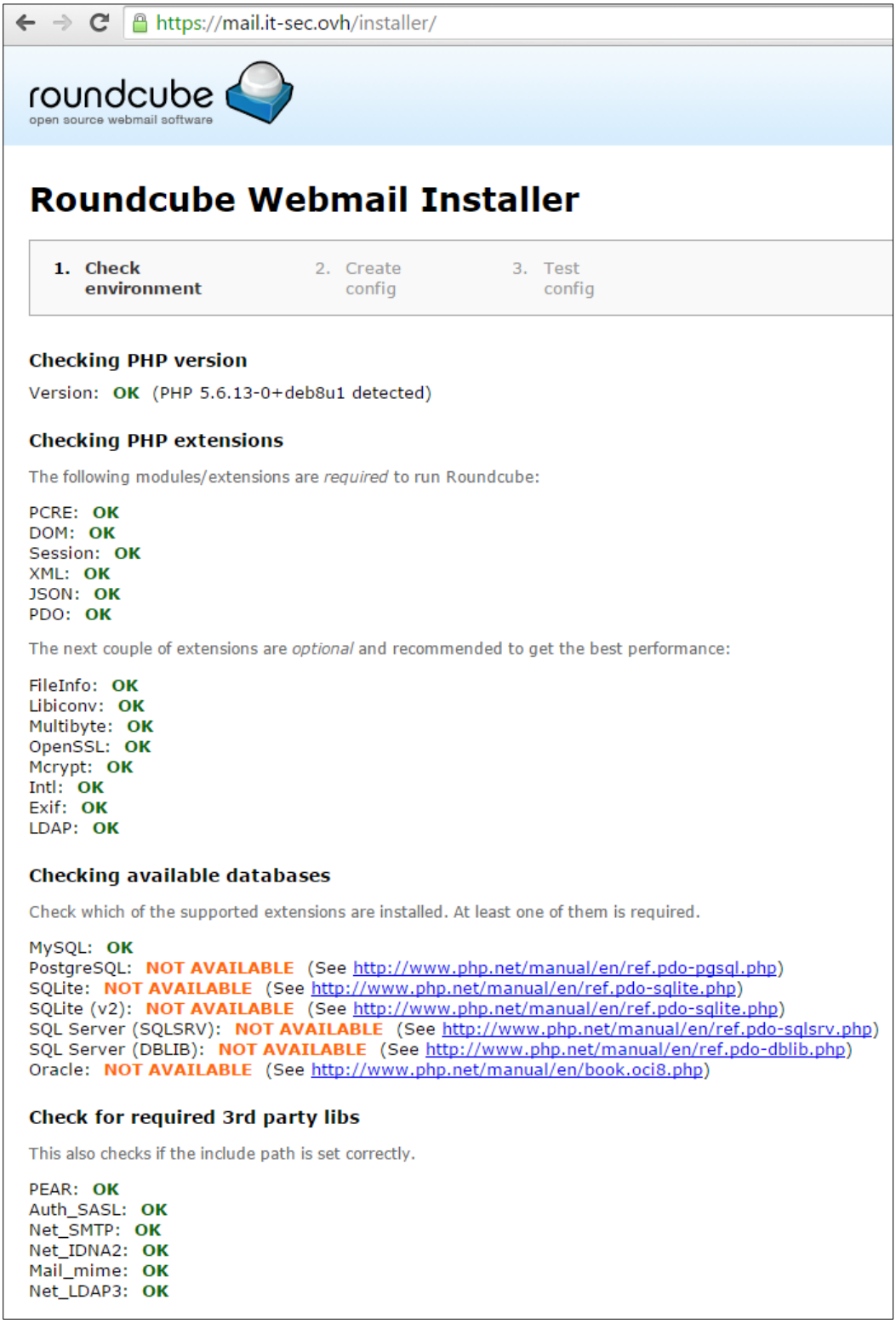


Abbildung 50: RoundCube Installer - Step 1 (Dependency-Check - #1)

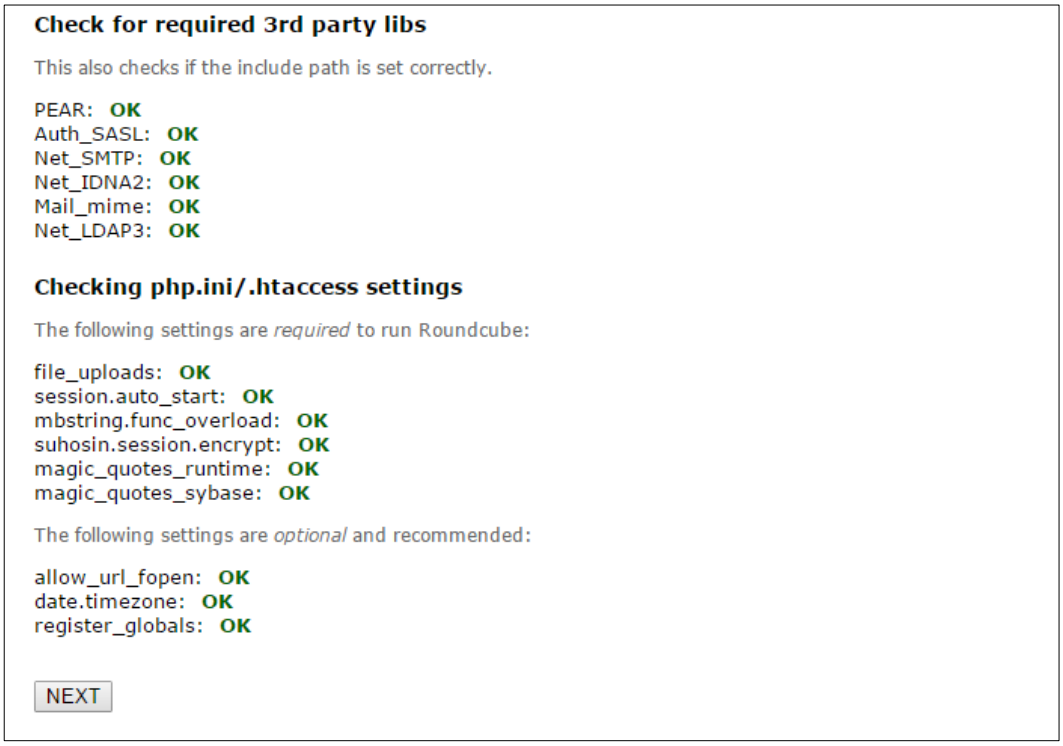


Abbildung 51: RoundCube Installer - Step 1 (Dependency-Check - #2)

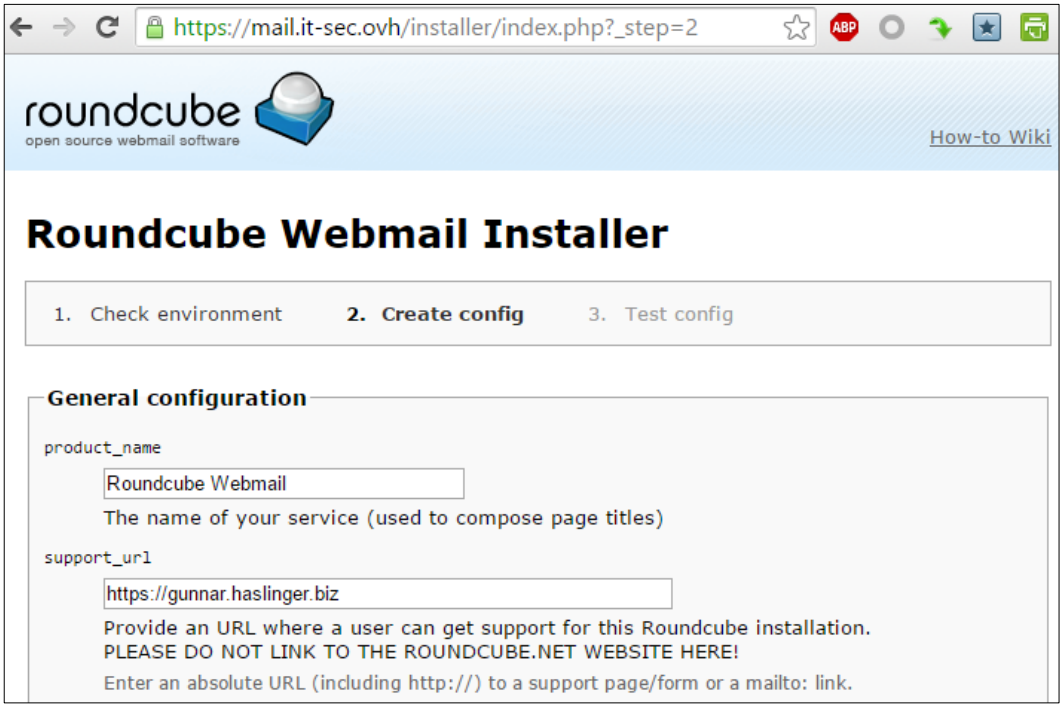


Abbildung 52: RoundCube Installer - Step 2a (Create Config)

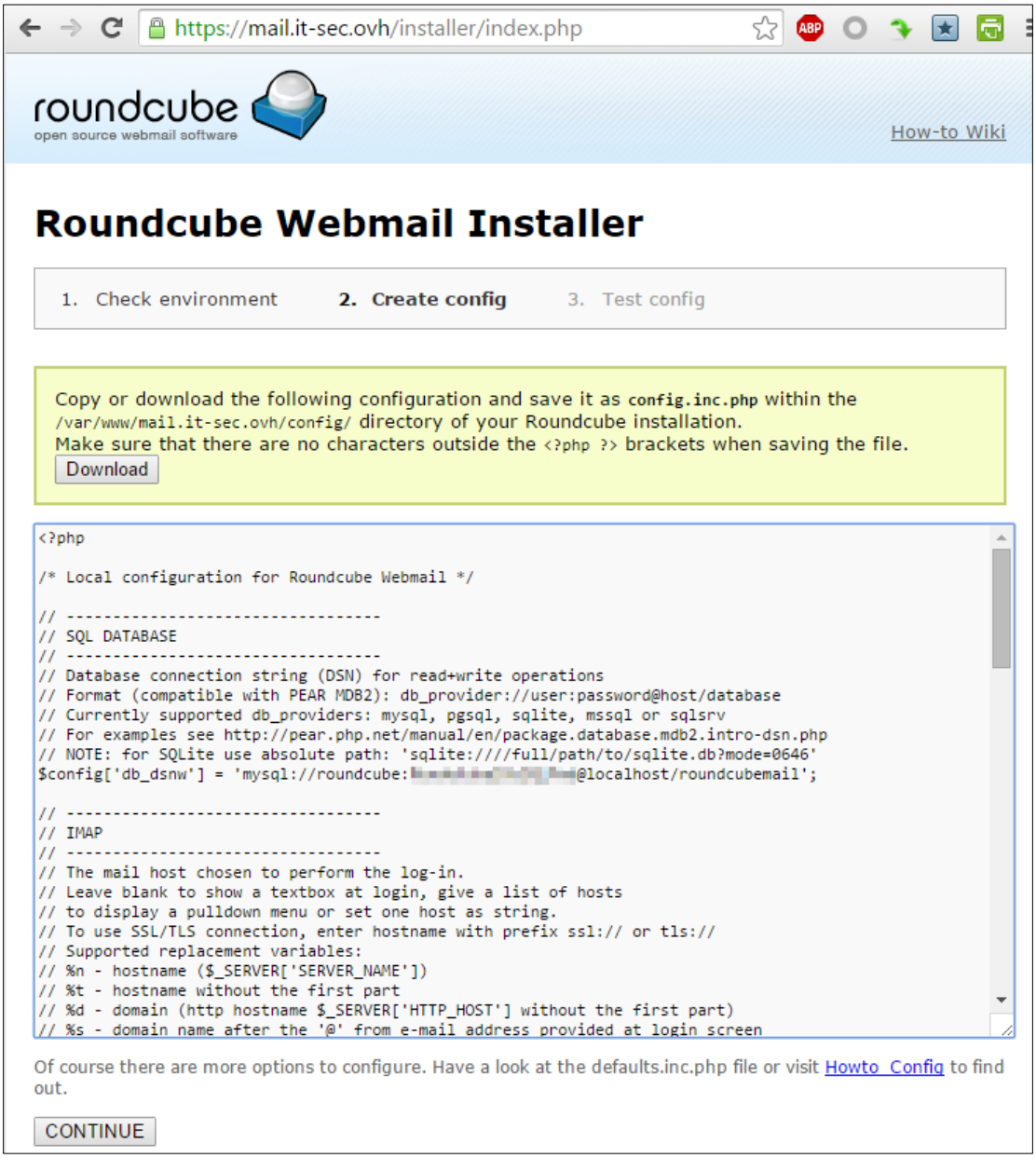


Abbildung 53: RoundCube Installer - Step 2b (Save Config)

Die in Abbildung 53 dargestellte, generierte Konfiguration wird nun manuell in das File `/var/www/mail.it-sec.ovh/config/config.inc.php` eingefügt.

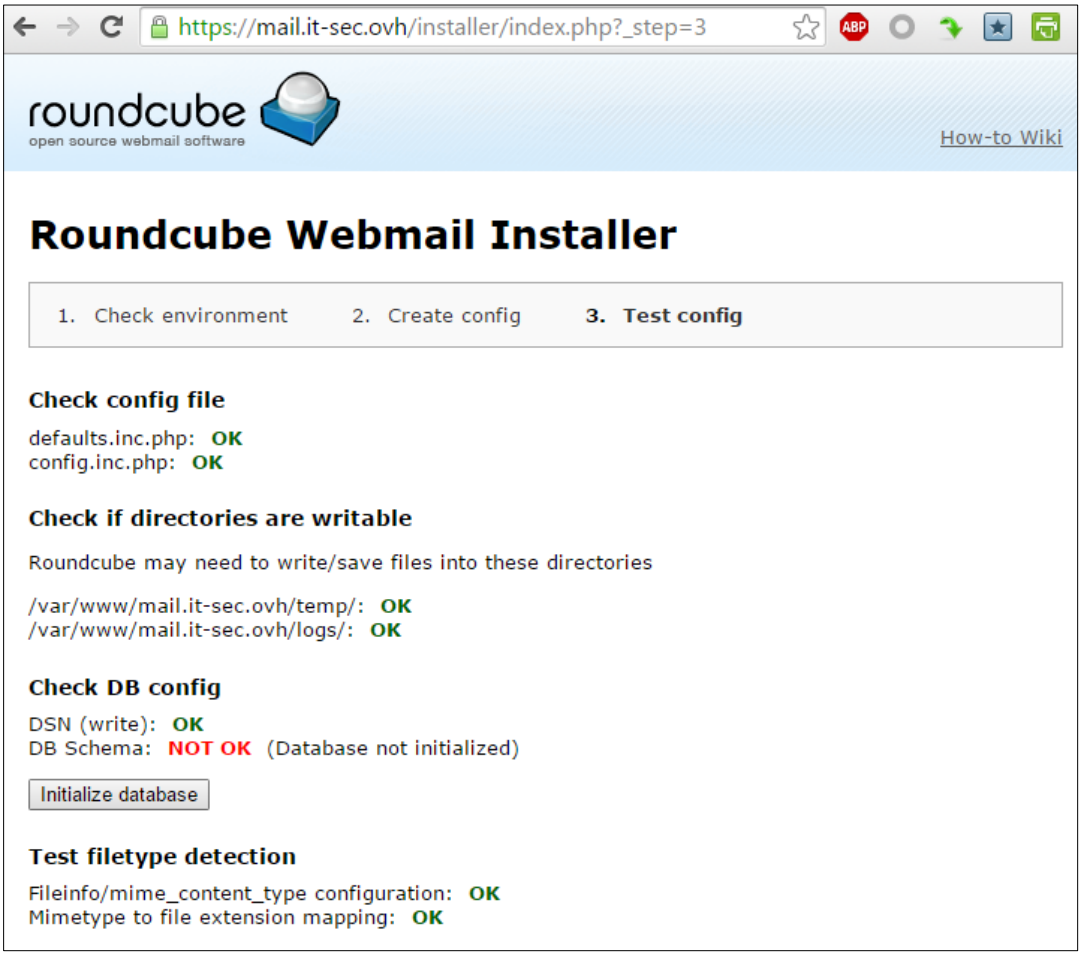


Abbildung 54: RoundCube Installer - Step 3 (Check Config)

Mittels „Initialize database“ (siehe Abbildung 54) wird zuletzt nun noch das Datenbank-Schema eingerichtet.

Zum Abschluss der Installation wird das Installer-Verzeichnis geschützt:

```
root@Sec-NS2:/var/www/mail.it-sec.ovh# chmod 000 installer
```

5.2.4. Funktionstest RoundCube-WebMail

Zugriff auf den WebMailer unter der URL https://mail.it-sec.ovh/, das Login erfolgt mittels der in Kapitel 4.4 angelegten Benutzerkonten.

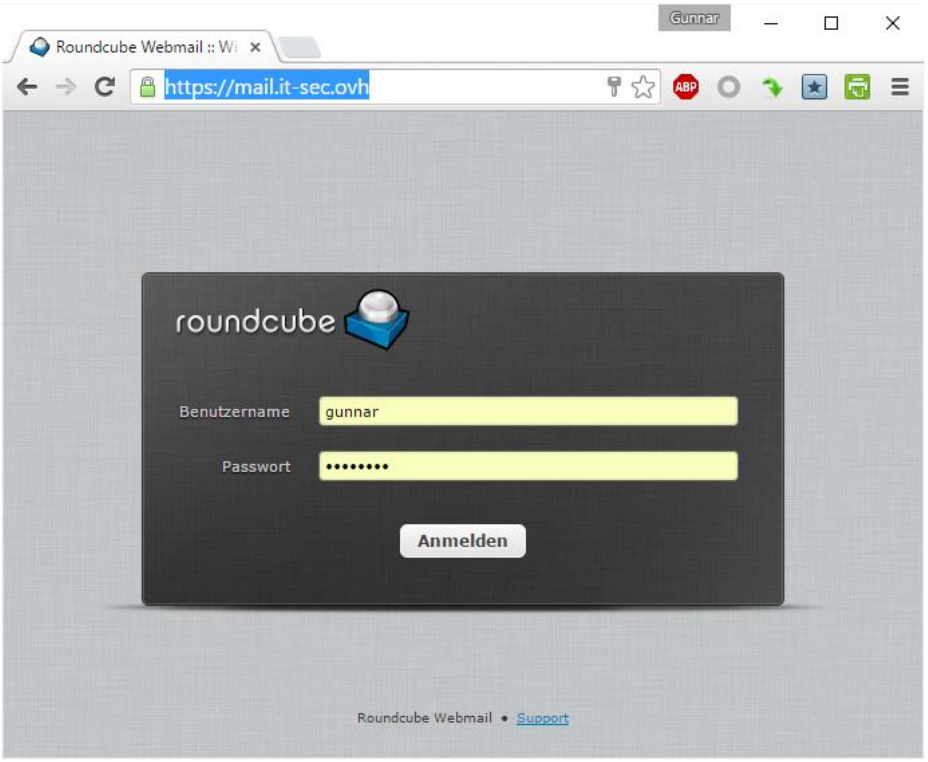


Abbildung 55: Funktionstest - Einstieg RoundCube WebMail System

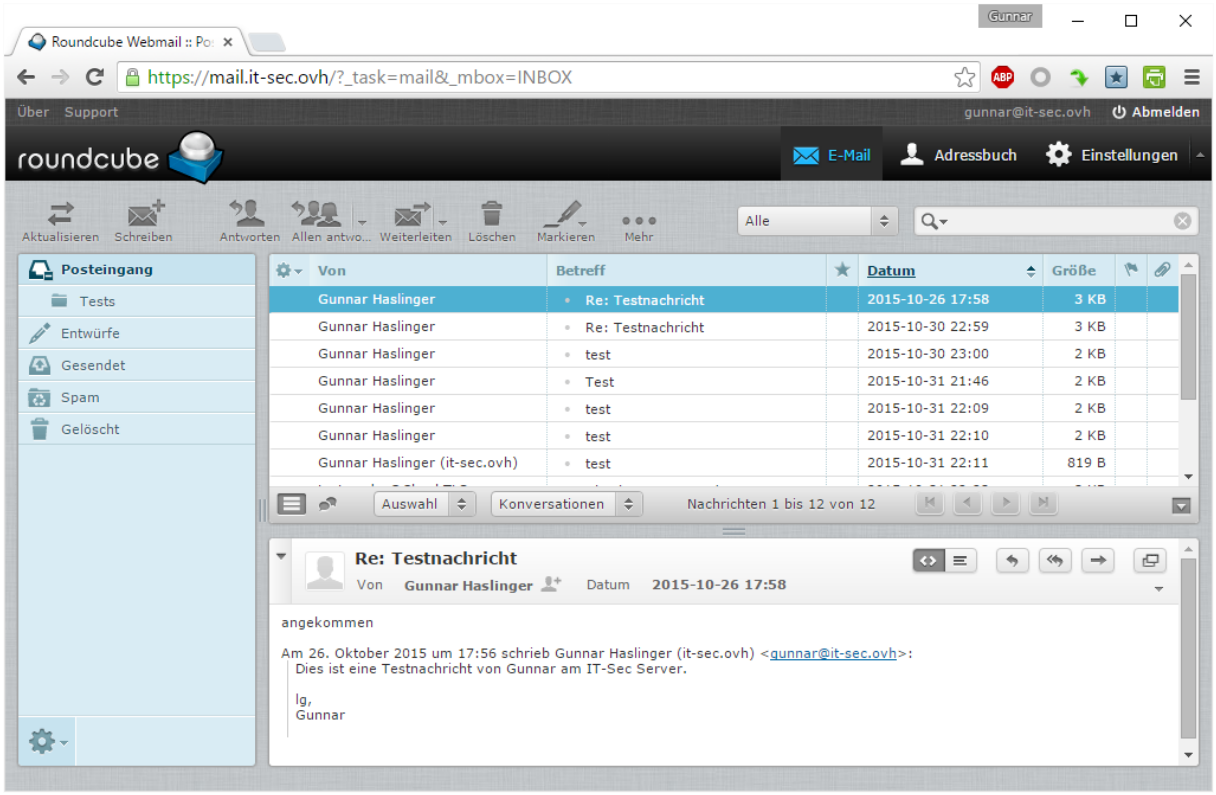


Abbildung 56: Funktionstest - Nutzung des RoundCube WebMail Systems

5.3. SSL-Checks

5.3.1. SSLLabs.com WebService

Eine ausführliche Analyse von SSL/TLS-Webservern ist mittels der von Qualys SSL Labs (Ivan Ristić) bereitgestellten Analysetools https://www.ssllabs.com/ssltest/ möglich. Wertvolle Informationen zur Konfiguration von SSL/TLS-Webservern sind nicht nur [BetterCrypo] sondern auch [IR-TLS] zu entnehmen, Hintergründe wie das SSL-Labs Rating zustande kommt sind in [IR-SrvRating] dokumentiert.

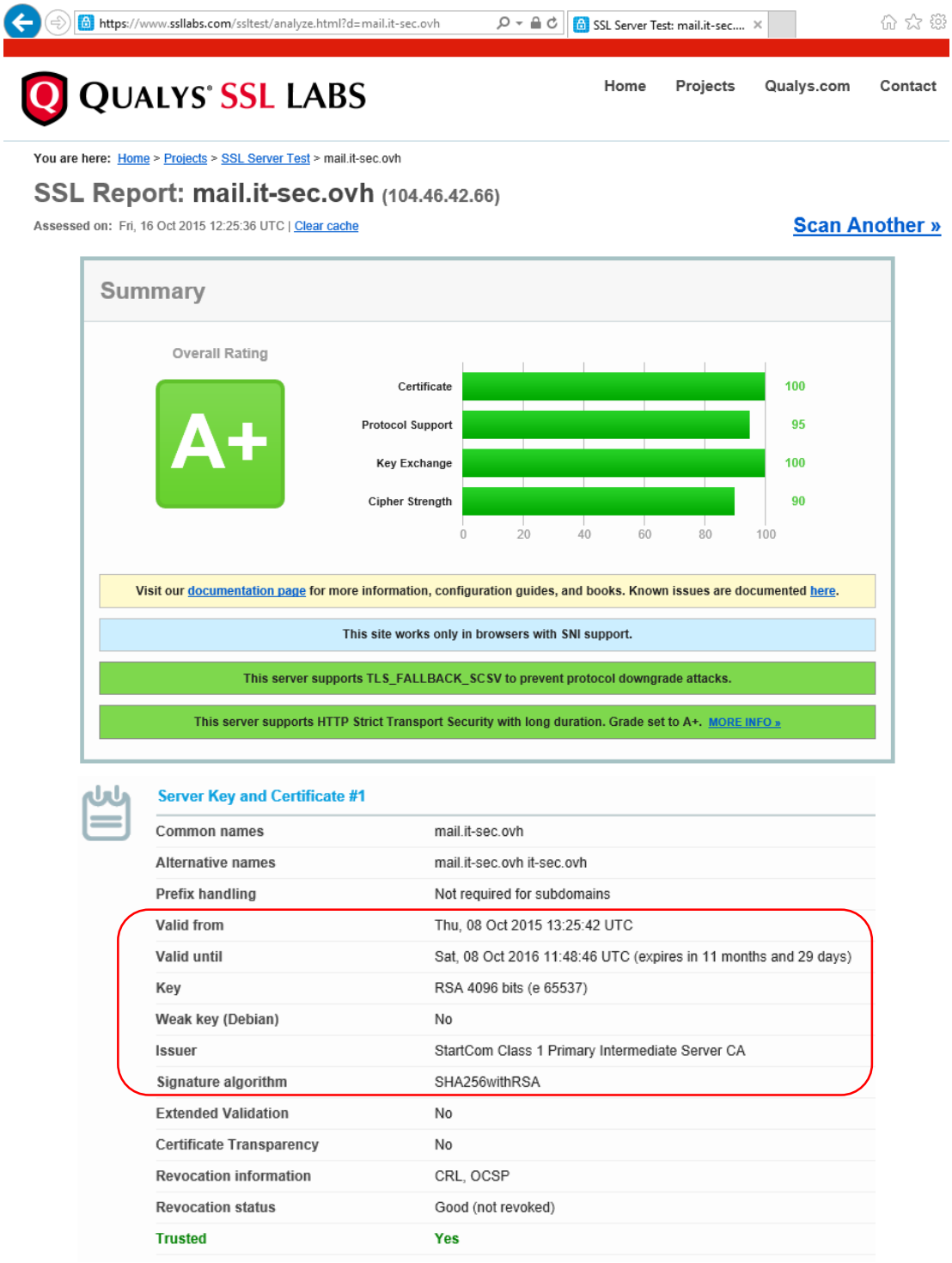


Abbildung 57: Server-Check mittels Qualys SSL Labs: Ergebnis

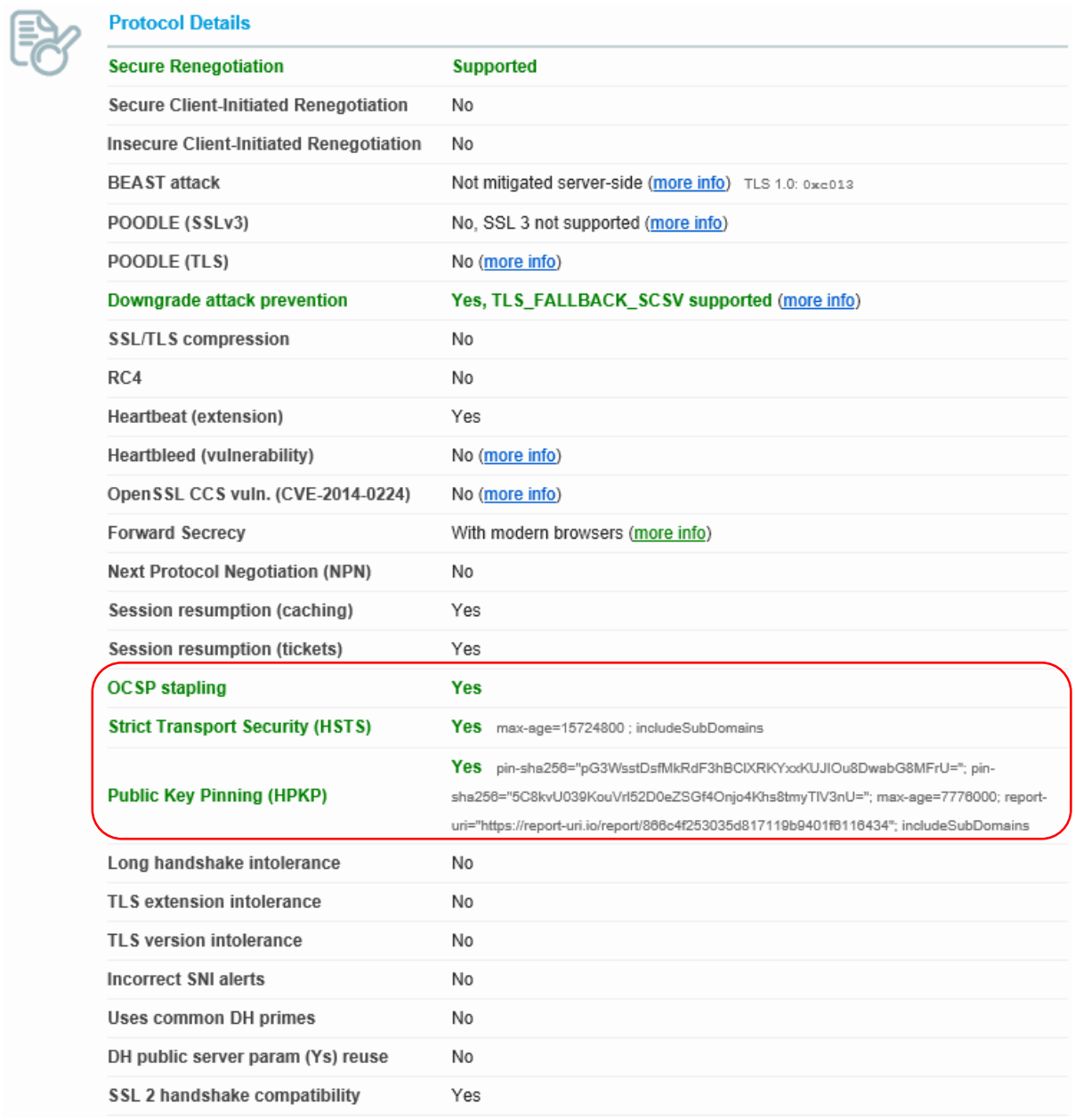
Protocols

TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

Cipher Suites (SSL 3+ suites in server-preferred order; deprecated and SSL 2 suites at the end)

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) ECDH 256 bits (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) ECDH 256 bits (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) ECDH 256 bits (eq. 3072 bits RSA) FS	128
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x67) DH 4096 bits (p: 512, g: 1, Ys: 511) FS	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	128
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	128
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)	128
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	128
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) ECDH 256 bits (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) ECDH 256 bits (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) ECDH 256 bits (eq. 3072 bits RSA) FS	256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) DH 4096 bits (p: 512, g: 1, Ys: 512) FS	256
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	256
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)	256
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	256

Abbildung 58: Server-Check mittels Qualys SSL Labs: Protokolle & Chiffren



Protocol Details

Secure Renegotiation	**Supported**
Secure Client-Initiated Renegotiation	No
Insecure Client-Initiated Renegotiation	No
BEAST attack	Not mitigated server-side (more info) TLS 1.0: 0xc013
POODLE (SSLv3)	No, SSL 3 not supported (more info)
POODLE (TLS)	No (more info)
Downgrade attack prevention	**Yes, TLS_FALLBACK_SCSV supported** (more info)
SSL/TLS compression	No
RC4	No
Heartbeat (extension)	Yes
Heartbleed (vulnerability)	No (more info)
OpenSSL CCS vuln. (CVE-2014-0224)	No (more info)
Forward Secrecy	With modern browsers (more info)
Next Protocol Negotiation (NPN)	No
Session resumption (caching)	Yes
Session resumption (tickets)	Yes
OCSP stapling	**Yes**
Strict Transport Security (HSTS)	**Yes** max-age=15724800 ; includeSubDomains
Public Key Pinning (HPKP)	**Yes** pin-sha256="pG3WsstDsfMkRdF3hBCIXRKYxxKUJIOu8DwabG8MFrU="; pin-sha256="5C8kvU039KouVrl52D0eZSGf4Onjo4Khs8tmyTlV3nU="; max-age=7776000; report-uri="https://report-uri.io/report/866c4f253035d817119b9401f6116434"; includeSubDomains
Long handshake intolerance	No
TLS extension intolerance	No
TLS version intolerance	No
Incorrect SNI alerts	No
Uses common DH primes	No
DH public server param (Ys) reuse	No
SSL 2 handshake compatibility	Yes

Abbildung 59: Server-Check mittels Qualys SSL Labs: Protokoll-Details

5.3.2. SSL-Check mittels A-SIT Browser-Plugin und/oder Chrome

Das „Zentrum für sichere Informationstechnologie – Austria“ (a-sit.at) stellt seit Juli 2015 ein Firefox-Plugin zur Anzeige von Sicherheits-Infos zu SSL-Websites bereit. Das Plugin ist unter https://demo.a-sit.at/firefox-plugin-zur-anzeige-von-sicherheitsinfos/ erhältlich und liefert nicht nur übersichtliche Informationen zum verwendeten Zertifikat und der Cipher-Suite, sondern auch zur Verwendung von HTTP Strict Transport Security und HTTP-Public-Key-Pinning (siehe Abbildung 60). Die Informationen zu HSTS und HPKP sind auch mittels der Google Chrome internen URL chrome://net-internals/#hsts auslesbar (Abbildung 61).

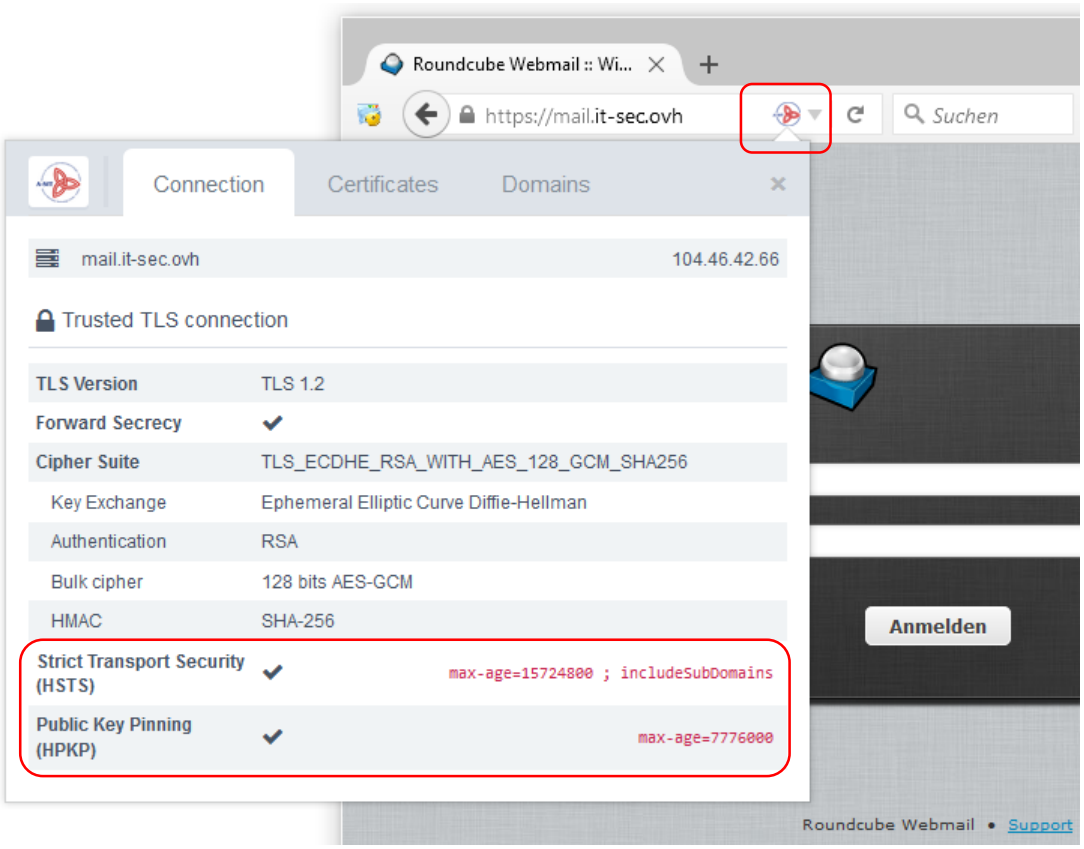


Abbildung 60: A-Sit Firefox-Browser PlugIn

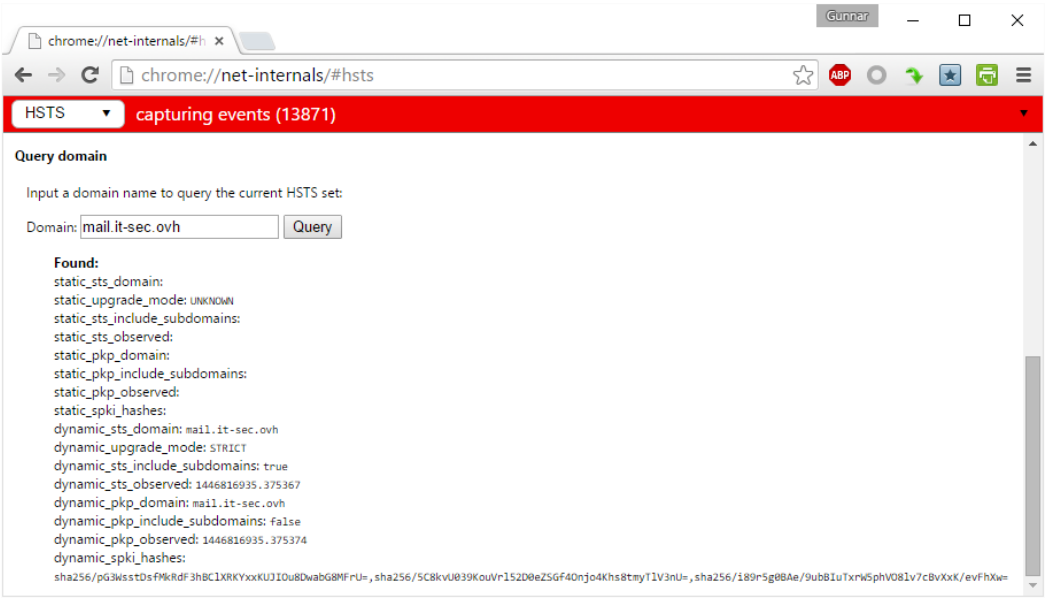


Abbildung 61: Google-Chrome zeigt HSTS und HPKP Infos - chrome://net-internals/#hsts

5.3.3. DANE-Check mittels SIDN-Labs WebSite

SIDN Labs (Stichting Internet Domain-Registry Netherlands) stellt einen DANE-Validator in Form einer WebSite online bereit: https://check.sidnlabs.nl/dane/. Dieser erlaubt die Prüfung einer URL, das Zertifikat wird hier gegen den in DNSSEC hinterlegten TLSA Record geprüft.

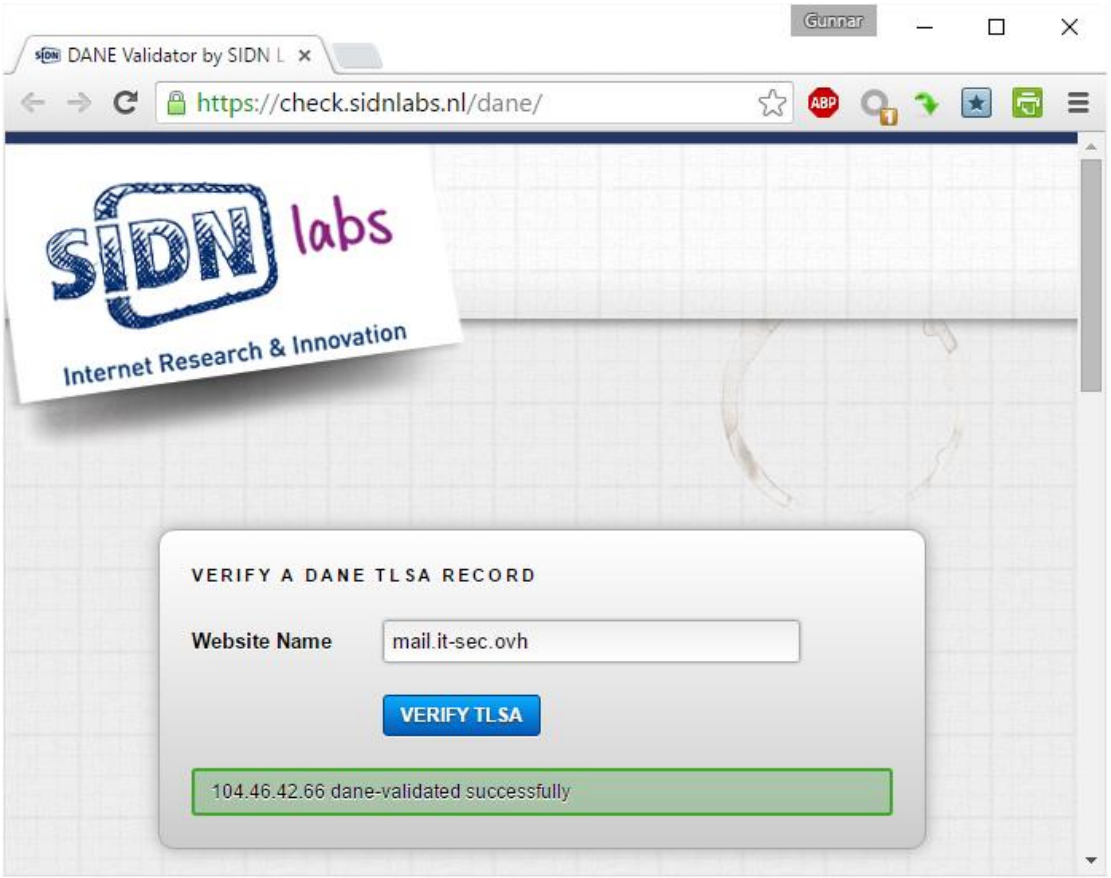


Abbildung 62: DANE Validator von SIDN Labs - https://check.sidnlabs.nl/dane/

5.3.4. DNSSec und DANE-Check mittels DNSSec-Validator PlugIn

Die tschechische Domain-Registry https://nic.cz bietet einen DNSSec-Validator in Form eines Webbrowser-Plugins für zahlreiche Browser (Internet Explorer, Firefox, Chrome, Safari, Opera, …) zum Download an: https://www.dnssec-validator.cz/

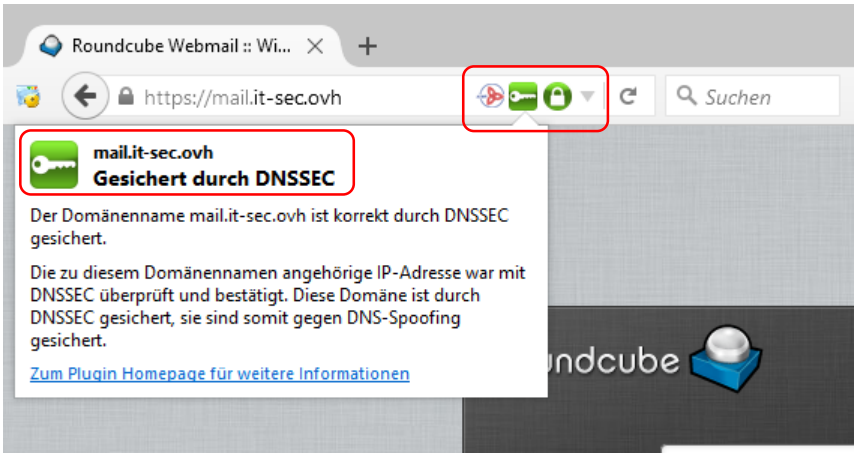


Abbildung 63: DNSSec Validierung mittels dnssec-validator.cz PlugIn

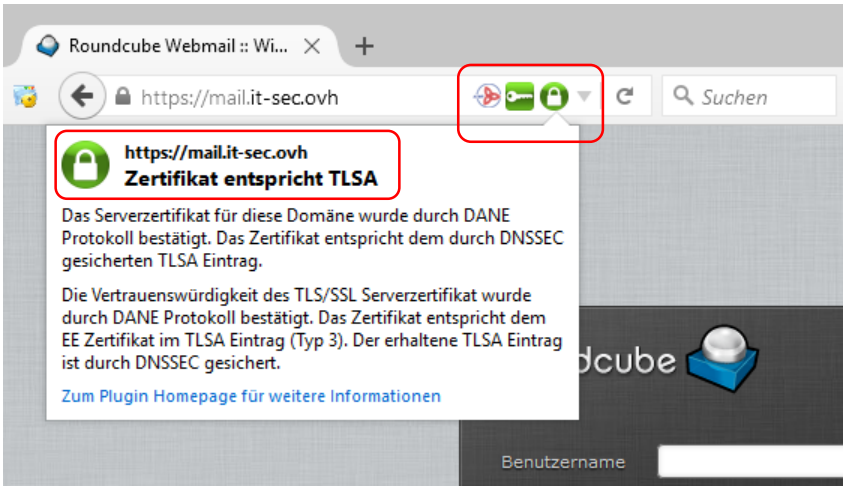


Abbildung 64: DANE / TLSA Validierung mittels dnssec-validator.cz PlugIn

Wird der WebServer mit einem anderen Zertifikat betrieben, so wird dies sofort bemerkt und bemängelt (siehe Abbildung 65).

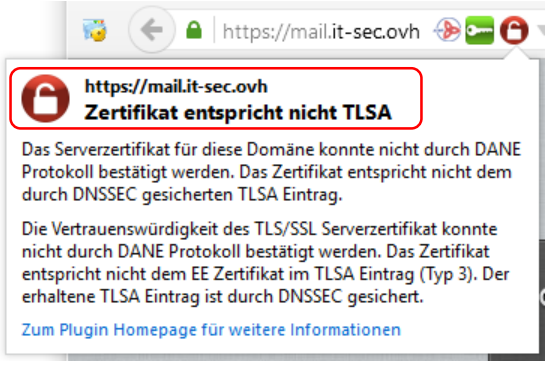


Abbildung 65: Fehlgeschlagene DANE Validierung mittels dnssec-validator.cz PlugIn

5.3.5. SSL-Check mittels TestSSL (OpenSSL-Script)

Das bereits in Abschnitt 4.8.1 vorgestellte Script TestSSL von Dirk Wetter kann auch zur Analyse des WebServers genutzt werden.

Download und Nutzung des Scripts sind wie folgt durchzuführen:

```
root@Sec-NS2:/tmp# wget -q https://testssl.sh/testssl.sh
root@Sec-NS2:/tmp# wget -q https://testssl.sh/mapping-rfc.txt
root@Sec-NS2:/tmp# chmod 755 ./testssl.sh
root@Sec-NS2:/tmp# ./testssl.sh https://mail.it-sec.ovh
```

Das Test-Resultat (siehe Abbildung 66) wird live auf der Konsole ausgegeben, oder kann in ein File umgeleitet und mittels Tools auch HTML-formatiert abgelegt werden.

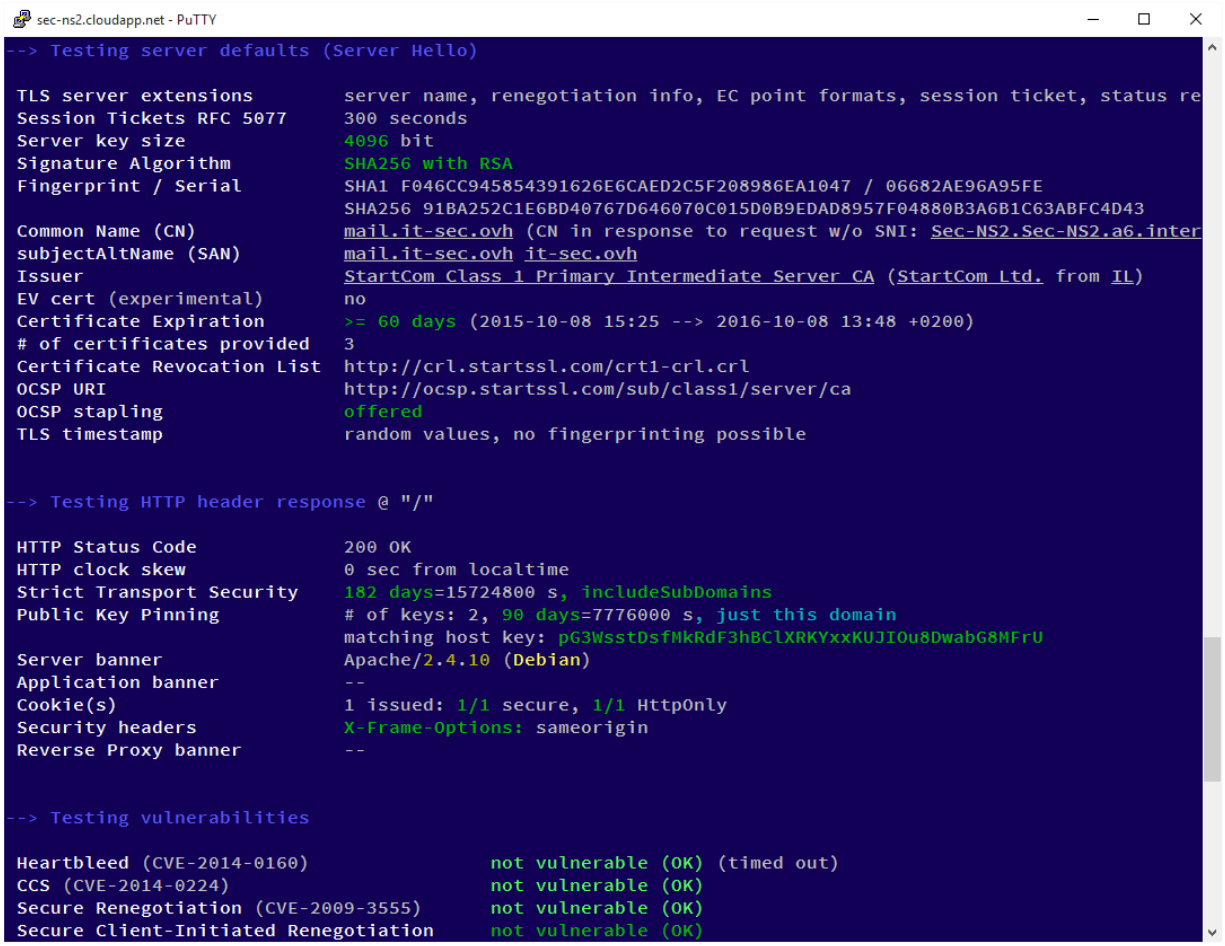


Abbildung 66: Test mittels TestSSL.sh Script und OpenSSL

Abbildungsverzeichnis

Anmerkung: Sofern nicht anderweitig gekennzeichnet wurden Abbildungen selbst erstellt bzw. mittels Screenshots erzeugt.

Literaturverzeichnis

Bücher

[Bauer-LinSrv] Michael D. Bauer: Linux Server-Sicherheit
O'Reilly Verlag GmbH & Co. KG; 2. Auflage, Juni 2005, ISBN 978-3-89721-413-2
Kapitel „Domain Name Services absichern“ als ebook im PDF-Format verfügbar:
http://www.oreilly.de/catalog/linuxss2ger/chapter/ch06.pdf

[IR-Apache] Ivan Ristić: Apache Security
Feisty Duck Limited, UK, Digital reprint published April 2010
https://www.feistyduck.com/library/apache-security/

[IR-OpenSSL] Ivan Ristić: OpenSSL Cookbook
Feisty Duck Limited, UK, Second Edition, March 2015
https://www.feistyduck.com/books/openssl-cookbook/

Zeitschriften

[heise-DANE] Heise Zeitschriften Verlag, Zeitschrift: c’t - Heft 2014/11
Artikel: Geleitschutz - DANE verbessert sicheren Transport zwischen Mailservern
Seite 194-197, PDF als Beilage verfügbar

Internet

[Acidx] Acidx’s Blog - Markus Klein:
Installing a Mailserver with Postfix, Dovecot, SASL, LDAP & Roundcube
http://acidx.net/wordpress/2014/06/
installing-a-mailserver-with-postfix-dovecot-sasl-ldap-roundcube/
Stand: 05.06.2014, abgerufen am 24.10.2015, PDF als Beilage verfügbar

[BetterCrypto] bettercrypto.org: Applied Crypto Hardening
Draft revision: 2079040, PDF-Dokument abrufbar: https://bettercrypto.org/
Stand: 18.06.2015, abgerufen am 13.09.2015, PDF als Beilage verfügbar

[BSI-Postfix] Deutsches Bundesamt für Sicherheit in der Informationstechnik (BSI):
Sicherer Betrieb von E-Mail-Servern mit Postfix 2.5
BSI-Checkliste zur Internet-Sicherheit (ISi-Check), PDF-Dokument abrufbar:
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Internetsicherheit/
isi_mail_server_checkliste_postfix_pdf.pdf
Stand: 2009, Version 1.0, abgerufen am 24.10.2015

[BSI-TR03108] Deutsches Bundesamt für Sicherheit in der Informationstechnik (BSI):
Technische Richtlinie TR-03108, Sicherer E-Mail-Transport
https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03108/index_htm.html
Stand: 20.08.2015, Version 0.9 Entwurf, abgerufen am 22.08.2015

[Google-Mail] Google: Transparenzbericht - E-Mail-Verschlüsselung bei der Übertragung
https://www.google.com/transparencyreport/saferemail/
Stand: Oktober 2015, abgerufen am 05.11.2015, PDF als Beilage verfügbar

[heise-DNSSec1] heise.de online – heise Netze, Ulrich Wieser:
Artikel: Domain Name System absichern mit DNSSEC
http://heise.de/-903318
Stand: 15.01.2010, abgerufen am 25.09.2015, PDF als Beilage verfügbar

[heise-Test] heise.de online – heise Netze, Dusan Zivadinovic – Artikel:
Verschlüsselter Mail-Transport: Neue DANE-Testseite hilft beim Aufsetzen
http://heise.de/-2517595
Stand: 15.01.2015, abgerufen am 25.10.2015, PDF als Beilage verfügbar

[IANA-DNSSec] Internet Assigned Numbers Authority (IANA):
Domain Name System Security (DNSSEC) Algorithm Numbers
http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml
Stand 31.03.2015, abgerufen am 27.09.2015, PDF als Beilage verfügbar

[IR-SrvRating] Qualys SSL Labs - Ivan Ristić: SSL Server Rating Guide
PDF-Dokument, abrufbar von: https://www.ssllabs.com/projects/rating-guide/
Version 2009j vom 20.05.2015, abgerufen 18.10.2015, PDF als Beilage verfügbar

[IR-TLS] Qualys SSL Labs - Ivan Ristić: SSL/TLS Deployment Best Practices
PDF-Dokument, abrufbar von: https://www.ssllabs.com/projects/best-practices/
Version 1.4 vom 08.12.2014, abgerufen 18.10.2015, PDF als Beilage verfügbar

[ISC-BIND9] Internet Systems Consortium, Inc.: BIND 9 Administrator Reference Manual
PDF-Dokument, abrufbar von: https://www.isc.org/downloads/bind/doc/bind-9-9/
Stand 17.01.2014, abgerufen am 13.09.2015, PDF als Beilage verfügbar

[ISC-DNSSec] Internet Systems Consortium, Inc.: BIND DNSSEC Guide
abrufbar von: http://users.isc.org/~jreed/dnssec-guide/dnssec-guide.html
sowie als PDF von: http://users.isc.org/~jreed/dnssec-guide/dnssec-guide.pdf
Stand 2015, Version 1.1, abgerufen am 13.09.2015, PDF als Beilage verfügbar

[NIST-81.2] National Institute of Standards and Technology: Special Publication 800-81-2
Secure Domain Name System (DNS) Deployment Guide
PDF-Dokument, abrufbar von: http://dx.doi.org/10.6028/NIST.SP.800-81-2
Stand: September 2013, abgerufen am 06.09.2015, PDF als Beilage verfügbar

[OWASP-PKP] The Open Web Application Security Project: Certificate and Public Key Pinning
https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
Stand: 17. Juni 2015, abgerufen am 16.10.2015, PDF als Beilage verfügbar

[Postfix-PFS] Postfix.org, Postfix Documentation: TLS Forward Secrecy in Postfix
http://www.postfix.org/FORWARD_SECRECY_README.html
abgerufen am 25.10.2015, PDF als Beilage verfügbar

[Postfix-TLS] Postfix.org, Postfix Documentation: Postfix TLS Support
http://www.postfix.org/TLS_README.html
abgerufen am 24.10.2015, PDF als Beilage verfügbar

[Postfix-Virtual] Postfix.org, Postfix Documentation: Postfix Virtual Domain Hosting Howto
http://www.postfix.org/VIRTUAL_README.html
abgerufen am 31.10.2015, PDF als Beilage verfügbar

[RFC-3655] The Internet Engineering Task Force (IETF): Request for Comments: 3655
Redefinition of DNS Authenticated Data (AD) bit
abrufbar von: https://tools.ietf.org/html/rfc3655
Stand: November 2003, abgerufen am 25.10.2015, PDF als Beilage verfügbar

[RFC-4033] The Internet Engineering Task Force (IETF): Request for Comments: 4033
DNS Security Introduction and Requirements
abrufbar von: https://tools.ietf.org/html/rfc4033
Stand März 2005, abgerufen am 10.10.2015, PDF als Beilage verfügbar

[RFC-4034] The Internet Engineering Task Force (IETF): Request for Comments: 4034 Resource Records for the DNS Security Extensions
abrufbar von: https://tools.ietf.org/html/rfc4034
Stand März 2005, abgerufen am 27.09.2015, PDF als Beilage verfügbar

[RFC-4035] The Internet Engineering Task Force (IETF): Request for Comments: 4035 Protocol Modifications for the DNS Security Extensions
abrufbar von: https://tools.ietf.org/html/rfc4035
Stand März 2005, abgerufen am 10.10.2015, PDF als Beilage verfügbar

[RFC-5155] The Internet Engineering Task Force (IETF): Request for Comments: 5155 DNS Security (DNSSEC) Hashed Authenticated Denial of Existence
abrufbar von: https://tools.ietf.org/html/rfc5155
Stand März 2008, abgerufen am 10.10.2015, PDF als Beilage verfügbar

[RFC-6066] The Internet Engineering Task Force (IETF): Request for Comments: 6066 Transport Layer Security (TLS) Extensions: Extension Definitions
abrufbar von: https://tools.ietf.org/html/rfc6066
Stand Jänner 2011, abgerufen am 31.10.2015, PDF als Beilage verfügbar

[RFC-6698] The Internet Engineering Task Force (IETF): Request for Comments: 6698 The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA
abrufbar von: https://tools.ietf.org/html/rfc6698
Stand August 2012, abgerufen am 10.10.2015, PDF als Beilage verfügbar

[RFC-6840] The Internet Engineering Task Force (IETF): Request for Comments: 6840 Clarifications and Implementation Notes for DNS Security (DNSSEC)
abrufbar von: https://tools.ietf.org/html/rfc6840
Stand Februar 2013, abgerufen am 10.10.2015, PDF als Beilage verfügbar

[RFC-6781] The Internet Engineering Task Force (IETF): Request for Comments: 6781 DNSSEC Operational Practices, Version 2
abrufbar von: https://tools.ietf.org/html/rfc6781
Stand Dezember 2012, abgerufen am 10.10.2015, PDF als Beilage verfügbar

[RFC-6797] The Internet Engineering Task Force (IETF): Request for Comments: 6797 HTTP Strict Transport Security (HSTS)
abrufbar von: https://tools.ietf.org/html/rfc6797
Stand November 2012, abgerufen am 16.10.2015, PDF als Beilage verfügbar

[RFC-6944] The Internet Engineering Task Force (IETF): Request for Comments: 6944 DNS Security (DNSSEC) DNSKEY Algorithm Implementation Status
abrufbar von: https://tools.ietf.org/html/rfc6944
Stand April 2013, abgerufen am 10.10.2015, PDF als Beilage verfügbar

[RFC-7469] The Internet Engineering Task Force (IETF): Request for Comments: 7469 Public Key Pinning Extension for HTTP
abrufbar von: https://tools.ietf.org/html/rfc7469
Stand April 2015, abgerufen am 16.10.2015, PDF als Beilage verfügbar

[RFC-7672] The Internet Engineering Task Force (IETF): Request for Comments: 7672 SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)
abrufbar von: https://tools.ietf.org/html/rfc7672
Stand Oktober 2015, abgerufen am 05.11.2015, PDF als Beilage verfügbar

[SIDN-NSEC3] SIDN Labs (Stichting Internet Domain-Registry Netherlands),
Miek Gieben & Matthijs Mekking: Authenticated Denial of Existence in the DNS
abrufbar: https://www.sidn.nl/fileadmin/docs/PDF-files_UK/wp-2011-0x01-v2.pdf
Stand Jänner 2012, abgerufen am 10.10.2015, PDF als Beilage verfügbar

[SBA-TLSmail] SBA Research Austria – W. Mayer, A. Zauner, M. Schmidecker, M. Huber:
No Need for Black Chambers: Testing TLS in the E-mail Ecosystem at Large
PDF-Dokument abrufbar von: http://arxiv.org/abs/1510.08646
Slides zum Dokument abrufbar: http://www.slideshare.net/a_z_e_t/no-need-for-black-chamers-testing-tls-in-the-email-ecosystem-at-large
Stand: v2 vom 01.11.2015, abgerufen am 05.11.2015, PDF als Beilage verfügbar

[SNE-DNSSec1] University of Amsterdam / System and Network Engineering,
Master Research Project - Anastasios Poulidis, Hoda Rohani: DNSSec Revisited
abrufbar von: https://www.os3.nl/archive/research_projects
https://www.os3.nl/_media/2013-2014/courses/rp2/p03_report.pdf
https://www.os3.nl/_media/2013-2014/courses/rp2/p03_presentation.pdf
Stand: Juli 2014, abgerufen am 11.10.2015, PDF als Beilage verfügbar

[SNE-DNSSec2] University of Amsterdam – System and Network Engineering:
Master Research Project – Nicolas Canceill:
Measuring the deployment of DNSSec over the Internet
abrufbar von: https://www.os3.nl/archive/research_projects
https://www.os3.nl/_media/2013-2014/courses/rp2/p14_report.pdf
https://www.os3.nl/_media/2013-2014/courses/rp2/p14_presentation.pdf
Stand: Juli 2014, abgerufen am 11.10.2015, PDF als Beilage verfügbar

[StatDNS] Frederic Cambus: statdns – DNS and Domain Name statistics and tools
TLD Zone File Statistics, October 2015 Reports
abrufbar von: http://statdns.com
Stand: Oktober 2015, abgerufen am 11.10.2015, PDF als Beilage verfügbar

[sys4-DANE] sys4 AG, Patrick Ben Koetter: SMTP-Sicherheit mit DANE
One Year of DANE- Tales and Lessons learned
PDF abrufbar von: https://sys4.de/download/dane-NCSC-en.pdf
Stand: 14.04.2015, abgerufen am 24.10.2015, PDF als Beilage verfügbar

[VB-TLS] Vincent Bernat: SSL/TLS & Perfect Forward Secrecy
http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html
Stand: 25.11.2011, abgerufen am 08.11.2015, PDF als Beilage verfügbar

Änderungen / Fehlerkorrekturen / Changelog

Version vom 21.11.2015

- Basisversion, final

Version vom 31.03.2016

- Layout/Format-Änderungen: Inhaltsverzeichnis
- Austausch (selbst gezeichnete Abbildungen):
    - Abbildung 3: DNSSec Einträge und deren Zusammenwirken
    - Abbildung 16: Abbildung 16: Man-in-the-Middle, Mailabgriff per Downgrade
    - Abbildung 18: Ablauf - SMTP mit DANE/TLSA/DNSSEC
    - Abbildung 25: Zusammenwirken eines typischen E-Mail Systems
    - Abbildung 27: Transport-Weg einer E-Mail

Version vom 22.04.2016

- Korrektur diverser (Tipp-)Fehler, keine inhaltlichen Änderungen

Version vom 28.04.2016

- Korrektur Abbildung 18: Ablauf - SMTP mit DANE/TLSA/DNSSEC