HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

strongMan Management-Tool-Erweiterung für grosse VPN Server

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik HSR, Rapperswil

Herbstsemester 2016

Studenten:	Raffael Fischer, Daniel Kolb, Josip Valencic
Betreuer:	Prof. Dr. Andreas Steffen, ITA
Gegenleser:	Prof. Dr. Markus Stolze, IFS
Experte:	Dr. Ralf Hauser
Zeitraum:	19.09.2016 - 23.12.2016

Inhaltsverzeichnis

1. Abstract

1.1. Ausgangslage

Für “strongSwan”, die IPsec-basierte und einzige Open-Source-VPN-Lösung mit vollständiger IKEv2 Implementation, existierte bereits eine Webanwendung namens “strongMan”. Sie bot die Möglichkeit, für Clients Zertifikate zu verwalten und Verbindungen zu einem VPN-Server aufzubauen.

Ziel dieser Bachelorarbeit war es, strongMan um einen Server-Modus für Administratoren zu erweitern. Dieser neu zu implementierende Server-Modus sollte somit das Gegenstück zum existierenden Client-Modus darstellen. Er soll Verbindungen von Clients, Verbindungen zu einem anderen strongSwan VPN-Server, Server-Zertifikate, Benutzernamen und Passwörter, sowie IP-Address-Pools verwalten können. Der Server-Modus soll am Ende in das bestehende strongMan Projekt integriert werden.

1.2. Vorgehen/Technologien

Das bestehende strongMan Projekt für VPN-Clients wurde in Python implementiert und verwendet das Web-Framework “Django”. Für die Persistenz der Daten wurde die leichtgewichtige Datenbank “SQLite” verwendet. Die Kommunikation mit dem strongSwan-Daemon, welcher das VPN-Backend darstellt, erfolgte via Unix-Socket.
Um eine möglichst einfache Integration zu garantieren, wurde im Server-Modus auf den gleichen Technologie-Stack gesetzt.

1.3. Ergebnis

Es entstand ein vollumfänglicher Server-Modus, welcher durch die gegebenen Anforderungen einer einfachen, übersichtlichen und sinnvollen Anwendung für Systemadministratoren dient. Als grosse Herausforderung stellte sich der Umgang mit vielen Verbindungen und deren übersichtliche Darstellung heraus. Dazu kam die Schwierigkeit der Visualisierung der Daten aus der eigenen Datenbank, kombiniert mit Live-Daten aus dem strongSwan-Daemon via Unix-Socket.
Dank flexiblen und detaillierten Filterfunktionen, welche auf den Listen der Connections agieren, können heute jedoch problemlos mehrere Hundert Verbindungen verwaltet und übersichtlich dargestellt werden. Eine übersichtliche Darstellung auch ohne Filterung wurde erreicht, indem zum Beispiel aufklappbare Tabellen für Detailinformationen oder Scrollbars für viele Einträge innerhalb einer aufgeklappten Untertabelle implementiert wurden.
Sollte das VPN-Backend, oder sogar der ganze VPN-Server, einmal neu gestartet werden, wird vom neuen Server-Modus direkt die komplette Konfiguration aller Verbindungen, Benutzernamen und Passwörter, Zertifikaten und Pools wieder in den strongSwan Daemon geladen und somit die aktuellen Status der Verbindungen und deren Clients in Echtzeit angezeigt.
Nebenbei kann vom Server-Modus einfach per Knopfdruck in den simplen Client-Modus und auch wieder zurück gewechselt werden.

Institute for Internet Technologies and Applications

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

Bachelorarbeit 2016

strongMan Management Tool für grosse VPN Server

Studenten: Raffael Fischer, Daniel Kolb, Josip Valencic

Betreuer: Prof. Dr. Andreas Steffen

Ausgabe: Montag, 19. September 2016

Abgabe: Freitag, 23. Dezember 2016

Ausgangslage

Im Frühlingssemester 2016 wurde im Rahmen einer HSR Bachelorarbeit die strongMan [1] Web-Anwendung auf der Basis Django/Python erstellt, welche es einem Benutzer ohne Spezialkenntnisse erlaubt, mühelos eine einfache IPsec Verbindung auf einem strongSwan VPN Client zu konfigurieren und zu starten. Die Kommunikation zwischen dem strongMan Management-Tool und dem strongSwan Daemon geschieht über das JSON-artige Versatile IKE Control Interface (VICI) [2].

Aufgabenstellung

In der ausgeschriebenen Arbeit soll strongMan für professionelle VPN-Server-Anwendungen erweitert werden. Das Ziel ist es, komplexe VPN-Netzwerk-Topologien aufsetzen zu können und mit Hilfe von Such- und Filter-Funktionen mehrere Hundert gleichzeitige VPN Tunnel zu überwachen und zu managen. Die Kommunikation zwischen strongMan und dem strongSwan VPN Gateway geschieht über die schon existierende Python VICI-Schnittstelle, so dass keine Neuentwicklung anfällt.

Ziele

- Der strongMan Server Modus soll die folgenden zwei Use Cases unterstützen:
  1. Passiver Responder in einem Remote Access Szenario
     mit einem `remote.id = %any` Connection-Template für eine beliebige Anzahl von Clients und folgenden Client/Server Authentisierungsarten:
     - Client Zertifikat / Server Zertifikat
       `remote.auth=pubkey` / `local.auth=pubkey`
     - Client EAP Username/Passwort / Server Zertifikat
       `remote.auth=eap-md5|eap-mschapv2|eap-ttls|eap-peap` / `local.auth=pubkey`
     - Client EAP Zertifikat / Server EAP Zertifikat (Mutual-EAP-only)
       `remote.auth=eap-tls|eap-ttls` / `local.auth=eap-tls|eap-ttls`
     - Client EAP / Server Zertifikat (Weiterleitung an Radius-Server)
       `remote.auth=eap-radius|eap-ttls` / `local.auth=pubkey`

2. Aktiver Initiator in einem Site-to-Site Szenario mit vorgegebenen `remote_addrs` / `remote.id` und folgenden Local/Remote Authentisierungsarten:
   - Lokales Zertifikat / Remote Zertifikat
     `local.auth=pubkey` / `remote.auth=pubkey`
   - Lokales EAP Zertifikat / Remote EAP Zertifikat (Mutual-EAP-only)
     `local.auth=eap-tls|eap-ttls` / `remote.auth=eap-tls|eap-ttls`

- Es sollen mehrere Remote-Access und/oder Site-to-Site Szenarien aufgesetzt werden können.
- Eine Site-to-Site Verbindung soll gestartet und gestoppt werden können.
- Jedes gespeicherte Szenario soll aktiviert/deaktiviert werden können.
- Der strongMan Server Modus soll die Konfiguration der `swanctl.conf` Parameter gemäss Appendix A unterstützen.
- Es sollen EAP Passwörter für eine beliebige Anzahl Remote-Access Benutzer verwaltet werden können. Mit der Einstellung `remote.auth = eap-radius` kann die User-Authentisierung an einen externen Radius-Server übergeben werden.
- Es sollen beliebig viele Pools für die Vergabe von IPv4 und/oder IPv6 Adressen definiert werden können (Pool-Name, Adressbereich, sowie Attribute wie z.B. DNS Server). Mit den reservierten Pool-Namen `pools = radius` und `pools = dhcp` soll die Pool-Verwaltung an einen externen Radius-Server, respektive DHCP-Server übergeben werden sollen.
- Wenn der strongSwan Daemon neu gestartet wird, sollen automatisch alle aktivierten, durch strongMan verwalteten Server-Szenarien, sowie die Pool-Definitionen und EAP Secrets via VICI-Schnittstelle in den Daemon geladen werden.
- Für jede aufgebaute IKE SA soll die Verbindungsinformation, die via den `list-sas` VICI Befehl angefordert werden kann, in einer Übersichtszeile visualisiert werden, die entweder aufgeklappt oder in einem separaten Fenster alle Details darstellt.
- Es soll mit einer Filterfunktion gezielt nach einzelnen oder einer Gruppe von IKE und CHILD SAs gesucht werden können. Mögliche Filterparameter sind IP Adresse, Zugeteilte Virtual IP Addresse, Identität, Verbindungsstatus, etc.
- Es sollen gezielt einzelne IKE oder CHILD SAs beendet werden können.

Optionale Ziele

- Implementierung der IKEv1 XAUTH-RSA Authentisierungsmethode.
  `remote.auth=pubkey, remote-xauth=xauth` / `local.auth=pubkey`
- Implementierung einer «History-Funktion» durch Speicherung der `ike-updown` und `child-updown` VICI Events. Dieses Feature bedingt wahrscheinlich einen ständig laufenden strongMan History-Daemon, der die entsprechenden VICI Events abonniert hat.
- Visualisierung der History mit Hilfe von Filterfunktionen.

Rapperswil, 19. September 2016

Prof. Dr. Andreas Steffen

Links

[1] strongMan Repository auf GitHub
https://github.com/strongswan/strongMan

[2] Spezifikation der strongSwan VICI Schnittstelle
https://github.com/strongswan/strongswan/blob/master/src/libcharon/plugins/vici/README.md

[3] VICI Konfigurationsbeispiele (benutzt swanctl.conf Konfigurationsdatei)
https://www.strongswan.org/testing/testresults/swanctl/

[4] strongTNC Policy Manager als beispielhafte Django Anwendung
https://github.com/strongswan/strongTNC

[5] RFC 7296 Internet Key Exchange Protocol Version 2 (IKEv2)
https://tools.ietf.org/html/rfc7296

Appendix A - Unterstützte swanctl.conf Parameter

- connections.<conn>.version (0, 1, 2)
- connections.<conn>.remote_addrs
- connections.<conn>.pools
- connections.<conn>.send_certreq
  if eap_only is used or remote certs are stored locally
- connections.<conn>.local<suffix>.round
- connections.<conn>.local<suffix>.certs
- connections.<conn>.local<suffix>.id
- connections.<conn>.local<suffix>.auth
  pubkey, eap-tls, eap-ttls, eap-peap
- connections.<conn>.remote<suffix>.round
- connections.<conn>.remote<suffix>.certs
- connections.<conn>.remote<suffix>.id
- connections.<conn>.remote<suffix>.auth
  pubkey, eap-md5, eap-mschapv2, eap-tls, eap-ttls, eap-peap, eap-radius
- connections.<conn>.remote<suffix>.cacerts
- connections.<conn>.children.<child>.local_ts
- connections.<conn>.children.<child>.remote_ts
- secrets.eap<suffix>.secret
- secrets.eap<suffix.id<suffix>
- pools.<name>.addrs
- pools.<name>.<attr>

Optionale Parameter:

- connections.<conn>.unique
- connections.<conn>.dpd_delay
- connections.<conn>.children.<child>.dpd_action
- connections.<conn>.remote<suffix>.auth = xauth

Teil I

Technischer Bericht

3. Einleitung und Übersicht

3.1. Vorbemerkung

In diesem Dokument ist die Bachelorarbeit von Raffael Fischer, Daniel Kolb und Josip Valencic dokumentiert, welche an der Hochschule für Technik Rapperswil (HSR) im Herbstsemester 2016 absolviert wurde. Die Arbeit wurde betreut duch Prof. Dr. Andreas Steffen (Institutsleiter Institut für Internettechnologien und Applikationen (ITA) und Dozent für Sicherheit und Kommunikation).

3.2. Aktuell

3.2.1. strongSwan

strongSwan[14] ist eine Open-Source, Internet Protocol Security (IPsec) basierte VPN Lösung für verschiedene Betriebssysteme. Es wird standardmässig per Konfigurationsdateien und Terminalanwendung verwaltet. Diese Vorgehensweise richtet sich hauptsächlich an versierte Systemadministratoren. Seit einiger Zeit steht nun eine einheitliche Schnittstelle für strongSwan (Versatile Internet Key Exchange (IKE) Control Interface (VICI)[2]) zur Verfügung und ermöglicht die Steuerung von strongSwan aus diversen verschiedenen Programmiersprachen.

3.2.2. strongMan

strongMan[10] ist ein Management Interface für strongSwan. Dadurch, dass strongMan auf Python und Django[4] basiert, bietet es eine moderne, benutzerfreundliche grafische Oberfläche und ist zudem Webbasiert. In besagtem Webinterface kann der Anwender IPsec-Verbindungen konfigurieren, verwalten und aufbauen. Unterstützt wird:

- RSA / ECDSA Public Key Authentisierung
- EAP mit Benutzername und Passwort
- EAP-TLS mit Zertifikaten
- mehrere Authentisierungsrunden

Die Applikation bietet momentan verschiedenste gebräuchliche Anwendungsfälle, welche nach simplem Aufsetzen von strongMan nur noch mit wenigen Klicks konfiguriert werden können.

3.3. Vision

Die Grundidee dieser Arbeit war, strongSwan um eine Benutzeroberfläche für versierte Systemadministratoren zu erweitern. Diese soll trotz grosser Last mit vielen VPN-Verbindungen weiterhin übersichtlich und performant bleiben. Ausserdem soll weitere Server-spezifische Funktionalität angeboten werden.

> *"Es ist ein Administratormodus sinnvoll, der weitere Authentisierungsmethoden ermöglicht, die Konfigurationsmöglichkeiten ausbaut, zusätzliche Statusinformationen visualisiert und das Management von mehreren Security Associations (SA) implementiert"* [11]

In der weiteren Dokumentation der Arbeit wird konkret auf die Planung der Architektur, die aufgetretenen Probleme und deren Lösungsansätze und das daraus entstandene Produkt eingegangen.

4. Ausgangslage

4.1. Komponenten

Folgende Komponenten werden in diesem Projekt verwendet:

4.1.1. strongMan

Das bestehende strongMan-Projekt, welches für den Client einer VPN-Verbindung entwickelt wurde, wird in dieser Arbeit als Grundlage verwendet um ein Server-Management-Interface anbieten zu können.

4.1.2. Webbrowser

Die Applikation wird dem Benutzer als Web-Applikation in seinem Browser präsentiert. Die Darstellung geschieht mit HTML und CSS, unterstützt durch JavaScript. Damit die Applikation auf unterschiedlichen Bildschirmauflösungen optimal präsentiert wird, und Templates für die Darstellung verwendet werden können, wird das quelloffene Frontend Web Framework Bootstrap[1] verwendet.

4.1.3. Django Server

Der Django Server ist quasi in der Mitte der Kommunikation positioniert. Neben des Verarbeitens der Business Logik und dem Anbieten der Daten an den Browser, liegt seine Aufgabe unter anderem auch in der Kommunikation mit der VICI-Schnittstelle und in der Persistierung der Daten.

4.1.4. VICI-Schnittstelle

Die VICI-Schnittstelle bietet eine einfache Abstraktion für strongSwan. Die Schnittstelle kann direkt in Python angesprochen werden und lädt Daten aus oder schreibt Daten und Konfigurationen in den strongSwan-Daemon. Sie erlaubt deshalb dem Entwickler, ohne detailliertere Kenntnisse des Daemons, und auch ohne wechseln der Programmiersprache, eine Kommunikation via Unix-Socket mit dem Backend. Die VICI-Schnittstelle wird zusätzlich in einen Wrapper gepackt, um die Abstrahierung deutlicher und einfacher zu gestalten.

4.1.5. strongSwan

strongSwan ist ein Daemon, welcher als VPN-Backend fungiert. Es kümmert sich um den Verbindungsaufbau der IPsec-Tunnels, und wird über die VICI-Schnittstelle angesprochen und konfiguriert.

4.1.6. Database

Die SQLite Datenbank[3] persistiert jegliche Konfigurationen, welche mit strongMan erstellt wurden, sowie Verbindungen, Pools, Zertifikate und auch die EAP Secrets. Da die Secrets sensitive Daten beinhalten, werden diese verschlüsselt abgelegt.

4.2. Sprachen / Technologien

4.2.1. Python

> *Python ist eine universelle, üblicherweise interpretierte höhere Programmiersprache. Sie will einen gut lesbaren, knappen Programmierstil fördern. Python unterstützt mehrere Programmierparadigmen, z. B. die objektorientierte, die aspektorientierte und die funktionale Programmierung. Ferner bietet es eine dynamische Typisierung. Wie viele dynamische Sprachen wird Python oft als Skriptsprache genutzt.* [21]

4.2.2. Django

> *Django ist ein in Python geschriebenes, quelloffenes Web Application Framework, das einem Model-View-Presenter-Schema folgt.* [18]

4.3. System Kontext

strongMan läuft beim Client im Browser und kommuniziert mit der strongMan-Serverkomponente, welche in Python/Django geschrieben ist. Die Datenhaltung ist mit einer SQLite Datenbank realisiert. Um mit dem dem strongSwan-Daemon zu kommunizieren wird der VICI Wrapper, welcher ebenfalls in Python implementiert ist, angesprochen. Diese spricht dann den Daemon via Unix Socket an.

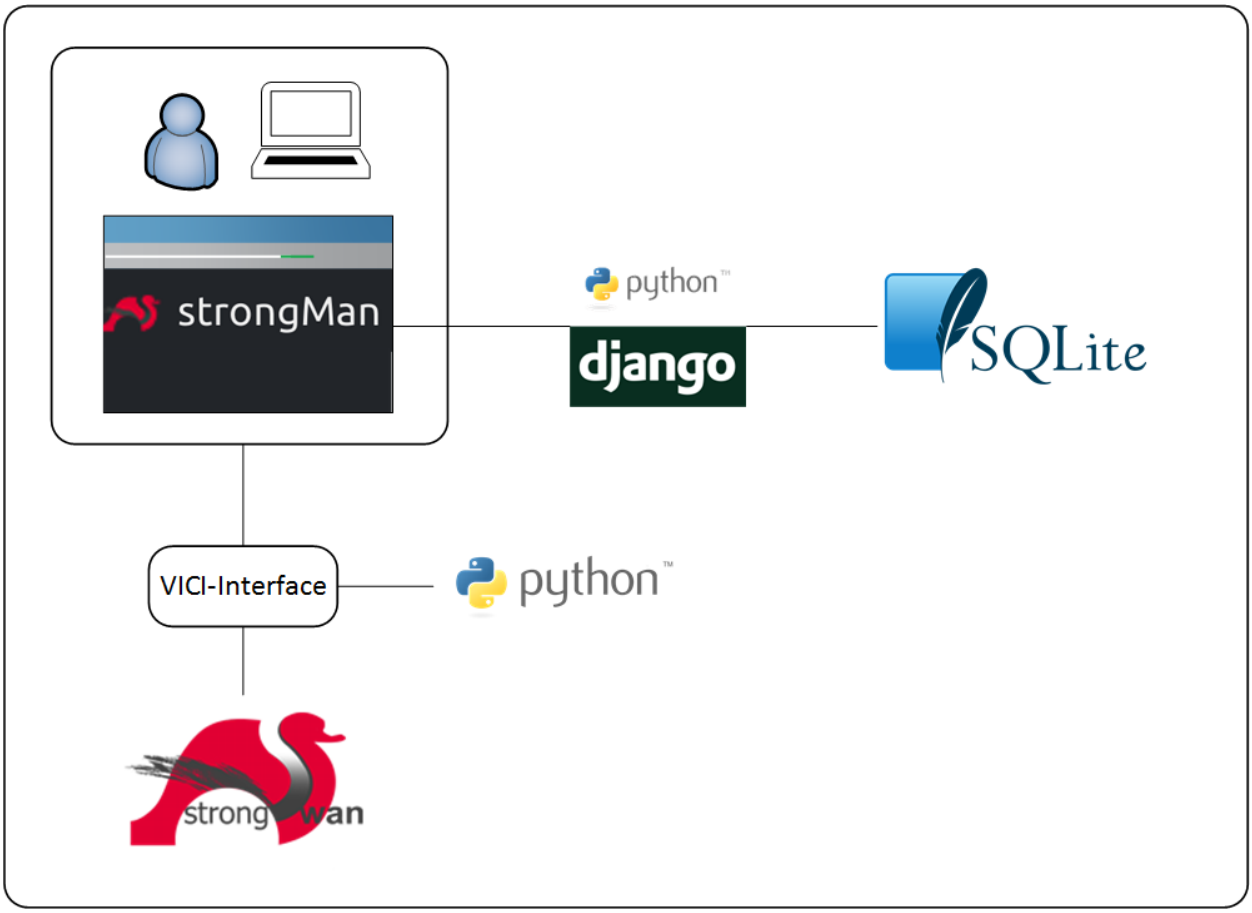


Abbildung 4.1.: Systemübersicht

4.4. Logische Architektur

strongMan ist nach dem MVC-Prinzip[20] aufgebaut. MVC beschreibt ein Model - View - Controller Layering der Architektur. Das Model steht hierbei für das Modell der Datenhaltung, der Controller für die Businesslogik, und die View für die Präsentation der aufbereiteten Daten. Die MVC-Schichtung wird in Django mit 'model', 'templates', 'views' beschrieben, wobei sich 'templates' auf die grafische Oberfläche, 'views' auf die Controller, oder auch Handler, und 'model' auf das Datenbankmodell, bezieht.

Dieses Layering existiert einmal für jede 'app'. Eine 'app' entspricht grundsätzlich einem Reiter im Menu der Benutzeroberfläche. Hiermit garantieren wir mit einem simplen Ansatz das S und O des SOLID-Principles[9] wie folgt:

Single Responsibility Principle
Die Verantwortlichkeiten einer Applikation sind strikt in sich gehalten, wobei die Business-Logik klar innerhalb einem Package der App gehalten wird.
Open/Closed Principle
Eine App ist in sich geschlossen und kann nach belieben ein- oder ausgeschaltet werden. Gegen aussen ist die ganze Applikation aber jederzeit offen für neue Apps.

Eine grobe Übersicht der Architektur mit den Apps und den Helpers ist in Abbildung 7.1 im Kapitel 7: Architektur ersichtlich.

4.5. Deployment

Das Deployment der Applikation wird beibehalten und läuft wie in Abbildung 4.2 ersichtlich ist ab. Sie wird auf einem Gunicorn-Webserver[7] unter Linux installiert und basiert auf dem Django-Framework für Python-Applikationen. Für die Persistierung wird eine leichtgewichtige SQLite-Datenbank verwendet. Über das HTTP Protokoll wird die Applikation dann dem Anwender im Browser präsentiert.

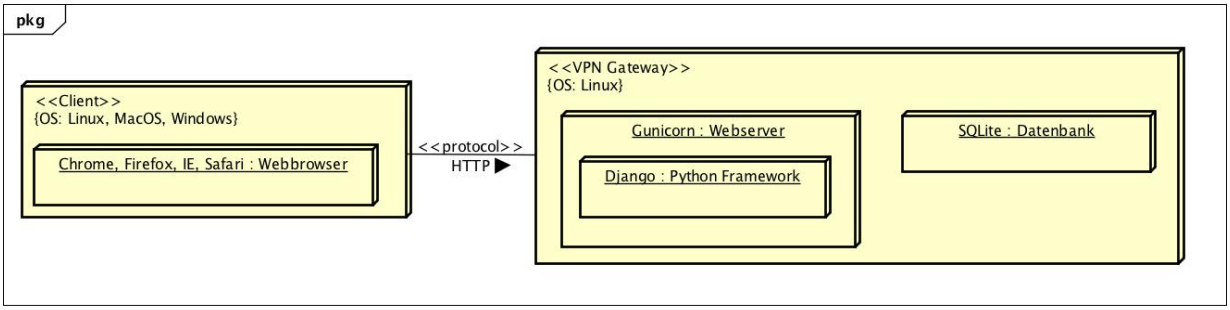
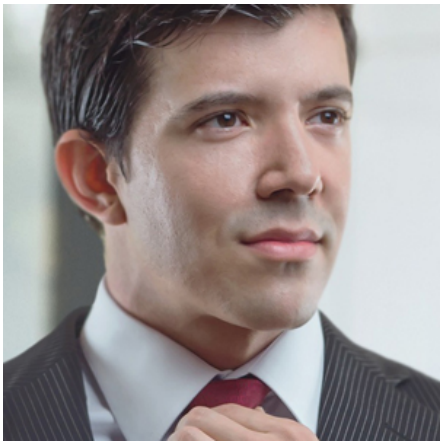


Abbildung 4.2.: Deployment des strongMan

5. Anforderungen

5.1. Benutzer und Personas

Persona Hans



Hans ist 35 Jahre alt und lebt in Zürich.

Er ist seit 9 Jahren Systemadministrator, wieder bei der Firma, wo er vor seinem Informatikstudium an der HSR seine Lehre absolviert hat. Hans ist seit zwei Jahren verheiratet und erwartet mit seiner Frau Claudia eine Tochter.

Technisches Verhalten Er ist nun seit ein paar Jahren verantwortlich für die VPN Funktionalität in seiner Firma. Vor seiner Zeit verwendete die Firma ein kommerzielles Produkt, jedoch hat sich Hans damals bei der Übernahme des Amtes für strongSwan entschieden. Seinen Anwendern hat er mittlerweile schon von strongMan erzählt, diese freuen sich auch schon auf dessen Einführung. Hans kommt ganz gut klar mit der Konsole, muss aber vor allem für spezifische Konfigurationen in der Dokumentation von strongSwan nachlesen gehen. Ausserdem fällt es ihm schwer, sich Übersicht über die mittlerweile mehrere Dutzend Verbindungen zu verschaffen.

Ziele Er liebäugelt deshalb seit einiger Zeit mit der Vorstellung, wie es wohl wäre, wenn ihm seine Arbeit auch durch eine moderne Benutzeroberfläche erleichtert werden würde, wie er dies bei strongMan sehen durfte.

5.2. Funktionale Anforderungen

5.2.1. Use Cases

Aus der Aufgabenstellung ergibt sich folgendes Use Case Diagramm. (Abb. 5.2)

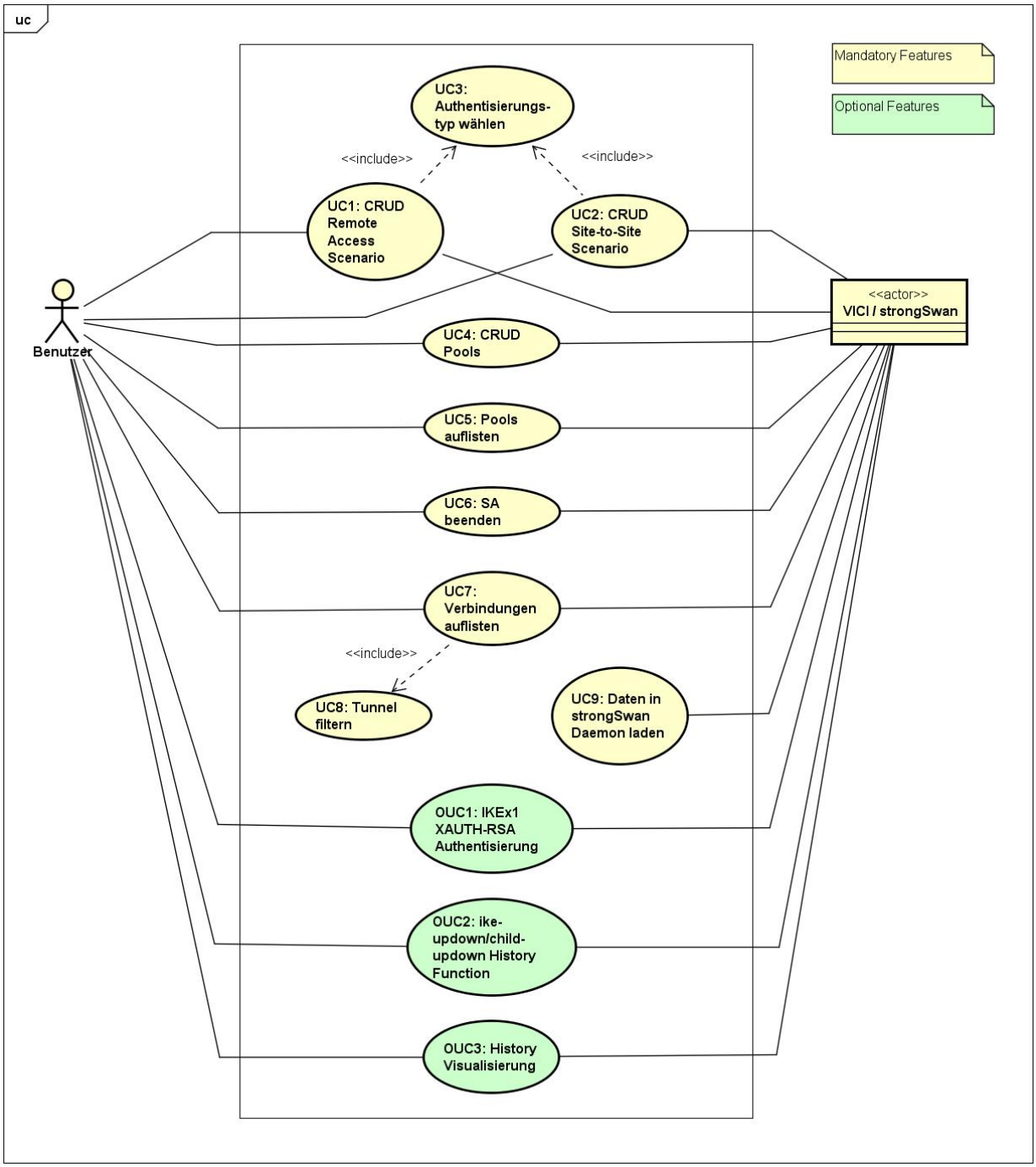


Abbildung 5.2.: UseCase Diagramm

UC1: CRUD Remote Access Scenario Der Benutzer möchte Verbindungen für ein Remote Access Scenario verwalten können, wobei der Server ein Passive Responder ist und Verbindungen von Clients akzeptieren soll.

UC2: CRUD Site-to-Site Scenario Der Benutzer möchte Verbindungen für ein Site-to-Site Scenario verwalten können. Dabei kann der Server entweder die Rolle eines Active Initiators oder eines Passive Responders übernehmen. Er initiiert also entweder die Verbindung zu einem anderen VPN-Gateway,

oder akzeptiert eine solche.

UC3: Authentisierungstyp wählen Der Benutzer möchte auswählen, welcher Authentisierungstyp verwendet wird.

Für Remote Access Scenario:

- Client Zertifikat / Server Zertifikat
- Client EAP Username/Passwort / Server Zertifikat
- Client EAP Zertifikat / Server EAP Zertifikat (Mutual-EAP-only)
- Client EAP / Server Zertifikat (Weiterleitung an Radius-Server)

Für Site-to-Site Scenario:

- Lokales Zertifikat / Remote Zertifikat
- Lokales EAP Zertifikat / Remote EAP Zertifikat (Mutual-EAP-only)

UC4: CRUD Pools Der Benutzer möchte Pools für folgende Zwecke verwalten können:

- Virtuelle IP Addressen konfigurieren
- DNS Server angeben
- DHCP Server angeben
- NBNS Server angeben
- Netzmasken konfigurieren
- Server Adresse konfigurieren
- Subnetze konfigurieren
- Split Includes / Excludes angeben

Die Pools werden als Strings dem VICI übergeben.

UC5: Pools auflisten Der Benutzer möchte die Pools, die vom strongSwan Daemon geladen werden, auflisten können.

UC6: Child_SA beenden Der Benutzer möchte gezielt einzelne IKE Security Associations (IKE SAs) oder Child Security Associations (Child SAs) beenden.

UC7: Verbindungen auflisten Der Benutzer möchte alle vorhandenen Verbindungen auflisten.

UC8: Tunnel filtern Der Benutzer möchte Filter auf IP-Addresse, Identity und gegebenfalls weitere Attribute setzen können, um schneller einen Tunnel zu finden.

UC9: Daten in StrongSwan Daemon laden Wenn der strongSwan Daemon neu gestartet wird, sollen automatisch alle aktivierten, durch strongMan verwalteten Server-Szenarien, sowie die Pool-Definitionen und EAP Secrets via VICI in den Daemon geladen werden.

5.2.2. Optional Use Cases

OUC1: IKEx1 XAUTH-RSA Authentisierung Implementierung der IKEx1 XAUTH-RSA Authentisierungsmethode.

OUC2: ike-updown/child-updown History Function Implementierung einer History-Funktion durch Speicherung der 'ike-updown' und 'child-updown' VICI-Events mithilfe eines History-Daemon.

OUC3: History Visualisierung Visualisierung der History mithilfe von Filterfunktionen.

5.2.3. Connection Parameters

connections.<conn>.version Die verwendete IKE Version. (0 = beide, 1 = IKEv1, 2 = IKEv2)

connections.<conn>.remote_addrs Die IP-Adresse des VPN-Servers, wenn local der Initiator ist. Muss nicht gesetzt werden, wenn local der Passive Responder ist, es sei denn, es soll eine klar definierte Site-to-Site Verbindung zwischen local und anderen aufgebaut werden. Dann dient die Remote Address als Einschränkung. Kann auch eine Range oder ein Subnetz sein.

connections.<conn>.pools IP-Pools für die Vergabe von internen Adressen. Pools werden in der Pools-Section global definiert und mit deren Namen identifiziert und referenziert. Es können auch weitere Konfigurationen für die Teilnehmer innerhalb des Pools, wie z.B. ein DNS-Server, angegeben werden. Reserved Words, wie z.B. DHCP und Radius, sollen nicht für benutzerspezifische Pool-Konfigurationen verwendet werden.

connections.<conn>.send_certreq Boolescher Wert, ob ein Certificate-Request gesendet werden soll. Kann 'false' sein, wenn eap_only oder die Remote-Zertifikate lokal gespeichert wurden.

connections.<conn>.local<suffix>.round Ganzzahlige Nummer, um die Reihenfolge der Authentification Rounds zu definieren. Mehrere Authentifizierungsverfahren können so nacheinander einsortiert werden.

connections.<conn>.local<suffix>.certs Eine Liste der lokal gespeicherten Zertifikaten.

connections.<conn>.local<suffix>.id Die Identität des Clients. Diese ist, sofern die lokale IP-Adresse oder die E-Mail Adresse nicht im Zertifikat gesetzt ist, der Distinguished Name. Ansonsten wird entweder die IP-Adresse oder E-Mail Adresse verwendet.

connections.<conn>.local<suffix>.auth Authentifizierungsmethoden, die akzeptiert werden. Z.B. pubkey, eap-tls, eap-ttls, eap-peap. Diese Einstellung wird grundsätzlich nicht dem User überlassen. Der Server sendet EAP-Proposals, wenn er initial keinen Auth-Payload erhält. Antwortet der Client jedoch mit einem 'NAck' (Not Acknowledged), wird keine Verbindung aufgebaut. Ansonsten authentifiziert sich der Server mit seinem public-key und der Client kann danach z.B. MSChapV2 verwenden, welches zwar in sich selbst als unsicher gilt, aber hier nicht mehr massgeblich ist, da eine Man-in-the-Middle Attacke durch das public-key-Verfahren bereits nicht mehr möglich ist.

connections.<conn>.remote<suffix>.round Analog 'round' für lokale Einstellung.

connections.<conn>.remote<suffix>.certs Wird nur im Spezialfall, z.B. bei einer Site-to-Site Verbindung, verwendet. Dann können die Zertifikate auch self-signed sein und bei beiden Endpunkten lokal gespeichert werden. Es muss dann nur die Signatur und die Identität gesendet werden. Die Prüfung zur Bestimmung der Richtigkeit des Zertifikates geschieht, indem die Identität verglichen wird.

connections.<conn>.remote<suffix>.id Analog 'id' für lokale Einstellung.

connections.<conn>.remote<suffix>.auth Analog 'auth' für lokale Einstellung.

connections.<conn>.remote<suffix>.cacerts Eine Liste von Certificate Authority (CA)-Zertifikaten. So kann vorgegeben werden, welche CA das Zertifikat ausgestellt haben muss.

connections.<conn>.children.<child>.local_ts Definition des Traffic Selectors für eine Child SA. Diese ist eine Beschreibung von Einem oder mehreren Netzen. Es können aber keine Ranges angegeben werden. Optional könnte man noch eine Protokoll-Angabe implementieren, wo z.B. zwischen UDP oder TCP gewählt werden kann.

connections.<conn>.children.<child>.remote_ts Definition des Remote-Traffic Selectors für eine Child SA. TS beschreiben die gekoppelten Netze für bestimmte Child SAs. Die remote_ts-Adresse ist eine virtuelle Adresse, welche innerhalb des Netzes von dem Routing verwendet werden kann.

connections.<conn>.children.<child>.start_action Action, welche nach dem Laden der Verbindung ausgeführt wird. Bei einem Unload der Verbindung oder beim Ändern einer Konfiguration mit gesetzter start_action, wird die Umkehr-Action ausgeführt. Konfigurationen mit `"start"` werden geschlossen und bei denen mit `"trap"` [15] wird die trap policy deinstalliert.

5.3. Nichtfunktionale Anforderungen

5.3.1. Kompatibilität

Synopsis	Die Applikation muss in den meist verwendeten Browsern mit aktueller Version (Chrome ab Version 55, Firefox ab Version 45, Internet Explorer ab Version 11) übersichtlich mit voller Funktionalität angezeigt werden.
Relevante QA	Compatibility
Messbarkeit	Die Applikation bietet alle funktionalen Anforderungen, die implementiert wurden, in den besagten Browsern an. Es wird ein User Acceptance Test durchgeführt.

5.3.2. Fehlertoleranz

Synopsis	Werden nicht erlaubte Inputs der Applikation übergeben, wird eine sinnvolle Fehlermeldung und ein Abfangen der Eingabe mithilfe Input-Validierung (z.B. Regex) erwartet. Um das Problem einzuschränken wird so oft wie möglich auf Auswahllisten anstatt Plain Text User Input gesetzt.
Relevante QA	Fault Tolerance
Messbarkeit	Im Suchfeld und anderen Eingabefeldern werden willkürliche Zeichen eingegeben. Zudem wird ein Request abgefangen und verändert und somit die Server-Input-Validierung getestet.

5.3.3. Erweiterbarkeit

Synopsis	Das System soll einfach mit weiteren Apps erweitert werden können.
Relevante QA	Extensibility
Messbarkeit	Der Implementationsaufwand weiterer ‘Apps‘ soll erheblich kleiner sein als der Ersten.
Offene Probleme	Kann nicht gemessen werden, da wir bereits auf der bestehenden Applikation strongMan aufbauen. Es kann jedoch gesagt werden, dass der Aufwand für das Hinzufügen einer App sehr gering ausfiel, da die Architektur des Produktes gut auf Erweiterbarkeit ausgelegt wurde.

5.3.4. Sicherheit

Synopsis	Die Sicherheit wird garantiert durch den Ort der Datenhaltung. Ausserdem wird Input Validierung client-side, sowie serverseitig vollzogen. Sensible Daten, wie zum Beispiel Passwörter, welche in der SQLite Datenbank persistiert werden, sind verschlüsselt. Ausserdem wird das Backend nicht durch einen Netzwerk-Socket, sondern lediglich durch einen Unix-Socket angesprochen.
Relevante QA	Security
Messbarkeit	Passwörter und andere sensible Daten in der Datenbank sind nicht Klartext.

5.3.5. Performance

Performance der Liste

Synopsis	Die neue Komponente verwaltet eine grosse Anzahl von Verbindungen. Es wird erwartet, dass keine nennenswerten Wartezeiten, um die Verbindungen aufzulisten, auftreten.
Relevante QA	Performance, Capacity
Messbarkeit	Die Wartezeit für das Anzeigen der Liste bei 1000 Verbindungen beträgt weniger als drei Sekunden.

Performance des Filters

Synopsis	Um die grosse Anzahl an Verbindungen übersichtlicher zu gestalten, wurde eine Filterfunktionalität implementiert.
Relevante QA	Performance, Capacity
Messbarkeit	Die Wartezeit für das Filtern der Liste bis zu 1000 Verbindungen beträgt weniger als drei Sekunden.

Die Tests und deren Ergebnisse sind im Kapitel 9.2: NFR Testing dokumentiert.

6. UI Concepts & Mockups

6.1. User Interface Design

Um eine gute User Experience zu garantieren, möchten wir uns an den Usability Heuristics von Nielson[8] orientieren.

Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
Es wird nicht nur im Fehlerfall, sondern auch im Erfolgsfall eine Rückmeldung gegeben, um den Status des Veränderten zu zeigen.

Match between system and the real world
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
Die verwendeten Begriffe des User Interfaces sind analog der Begriffe aus der bekannten strongSwan Konfiguration. Zusätzlich wird eine Eingabehilfe angeboten.

User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue.
Für alle Eingabeformulare existiert entweder ein 'Cancel'-Button oder man kann einfach über das Menu oder den Browser-Back-Button zurückkehren und somit die Änderungen verwerfen.

Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
Innerhalb des ganzen Projektes wird eine 'ubiquitous language'[5] verwendet. Ein Fachmann soll sich mit diesem Wortschatz mit seinen Kollegen austauschen können.

Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
Textfelder werden gemäss ihrem Typ designed und verifiziert. Pflichtfelder werden entsprechend markiert.

Recognition rather than recall
Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
Bereits Konfiguriertes wird bei Wiederverwendung in einem anderen Formular in einer Dropdown-Liste, und nicht als Textfeld, angeboten.

Flexibility and efficiency of use
Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such

that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
Definitionen von Pools und das Hochladen von Zertifikaten kann entweder direkt im Connection Setup Wizard oder in einem separaten Formular erledigt werden.

Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
Die Übersicht von Connections, Certificates oder auch Pools wird knapp gehalten. Bei Bedarf kann zu jedem der Einträge in derselben Form eine Detailansicht aufgeklappt werden oder während des Setups ein Hilfetext angezeigt werden. Ausserdem existiert für die Server Connections eine Read-Only-View, um schnell einen Überblick zu erhalten.

Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
Errormeldungen werden so präzis wie möglich angeboten und klar visuell von Erfolgs- oder Informationsmeldungen unterschieden.

Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.
Zusätzlich zu der Dokumentation wird ein User Guide (Anhang C) erstellt, welcher mithilfe von Screenshots die Use Cases aufzeigt.

6.2. UI Mockups

Die folgende Abbildung (Abb. 6.1) zeigt die Übersicht über die Server Connections. Die äusserste Tabelle listet die Connections auf und zeigt deren Typ, sowie andere wichtige Informationen. Eine Connection lässt sich hier direkt ein- und ausschalten.
Zudem lässt sich mithilfe einer aufklappbaren Subtabelle die Liste der IKE SA Verbindungen zeigen, und unter diesen die einzelnen Child SAs. In allen Tabellen lassen sich die entsprechenden Einträge jeweils terminieren und wieder aktivieren.
Auf der Ebene der Connection wird ausserdem ein Button für das Anzeigen der Read-Only-View angeboten, um sich einen schnellen Überblick der Konfigurationen der Verbindung zu erschaffen.

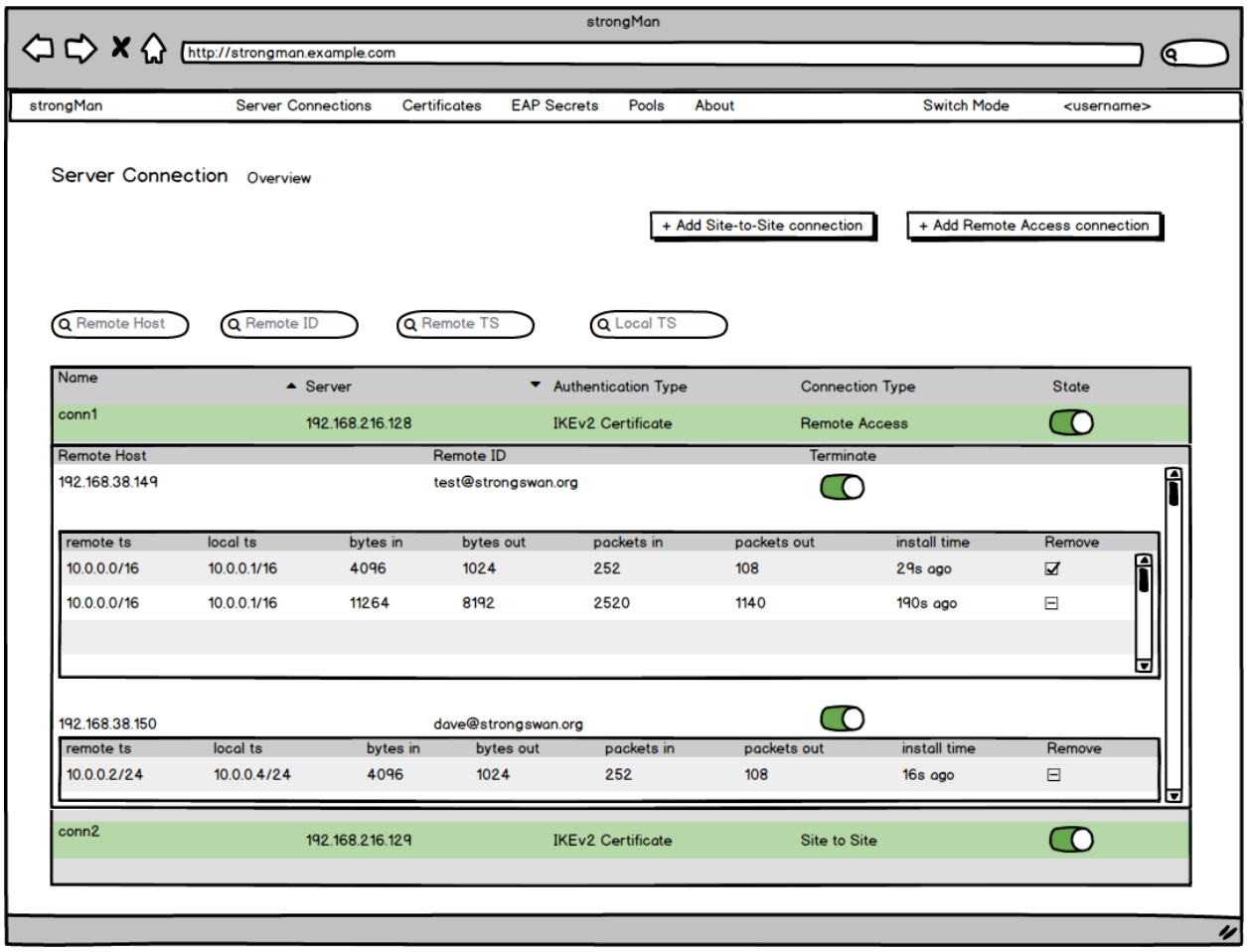


Abbildung 6.1.: Server Connection Übersicht

Abbildungen 6.2 und 6.3 zeigen den Prozess um eine Server Connection zu erstellen auf. Initial kann zwischen Site-to-Site Connection oder Remote Access Connection gewählt werden, wie in obigem Bild der Connection Overview ersichtlich ist. Danach führt der Wizard den Benutzer durch die nötigen Konfigurationen für die entsprechende Connection, wie sie in der Aufgabenstellung beschrieben wurden.

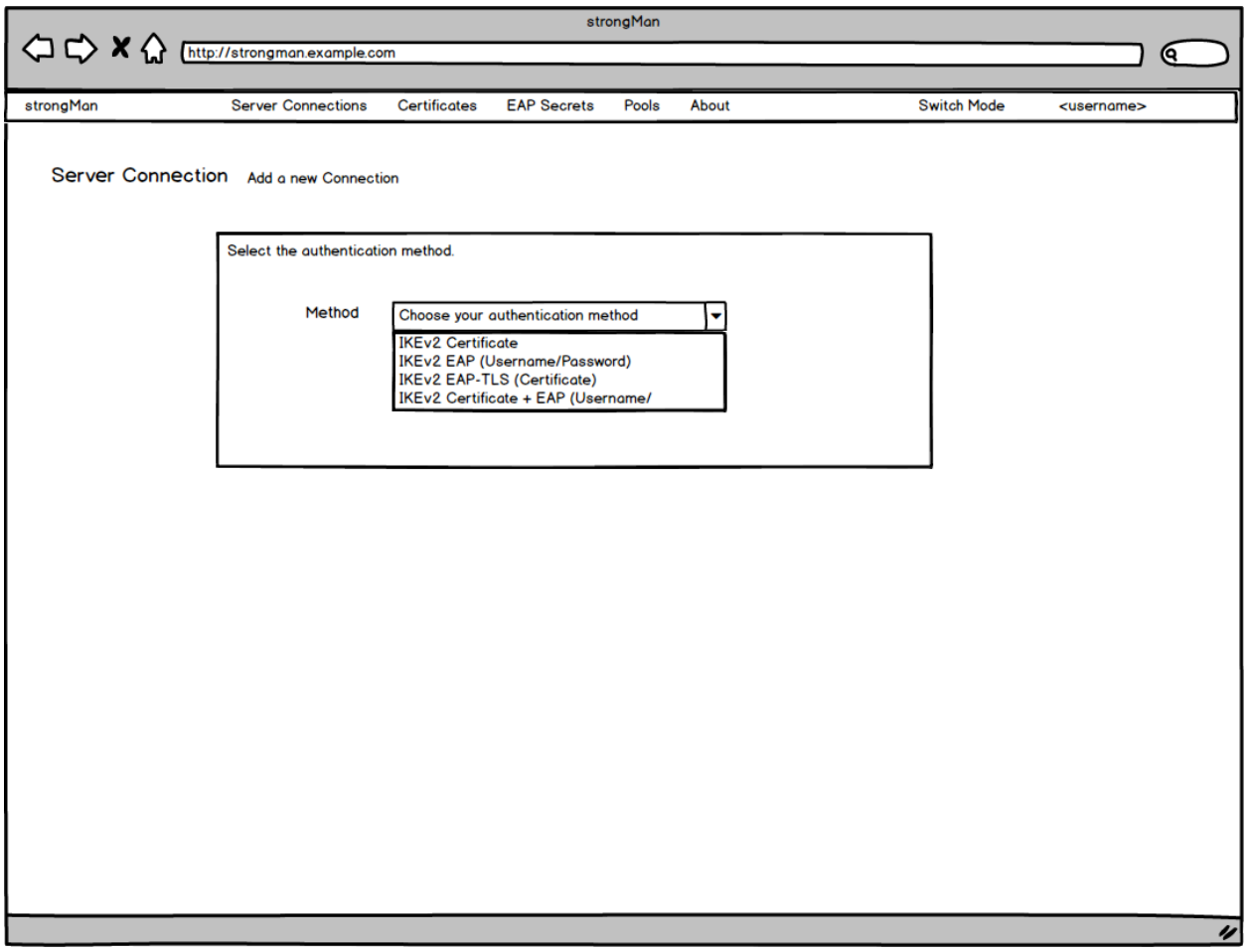


Abbildung 6.2.: Wählen der Authentisierungsmethode für eine neue Server Connection

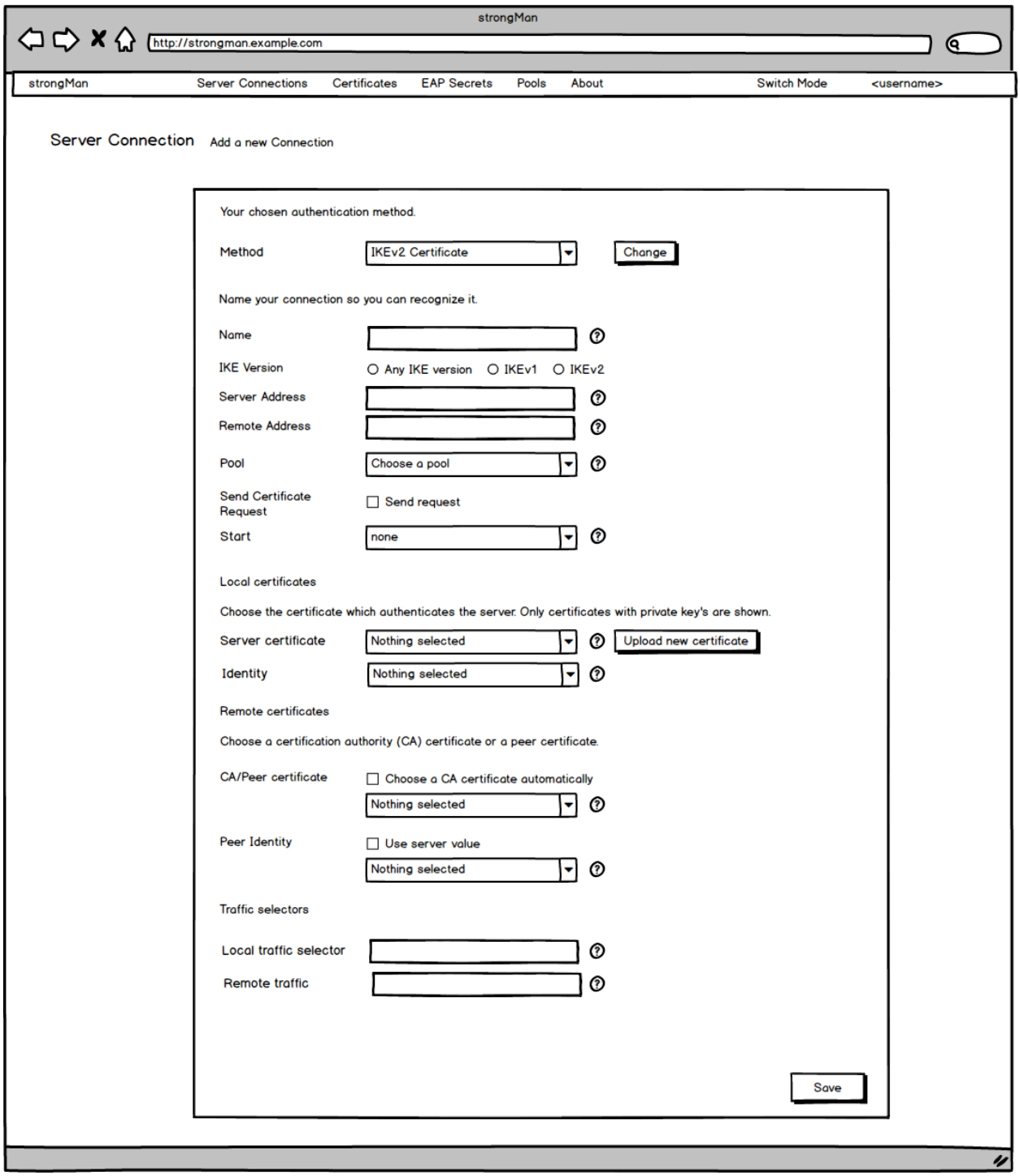


Abbildung 6.3.: Erstellen einer Server Connection

In der Abbildung der EAP Secrets Übersicht (Abb. 6.4) wird die Darstellung der Übersicht über die vorhandenen EAP Secrets visualisiert. Des Weiteren kann aus dieser Ansicht das Formular für das Hinzufügen und Ändern (Abb. 6.5) eines Secrets erreicht werden. Die Einträge lassen sich filtern und direkt löschen.

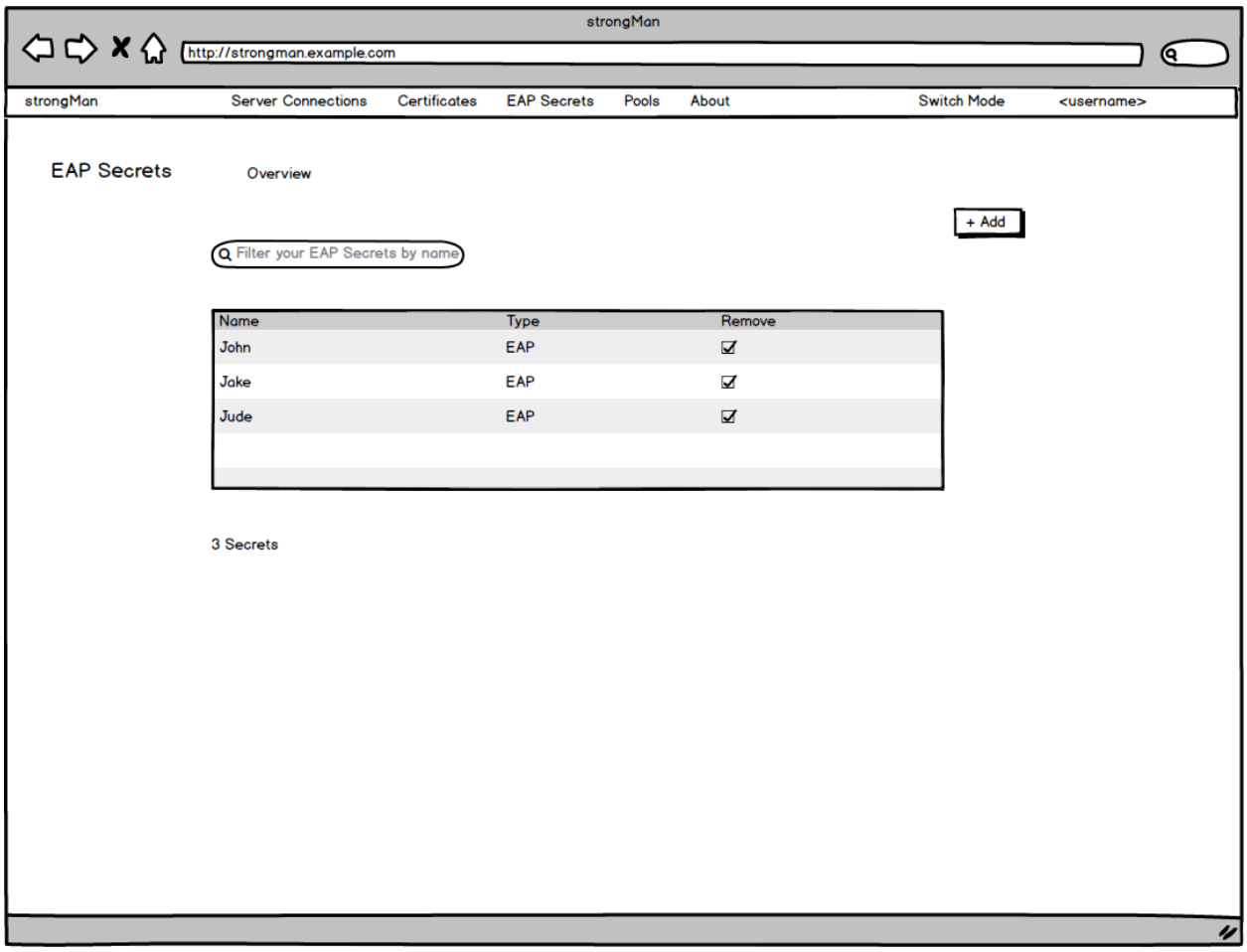


Abbildung 6.4.: EAP Secrets Übersicht

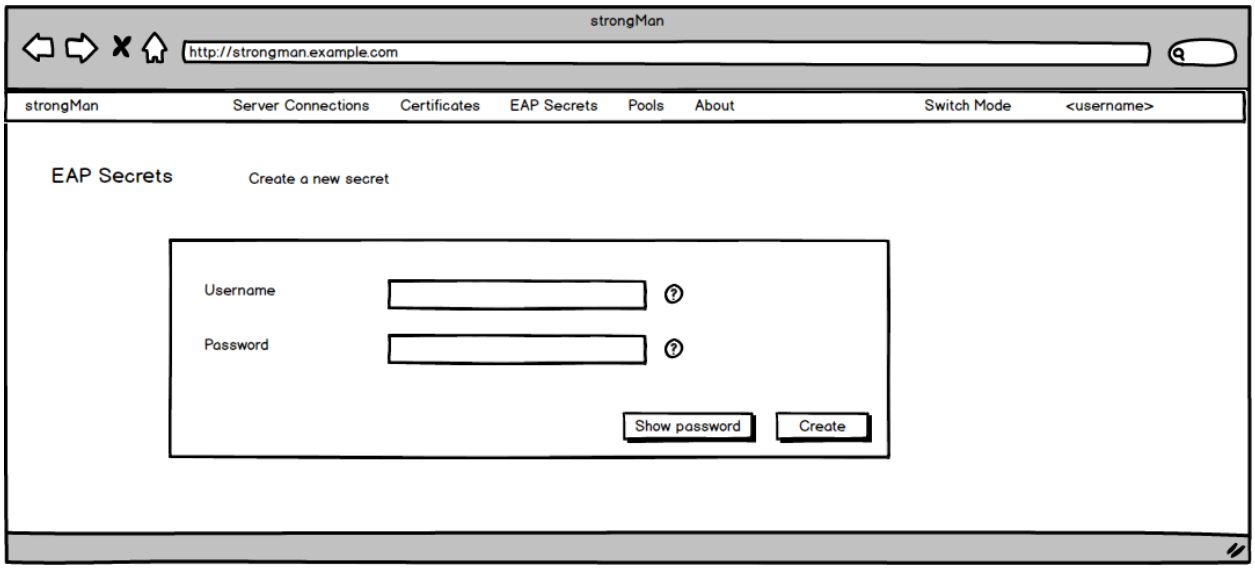


Abbildung 6.5.: EAP Secrets erstellen

Die Pool Übersicht (Abb. 6.6) präsentiert die Pools mit ihren wichtigsten Attributen. Für weitere Attribute, wie zum Beispiel die Live-Ansicht der Anzahl Leases und die Identitäten der Leases, kann

ein einzelner Pool aufgeklappt werden. Ausserdem lassen sich Pools direkt löschen, sofern sie nicht in Verwendung sind. Aus dieser Ansicht erreicht man das Formular zur Erstellung und Bearbeitung eines Pools (6.7).

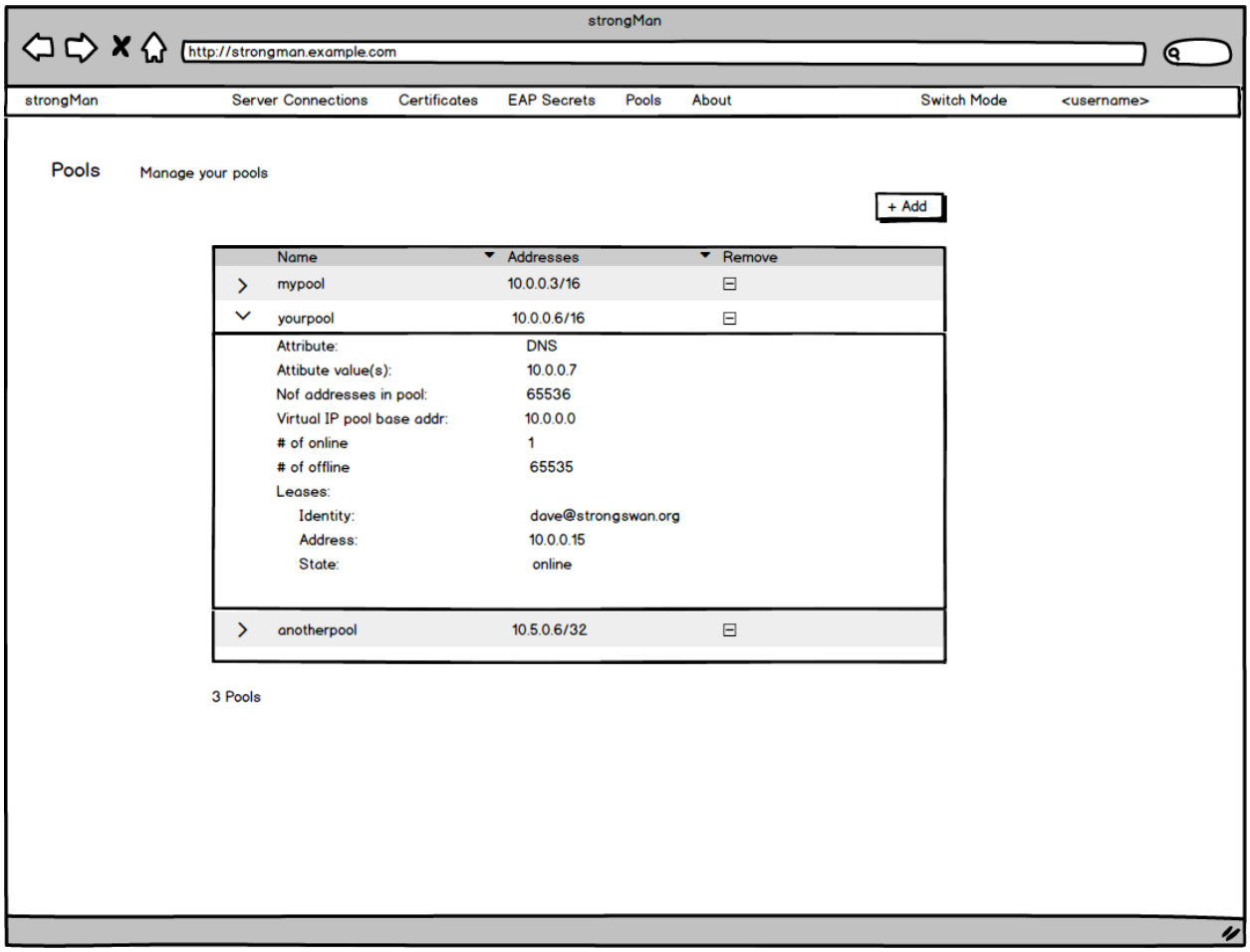


Abbildung 6.6.: Pools Übersicht

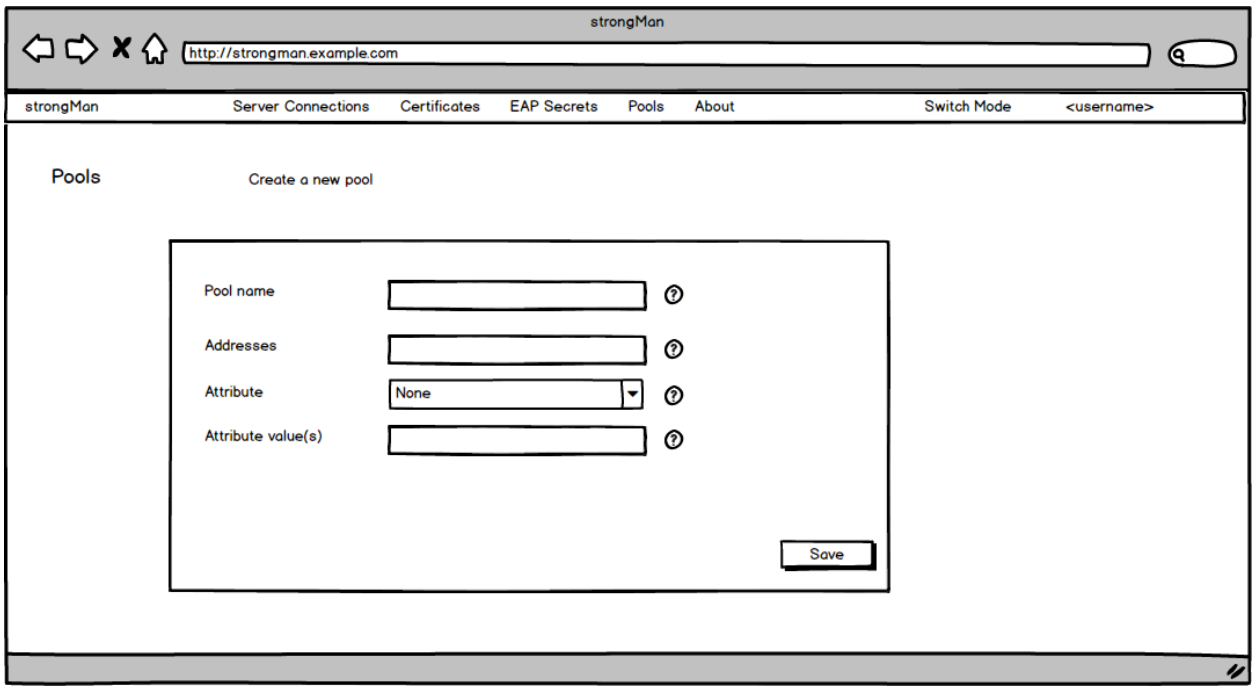


Abbildung 6.7.: Erstellen eines Pools

7. Architektur

7.1. Anforderungen

Aus den funktionalen und nicht funktionalen Anforderungen für das Gesamtsystem ergeben sich für den strongMan-Server folgende Architekturentscheidungen:

Wir möchten uns weiterhin an das MVC-Design halten, da dies für eine Webapplikation dieser Art sinnvoll ist und für Django Applikationen der Best Practice entspricht.
Neben den Apps wird ein Package ‘helper_apps‘ eingeführt, welches unter anderem die Packages ‘encryption‘ und ‘vici_wrapper‘ enthält. Dieses Package dient als Utility für alle Apps und wird deshalb nicht unter, sondern neben den ‘apps‘ einsortiert. Die bestehende VICI-Schnittstelle wurde bis anhin durch einen Wrapper angeboten, was wir beibehalten möchten.

In folgendem Diagramm (Abb. 7.1) wird eine Übersicht über das Zusammenspiel von Apps, deren Aufbau und den helper_apps, gezeigt. ’server_connections’, ’pools’ und ’eap_secrets’ sind Packages der Use-Cases aus diesem Projekt, während ’certificates’ und ’connections’ davor bereits existierten. Innerhalb jedes Packages, welches eine ’app’ darstellt, ist die MVC-Schichtung in weiteren Packages abgebildet.

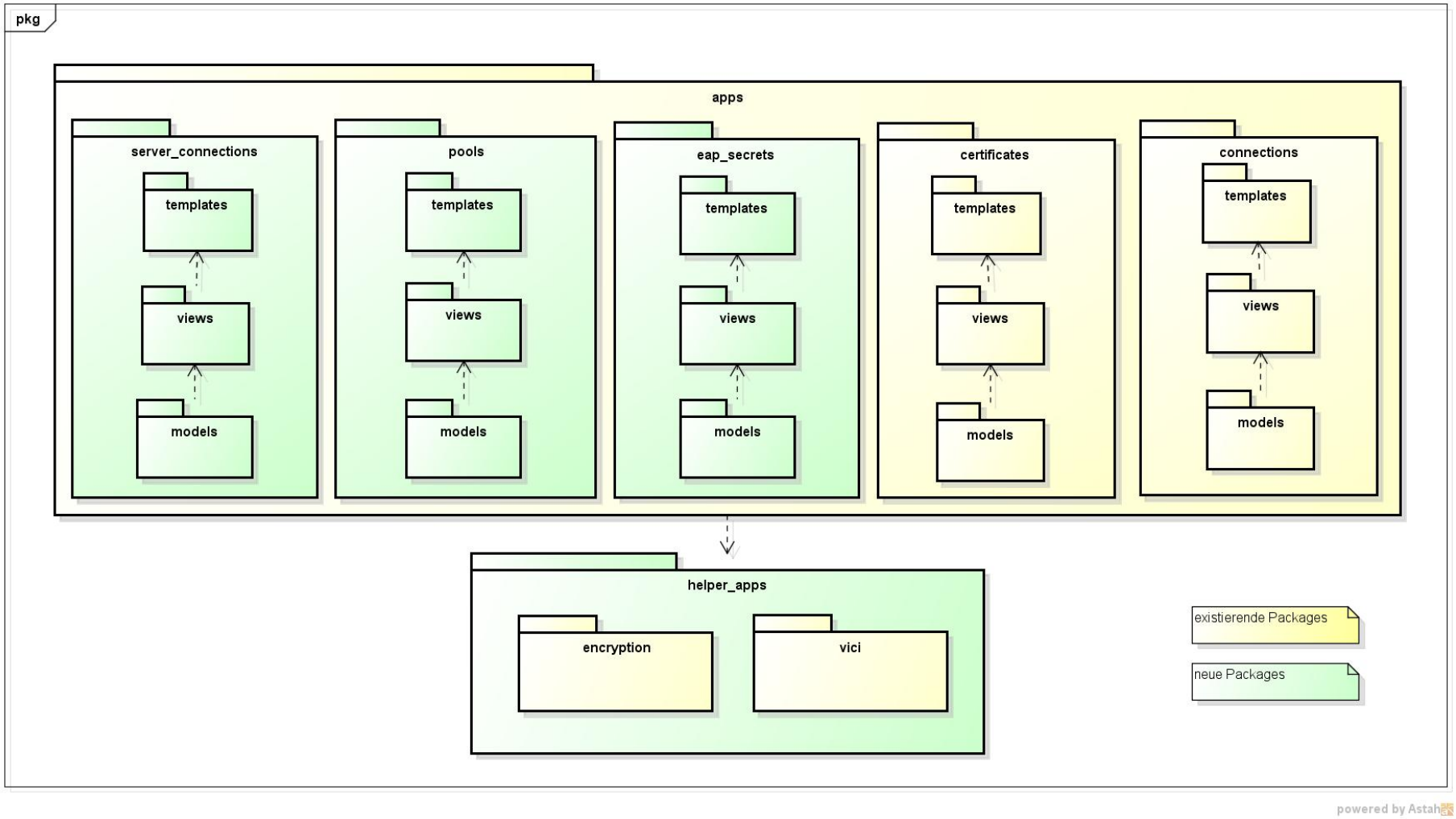


Abbildung 7.1.: strongMan Architekturübersicht

Der detaillierte Aufbau der einzelnen Packages innerhalb einer App setzt sich wie in Abb. 7.2 gezeigt, zusammen. Hier ist gut ersichtlich, wie sich das User Interface klar von der Logik und auch von den Daten trennt.

static Im Package 'static' befinden sich für jede App ihre CSS Stylesheets und die JavaScript-Files, welche die visuelle Darstellung der Benutzeroberfläche definieren.

templates Unter 'templates' sind die in HTML und der Django-eigenen Template-Sprache geschriebenen User Interface Dateien eingeordnet, die für den Aufbau, die Ordnung und auch für UI-Logik verantwortlich sind. Wir unterscheiden hier zwischen 'forms' und 'widgets', wobei 'forms' für eigenständige Formulare und 'widgets' für Helper dieser Formulare steht. Widgets können zum Beispiel als Popover, mit Hilfestellungen zu der unterliegenden Eingabemaske, als Tabellendefinition, oder auch als Statusmeldungen dienen.
Mithilfe der besagten Template-Sprache kann zum Beispiel eine Business Variable aus dem Package 'views' oder 'forms' vom User Interface verwendet werden. Auf diese Weise kann dem UI zum Beispiel ein Python Dictionary angeboten werden, welches direkt als Tabelle angezeigt wird.

models Die Models sind in Pyhon geschrieben und bilden das Datenbankmodell ab. Von hier aus übernimmt Django die Arbeit als OR-Mapper und erstellt die Tabellen für die Datenbank gemäss diesen Models. Mithilfe dieses Mappers können sämtliche CRUD-Operationen[17] (Create-Read-Update-Delete) direkt auf Objekte angewendet werden.

migrations Migrations ist ein von Django generiertes Paket und Django's Art, die Änderungen der Models in das Datenbank-Schema zu propagieren. Diese beinhalten zum Beispiel das Hinzufügen eines Feldes oder auch das Löschen eines Models.

forms Die Forms übernehmen das Mapping der Models auf das User Interface. Sie geben also an, welches Feld des Models auf welches Feld im UI passt. Hier können zum Beispiel Angaben zu den initialen Werten innerhalb einer Form und deren Bedingungen (Constraints) gemacht werden. So kann einem Passwort-Eingabefeld vorgegeben werden, dass die Eingabe nicht klartext sein soll, oder auch per RegEx definiert werden welche Zeichen ein Feld akzeptieren darf. Diese Angaben werden dann beim Absenden (POST Action) der Form automatisch geprüft.

views Dieses Paket beinhaltet die Handler für die Forms. Die Handler sind verantwortlich für Logikoperationen auf den Forms. Sie können beliebige Funktionen übernehmen, wobei ihnen häufig CRUD-Operationen oder auch das Behandeln des Logging aufgetragen wird.

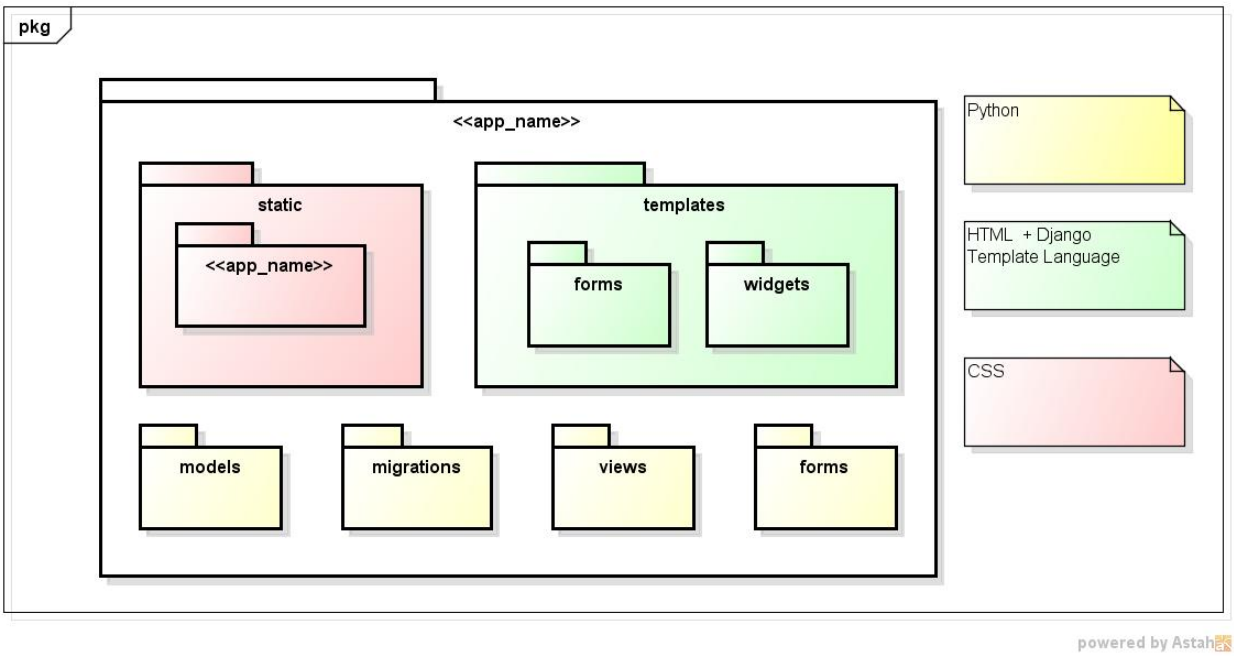


Abbildung 7.2.: Detailansicht: Packages pro App

7.2. Domain Model

Aufgrund der funktionalen Anforderungen wurde das folgende Klassendiagramm erarbeitet (Abb. 7.3). Dieses unterteilt sich grundsätzlich in 'server_connections', 'pools', 'certificates' und 'eap_secrets'. Eine Server Connection besteht hauptsächlich aus den Klassen 'ServerConnection', 'Child' und 'Authentication'.

Eine Connection kann mehrere Childs beinhalten. Childs entsprechen einer Child SA, wobei Server Connection die IKE SA darstellt. Ein Pool wird in einer Server Connection erstellt und von dieser verwendet. Er kann aber auch global und unabhängig erstellt und konfiguriert werden.

Im Gegensatz zu den Pools werden EAP Secrets nicht einer Connection zugeordnet, sondern bestimmen global, welche User sich bei diesem Server mit einem Passwort authentifizieren dürfen. Die meisten der restlichen Attribute entsprechen den Configuration Parameters aus den Anforderungen.

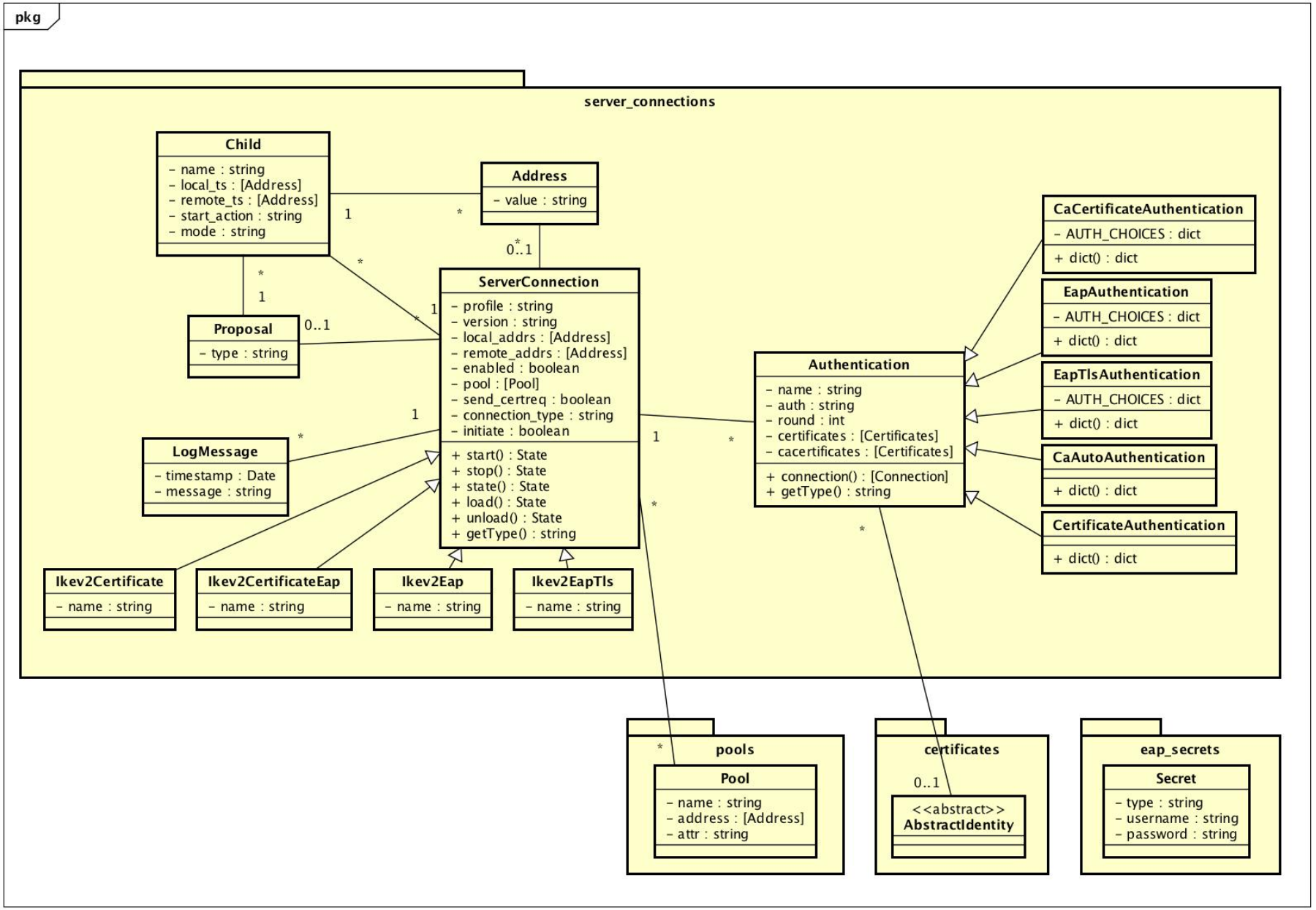


Abbildung 7.3.: Klassendiagramm des strongMan Servers

7.3. Integration

Die strongMan-Erweiterung, welche in diesem Projekt erarbeitet wird, basiert auf dem bestehenden strongMan-Projekt und wird komplett in dieses Integriert, sobald diese Erweiterung ausreichend Features beinhaltet. Dies wird erreicht, indem ein Prototyp (Kap.: 7.4) erstellt wird.

7.4. Prototyp

Um die Risiken zu minimieren und auch das bestehende Projekt nicht zu verändern, bevor eine konkrete Erweiterung besteht, wurde ein neuer Branch "prototype" für den Prototyp erstellt, welcher nach einem erfolgreichen Abschliessen des Projektes in strongMan integriert wird. Der Prototyp dient auch der Verifizierung für die Umsetzbarkeit der Requirements.

7.5. Szenarien

7.5.1. Sequence Diagram: Create and start a server connection

In folgendem Sequenzdiagramm (7.4) wird die Konfiguration und der Aufbau einer Server Verbindung in Stil eines Raum-Zeit-Diagramms gezeigt.

1. Der Benutzer öffnet im Menu die Server Connections. Es werden ihm alle konfigurierten Verbindungen angezeigt.
   1.1. strongMan lädt alle Verbindungen aus der Datenbank.
   1.2. strongMan ruft alle verbundenen SA's vom strongSwan Daemon ab.
2. Der Benutzer klickt auf 'Add Remote-Access' oder 'Add Site-to-Site'. Es werden ihm verschiedene Authentifizierungsmethoden angezeigt.
3. Der Benutzer wählt eine Authentifizierungsmethode aus. Daraufhin wird ihm das entsprechende Formular angezeigt. Je nach ausgewählter Methode beinhaltet dieses Formular unterschiedliche Elemente.
4. Der Benutzer gibt die Formulardaten ein.
5. Der Benutzer speichert das Formular.
   5.1. Die Connection wird in der Datenbank abgespeichert.
6. Zurück in der Connection Übersicht startet der Benutzer die Verbindung, indem er auf 'On' klickt.
   6.1. Die Connection wird über VICI in den strongSwan Daemon geladen.
   6.2. Falls die Connection ein Active Initiator ist, wird diese über VICI initiiert.

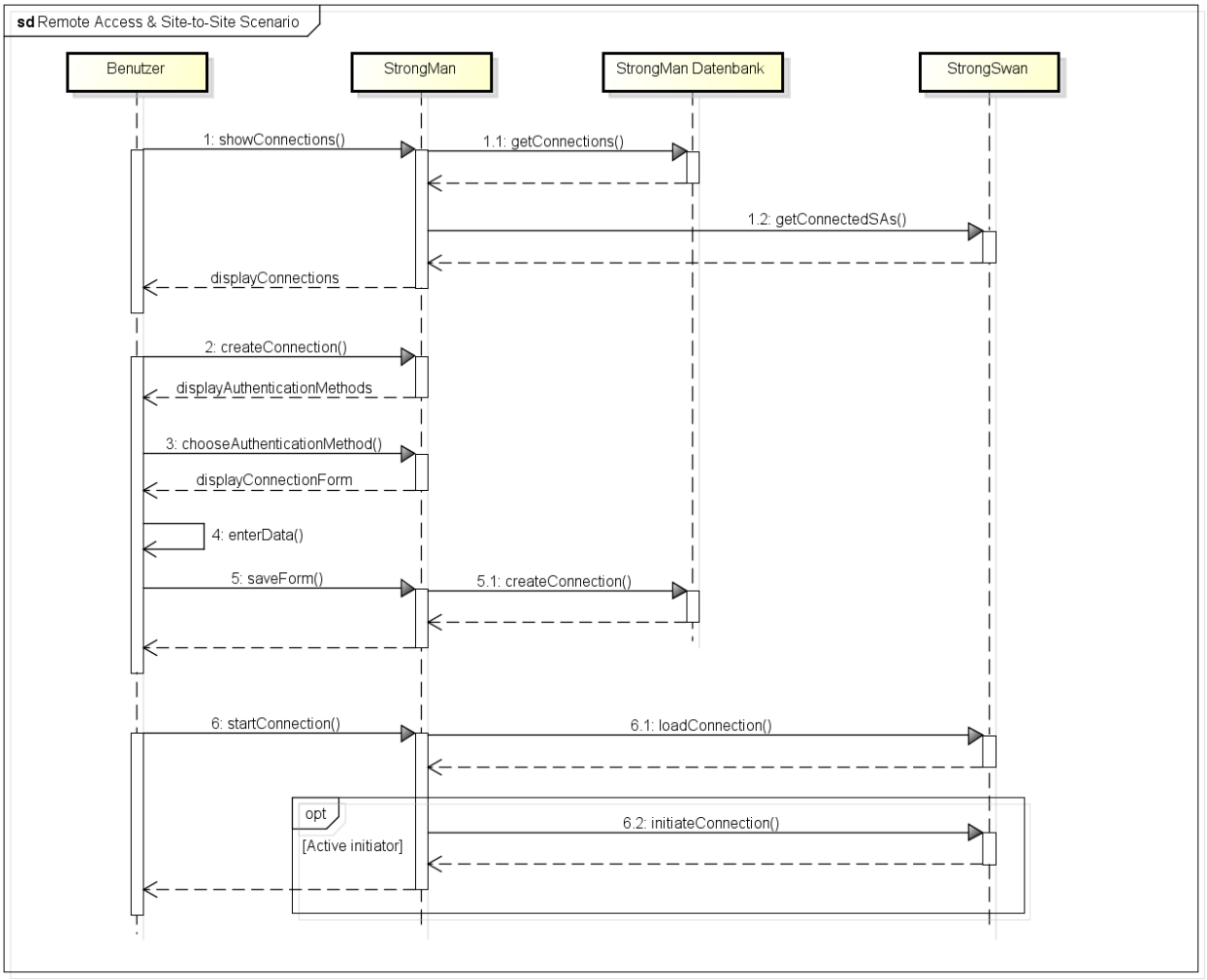


Abbildung 7.4.: Sequenz Diagramm Connection

7.5.2. Sequence Diagram: Push strongMan Data to StrongSwan Daemon

In folgendem Sequenzdiagramm (7.5) wird gezeigt, wie bei einem strongSwan Daeomon Restart die Daten aus der strongMan Datenbank in den Daemon geladen werden.

1. Der Daemon möchte, dass strongMan alle seine Daten zu ihm lädt.
   1.1. strongMan holt alle Secrets von der Datenbank.
   1.2. strongMan lädt die Secrets in den Daemon.
   1.3. strongMan holt alle Keys von der Datenbank.
   1.4. strongMan lädt die Keys in den Daemon.
   1.5. strongMan holt alle Zertifikate von der Datenbank.
   1.6. strongMan lädt die Zertifikate in den Daemon.
   1.7. strongMan holt alle Pools von der Datenbank.
   1.8. strongMan lädt die Pools in den Daemon.
   1.9. strongMan holt alle Connections von der Datenbank.

1.10. strongMan lädt die Connections in den Daemon, falls diese im strongMan aktiviert waren.

1.11. strongMan lässt den strongSwan Daemon die (Active Initiator) Connections initieren, welche im strongMan als bereits initiiert gekennzeichnet sind.

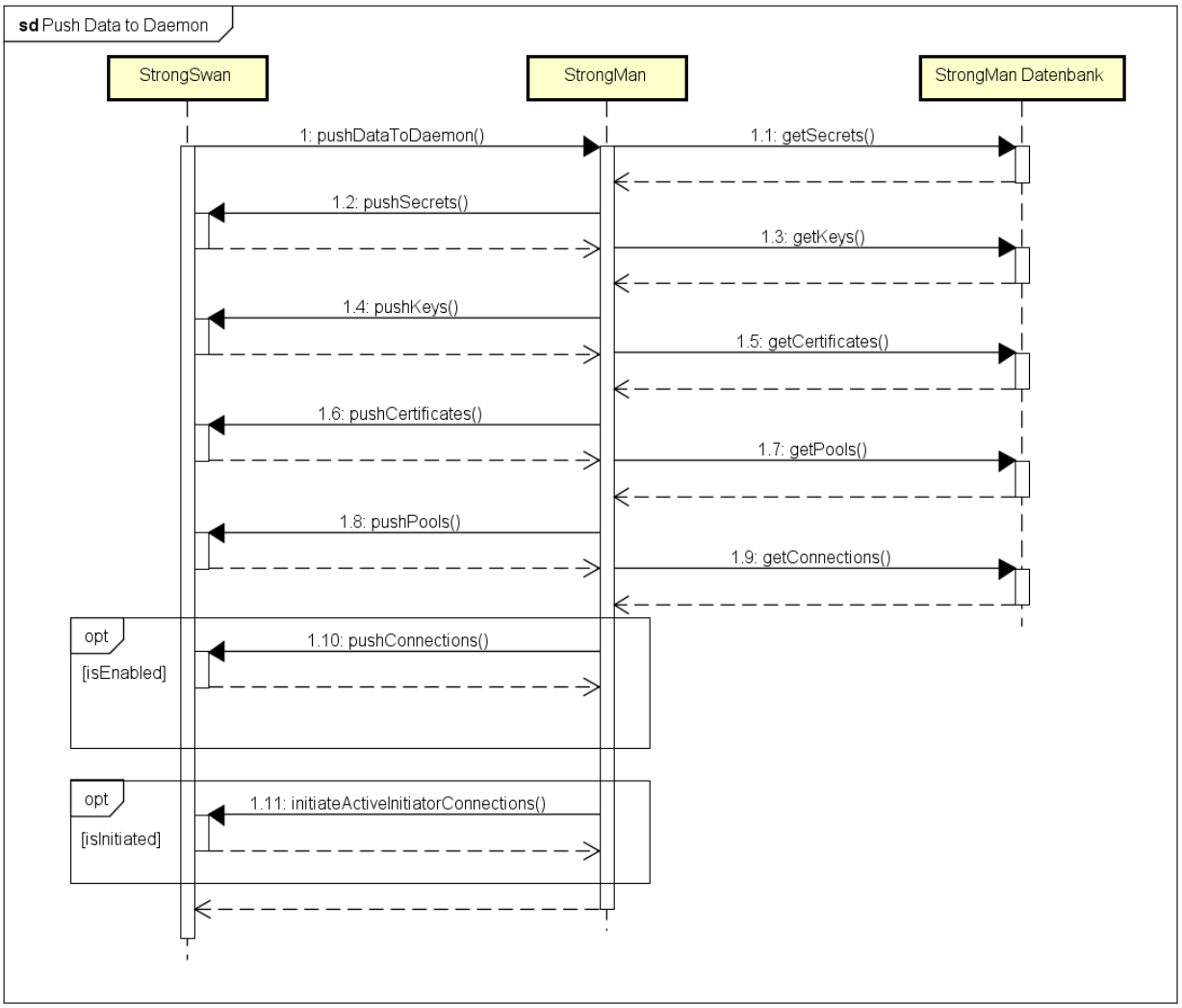


Abbildung 7.5.: Daten in strongSwan Daemon laden

8. Umsetzung und Integration

8.1. Qualität

Um einen hohen Qualitätsstandard zu garantieren wurden diverse Tests durchgeführt. Diese sind im Kapitel 9 protokolliert. Das Testen war ein äusserst wichtiger Bestandteil dieser Arbeit, da so einige Probleme erkannt und auch behoben werden konnten.

8.2. Verwendete Libraries

Für die Filterfunktionalität wurde eine neue Library namens "jQuery.FilterTable"[16] verwendet.

8.3. Neue Features

Server Connection Management

- Es können Remote Access und Site-to-Site Verbindungen erstellt, bearbeitet und gelöscht werden. Ist eine Connection geladen und wird bearbeitet, wird dem Benutzer gezeigt, dass für ein Speichern der Änderungen ein Reload der Connection nötig ist.
- Verbindungen können in einer Read-Only Ansicht angezeigt werden. Diese dient dazu, die Konfiguration einer Verbindung anzuzeigen, ohne ungewollt Einstellungen zu verändern.
- Alle Verbindungen werden in einer Tabelle aufgelistet und die geladenen Verbindungen werden grün markiert.
- Ist eine Verbindung geladen und somit aktiv, werden unterhalb der Verbindung die dazugehörigen IKE SAs sowie deren Child SAs gelistet und deren Informationen angezeigt.
- Die Auflistung der IKE SAs und Child SAs kann ein- und ausgeblendet werden.
- IKE SAs und Child SAs können terminiert werden.
- Es kann mit Hilfe von Filtern gezielt nach `"Remote host"`, `"Remote id"`, `"Remote Traffic Selector"` und `"Local Traffic Selector"` gefiltert werden.
- In einem Logfenster werden Logdaten beim Initiieren einer Site-to-Site Verbindung und beim Terminieren von IKE SAs und Child SAs angezeigt.

Pool Management

- Es können Pools erstellt, bearbeitet und gelöscht werden. Dies einerseits in einem eigenständigen Formular, andererseits aber auch integriert in die Erstellung oder Bearbeitung einer Server Connection.
- Jedem Pool können Attribute zugewiesen werden (z.B. DNS Server), welche den Clients neben den Adressen zugeordnet werden.

- Im Adressfeld eines zu ladenden Pools kann ein Subnetz in CIDR-Notation [6] oder ein Range (IP-IP) übermittelt werden. Eine einzelne IP wird als Spezialfall der CIDR-Notation behandelt (i.e. 32 für IPv4).
- Wird ein Pool erstellt, dessen Name bereits für ein Pool verwendet wird, wird ein Fehler zurückgegeben und der Pool wird nicht gespeichert.
- Beim Ändern eines Pools wird der Name read-only angezeigt.
- Möchte der User ein Pool löschen, wird geprüft, ob dieser noch Online Leases hat oder von einer Connection verwendet wird. Ist das der Fall, wird eine entsprechende Fehlermeldung zurückgegeben und der Pool nicht gelöscht.
- Bei der Auflistung der Pools wird unterschieden zwischen Basic und Extended Information. Extended Information z.B. über die Online Leases wird in einer aufklappbaren Untertabelle dargestellt.
- Wird einer der reservierten Poolnamen ('dhcp' und 'radius') beim Erstellen eines Pools angegeben, wird ein Fehler zurückgegeben, da der Daemon von diesen sowieso keine Leases liefern kann, weil diese nicht von ihm verwaltet werden. Der Pool wird also nicht gespeichert, und dem Benutzer wird erklärt, wie er diese Keywords verwenden kann. Im Wizard der Server Connections können diese beiden Poolnamen direkt aus dem Dropdown-Menu referenziert werden.

EAP Secrets Management

- Es können Username-Password Tupel erstellt, bearbeitet und gelöscht werden.
- Beim Ändern eines Secrets wird der Username read-only angezeigt.
- Die Secrets sind eigenständig und gehören nicht zu einer bestimmten Server Connection. Wenn jemand ein Username und Passwort kennt, kann er sich mit jeder Server Connection verbinden, die EAP als Authentifizierung verwendet.
- Ein Username besteht aus Buchstaben und Zahlen. Das Passwort ist ein Pflichtfeld, hat sonst jedoch keine Mindestanforderung.
- Der Administrator hat die Möglichkeit sich die Passwörter anzeigen zu lassen.
- Bei einer grossen Anzahl Secrets können diese mithilfe der Filterfunktion schnell gefunden werden.

Configuration Loader

Um eine Datenkonsistenz zwischen strongMan und dem strongSwan-Daemon zu garantieren wurde ein Configuration Loader implementiert. Spezifisch soll dieser die im strongMan User Interface erfassten, und somit in der Datenbank persistierten Daten nach einem Neustart des Daemons in diesen zurück schreiben. Der Configuration Loader wird in der Datei /etc/strongswan.d/strongman.conf als Startup-Skript konfiguriert. Bei einem Neustart wird diese Datei eingelesen und der Configuration Loader somit ausgeführt.

8.4. Deployment

Das Deploymentdiagramm in Abbildung 8.1 zeigt den Stand des Deployments am Ende des Projektes.

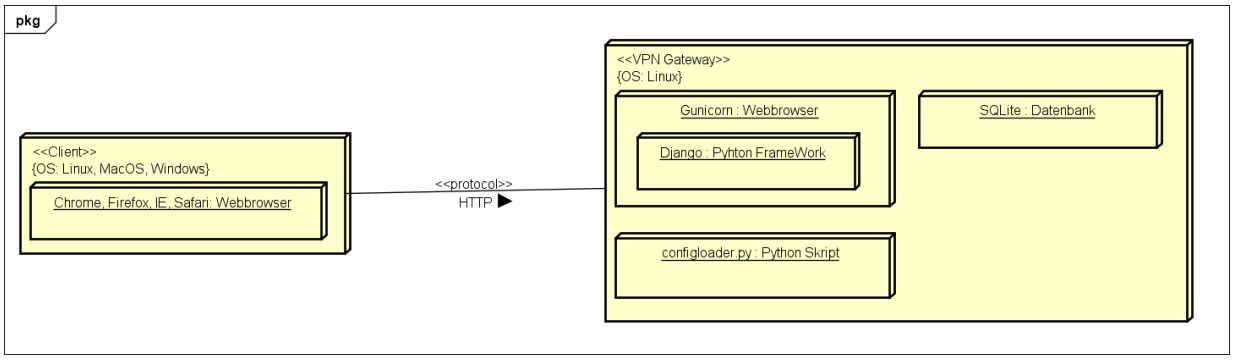


Abbildung 8.1.: Deployment des strongMan

8.5. Continuous Integration

Um kontinuierlich Integration Tests und Unit Tests durchzuführen, verwendeten wir TravisCI, wo wir auf dem bestehenden Setup des strongMan Client Projektes aufbauten. Konkret wird also bei jedem Commit via GitHub TravisCI angestossen. Dort werden zwei Docker Container gebildet, welche eine strongSwan Server-Client Umgebung darstellen. So können z.B. Connections mithilfe von Unit Tests realitätsgetreu getestet werden.

Abbildung 8.2 zeigt einen Report der Testdurchführung, wie er nach jedem Commit aussieht. Zusätzlich erhält man auch pro Docker Container die Log Ausgaben, wo allfällig fehlgeschlagene Tests detailliert aufgeführt werden.

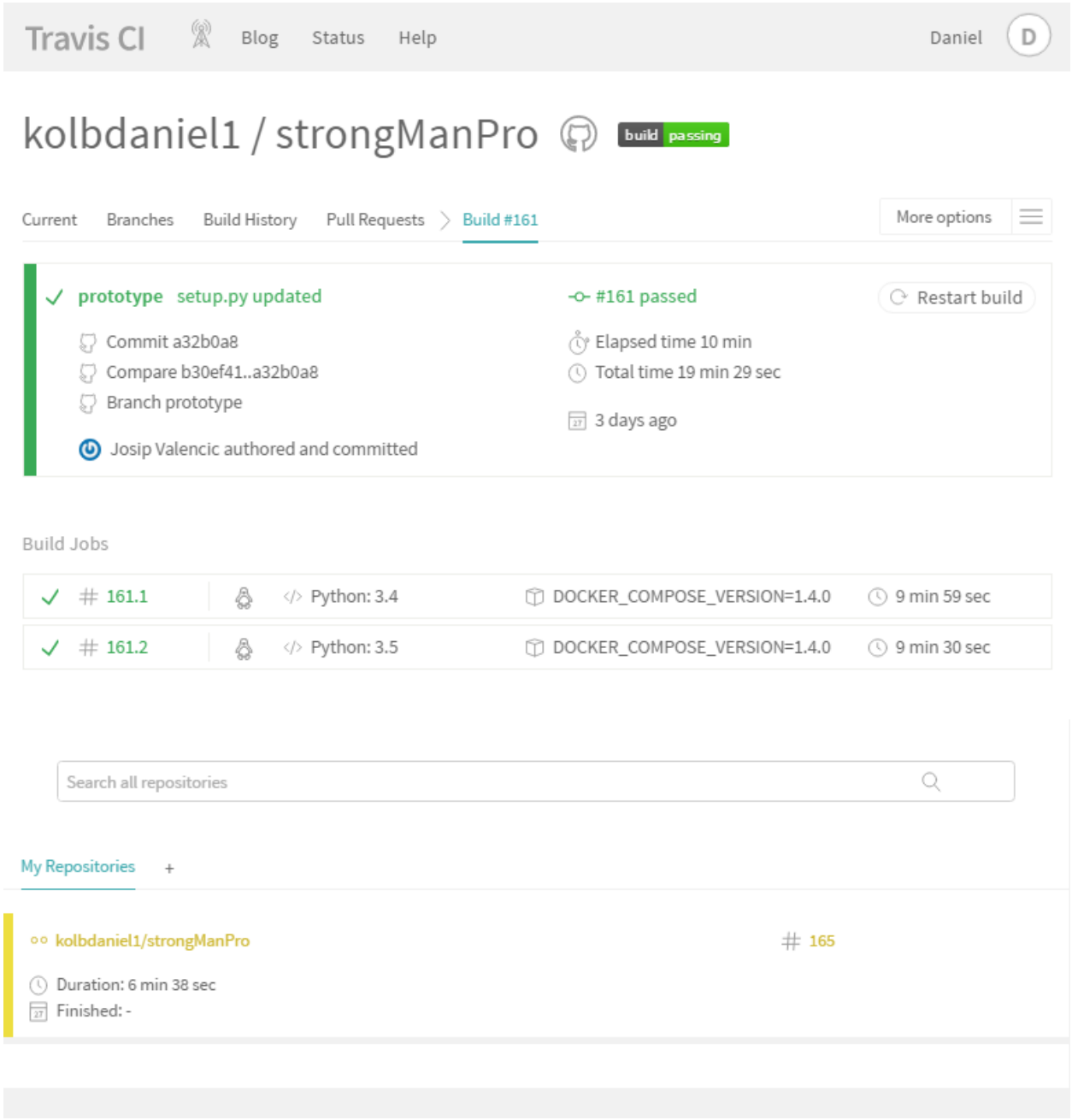


Abbildung 8.2.: Durchführung der Tests für einen Commit

9. Tests

In diesem Kapitel ist dokumentiert, wie die Tests durchgeführt wurden, was getestet wurde und wie mit den Testergebnissen umgegangen wurde.

9.1. Perfomance Testing

Um die Perfomance bei einer grosser Anzahl an Verbindungen zu testen, wurde der Charon Load-Tester eingesetzt. Der Load-Tester ist ein Plugin von strongSwan, dessen Funktion es ist, eine beliebige Anzahl von VPN-Tunnels zum strongSwan Server aufzubauen.

9.1.1. Setup

Der Load-Tester ist standardmässig nicht aktiviert. Um ihn zu aktivieren, muss dies bereits bei der Installation[12] von strongSwan angegeben werden. Darauf muss der Load-Tester[13] bei einem Client konfiguriert werden.

- Füge –enable-load-tester zu den ./configure Optionen hinzu
- Konfiguriere den Load-Tester in der Datei /etc/strongswan.d/loadtester.conf wie in folgendem Beispiel

```
charon {
    reuse_ikesa = no
    threads = 32

    plugins {
        load-tester {
            enable = yes
            initiators = 10
            iterations = 1000
            delay = 100
            responder = 1.2.3.4 # IP Adresse des Servers
            proposal = aes128-sha1-modp1024
            initiator_auth = psk
            responder_auth = psk
            request_virtual_ip = yes
            ike_rekey = 0
            child_rekey = 60
            delete_after_established = no
            shutdown_when_complete = no
        }
    }
}
```

Listing 9.1: Beispielkonfiguration: strongswan.conf

Die oben beschriebenen Load-Test-Konfigurationen werden beim Daemon-Startup ausgeführt. Möchte man anschliessend noch mehr Verbindungen aufbauen, kann man dafür das 'ipsec load-tester' Tool verwenden.

- Starte den IKEv2 Daemon mit dem Befehl 'sudo ipsec start –nofork'
- Für zusätzliche Verbindungen starte das Load-Tester Tool mit dem Befehl 'sudo ipsec load-tester initiate <Anzahl Tunnel> <Verzögerung zwischen Initiationen>'

Um die Verbindungen des Load-Testers in strongMan zu sehen, muss eine Server Connection mit dem richtigen Zertifikat und dem dazugehörigen Key konfiguriert werden. Das Zertifikat befindet sich hartcodiert im Load-Tester Plugin[22]

—BEGIN CERTIFICATE—
MIIB9DCCAV2gAwIBAgIBADANBgkqhkiG9w0BAQUFADA3MQwwCgYDVQQDEwNzcnYx
EjAQBgNVBAsTCWxvYWQtdGVzdDETMBEGA1UEChMKc3Ryb25nU3dhbjAeFw0wODEy
MDgxODU4NDhaFw0xODEyMDYxODU4NDhaMDcxDDAKBgNVBAMTA3NydjESMBAGA1UE
CxMJbG9hZC10ZXN0MRMwEQYDVQQKEwpzdHJvbmdTd2FuMIGfMA0GCSqGSIb3DQEB
AQUAA4GNADCBiQKBgQDQXr7poAPYZLxmTCqR51STGRuk9Hc5SWtTcs6b2RzpnP8E
VRLxJEVxOKE9Mw6n7mD1pNrupCpnpGRdLAV5VznTPhSQ6k7ppJJrxosRYg0pHTZq
BUEC7nQFwAe10g8q0UnM1wa4lJzGxDH78d21cVweJgbkxAeyriS0jhNs7gO5nQID
AQABoxAwDjAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBQUAA4GBAF39Xedyk2wj
qOcaaZ7ypb8RDlLvS0uaJMVtLtlhtb2weMMlgdmOnKXEYrJL2/mbp14Fhe+XYME9
nZLAnmUnX8bQWCsQlajb7YGE8w6QDMwXUVgSXTMhRl+PRX2CMIUzU21h1Elx65Po
CwMLbJ7vQqwPHXRitDmNkEOK9H+vRnDf
—END CERTIFICATE—

- Speichere das Zertifikat in loadtestCert.pem
- Führe den Befehl 'openssl x509 -in loadtestCert.pem -out loadtestCert.der -outform DER' aus.
- Lade loadtestCert.der in den strongMan Zertifikat Manager.

Der dazugehörige Key befindet sich ebenfalls hardcodiert im Load-Tester Plugin:

—BEGIN RSA PRIVATE KEY—
MIICXQIBAAKBgQDQXr7poAPYZLxmTCqR51STGRuk9Hc5SWtTcs6b2RzpnP8EVRLx
JEVxOKE9Mw6n7mD1pNrupCpnpGRdLAV5VznTPhSQ6k7ppJJrxosRYg0pHTZqBUEC
7nQFwAe10g8q0UnM1wa4lJzGxDH78d21cVweJgbkxAeyriS0jhNs7gO5nQIDAQAB
AoGACVACtkxJf7VY2jWTPXwaQoy/uIqYfX3zhwl9i6eTbDlxCE+JDi/xzpKaWjLa
99RmjvP0OPArWQB239ck03x7gAm2obutosGbqbKzJZS5cyIayzyW9djZDHBdt9Ho
quKB39aspWit3xPzkrr+QelkiggtmBKALTBxTwxAU+P6euECQQD4IPdrzKbCrO79
LKvoPrQQtTjL6ogag9rI9n2ZuoK3/XVybh2byOXT8tA5G5jSz9Ac8XeVOsnH9gT5
3WXeaLOFAkEA1vrm/hVSEasp5eATgQ7ig9CF+GGKqhTwXp/uOSI/h3IRmStu5J0C
9AkYyx0bn3j5R8iUEX/C00KSE1kQNh4NOQJAVOsLYIRG2idPH0xThQc4nuM2jes1
K0Xm8ZISSDNhm1BeCoyPC4rExTW7d1/vfG5svgsRrvvQpOOYrl7MB0Lz9QJBALhg

AWJiyLsskEd90Vx7dpvUaEHo7jMGuEx/X6GYzK5Oj3dNP9NEMfc4IhJ5SWqRJ0KA
bTVA3MexLXT4iqXPSkkCQQDSjLhBwvEnSuW4ElIMzBwLbu7573z2gzU82Mj6trrw
Osoox/vmcepT1Wjy4AvPZHgxp7vEXNSeS+M5L29QNTp8
—–END RSA PRIVATE KEY—–

- Speichere den RSA Key in loadtestKey.pem
- Führe den Befehl 'openssl rsa -in loadtestKey.pem -out loadtestKey.der -outform DER' aus.
- Lade loadtestKey.der in den strongMan Zertifikat Manager.

Zuletzt muss nun die Server Connection erstellt werden. Eine Anleitung dazu befindet sich im Anhang C.3. Wichtig ist, dass die davor erstellten Zertifikate verwendet werden und ein genügend grosser Pool definiert ist. Die Anleitung für den Pool-Manager befindet sich im Anhang C.5.

9.1.2. Test

Die Tests gelten als erfüllt, wenn die nichtfunktionale Anforderung 'Performance' eingehalten wird. Gemessen wird die Wartezeit, um Verbindungen aufzulisten, und nicht die, um die Tunnels aufzubauen.

Load-Tester Einstellung	**Anforderung**	**Ergebnis**
Anzahl Tunnel:1000	<3s Wartezeit	siehe Kap. 9.2.4

9.2. NFR Testing

9.2.1. Kompatibilität

Die Funktionalität bezüglich Browser-Kompatibilität wurde komplett getestet und es ergaben sich, wie erwartet, keine Einbussen. Auch das User Interface sieht wie in Abb: 9.1, 9.2 und 9.3 gezeigt wird, in allen Browsern praktisch identisch und somit korrekt aus.

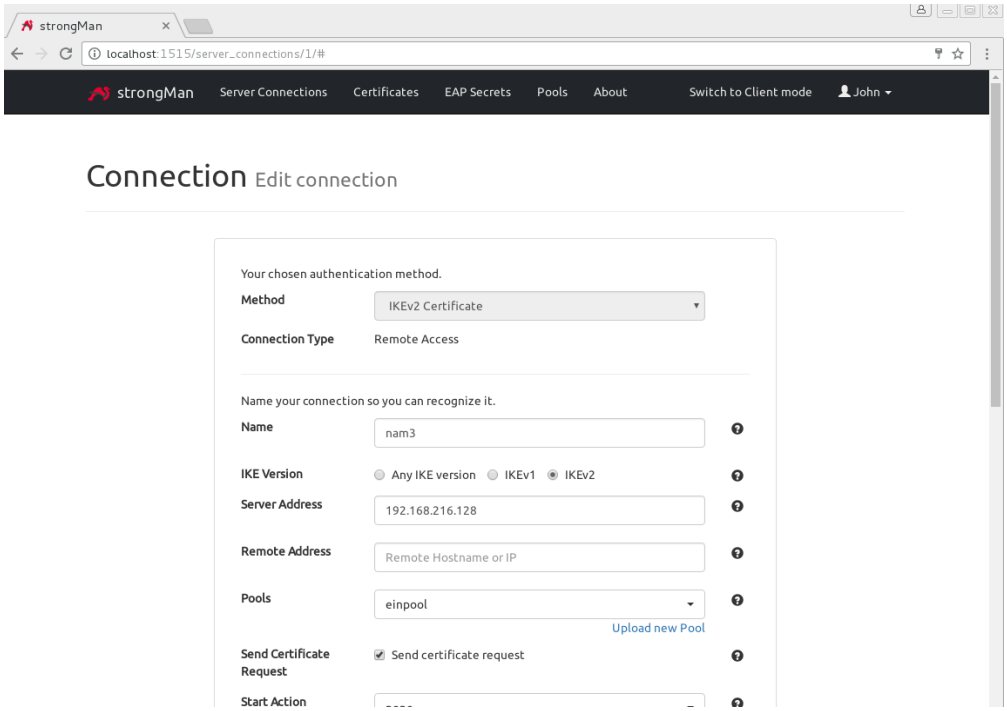


Abbildung 9.1.: Erstellen einer Server Connection in Google Chrome Version 55

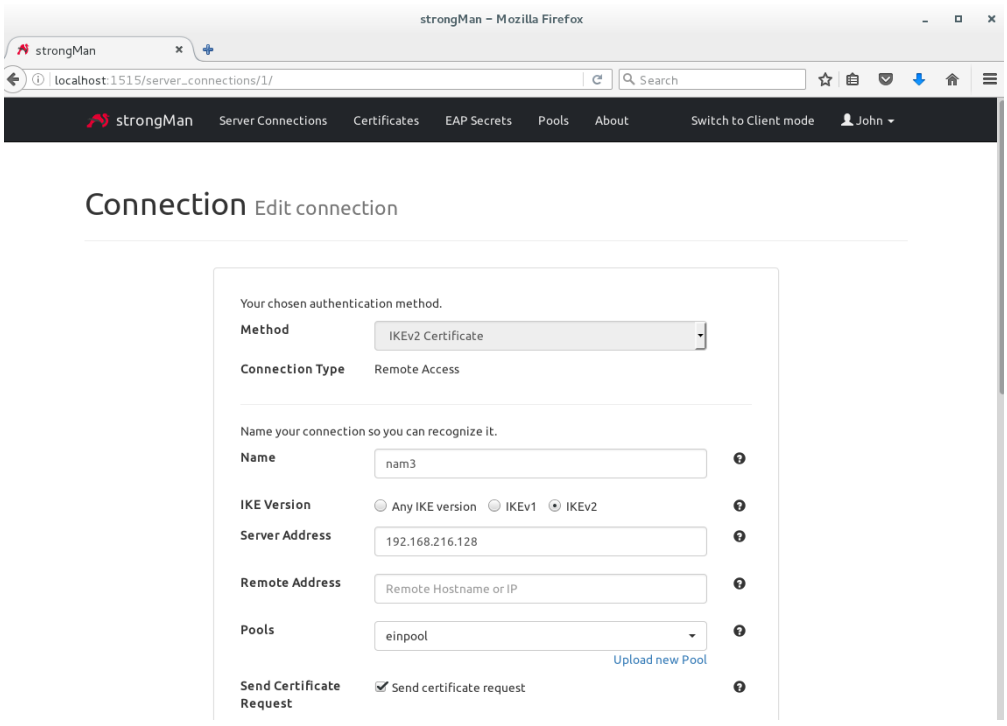


Abbildung 9.2.: Erstellen einer Server Connection in Firefox Version 45

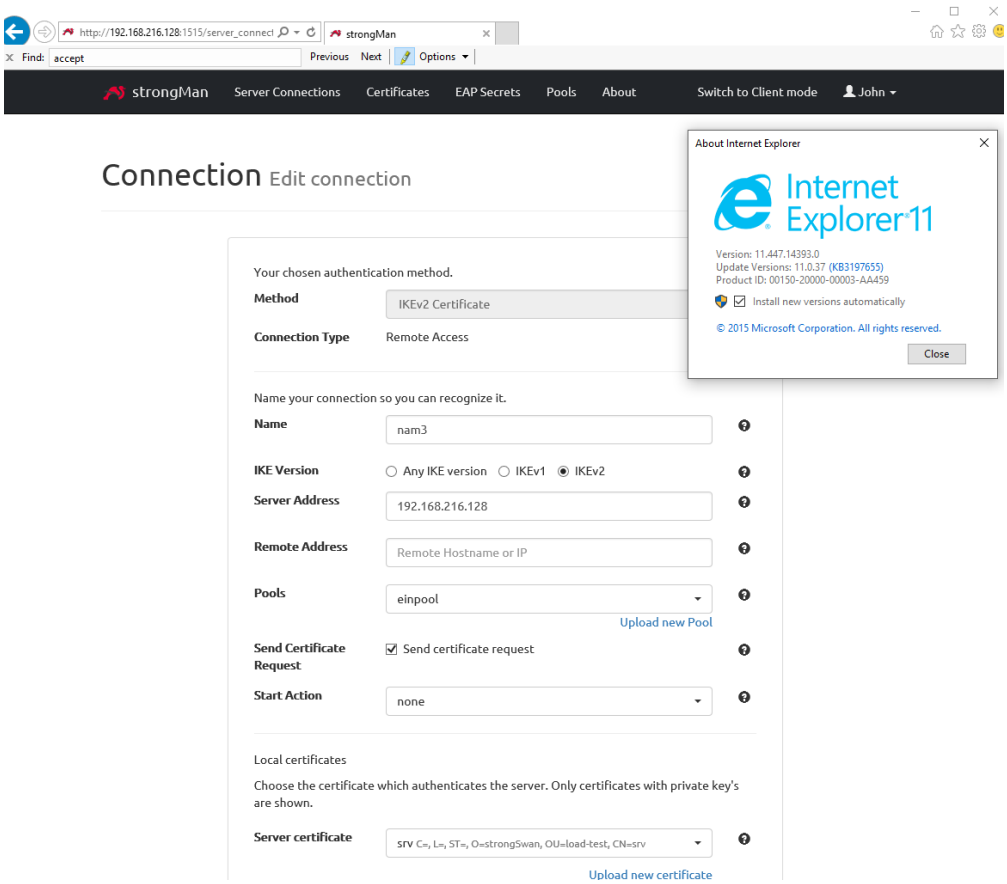


Abbildung 9.3.: Erstellen einer Server Connection im Internet Explorer 11

9.2.2. Fehlertoleranz und Sicherheit

Die Inputvalidierung wurde mithilfe von Regular Expressions (RegEx) durchgeführt. Bei nicht erlaubten Zeichen erhält der User eine entsprechende Fehlermeldung. Der Code um den Input zu validieren sieht wie folgt aus:

```
username = forms.RegexField(max_length=50, initial="", regex=r'^[0-9a-zA-Z]+$')
```

Listing 9.2: Inputvalidierung mit RegEx

Gibt der Benutzer unerlaubte Zeichen ein, erhält er folgende Fehlermeldung:

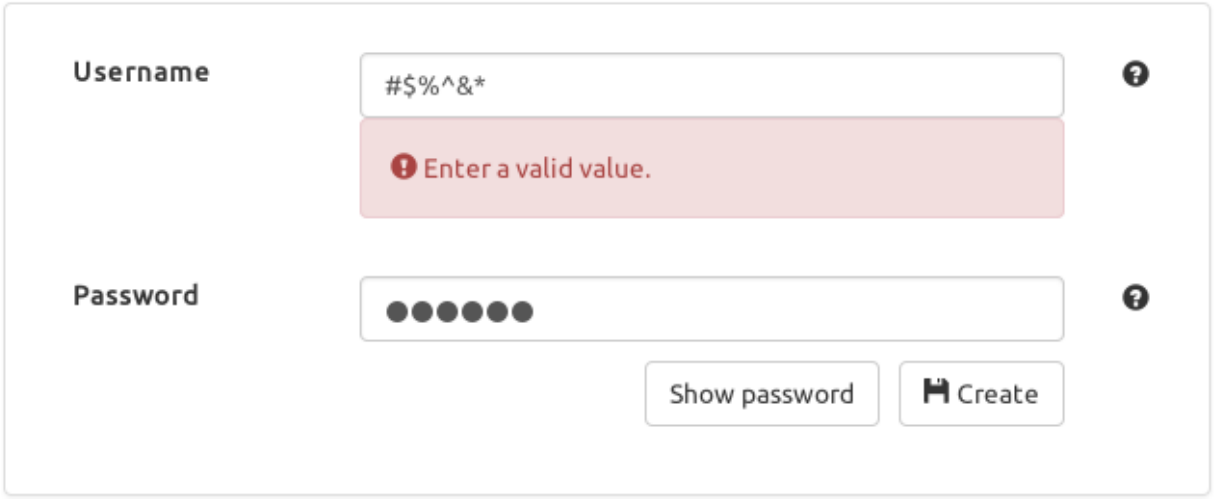


Abbildung 9.4.: Inputvalidierung mit Fehlermeldung

Um zu zeigen, dass die Validierung nicht nur client-seitig erfolgt, wurde ein Request manipuliert. Konkret wurde der Benutzername unterwegs ersetzt durch eine zufällige Zeichenfolge mit nicht erlaubten Zeichen. Das Ergebnis hierbei war dasselbe, wie wenn der Input direkt im Browser geschehen würde.

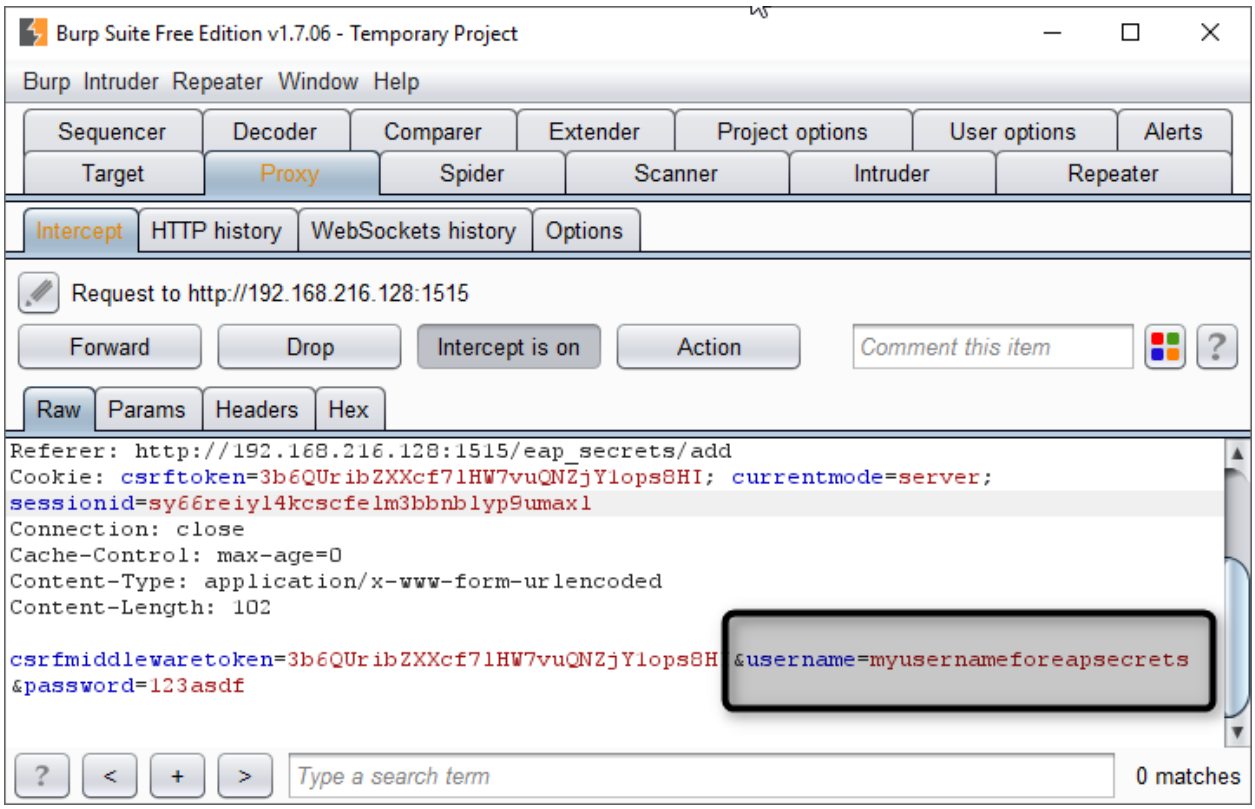


Abbildung 9.5.: Requestmanipulation mit Burp

9.2.3. Sicherheit

Der folgende Screenshot (Abb: 9.6) zeigt die Datenbanktabelle der EAP Secrets. Die Passwörter werden verschlüsselt in der Datenbank gespeichert.

eap_secrets_secret

id	username	type	password	salt
	Filter	Filter	Filter	Filter
	joe	EAP	��9Z□h□,□4w□□.! □□□□^□□□f}□,F1	0F1XpZiQAovTylhYChl...
	jack	EAP	��Me□1□^□V□)□□&...	L7EglY0R3wv6w2loKiT...
	Lucy	EAP	��;A□Ws□□/□B□□<! □Mq/□□g□□□□4L□,F1	GN3Cr8V3ppyO1YswQ...

Abbildung 9.6.: Passwörter in der Datenbank

9.2.4. Performance der Liste und des Filters

Folgende Messung (Abb: 9.7) zeigt die Performance der Liste mit 1000 offenen Verbindungen. Die Wartezeit soll drei Sekunden nicht überschreiten. Für das Filtern soll ebenfalls nicht mehr als drei Sekunden verstreichen. Da für das Visualisieren der Liste Rechenleistung auf dem Client benötigt wird, kann diese Messung nicht auf alle Clients projiziert werden. Hier wurde in einer nicht allzu starken virtuellen Maschine mit 4 GB Arbeitspeicher gemessen.

Abbildung 9.7.: Performance der Server Connections Liste

Problem

Das Laden der Liste mit 1000 Server Connections dauert im Schnitt 1.5 Sekunden länger als in den NFR definiert wurde. Dieses Problem entsteht nicht nur weil das Anzeigen (Rendering) viel Zeit in Anspruch nimmt, sondern auch, weil mehr Daten von der Schnittstelle geladen werden, als effektiv angezeigt werden müssten. Was sicher auch eine Rolle spielt, ist die Komplexität der Tabelle, da unterhalb der Haupttabelle zwei weitere Untertabellen existieren.

Lösungsansätze

Um dieses Problem zu beheben könnten folgende Massnahmen ergriffen werden:

- Es werden nur die Connections im User Interface gezeichnet, die auch sichtbar sind.
- Die Schnittstelle wird so angepasst, dass sich sich einzelne Child SAs von der VICI-Schnittstelle laden lassen.

Somit könnte jeweils beim Scrollen "on demand" weitere Connections nachgeladen und gezeichnet werden (Lazy Loading).

9.3. System Testing

Der strongMan Server Teil besteht aus Connections, Certificates, EAP-Secrets und den Pools. Jeder Teil lässt sich für sich separat konfigurieren. Doch am Schluss, wenn eine Verbindung aufgebaut wird, müssen alle Teile zusammenarbeiten. Die Pools und die Zertifikate werden schon direkt innerhalb der Connection verwendet. Die EAP Secrets hingegen bestehen komplet für sich alleine und sind indirekt für die Connections da, wenn jemand sich mit einer EAP Methode verbindet. Das richtige Zusammenspiel dieser Teile wird in diesem Kapitel im Detail getestet.

9.3.1. Pool

Bei diesem Test werden Pools erstellt und gelöscht. Zur Kontrolle, ob diese auch in strongSwan erstellt und gelöscht werden, wird der Befehl

```
sudo swanctl --list-pools
```

verwendet.

Aufgabe	Anforderung	Ergebnis
Erstelle einen Pool mit: Pool name = Pool1; Addresses = 192.168.0.1/24	Erfolgreich erstellt	ok
Erstelle nochmals den gleichen Pool: Pool name = Pool1, Addresses = 192.168.0.1/24	Fehlermeldung erscheint	ok
Erstelle einen Pool mit: Pool name = Pool2; Addresses = 192.168.5.1/24; Attribute: dns; Attribute values: 192.168.6.1	Pool erfolgreich erstellt	ok
Lösche Pool2	Pool erfolgreich gelöscht	ok

9.3.2. EAP Secret

Bei diesem Test sollen EAP Secrets erstellt und gelöscht werden. Es gibt leider keinen entsprechenden swanctl-Befehl, um zu überprüfen, ob das Secret tatsächlich erstellt wurde. Allerdings wird dieses Ereignis in /var/log/syslog geschrieben:

```
sudo tail -fn 10 /var/log/syslog
```

Aufgabe	Anforderung	Ergebnis
Erstelle ein Secret mit: Username = Secret; Password = SafePassword	Erstellung erfolgreich	ok
Erstelle nochmals das gleiche Secret: Username = Secret; Password = SafePassword	Fehlermeldung erscheint	ok
Lösche das Secret	Secret erfolgreich gelöscht	ok
Erstelle das Secret erneut für späteren Gebrauch	n/a	Erledigt

9.3.3. Certificate Connection

Voraussetzung: Es wurden Zertifikate und ein Pool erstellt.
In Abbildung 9.8 sieht man die Read-Only-View der vorgenommenen Connection Einstellung.

Server Connection Einstellung

Connection Settings: Cert

Method	IKEv2 Certificate
Name	Cert
IKE Version	2
Server Address	192.168.38.150
Remote Address	-
Pool Name	Pool1
Pool Addresses	192.168.0.1/24
Send Certificate Request	True
Start Action	-
Server Certificate	C=CH, L=, ST=, O=strongSwan, OU=, CN=peer
Identity	peerID
Identity Type	subjectAltName
CA/Peer Certificate	-
CA Identity	-
Local traffic selector	10.1.0.0/16
Remote traffic selector	-

Abbildung 9.8.: Certificate Connection Test Einstellung

Client Connection Einstellung

```
connections {
  home {
      local_addrs = 192.168.38.149
      remote_addrs = 192.168.38.150
      vips = 0.0.0.0

      local{
          auth = pubkey
          id = peerID
          certs = peerCert.der
       }
       remote{
```

```
            auth = pubkey
        }
        version = 2
        proposals = aes128-sha256-modp2048
        children {
            net {
                remote_ts = 10.1.0.0/16
                esp_proposals = aes128gcm128-modp2048
            }
        }
    }
}
```

Durchführung

Aufgabe	Anforderung	Ergebnis
Erstelle in strongMan die Connection gemäss Server Connection Einstellung und lade sie gleich	n/a	Erledigt
Erstelle bei einem Client eine SwanCtl Konfiguration gemäss Client Connection Einstellung	n/a	Erledigt
Starte die Verbindung vom Client aus	und Child SA werden erfolgreich aufgebaut	ok

9.3.4. EAP Connection

Voraussetzung: Es wurden Zertifikate, ein Secret und ein Pool erstellt.
In Abbildung 9.9 sieht man die Read-Only-View der vorgenommenen Connection Einstellung.

Server Connection Einstellung

Method	IKEv2 EAP (Username/Password)
Name	EAP
IKE Version	2
Server Address	192.168.38.150
Remote Address	-
Pool Name	Pool1
Pool Addresses	192.168.0.1/24
Send Certificate Request	True
Start Action	-
Remote Authentication	eap-md5
Server Certificate	C=CH, L=, ST=, O=strongSwan, OU=, CN=peer
Identity	C=CH, L=, ST=, O=strongSwan, OU=, CN=peer
Identity Type	distinguishedName
CA/Peer Certificate	-
CA Identity	-
Local traffic selector	10.1.0.0/16
Remote traffic selector	-

Abbildung 9.9.: Certificate + EAP Connection Test Einstellung

Client Connection Einstellung

```
connections {
  home {
      local_addrs = 192.168.38.149
      remote_addrs = 192.168.38.150
      vips = 0.0.0.0

      local{
          auth = eap-md5
          id = Secret
       }
       remote{
          auth = pubkey
       }
       version = 2
       proposals = aes128-sha256-modp2048
       children {
          net {
              remote_ts = 10.1.0.0/16
              esp_proposals = aes128gcm128-modp2048
          }
       }
   }
}

secrets {
  eap-md5 {
    secret = "SafePassword"
    id = Secret
  }
}
```

Durchführung

Aufgabe	Anforderung	Ergebnis
Erstelle in strongMan die Connection gemäss Server Connection Einstellung und lade sie gleich	n/a	Erledigt
Erstelle bei einem Client eine SwanCtl Konfiguration gemäss Client Connection Einstellung	n/a	Erledigt
Starte die Verbindung vom Client aus	IKE SA und Child SA werden erfolgreich aufgebaut	ok

9.3.5. Certificate + EAP Connection

Voraussetzung: Es wurden Zertifikate, ein Secret und ein Pool erstellt.
In Abbildung 9.10 sieht man die Read-Only-View der vorgenommenen Connection Einstellung.

Server Connection Einstellung

Method	IKEv2 Certificate + EAP (Username/Password)
Name	CertEAP
IKE Version	2
Server Address	192.168.38.150
Remote Address	-
Pool Name	Pool1
Pool Addresses	192.168.0.1/24
Send Certificate Request	True
Start Action	-
Remote Authentication	eap-md5
Server Certificate	C=CH, L=, ST=, O=strongSwan, OU=, CN=peer
Identity	peerID
Identity Type	subjectAltName
CA/Peer Certificate	-
CA Identity	-
Local traffic selector	10.1.0.0/16
Remote traffic selector	-

Abbildung 9.10.: Certificate + EAP Connection Test Einstellung

Client Connection Einstellung

```
connections {
  home {
      local_addrs = 192.168.38.149
      remote_addrs = 192.168.38.150
      vips = 0.0.0.0
      version = 2

      local-cert {
          round = 1
          auth = pubkey
          certs = peerCert.der
          id = peerID
      }
      local{
          round = 2
          auth = eap-md5
          id = Secret
       }
       remote{
          round = 1
          auth = pubkey
       }
       proposals = aes128-sha256-modp2048
       children {
          net {
              remote_ts = 10.1.0.0/16
              esp_proposals = aes128gcm128-modp2048
          }
       }
   }
}
```

```
secrets {
  eap-md5 {
    secret = "SafePassword"
    id = Secret
  }
}
```

Durchführung

Aufgabe	Anforderung	Ergebnis
Erstelle in strongMan die Connection gemäss Server Connection Einstellung und lade sie gleich	n/a	Erledigt
Erstelle bei einem Client eine SwanCtl Konfiguration gemäss Client Connection Einstellung	n/a	Erledigt
Starte die Verbindung vom Client aus	IKE SA und Child SA werden erfolgreich aufgebaut	ok

9.3.6. EAP-TLS Connection

Voraussetzung: Es wurden Zertifikate, ein Secret und ein Pool erstellt.
In Abbildung 9.11 sieht man die Read-Only-View der vorgenommenen Connection Einstellung.

Server Connection Einstellung

Method	IKEv2 EAP-TLS (Certificate)
Name	EAPTLS
IKE Version	2
Server Address	192.168.38.150
Remote Address	-
Pool Name	Pool1
Pool Addresses	192.168.0.1/24
Send Certificate Request	False
Start Action	-
Remote Authentication	eap-tls
Server Certificate	C=CH, L=, ST=, O=strongSwan, OU=, CN=peer
Identity	peerID
Identity Type	subjectAltName
CA/Peer Certificate	-
CA Identity	peerID
Local traffic selector	10.1.0.0/16
Remote traffic selector	-

Abbildung 9.11.: EAP-TLS Connection Test Einstellung

Client Connection Einstellung

```
connections {
  home {
```

```
        local_addrs = 192.168.38.149
        remote_addrs = 192.168.38.150
        vips = 0.0.0.0

        local{
            auth = eap-tls
            id = peerID
            eap_id = Secret
         }
         remote{
            auth = eap-tls
            id = peerID
         }

         version = 2
         proposals = aes128-sha256-modp2048
         children {
            net {
                remote_ts = 10.1.0.0/16
                esp_proposals = aes128gcm128-modp2048
            }
         }
    }
}

secrets {
  eap-tls {
    secret = "SafePassword"
    id = Secret
  }
}
```

Durchführung

Aufgabe	Anforderung	Ergebnis
Erstelle in strongMan die Connection gemäss Server Connection Einstellung und lade sie gleich	n/a	Erledigt
Erstelle bei einem Client eine SwanCtl Konfiguration gemäss Client Connection Einstellung	n/a	Erledigt
Starte die Verbindung vom Client aus	IKE SA und Child SA werden erfolgreich aufgebaut	ok

9.3.7. Certificate Connection: Site-to-Site

Voraussetzung: Es wurden Zertifikate und ein Pool erstellt.
In Abbildung 9.12 sieht man die Read-Only-View der vorgenommenen Initiator Connection Einstellung und in Abbildung 9.13 die des Responders.

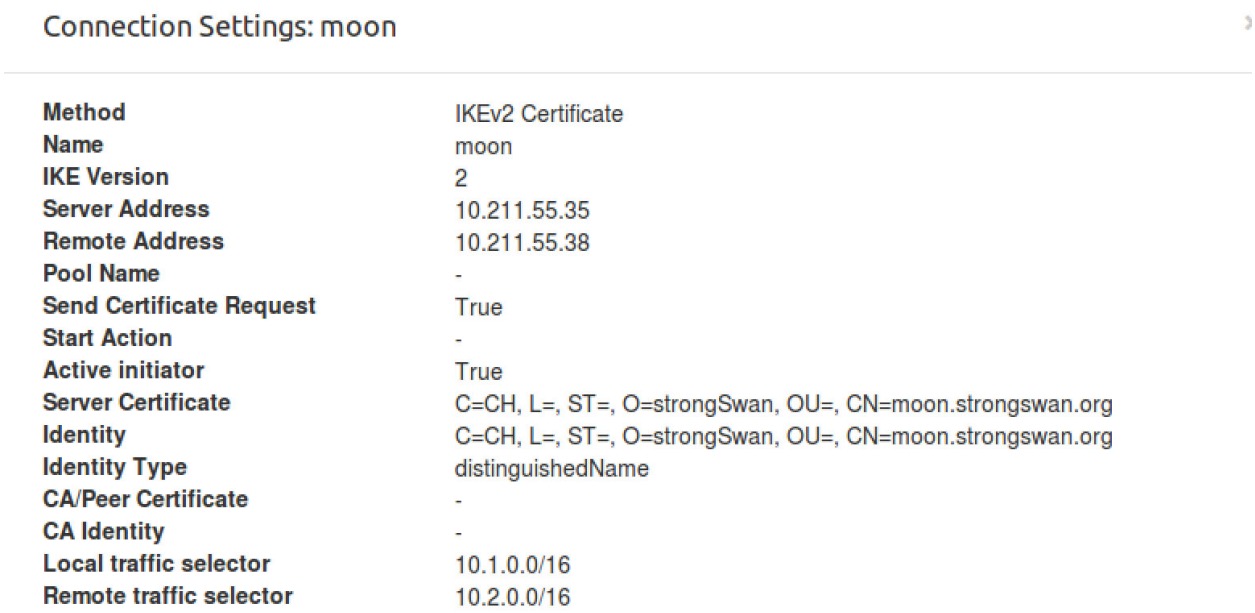


Abbildung 9.12.: Certificate Site-to-Site (Initiator) Connection Test Einstellung

Client Connection Einstellung

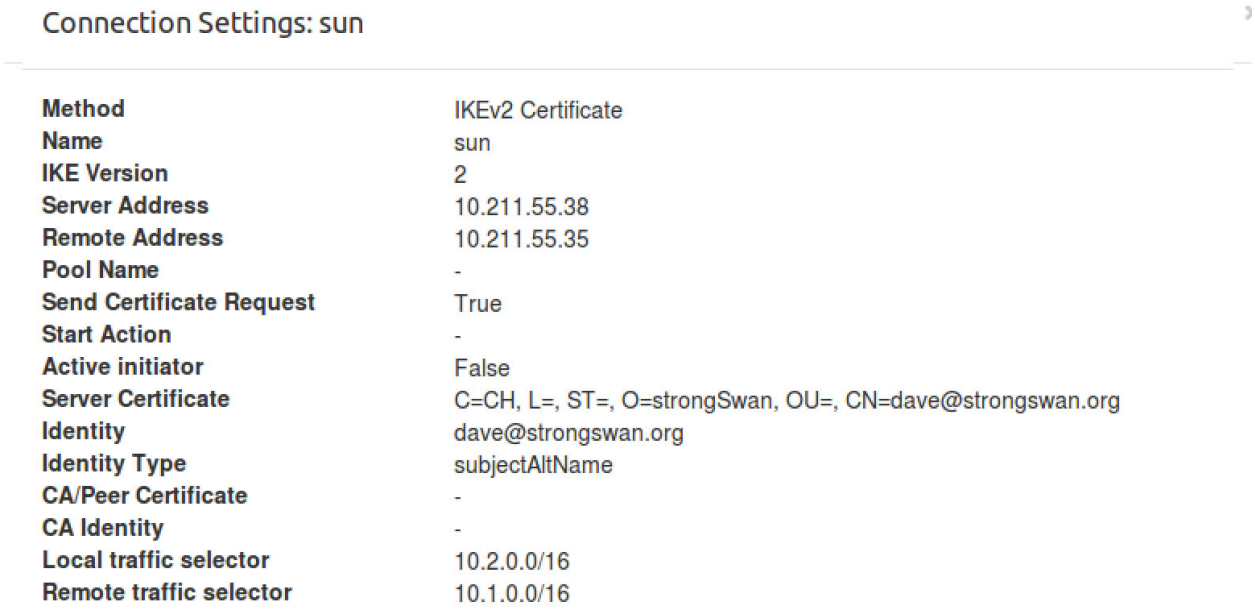


Abbildung 9.13.: Certificate Site-to-Site (Responder) Connection Test Einstellung

Durchführung

Aufgabe	Anforderung	Ergebnis
Erstelle in strongMan die Connection gemäss Site-to-Site Responder Einstellung und lade sie gleich	n/a	Erledigt
Erstelle in strongMan die Connection gemäss Site-to-Site Initiator Einstellung und lade sie gleich	n/a	Erledigt
Starte die Verbindung vom Initiator aus	IKE SA und Child SA werden erfolgreich aufgebaut	ok

9.4. User Acceptance Testing

Nach Fertigstellung der Features wurde ein User Acceptance Test mit einem Systemadministrator durchgeführt. Dieser arbeitet sonst mit Linux Serversystemen, verwendet jedoch kein strongSwan. Er bevorzugt in den meisten Fällen im Alltag die Konsole.
Ihm wurde für das Testing der Auftrag erteilt, einen VPN Server zu konfigurieren, der Clientverbindungen akzeptiert. Die aufgelisteten "Test Cases" haben sich in der aufgeführten Reihenfolge ereignet. Dem Tester wurde also freien Lauf gelassen und dazu wurde beobachtet was geschah und welches Feedback wir erhielten.

Test Cases

Getesteter Task	Resultat	Lösung (wenn Fehler)
Login	erfüllt	-
Pool Erstellung	Help für Attribute und Attribute Value ist unklar	Help Texte angepasst
Pool Bearbeitung	Attribut mit und ohne Attribute Value wurde hinzugefügt: erfüllt, Fehlermeldungen klar	-
EAP Secret Erstellung	erfüllt	-
Zertifikat und Key Upload im Menu	Textwahl für Summe der Zertifikate irritierend	Text angepasst
Server Connection Erstellung	Teilweise Help Texts unklar (Server Address) oder sogar fehlend (Traffic Selectors)	Text angepasst
Geladene Connection editieren	Sollte klar sein, dass Änderungen die Connection neu starten werden	Edit Buttons Naming verbessert
Connection Filtering	Help Text Popover zu schmal	Breiteres Popover
Read-Only-View öffnen	Button Platzierung nicht optimal	Button nach links verschoben
Pool Address Help	Mögliche Eingaben klarer beschreiben	Text angepasst
Read-Only-View Ansicht	Pool Adressen zusätzlich anzeigen	Adressen eingefügt
Pool Erstellung in Server Connection Erstellung	"Upload Pool" unklar, sollte "Create Pool" heissen	Text angepasst
Pool Erstellung in Server Connection Erstellung	Help Texts für Pool Erstellung in der Server Connection Erstellung und in Pool Menu sollen gleich sein	Texte abgeglichen

Diverse Erkenntnisse

Beschreibung	Fix	Resultat / Erkenntnis
User Switch wurde intuitiv vorgenommen	nein	Erfolg: Feature und Platzierung im UI sind sinnvoll
Browser's back button wurde verwendet und funktionierte wie erwartet	nein	Erfolg: Redirect Handling ist korrekt
Pool Erstellung und Zertifikat Upload direkt in der Connection Erstellung wurde nicht erwartet, war aber ein beliebtes Feature	nein	Erfolg
Help Texts wurden oft verwendet	nein	Help Texte müssen aussagekräftig sein
Nach dem Speichern einer Connection soll man direkt in die Connection Overview weitergeleitet werden	ja	Redirect ändern

Fazit

Das Fazit des Testers war, dass die Anwendung für einen versierten Systemadministrator aus dem Linux-Bereich die Konsole nicht ersetzen würde. Grund dafür ist, dass das Kopieren einer Connection-Konfiguration eine praktische Funktionalität bieten würde. Dies ist momentan nicht implementiert. Im Kapitel 10.2.1 ist diese Funktionalität deshalb als mögliche Erweiterung aufgeführt. Ausserdem fehlt für eine schnelle Lokalisierung eines Fehlers beim Erstellen einer Konfiguration ein Systemlog. Auch dieser ist im Kapitel 10.2.1 aufgelistet.

Abschliessend betonte der Tester jedoch, dass ein Systemadministrator, welcher hauptsächlich mit Windows Servern arbeitet und wenig Erfahrung in einer Linux-Konsole hat, definitiv froh um dieses User Interface wäre, da er somit einiges schneller und vor allem intuitiver arbeiten könne.

10. Zusammenfassung und Ausblick

10.1. Zusammenfassung der Ergebnisse

10.1.1. Neue Features

Pool Management

Das Pool Management erlaubt dem Anwender, IP-Address-Pools zu definieren, die er danach im Server Connection Management verwenden kann. Hier werden aktuelle Leases der Adressen und diverse andere detaillierte Informationen angezeigt.

EAP Secrets Management

Benutzer, welche sich mit Username und Password authentifizieren, lassen sich hier einfach eintragen und verwalten.

Server Connections Management

Das Server Connections Management bietet nicht nur grundsätzliche Operationen wie Erstellung, Bearbeitung, Löschung einer Verbindung, sondern auch das Laden, respektive das Initiieren einer Verbindung. Neben der Möglichkeit den VPN-Server als Endpoint zu verwenden, kann auch eine Verbindung zu einem anderen VPN-Server aufgebaut werden (Site-to-Site).
Während dem Erstellen der Connection können direkt Zertifikate hochgeladen und Pools erstellt werden.

Configuration Loader

Müsste man strongSwan einmal neu starten, wird der Daemon direkt nach dem Starten mit den in strongMan erfassten Daten gespiesen.

10.2. Ausblick

10.2.1. Empfehlung Weiterentwicklung

Diese Liste soll dazu dienen, Features zu identifizieren, die nicht als Ziele dieser Arbeit definiert waren, aber während der intensiven Entwicklung und Verwendung von strongMan und während dem User Acceptance Test als sinnvolle Erweiterungen eingeschätzt wurden.

Connection Templates

Die Templates sollen dazu dienen, Verbindungstypen zu klassifizieren und daraus Vorlagen zu erstellen. Aus diesen Vorlagen soll danach eine Connection generiert werden können, and der man lediglich noch grobe Änderungen, wie zum Beispiel das Setzen eines Namens, vornehmen muss.

Connections kopieren

Als Erweiterung für die Templates, oder als Ersatz dafür, soll eine Kopierfunktionalität für Connections existieren. Dies würde den Aufwand des Erstellens vieler Connections massiv verringern.

System Log Ansicht

Um potenzielle Errors der Konfiguration schneller und einfacher identifizieren zu können, wäre eine zusätzliche Ansicht der letzten Einträge des System Logs sinnvoll.

Login Verwaltung

Die momentane Applikation unterstützt keine Benutzerverwaltung. Sinnvoll wäre neben der Benutzerverwaltung auch eine Rollenzuweisung, um gewisse Ansichten nur Benutzern mit ausreichend Rechten anzuzeigen oder für das Bearbeiten anzubieten oder auch die Funktionalität einzuschränken.

10.3. Schlussfolgerung

10.3.1. Beurteilung des Resultats

Die Applikation bietet heute nicht nur was gefordert wurde, sondern auch kleine zusätzliche "nice-to-have"-Features. Die optionalen Ziele konnten nicht mehr angestrebt werden, da diese erheblich mehr Aufwand bedeutet hätten und somit nicht mehr in den Zeitplan passten. Verglichen mit dem strongMan-Projekt für Clients, welches die Basis für dieses Projekt war, kann die Server-Version gut mithalten. Die beiden Projekte ergänzen sich sinnvoll und können einfach installiert und verwendet werden.

Teil II

Anhänge

A. Projektplan

A.1. Einführung

Zweck dieses Dokuments

Dieses Dokument dient der Projektplanung für das Projekt "strongMan Management-Tool-Erweiterung für grosse VPN Server", welches an der HSR im Rahmen der Bachelorarbeit durchgeführt wird. Diese Planung wird regelmässig angepasst und vervollständigt.

Zweck und Ziel dieser Arbeit

Die strongSwan-Umgebung mit strongMan als Management-Interface bietet momentan diverse Konfigurationsmöglichkeiten um eine VPN-Verbindung zu einem strongSwan VPN-Server aufzubauen. Bisher fehlt jedoch ein sogenannter Administratormodus, welcher detailliertere und spezifischere Konfigurationsmöglichkeiten anbietet und sinnvoll darstellt. Ziel dieser Arbeit ist es deshalb, diese Komponente und deren Architektur, sowie Integration basierend auf dem strongMan-Projekt zu planen, zu implementieren und nachhaltig zu dokumentieren.

Annahmen und Einschränkungen

Wir gehen davon aus, mindestens die als 'mandatory' deklarierten Anforderungen aus der Aufgabenstellung, respektive aus den Requirements und der Requirementsanalyse als Prototyp anzubieten und umfänglich getestet, sowie dokumentiert abzuliefern. Sollte die Zeit ausreichen, wird natürlich an den optionalen Anforderungen weiter gearbeitet.

A.2. Managementabläufe

A.2.1. Arbeitspakete und Zeiterfassung

Die Arbeitspaket- und Zeiterfassung erfolgt im Projektmanagementtool Redmine, welches auf einem virtuellen Server der HSR angeboten wird.

A.2.2. Allgemeine Bemerkungen zur Zeitplanung

Jedes Teammitglied hat gemäss den 12 ECTS Punkten eine Gesamtarbeitszeit von 360h zu leisten. Dies entspricht einem wöchentlichen Aufwand von circa 26 Stunden, respektive ungefähren drei Arbeitstagen.

A.2.3. Projektphasen

Die Projektphasen werden gemäss RUP geplant. Die Construction-Phasen dauern jeweils ca. drei Wochen. Die detaillierten Aufgaben innerhalb der Construction-Phasen werden agil als Arbeitspakete im Redmine geplant und auch so ausgeführt.

Phase	**Datum**	**Ziele / Aufgaben**
Inception	12.09.2016 - 19.09.2016	Themenzuweisung, Kickoff
Elaboration	19.09.2016 - 10.10.2016	Dokumentation und Arbeiten zu Beginn (Einarbeitung, Analyse, Backup, Entwicklungsumgebung, CI, Wireframes, UI-Prototype)
Construction 1	10.10.2016 - 31.10.2016	EAP User und Secrets erfassen, Pools erfassen, Remote-Access Connection Parameter speichern, charon load-tester plugin config, Testing
Construction 2	1.11.2016 - 21.11.2016	Connection Remote-access auflisten, Connection site-to-site, charon load-tester plugin testen, Testing
Construction 3	22.11.2016 - 12.12.2016	advanced Features (List Filtering) final, Bug Fixing, Code Freeze, Testing
Transition	13.12.2016 - 23.12.2016	Testing, Verifikation, Dokumentation, Abgabe

Tabelle A.1.: Phasenplanung

A.3. Risikomanagement

Das resultierende Risiko errechnet sich wie folgt:
Risiko = Wahrscheinlichkeit x resultierender Schaden

Risiko	Auswirkung	Wahrscheinlichkeit	Schaden	Risiko	Massnahme	Konsequenzen
(1) Einarbeitung in strongMan	Nichteinhaltung der Zeitschätzung	0.6	0.5	0.30	Know-How erwerben, entsprechende Personen anfragen	Mehraufwand
(2) Schwierigkeiten bei Integration in strongMan	Nichteinhaltung der Zeitschätzung	0.4	0.5	0.20	Systemanalyse	Mehraufwand
(3) Neue Technologien: Django / Python	Nichteinhaltung der Zeitschätzung	0.8	0.6	0.48	Know-How Erarbeitung	Mehraufwand
(4) Probleme mit Testumgebung	Nichteinhaltung der Zeitschätzung	0.8	0.3	0.24	Einarbeitung	Mehraufwand
(5) Probleme mit der Performance	Nichteinhaltung der Zeitschätzung	0.6	0.3	0.18	Frühzeitiges Erkennen mithilfe loadtester	Mehraufwand / Technical Debt

Tabelle A.2.: Risiken

A.3.1. Umgang mit Risiken

Die Risiken werden fortlaufend überwacht und ergänzt. Es sollen stets konkrete Massnahmen zur Minimierung der Risiken ergriffen werden. Sollten die Massnahmen trotzdem nicht ausreichen und es tritt ein Schaden ein, muss unter Umständen mit Mehraufwand oder einer Korrektur des Projektplans gerechnet werden.

A.3.2. Risiken: Retrospektive

Im Laufe des Projektes haben sich vor allem drei der oben gelisteten Risiken bestätigt, wobei wir folgende Erfahrungen gemacht haben: Grundsätzlich haben wir die Risiken korrekt identifiziert, da es sich auf Besagte beschränkte. Auch haben wir die Eintrittswahrscheinlichkeiten verhältnismässig richtig geschätzt.
Es stellte sich jedoch heraus, dass die Einarbeitung in die neuen Technologien (Risiko 3) eine massiv höhere Wahrscheinlichkeit traf, als die restlichen Risiken. Leider zeigte sich dieser Schaden nicht sofort, und zwar jeweils erst, wenn Defizite in der Entwicklung spürbar wurden und somit den Prozess ausbremsten.
Risiko 1, welches als zweithöchstes Risiko identifiziert wurde, wurde ebenfalls leicht unterschätzt. Aufgrund der Defizite der neuen Technologien war auch das Zusammenspiel der einzelnen Packages

schwieriger zu erkennen. Somit fehlte auch ausreichendes Verständnis des Projektes 'strongMan'. Diese beiden Risiken haben sich also gegenseitig noch verstärkt.
Bessere Erfahrungen machten wir mit den Risiken 4 und 5. Durch die vorhandene Testumgebung erfuhren wir keinen Mehraufwand, sondern sogar eine Zeiteinsparung. Auch die Performance, welche im Nachhinein betrachtet höher eingeschätzt werden sollte, traf uns erstaunlich gering. In den in den nichtfunktionalen Anforderungen (NFR) beschriebenen Testszenarien und deren definierten Testwerten konnten wir befriedigende Ergebnisse liefern und somit die NFRs beinahe erfüllen.

A.4. Qualitätsmanagement

A.4.1. Dokumentation

Die Dokumentation wird auf der ShareLatex-Plattform (www.sharelatex.com) verfasst. Sie bietet einen Plaintext LaTeX Editor, sowie einen Rich-Text-Editor und eine Live-Vorschau des erstellten Dokumentes. Es existiert ausserdem eine Versionsverwaltung, welche direkt mit GitHub verknüpft werden kann und aus dem Editor hinaus Commits angestossen werden können.

A.4.2. Entwicklung

Für den Quellcode steht eine GitHub Repository bereit, wo während der Entwicklungsphase auf verschiedenen Branches gearbeitet wird um keine aktiven Systeme zu beeinflussen. Nach erfolgreichem Testen werden bei Bedarf und Fertigstellung einer Komponente diese Branches zusammengefügt.

A.4.3. Infrastruktur

- VMWare Workstation 12 Player / Parallels Desktop 11
- Debian 8.5 / Xubuntu 16
- pyCharm Pro 2016
- ShareLatex, Online LaTeX Editor
- GitHub, Code und Dokumentation Versionierung
- travisCI, Automatisiertes Testing, CI
- Google Drive, Dateiablage
- Redmine, Projektmanagement
- Balsamiq Mockups 3, UI Wireframes
- Astah Professional, UML Diagramme
- MS Visio 2013, Diagramme

A.4.4. Unit Testing

Es wird für jede Methode, welche Logik enthält und somit sinnvoll testbar ist, Tests geschrieben. Dies ist nicht nur eine Sicherstellung der Korrektheit des Codes unsererseits, sondern wurde für strongSwan seit jeher sehr strikt so behandelt.

A.4.5. Code Style Guidelines

- PEP 8: Python Guidelines

B. Installation guide

B.1. Requirements

- strongSwan with VICI plugin (>= v5.4.0)
- >= python3
- >= pip3
- git
- virtualenv
- systemd (optional)

B.1.1. Requirements (installed by setup.py)

- Django 1.9.6
- oscrypto 0.15.0
- asn1crypto 0.17.1
- Pyaes 1.3.0
- Django-tables2 1.2.1
- VICI 5.4.1dev3
- Gunicorn 19.5.0
- dj_static 0.0.6

B.2. Installation

```
git clone https://github.com/strongswan/strongMan.git
cd strongMan
sudo ./setup.py install
```

B.3. Configuration Loader

To guarantee data consistency between strongMan and strongSwan, configure a script in the strongSwan configuration, which will be executed on the startup of strongSwan.

B.3.1. Option 1

If you aren't planning on setting up a systemd service (B.5), do the following: Save this configuration in `"/etc/strongswan.d/strongman.conf"`. Replace 'pathToStrongMan' with the path, where you installed strongMan.

```
charon {
    start-scripts {
        strongman = python3 /pathToStrongMan/configloader.py
    }
}
```

B.3.2. Option 2

If you will configure strongMan with a systemd service (B.5), follow these instructions to get the Configuration Loader running.
Put these lines into " /strongswan/init/systemd-swanctl/strongswan-swanctl.service.in"

```
charon {
    start-scripts {
        strongman = python3 /pathToStrongMan/configloader.py
    }
}
```

B.4. Run

```
sudo ./run.py
```

B.5. Add a systemd service (optional)

Adds a systemd service which starts strongMan automatically on every system startup. Additonally to the service, a launcher icon will be added.

```
sudo ./setup.py add service
```

B.6. Uninstallation

B.6.1. Remove Systemd Service (Optional)

Removes the systemd service and the launcher icon.

```
sudo ./setup.py remove service
```

B.6.2. Remove application folder

To remove strongMan completely, just remove the application directory.

```
rm -rf strongMan/
```

C. User Guide

C.1. Set up a Certificate Authority

Set up a Certificate Authority using strongSwan's PKI tool. For strongMan, you need binary DER encoded certificates and keys. The following commands generates a CA private key and self-signs a CA certificate. The CA certificate needs to be stored in the "/etc/swanctl/x509ca/" directory. Make sure to keep the private key caKey.der of the Certificate Authority save and not to save it on a host with constant direct access to the Internet, since a theft of this master signing key will completely compromise your PKI.

```
# CA Certificate
ipsec pki --gen > caKey.der
ipsec pki --self --in caKey.der --dn "C=CH, O=Linux strongSwan, CN=strongSwan Root
    CA" --ca > caCert.der

sudo cp caCert.der /etc/swanctl/x509ca/
```

Listing C.1: Generate CA Certificate

Generate for each peer, i.e. for all VPN clients and VPN gateways in your network, an individual private key and issue a matching certificate using your CA. Store the certificates into the directory "/etc/swanctl/x509/" and the keys into "/etc/swanctl/rsa/".

```
# End Entity Certificates
ipsec pki --gen > moonKey.der
ipsec pki --pub --in moonKey.der | ipsec pki --issue --cacert caCert.der --cakey
    caKey.der --san "moon.strongswan.org" --dn "C=CH, O=strongSwan, CN=moon.
    strongswan.org" > moonCert.der

ipsec pki --gen > daveKey.der
ipsec pki --pub --in daveKey.der | ipsec pki --issue --cacert caCert.der --cakey
    caKey.der --san "dave@strongswan.org" --dn "C=CH, O=strongSwan, CN=
    dave@strongswan.org" > daveCert.der

sudo cp moonCert.der /etc/swanctl/x509/
sudo cp daveCert.der /etc/swanctl/x509/
sudo cp moonKey.der /etc/swanctl/rsa/
sudo cp daveKey.der /etc/swanctl/rsa/
```

Listing C.2: Generate End Entity Certificates

C.2. Server and Client mode

strongMan is running on port 1515 and can be accessed using the following URL: "http://localhost:1515/". On startup, strongMan is running in server mode. By clicking on the button "Switch to Client Mode" respectively "Switch to Server Mode" in the navigation bar, you can switch between client and server mode. The server mode gives you access to manage server connections, certificates, EAP secret and pools.

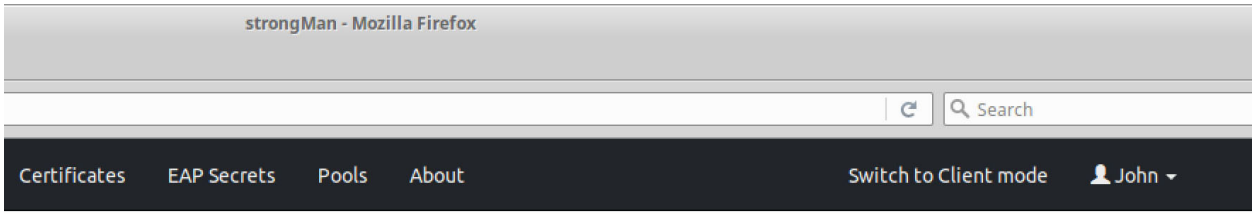


Abbildung C.1.: Home screen with Client / Server Mode

C.3. Server Connections

Connections can be created for a Remote Access or Site-to-Site scenario. All existing connections are listed in the main view. When a connection is loaded and active, the connection is colored green. For active connections the IKE SAs and their Child SAs will be shown in a table view below the related connection. For an IKE SA the remote host and the remote ID are displayed. Below an IKE SA the related Child SAs are displayed in a table with remote and local traffic selectors, the incoming and outgoing bytes as well as incoming and outgoing number of packets and the install time of this Child SA. The table with all SAs of the corresponding connection can be hidden by clicking on the arrow icon on the left of the connection row. IKE SAs and Child SAs can be terminated individually by clicking on the terminate button. Terminating needs a confirmation by clicking on the appearing red terminate button. All SA informations will be updated every ten seconds, except you apply filters on the table. A connection can be edited by clicking on the connection name.

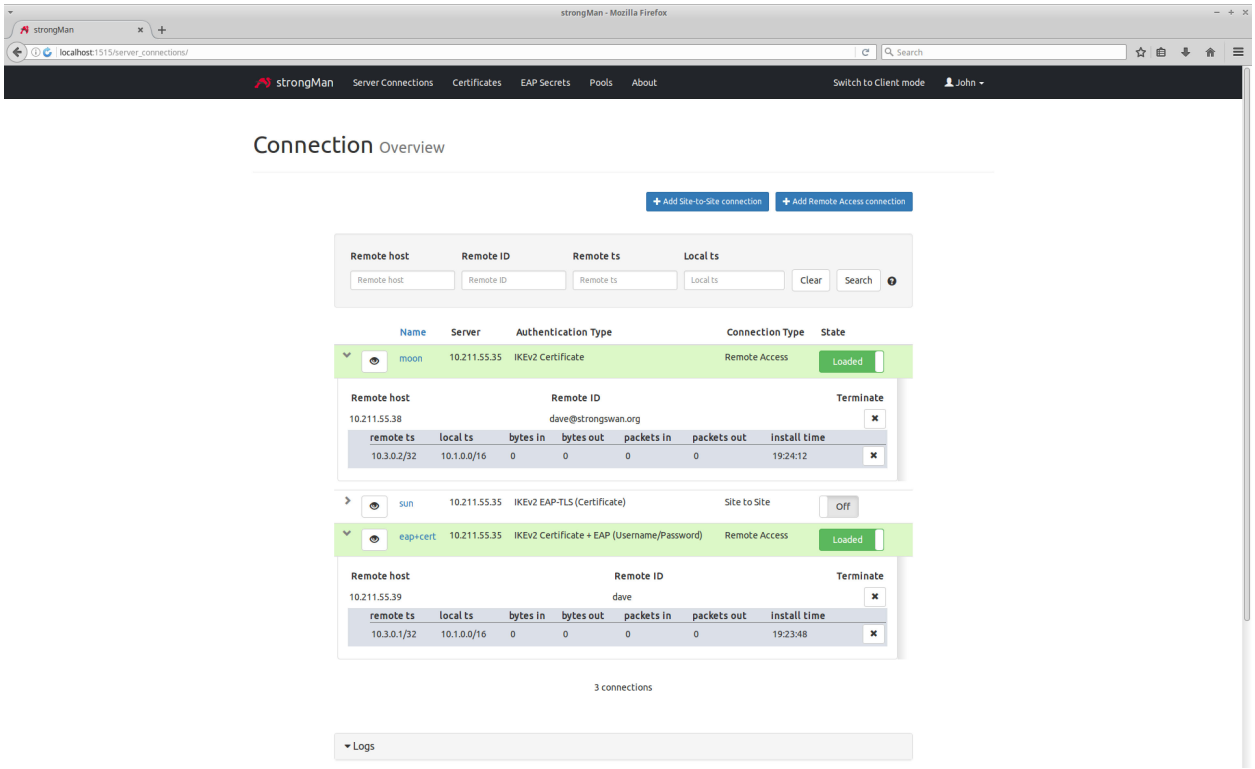


Abbildung C.2.: Server connections overview

Filter functions allow you to filter over all connections and their IKE SAs as well as Child SAs. The filter field complement each other. Filters can be applied on the IKE SA columns remote host and remote ID and on the Child SA columns remote and local traffic selectors. While filters are applied, the SA information won't be updated until the filters are cleared.

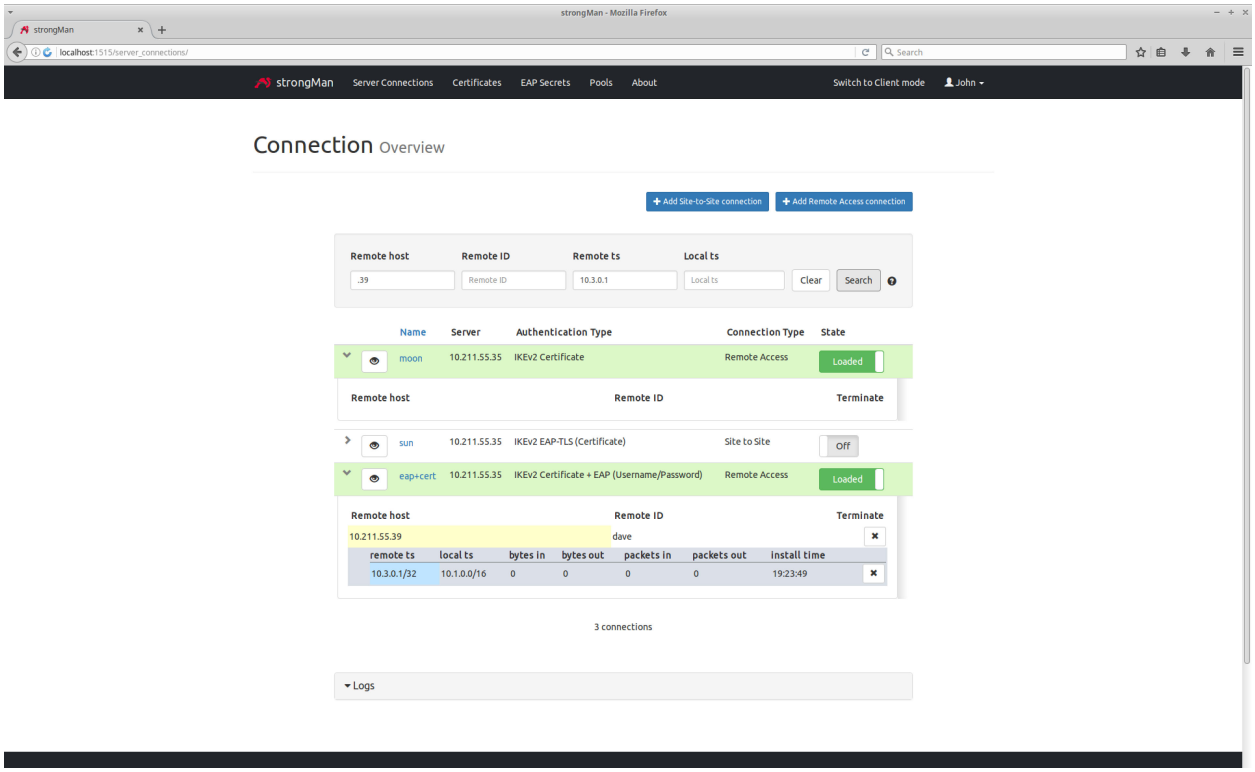


Abbildung C.3.: Server connections filter

Configuration of a loaded connection can be displayed in a readonly view by clicking on the eye icon in a connection row.

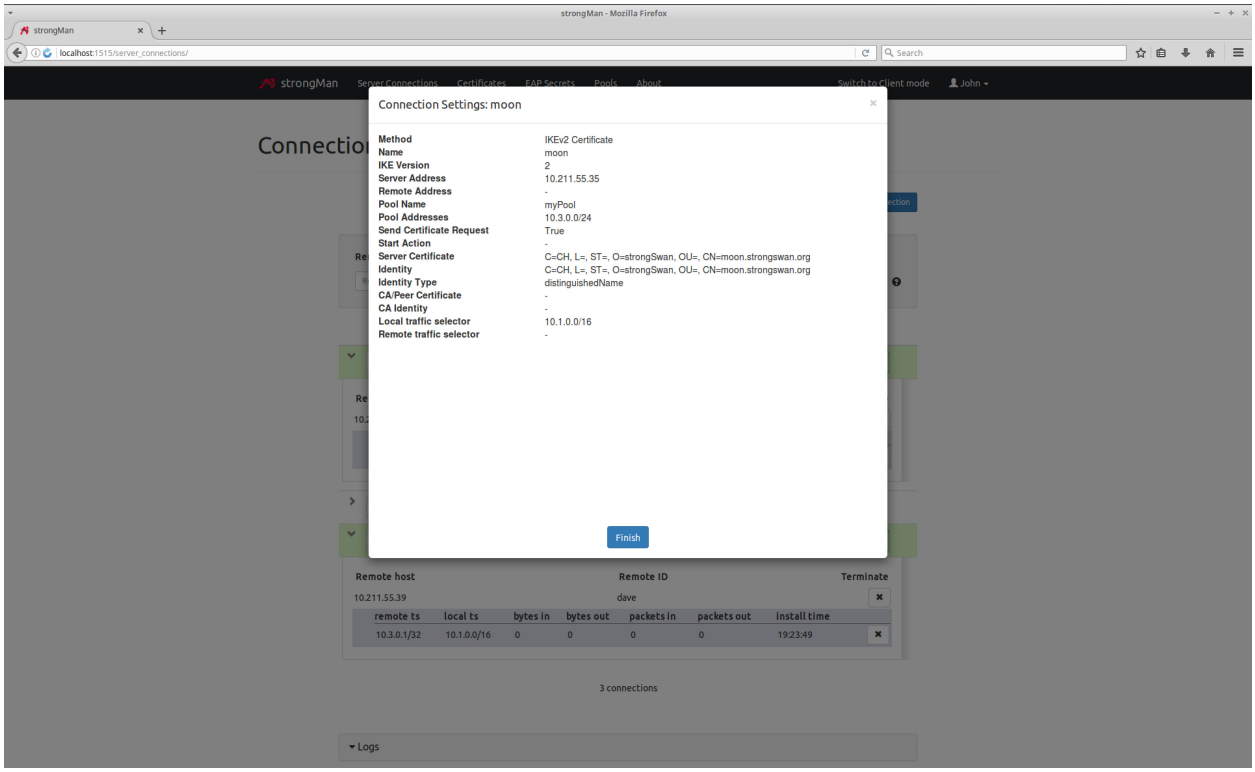


Abbildung C.4.: Server connection readonly view

Depending on the connection type, the available form field for creating a connection will be different and the constraints, if a field is required, changes as well.

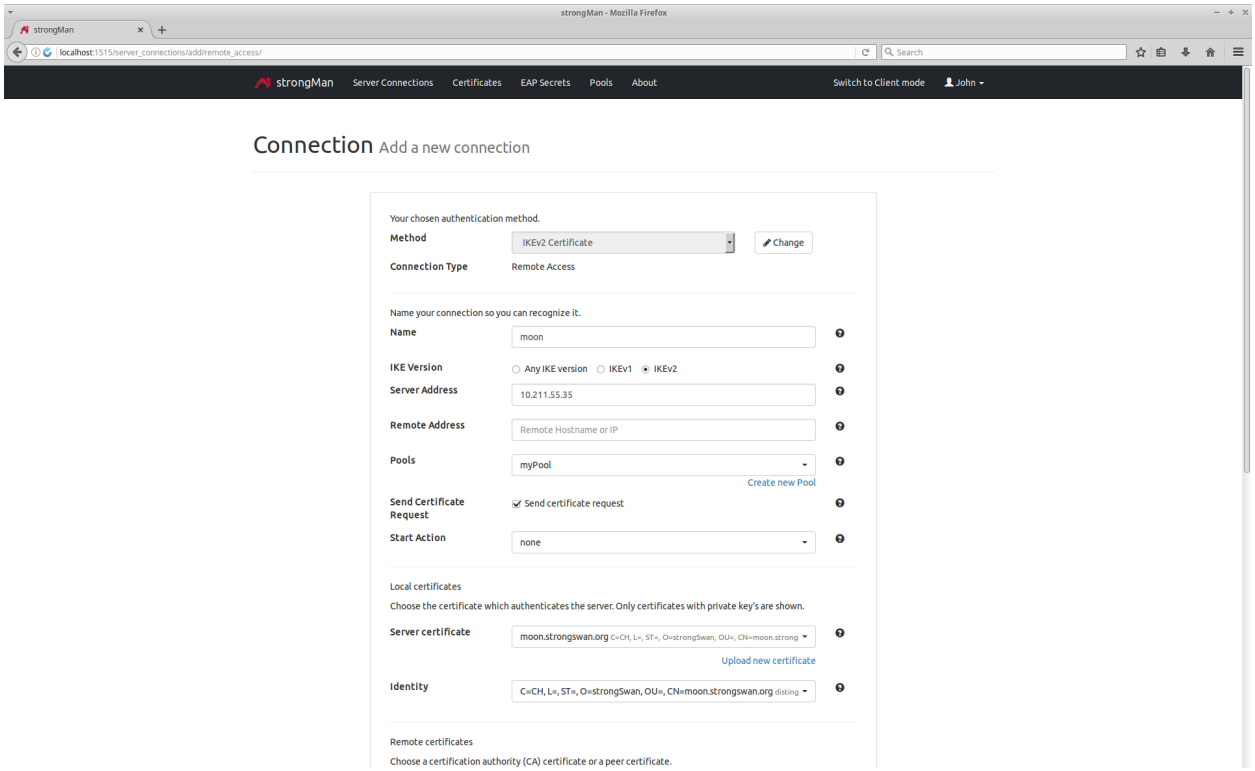


Abbildung C.5.: Server connection form

C.4. EAP Secrets

A filter function allows you to filter usernames. EAP secrets not matching the filter will be hidden. Secrets can be deleted by clicking on the remove button. Removing needs a confirmation by clicking on the appearing red remove button.

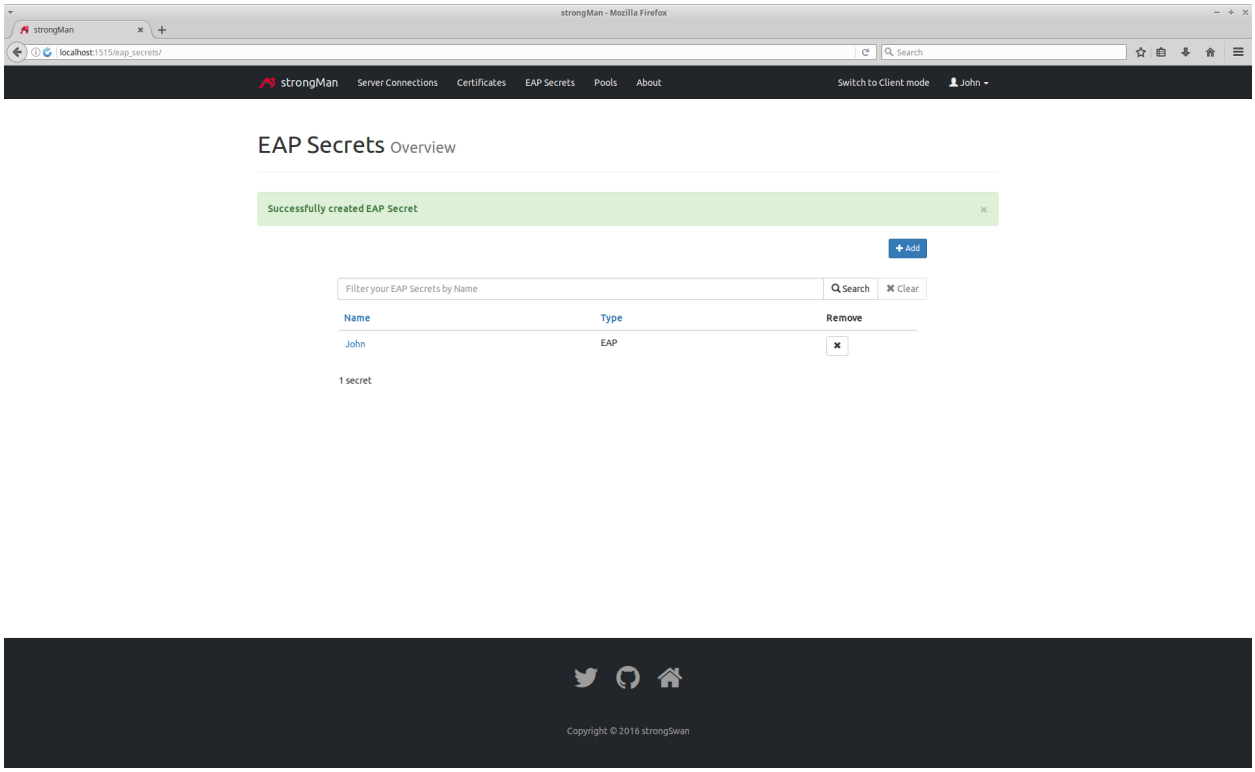


Abbildung C.6.: EAP secrets overview

EAP Secrets can be defined using a username and a password. With the button `"Show Password"`, the password will be shown in plain text to verify you enter(ed) the password correctly.

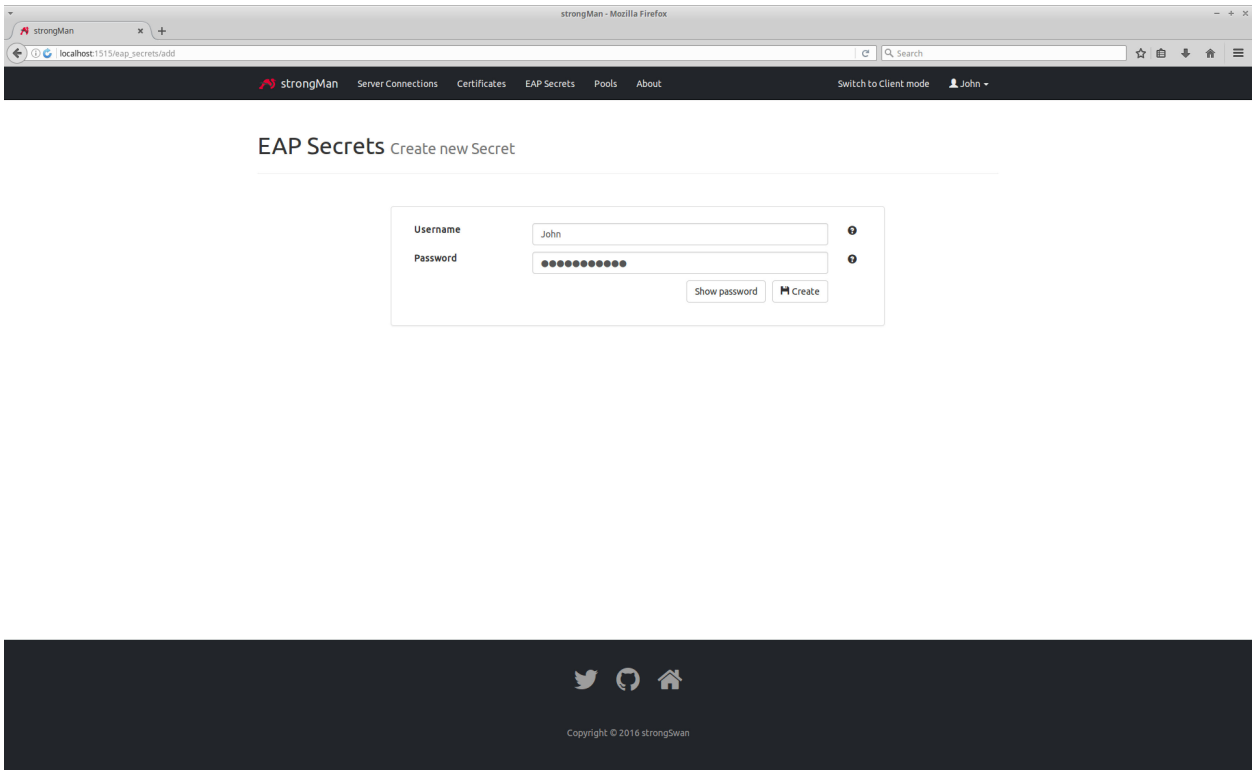


Abbildung C.7.: EAP secrets form

C.5. Pools

All created pools are displayed in a table view. Pool information can be seen by unfolding the information view, which shows the attribute with value, number of addresses in this pool, the IP pool base address and the number of offline and online leases.

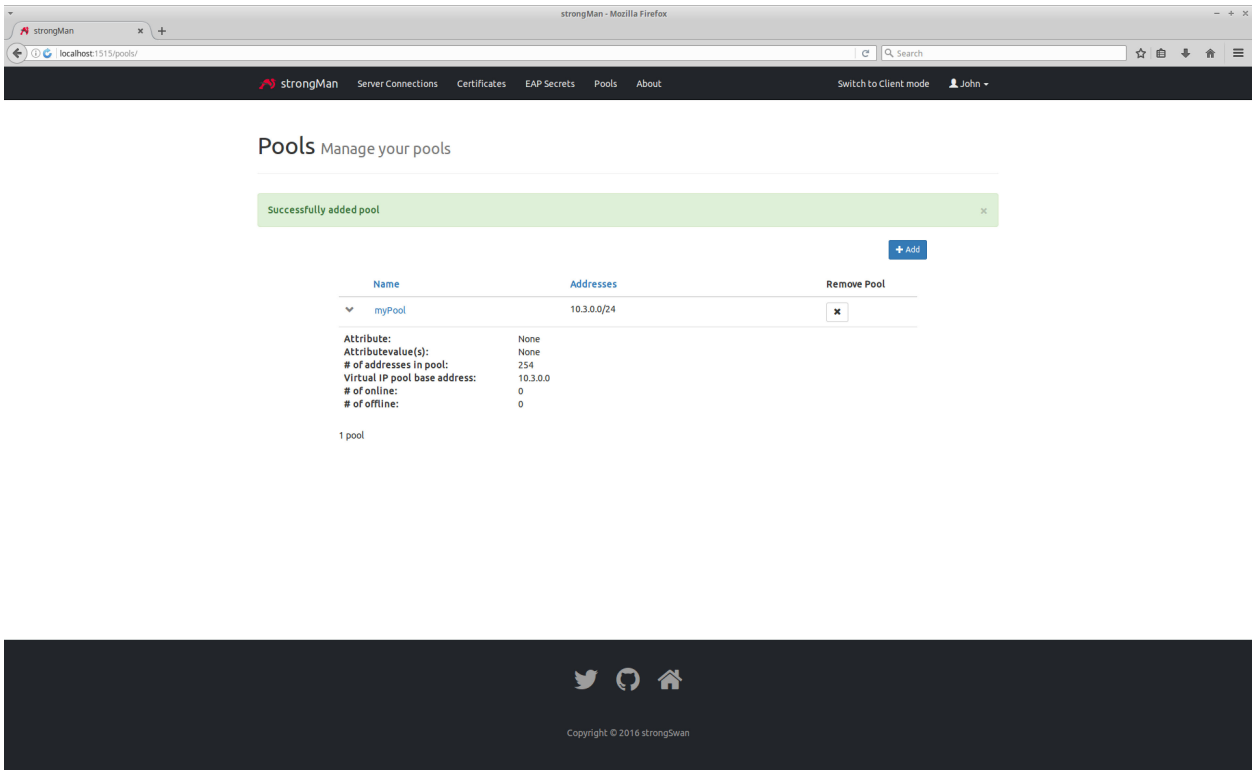


Abbildung C.8.: Pools overview

Pools can be created with at least a pool name and an IP address or address range. The attribute and its value are optional.

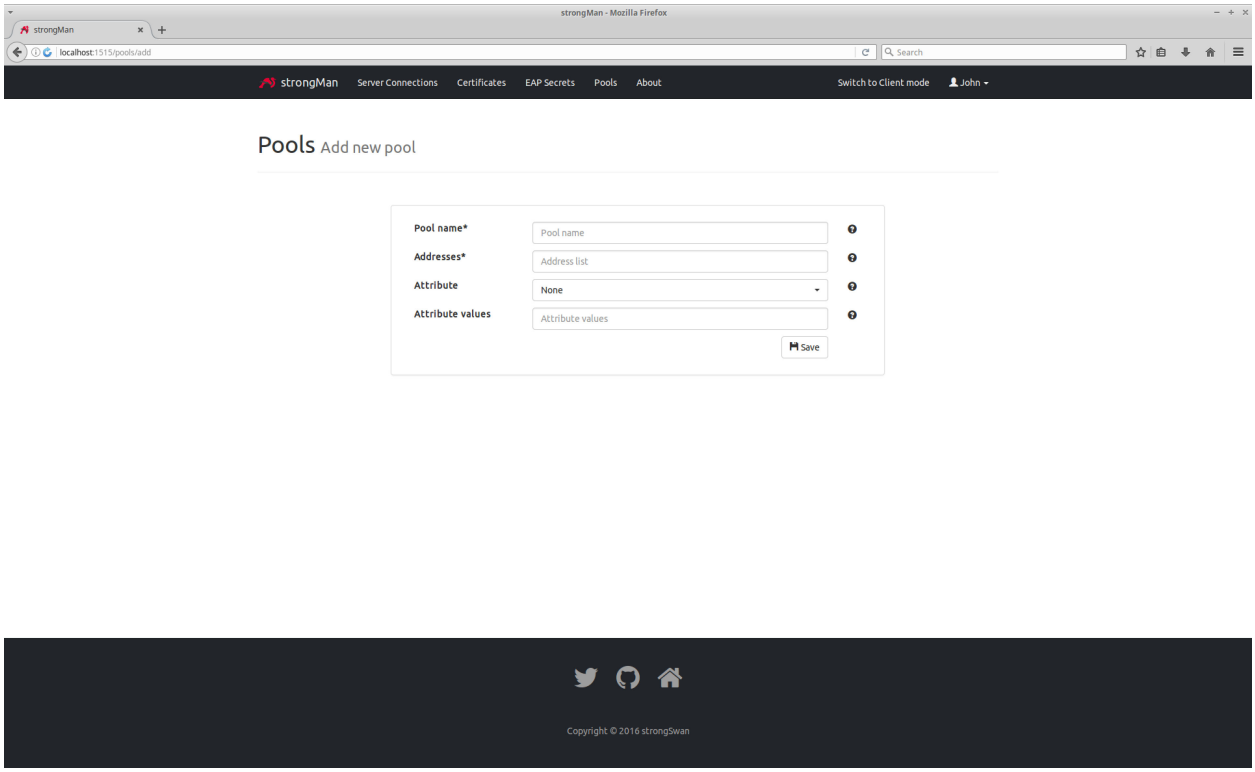


Abbildung C.9.: Pools form

D. Literaturverzeichnis

[1] Bootstrap. *Bootstrap*. http://getbootstrap.com/. Letzter Zugriff am 19. Dezember 2016.

[2] Tobias Brunner. *The Versatile IKE Control Interface (VICI) protocol*. https://github.com/strongswan/strongswan/tree/master/src/libcharon/plugins/vici. Letzter Zugriff am 19. Dezember 2016.

[3] The SQLite Consortium. *SQLite*. https://sqlite.org/. Letzter Zugriff am 19. Dezember 2016.

[4] Django Software Foundation. *Django*. https://www.djangoproject.com/. Letzter Zugriff am 19. Dezember 2016.

[5] Martin Fowler. *Ubiquitous Language*. http://martinfowler.com/bliki/UbiquitousLanguage.html. Letzter Zugriff am 19. Dezember 2016.

[6] V. Fuller. *CIDR-Notation*. https://tools.ietf.org/html/rfc4632. Letzter Zugriff am 20. Dezember 2016.

[7] Gunicorn. *Gunicorn: Python WSGI HTTP Server for UNIX*. http://gunicorn.org/. Letzter Zugriff am 19. Dezember 2016.

[8] Nielsen. *Usability Heuristics*. http://www.designprinciplesftw.com/collections/10-usability-heuristics-for-user-interface-design. Letzter Zugriff am 19. Dezember 2016.

[9] Samuel Oloruntoba. *S.O.L.I.D: The First 5 Principles of Object Oriented Design*. https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design. Letzter Zugriff am 19. Dezember 2016.

[10] S. Kurath S. Bühler. *strongMan*. https://github.com/strongswan/strongMan. Letzter Zugriff am 19. Dezember 2016.

[11] S. Bühler und S. Kurath. *Bachelorarbeit 'strongMan, Frühlingssemester 2016*. https://eprints.hsr.ch/525/. Letzter Zugriff am 19. Dezember 2016.

[12] strongSwan. *Installation*. https://wiki.strongswan.org/projects/strongswan/wiki/InstallationDocumentation. Letzter Zugriff am 20. Dezember 2016.

[13] strongSwan. *Load-Tester*. https://wiki.strongswan.org/projects/strongswan/wiki/LoadTests. Letzter Zugriff am 20. Dezember 2016.

[14] strongSwan. *strongSwan: the OpenSource IPsec-based VPN Solution*. https://www.strongswan.org/. Letzter Zugriff am 19. Dezember 2016.

[15] strongSwan. *Trap Configuration*. https://wiki.strongswan.org/projects/strongswan/wiki/Swanctlconf#Settings. Letzter Zugriff am 20. Dezember 2016.

[16] sunnywalker. *Filter Library*. https://github.com/sunnywalker/jQuery.FilterTable. Letzter Zugriff am 20. Dezember 2016.

[17] Wikipedia. *CRUD*. https://en.wikipedia.org/wiki/Create,_read,_update_and_delete. Letzter Zugriff am 19. Dezember 2016.

[18] Wikipedia. *Django: Wikipedia*. https://de.wikipedia.org/wiki/Django_(Framework). Letzter Zugriff am 19. Dezember 2016.

[19] Wikipedia. *IPsec: Wikipedia*. https://de.wikipedia.org/wiki/IPsec. Letzter Zugriff am 19. Dezember 2016.

[20] Wikipedia. *MVC*. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. Letzter Zugriff am 19. Dezember 2016.

[21] Wikipedia. *Python: Wikipedia*. https://de.wikipedia.org/wiki/Python_(Programmiersprache). Letzter Zugriff am 19. Dezember 2016.

[22] Martin Willi. *Load-tester Creds*. https://github.com/strongswan/strongswan/tree/master/src/libcharon/plugins/load_tester/load_tester_creds.c. Letzter Zugriff am 20. Dezember 2016.

E. Akronyme

CA Certificate Authority. 18, 67, *Glossar:* Certificate Authority

Child SA Child Security Association. 17, 19, 22, 32, 37, 48, 50, 51, 53, 54, 56, 68, 69, *Glossar:* Child Security Association

IKE Internet Key Exchange. 9, 17, 18, 42, 77, 78, *Glossar:* Internet Key Exchange

IKE SA IKE Security Association. 17, 22, 32, 37, 51, 53, 54, 56, 68, 69, *Glossar:* IKE Security Association

IPsec Internet Protocol Security. 9, 11, 78, *Glossar:* Internet Protocol Security

VICI Versatile IKE Control Interface. 9, 11, 12, 17, 18, 29, 34, 48, 65, *Glossar:* Versatile IKE Control Interface

F. Glossar

IKE Security Association Eine IKE Security Association beschreibt die Art der Authentisierung und welche Diffie-Hellman Gruppe verwendet wird. 17, 77

Bootstrap Ein freies CSS-Framework von Twitter entwickelt. 11

Certificate Authority Institution mit Berechtigung Zertifikate auszustellen. 18, 67, 77

Child Security Association Eine Child Security Association beschreibt die gemeinsamen Sicherheitsattribute zwischen zwei kommunizierenden Punkten oder Netzen. Sie ist innerhalb einer IKE Verbindung aufgebaut und kann ohne, dass der VPN-Tunnel abgebaut wird, deaktiviert werden. 17, 77

Django In Python geschriebenes quelloffenes Web Application Framework, das einem Model-View-Presenter-Schema folgt[18]. 9, 11–14, 29, 31, 63, 65

Internet Key Exchange Protokoll zur automatischen Schlüsselverwaltung für IPsec[19]. 9, 77

Internet Protocol Security Eine Protokoll-Suite, die eine gesicherte Kommunikation über potentiell unsichere IP-Netze wie das Internet ermöglichen soll[19]. 9, 77

strongMan Management Interface für strongSwan. 9, 11–13, 15, 17, 19, 29, 34–36, 38, 39, 42, 43, 48, 50, 51, 53, 54, 56, 58, 59, 61, 63–67

strongSwan Open-Source, IPsec basierte VPN Lösung. 9, 11, 12, 15, 17, 21, 34–36, 38, 39, 41, 48, 56, 58, 61, 64, 65, 67, 78

Traffic Selector Detailliertere Spezifikation der Sub-/Netze der zu verbindenden Endpunkte. Kann für ‘narrowing‘ verwendet werden. 19, 37

Versatile IKE Control Interface Plugin, welches die Serverseite des IKE Daemon Charon implementiert[2]. 9, 77

Zertifikat (auch: Digitales Zertifikat)] Durch kryptografische Verfahren hergeleitete Bescheinigung für die Bestätigung der Authentizität und Identiät eines Kommunikationsteilnehmers. 18

Abbildungsverzeichnis