HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

INSTITUTE FOR INTERNET
TECHNOLOGIES AND APPLICATIONS

Bachelorarbeit, Abteilung Informatik

strongMan

HSR Hochschule für Technik Rapperswil

Frühjahrssemester 2016
18. Juni 2016

Autoren:	Bühler Severin & Kurath Samuel
Betreuer:	Prof. Dr. Andreas Steffen
Gegenleser:	Prof. Peter Sommerlad
Experte:	Dr. Ralf Hauser
Arbeitsperiode:	22.02.2016 - 18.06.2016

0.1 Abstract

Die strongSwan Open Source VPN Software ist weltweit im Einsatz. Es besteht schon länger die Nachfrage nach einer grafischen Management-Oberfläche, welche das Konfigurieren und Starten von IPsec Verbindungen erleichtert.
Zur Umsetzung dieses Problems stellt strongSwan das Versatile IKE Configuration Interface (VICI) zur Verfügung, welches für diverse Programmiersprachen eine JSON-artige Schnittstelle bietet.
Im Rahmen dieser Bachelorarbeit ist die Applikation strongMan entstanden, die auf dem Python Webframework Django basiert.

strongMan unterstützt die benutzerfreundliche Konfiguration diverser Internet Key Exchange (IKEv2) Authentisierungsmethoden, welche in einer Datenbank persistiert werden. Diese können auf der Hauptübersicht bearbeitet werden. Weiter lassen sich diese Verbindungen auf- und abbauen. Das Starten hat zur Folge, dass der strongMan die Konfigurationsdaten in ein Dictionary umwandelt und dem strongSwan per Unix Socket über die VICI-Schnittstelle übergibt. Ist die Verbindung erfolgreich etabliert, werden Statusinformationen zu Traffic Selectoren, sowie Datenin- und Output dargestellt. Parallel dazu werden die Logmessages des strongSwan ausgelesen und visualisiert.

Die interne Zertifikatsverwaltung ermöglicht den Upload von PKCS1, PKCS8, PKCS12 und X.509 Schlüsselkontainern. Diese werden in der Datenbank gespeichert, wobei alle sensitiven Daten verschlüsselt abgelegt werden. Nach dem Erfassen können wichtige Felder wie der Canonical Name (CNAME) direkt in der Zertifikatsvorschau eingesehen werden. Weiter werden auch alle durch den strongSwan verwalteten Zertifikate in der strongMan Applikation dargestellt.

Zusätzlich bietet eine Informationsseite Übersicht über die verwendete strongSwan Version und allen installierten Plugins.

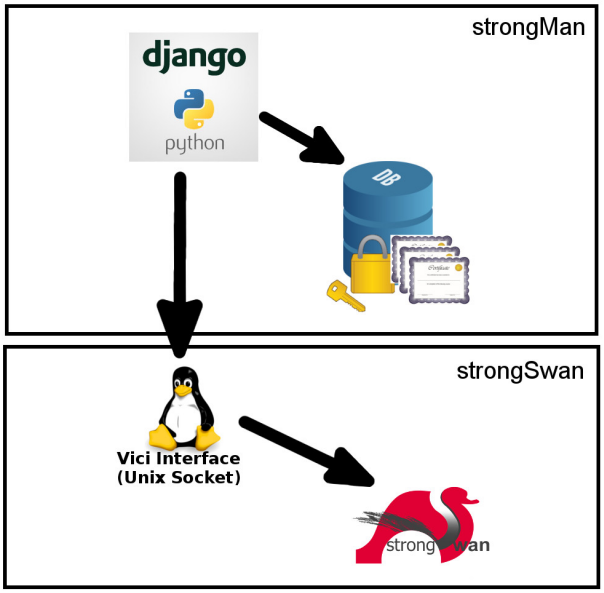


Abbildung 1: Übersicht

Der Fokus der strongMan Applikation zielt aktuell auf die Clientseite. In einem nächsten Schritt ist es möglich, die Anwendung so zu erweitern, dass auch Serverfunktionalitäten hinzukommen. Dazu ist ein Administratormodus sinnvoll, der weitere Authentisierungsmethoden ermöglicht, die Konfigurationsmöglichkeiten ausbaut, zusätzliche Statusinformationen visualisiert und das Management von mehreren Security Associations (SA) implementiert.

Inhaltsverzeichnis

0.1 Abstract . . . 2
0.2 Aufgabenstellung . . . 5

1 Analyse **7**
1.1 Einführung . . . 8
1.2 Anforderungsspezifikation . . . 9
1.2.1 Allgemeine Beschreibung . . . 9
1.2.2 Use Case . . . 10
1.2.3 Architektur . . . 12
1.2.4 Domain Model . . . 13
1.2.5 Nichtfunktionale Anforderungen . . . 15
1.3 Mockups . . . 17
1.4 Evaluation Usermanagement . . . 19
1.5 Evaluation Zertifikatsbibliothek . . . 20

2 Technischer Bericht **24**
2.1 Einführung . . . 25
2.2 Verbindungskonfigurationen . . . 26
2.2.1 Automatisch konfigurierte Parameter . . . 27
2.2.2 Manuell konfigurierte Parameter . . . 27
2.3 Implementation . . . 36
2.3.1 Einführung Django . . . 36
2.3.2 Struktur strongMan . . . 38
2.3.3 Kapselung Versatile IKE Configuration Interface . . . 39
2.3.4 Log Messages . . . 40
2.3.5 Statusinformationen . . . 41
2.3.6 Subklassen . . . 42
2.3.7 Verschlüsselte Datenbankfelder . . . 45
2.3.8 Zertifikate . . . 46
2.3.9 Webserver . . . 47
2.4 Testing . . . 49
2.4.1 Continuous Integration (CI) . . . 49
2.4.2 Usability Tests . . . 51
2.5 Abhängigkeiten . . . 55
2.5.1 Python . . . 55
2.5.2 Javascript/CSS . . . 55
2.6 Codestatistik . . . 56

2.7 Resultate . . . 57
2.7.1 Management-Oberfläche für VPN Clients . . . 57
2.7.2 Zertifikatsverwaltung . . . 57
2.7.3 Ausblick . . . 58
2.7.4 Auswertung . . . 59

3 Projektmanagement **60**
3.1 Rollen und Verantwortlichkeiten . . . 61
3.2 Entwicklungsumgebung und Infrastruktur . . . 62
3.3 Planung . . . 63
3.4 Risiken . . . 65
3.5 Arbeitsaufwand . . . 67
3.6 Erfahrungsbericht . . . 69

4 Softwaredokumentation **70**
4.1 Installation script setup.py . . . 71
4.2 Benutzerhandbuch . . . 73
4.2.1 Installation . . . 73
4.2.2 Uninstallation . . . 73

Anhang

A Mockups **74**

B Inhalt der CD **84**

C Eigenständigkeitserklärung **85**

D Glossar **86**

E Literaturverzeichnis **88**

F Abbildungsverzeichnis **90**

G Tabellenverzeichnis **92**

H Verzeichnis der Entscheide **93**

0.2 Aufgabenstellung

Institute for Internet Technologies and Applications

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

Bachelorarbeit 2016

Django-basiertes Management Tool für strongSwan

Studenten: Severin Bühler und Samuel Kurath

Betreuer: Prof. Dr. Andreas Steffen

Ausgabe: Montag, 22. Februar 2016

Abgabe: Freitag, 17. Juni 2016

Ausgangslage

Die strongSwan Open Source VPN Software wurde vor zwei Jahren mit dem neuen Versatile IKE Configuration Interface (VICI) ausgestattet, eine JSON-artige Schnittstelle, welche es erlaubt eine ManagementAnwendung über ein C, Ruby, Python oder Perl Binding an den Charon IKE Daemon anzubinden.

Aufgabenstellung

In dieser Arbeit soll auf der Basis von Django/Python eine grafische Management-Anwendung für strongSwan geschaffen werden, welche es erlaubt, IPsec Verbindungen über ein Web-Interface zu definieren, in einer Datenbank zu speichern und via die VICI-Schnittstelle an den IKE Daemon zu übermitteln. Weiter soll der Stand der aktuellen IPsec Verbindungen und andere statistische Daten abgefragt werden können.

Ziele

- Implementation einer grafischen Management-Oberfläche mit Django für strongSwan zur Konfiguration eines VPN Clients für folgende vier IKEv2 Authentisierungsmethoden:
  1) X.509 Zertifikat und privater RSA/ECDSA Schlüssel
  2) EAP mit Benutzername/Passwort
  3) Zweirunden-Authentisierung mit Methode 1) gefolgt von Methode 2)
  4) EAP-TLS mit X.509 Zertifikat und privatem RSA/ECDSA Schlüssel
- Oberfläche zur Verwaltung von X.509 End Entity und CA Zertifikaten, sowie privater RSA/ECDSA Schlüssel.
- Persistierung der Konfigurationsdaten in einer Datenbank.
- Verschlüsselte Ablage der RSA/ECDSA Authentisierungsschlüssel in der Datenbank.
- Starten und Stoppen von konfigurierten VPN Verbindungen
- Darstellung von Statusinformation über aktive VPN Verbindungen
- **Optional**: Oberfläche zur Konfiguration eines VPN Gateways

Links

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

[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

Rapperswil, 22. Februar 2016

Prof. Dr. Andreas Steffen

Kapitel 1

Analyse

1.1 Einführung

Vorbemerkung Bei diesem Dokument handelt es sich um die Bachelorarbeit von Samuel Kurath und Severin Bühler, erstellt an der Hochschule für Technik Rapperswil (HSR) im Studiengang Informatik. Betreut und begleitet wurde die Arbeit durch Prof. Dr. Andreas Steffen, Institut für Internettechnologien und Applikationen (ITA). Die Inhalte dieser Arbeit wurden von den Studenten zu gleichen Teilen erarbeitet.

Aktuell strongSwan ist eine Open-Source, IPsec basierte VPN Lösung für verschiedene Betriebssysteme. Es wird standardmässig per Konfigurationsdateien verwaltet und via Terminalanwendung gemanagt. Diese Vorgehensweise richtet sich hauptsächlich an versierte Systemadministratoren. Seit einiger Zeit steht eine einheitliche Schnittstelle für strongSwan (VICI) zur Verfügung und ermöglicht die Steuerung von strongSwan aus vielen verschiedenen Programmiersprachen.

Vision Im Verlaufe dieses Projekts soll eine Benutzeroberfläche zur Steuerung von VPN-Verbindungen entstehen. Die Applikation soll mehrere vordefinierte Authentisierungsmethoden unterstützen und entsprechende Statusinformationen anzeigen können. Es soll für einen normalen Benutzer mithilfe der Oberfläche möglich sein eine VPN Verbindung zu öffnen und zu verwalten.

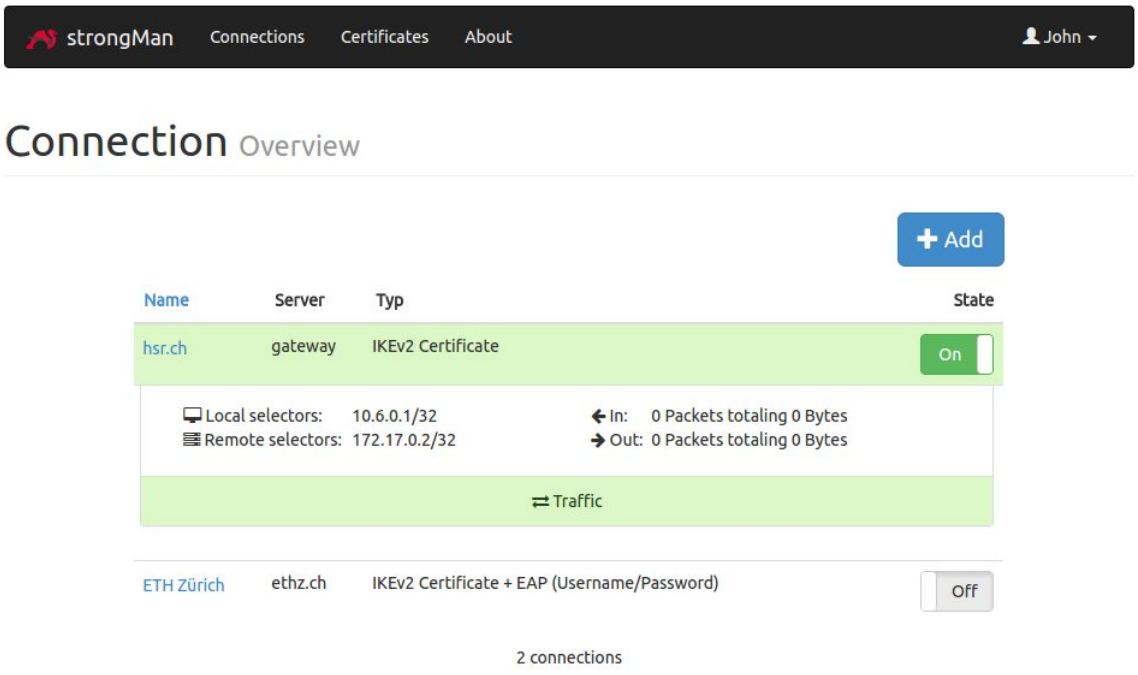


Abbildung 1.1: strongMan Verbindungsübersicht

1.2 Anforderungsspezifikation

1.2.1 Allgemeine Beschreibung

Generell sind zwei Anwendungsszenarien denkbar:

- VPN-Client
- VPN-Server

Dabei ist der VPN-Client eine **Muss**-Anforderung und der VPN-Server eine **Kann**-Anforderung.

VPN-Client

Die Applikation wird von einem Standard-Nutzer verwendet. Dieser soll mit wenigen Eingaben eine Verbindungskonfiguration zum Auf- und Abbau von VPN-Tunnels zu einem Server erstellen können. Die Konfigurationsmöglichkeiten sind beschränkt, als Vorlage wird der strongSwan Android Client verwendet.

VPN-Server

Der Server ist auf Systemadministratoren ausgerichtet. Es soll möglich sein Verbindungskonfigurationen aus Serversicht zu erstellen. Mithilfe dieser Konfigurationen können VPN-Tunnels aufgebaut werden. Dabei soll es möglich sein, mehrere VPN-Tunnel pro Konfiguration zu etablieren.

1.2.2 Use Case

Use Case Diagramm

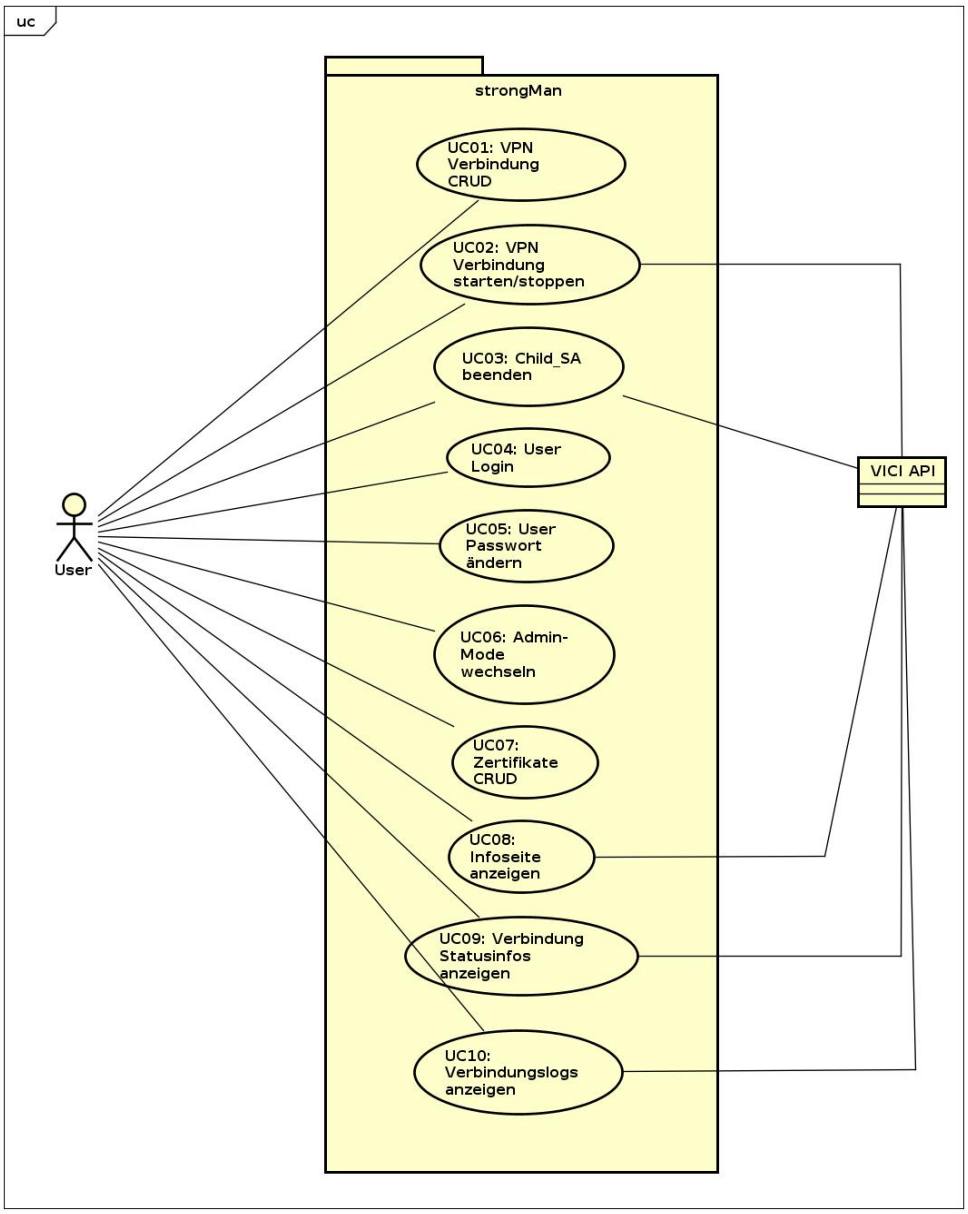


Abbildung 1.2: Use Case Diagramm

Use Cases Brief

Alle hier definierten Use Cases haben auch ein entsprechendes Mockup im Anhang.

UC01: VPN-Verbindung CRUD
Der User kann eine VPN-Verbindung erfassen / konfigurieren. Dabei hat er eine Auswahl von verschiedenen vordefinierten Verbindungstypen. Jeder Verbindungstyp hat vordefinierte Konfigurationsfelder, die der User ausfüllen muss. Die Verbindungs-Übersichtsseite stellt die Hauptseite der Applikation dar. Dort können die Verbindungen bearbeitet und gelöscht werden.

UC02: VPN-Verbindung starten/stoppen
Der User kann eine erfasste VPN-Verbindung starten und stoppen. Dabei wird die Konfiguration über die VICI-Schnittstelle geladen. Falls eine VPN-Verbindung nicht aufgebaut werden kann, soll eine passende Fehlermeldung angezeigt werden.

UC03: Child_SA beenden
Optional: Jede VPN-Verbindung kann mehrere Child_SA aufbauen. Diese werden in der Hauptseite angezeigt und können vom User beendet werden. Auch dieser Use Case interagiert mit der VICI-Schnittstelle. Dies ist ein Use Case des VPN-Servers.

UC04: User Login
Der User loggt sich zu Beginn des Webseiten Aufrufs mit einem Usernamen und Passwort ein. Es existiert nur ein Benutzer.

UC05: User Passwort ändern
Sobald der User eingeloggt ist, hat er die Möglichkeit, sein Passwort zu ändern. Dabei gibt er sein altes Passwort einmal und sein neues Passwort zweimal an.

UC06: Admin-Mode wechseln
Optional: Das Userinterface unterscheidet zwischen zwei Modis: User- und Admin-Mode. Der Modus soll über das Userinterface wechselbar sein. Der Admin-Mode stellt einige Server-spezifische Funktionalitäten zusätzlich zur Verfügung, welche der User zur einfacheren Bedienung nicht sieht.

UC07: Zertifikate CRUD
Dem User wird eine Zertifikatsverwaltung zur Verfügung gestellt. Er kann Zertifikate und private Schlüssel in den gängigen Formaten uploaden, anschauen und wieder löschen. Die Dateien können mit einem Passwort verschlüsselt sein.

UC08: Infoseite anzeigen
Die Informationsseite zeigt dem eingeloggten User verschieden Informationen des strongSwans zum Beispiel Version, installierte Plugins usw.

UC09: Verbindung Statusinfos anzeigen
Die Statusinformationen zu einer Verbindung zeigt dem User Traffic Selektoren und den Input/Output Traffic.

UC10: Verbindungslogs anzeigen
Zeigt die Loginformationen an, die beim Verbindungsauf- und abbau von strongSwan empfangen werden.

1.2.3 Architektur

Übersicht

Die strongMan Applikation ist webbasiert, der Aufbau gliedert sich in folgende Komponenten.

Webbrowser ist der Einstiegspunkt für den Benutzer und übernimmt die Darstellung mithilfe von HTML und CSS. Die Kommunikation findet per HTTP statt. Durch den Einsatz von Javascript wirkt das Arbeiten mit der Oberfläche flüssiger.

Django Server beinhaltet die eigentliche Business Logik, stellt dem Webbrowser den Inhalt zur Verfügung, regelt die Kommunikation mit strongSwan per Unix Socket und nutzt zur Persisterung der Daten die Datenbank.

Database beinhaltet die Konfigurationsangaben, sowie die Zertifikate. Sensitive Daten werden verschlüsselt gespeichert.

strongSwan baut die IPsec-Tunnels auf und ab. Der Django Server nutzt die VICI-Schnittstelle um Verbindungskonfiguration zu übergeben und die Verbindungen zu starten beziehungsweise zu stoppen.

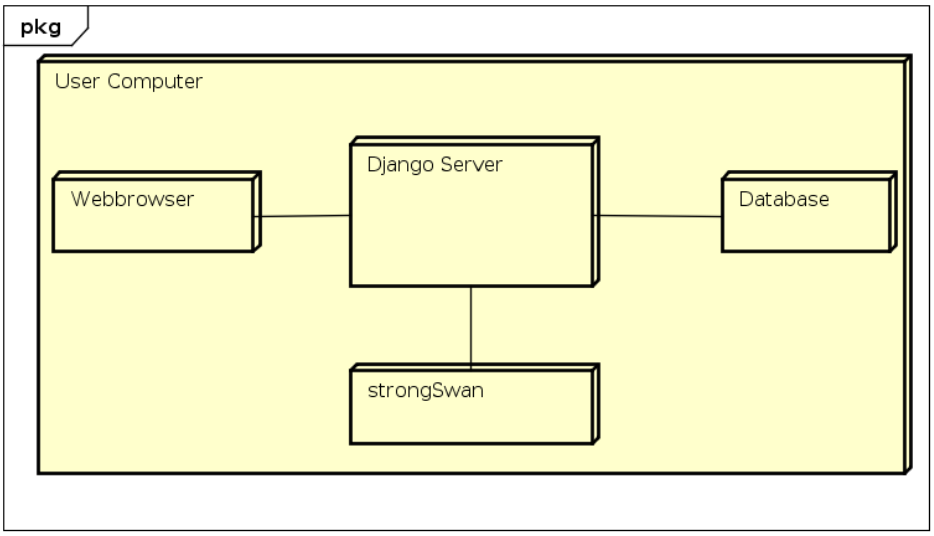


Abbildung 1.3: Deployment

1.2.4 Domain Model

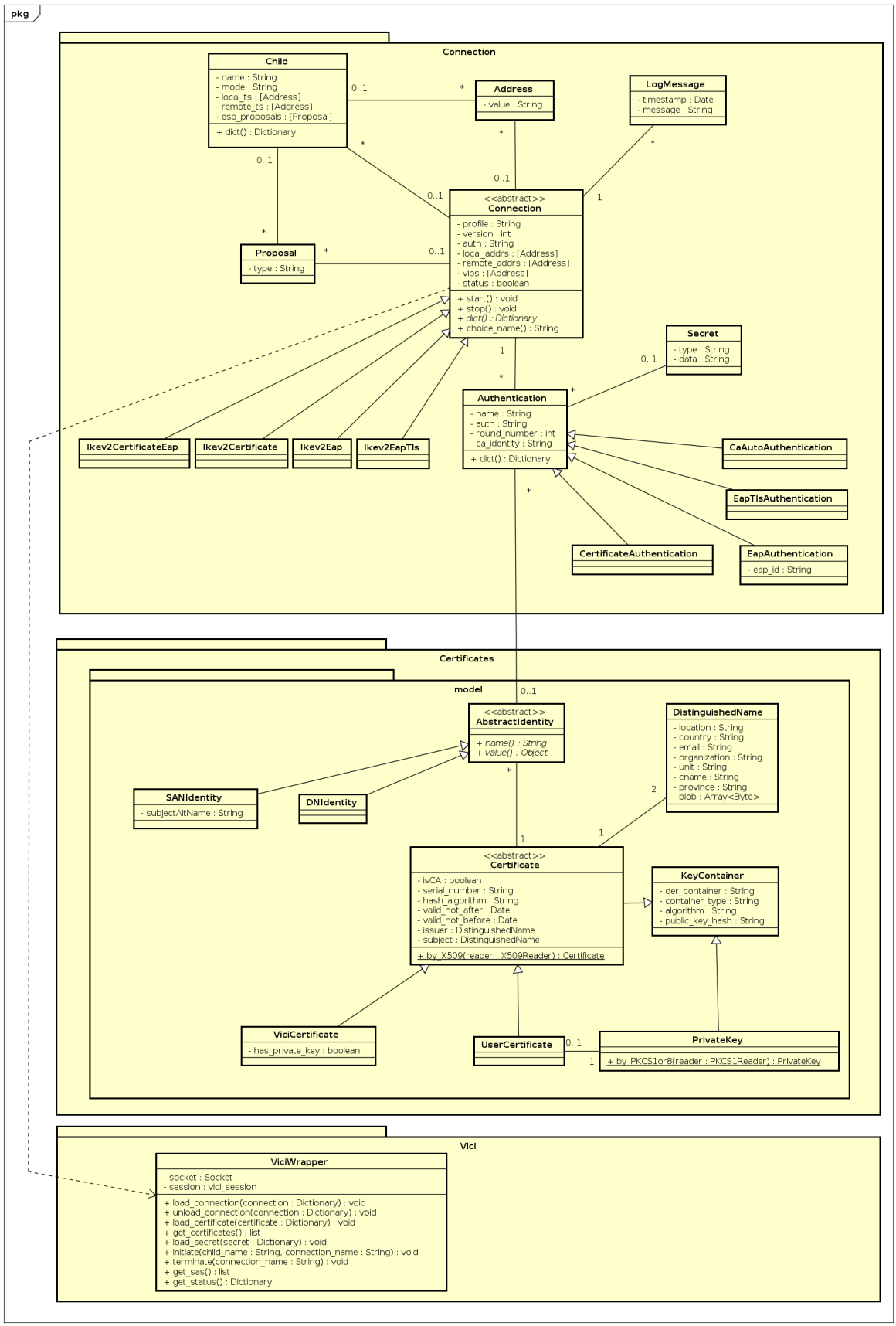


Abbildung 1.4: Domain Model

Das Domain Model teilt sich in drei Teile VICI, Certificates und Connections auf. Diese bauen dabei hierarchisch aufeinander auf.

VICI

Das VICI Package beinhaltet den ViciWrapper, der verschiedene Befehle aus der VICI Bibliothek bündelt, sowie die Exceptions managet. Der Wrapper ist selbst für den Verbindungsauf- und Abbau zuständig.

Certificates

Das Package Certificates beinhaltet die Klassen zur Zertifikatsverwaltung. Alle Schlüsselcontainer, die im Uploadfeld hinaufgeladen werden, werden mithilfe des ORM's in dieses Package abgespeichert. Die Hauptklasse des Packages sind UserCertificate, ViciCertificate und PrivateKey.

UserCertificate Stellt ein X.509 Zertifikat dar, das vom Benutzer hochgeladen wird.

ViciCertificate Ein X.509 Zertifikat, das aus der VICI-Schnittstelle gelesen wird.

PrivateKey Privater Schlüssel, der vom Benutzer geuploaded wird und eine Verbindung zum entsprechenden UserCertificate hat.

Identity Eine Identity ist entweder ein Subject Alternative Name oder ein Distinguished Name. Es dient als Verbindungsstück zum Connection Package.

Connections

Das Connection Package ist im Allgemeinen eine Abstraktion der VICI-Schnittstelle. Um die verschieden Verbindungstypen abzubilden, wird auf Vererbung gesetzt. Die Methode dict() stellt alle verbindungsrelevanten Einträge aus der Datenbank in einem ordered Dictionary zusammen, welche über den ViciWrapper an strongSwan übergeben werden. Durch start() und stop() werden die übertragenen IPsec Verbindungen auf- und abgebaut.

LogMessage Wird verwendet um LogMessages über mehrere Requests zu verteilen.

Secret Bildet die Secret Section der VICI-Schnittstelle ab, dabei wird das Data Feld verschlüsselt in der Datenbank abgelegt.

1.2.5 Nichtfunktionale Anforderungen

Funktionalität

Sicherheit Änderung an den Daten der Anwendung dürfen nur authentifizierte Benutzer machen. Nicht authentifizierte Benutzer werden auf die Anmeldeseite weitergeleitet.

Sensitive Daten Kritische Daten sind das Passwort für den Benutzer/Adminstrator, die Private Keys und Secret Konfigurationsdaten (z.B. EAP Passwort). Diese müssen verschlüsselt in der Datenbank gespeichert werden.

Passwort Anforderungen

- minimal 8 Zeichen
- maximal 60 Zeichen
- minimal 1 Sonderzeichen und 1 Zahl
- minimal 1 Grossbuchstaben
- minimal 1 Kleinbuchstaben

Weitere Schutzmassnahmen

- Die Applikation muss vor SQL Injection sicher sein
- Es sollen CSFR Tokens verwendet, um vor Cross Site Attacken sicher zu sein.

Interoperabilität Als externe Schnittstelle ist das VICI Interface aufzuzählen, dabei wir über Unix Sockets kommuniziert. strongSwan bietet dazu schon eine Python Implementation an.

Zuverlässigkeit

Wiederherstellbarkeit Nach einem Systemabsturz oder Stopp, soll die Anwendung ohne Komplikationen wieder gestartet werden können. Der User muss sich neu einloggen.

Fehlertoleranz Bei einem Anwendungsfehler sollte nur die Operation des verursachten Benutzers abgebrochen werden. Andere Benutzer sollten vom Fehler nicht betroffen sein.
Bsp.: Will man ein Zertifikat löschen, welches noch von einer Verbindungskonfiguration verwendet wird, ist dieses nicht löschbar.

Benutzbarkeit

Erlernbarkeit Neue Benutzer sollten die grundlegenden Funktionen durch Verwenden der Anwendung und eines Tutorials erlernen können.

Bedienbarkeit Die Applikation sollte sowohl auf einem Smartphone, wie auch auf einem Desktopcomputer bedienbar sein, weshalb wir mit einem responsiven Webdesign arbeiten.

Minimale Auflösung: 640 × 960 px
Maximale Auflösung: 1920 x 1080 px

Wir verwenden dabei zwei Skalierungsstufen (Smartphones, Desktopcomputer).

Robustheit Jedes Eingabefeld soll durch die Webseite validiert werden. Falsche Eingaben werden angezeigt, damit der Benutzer seine Eingabe korrigieren kann. Für jede falsche Eingabe soll eine kleiner Hilfetext erscheinen.

Effizienz Die Anwendung wird für einen zur gleichen Zeit arbeitenden User spezifiziert. Es sollten theoretisch mehrere User auf der Anwendung arbeiten können, es existiert aber nur ein Userlogin und es können im schlimmsten Fall Race Conditions auftreten. Die Ladezeit für die Anwendung soll für jede Operation unter einer Sekunde bleiben. Bezüglich Geschwindigkeit muss das Projekt nicht spezifisch auf "langsame Devices" wie Router oder Raspberry Pi's achten.

Änderbarkeit

Die Applikation soll auch nach Projektabschluss betreubar sein. Daher sollen folgende Anforderungen erfüllt werden:

- Der Coding-Standard PEP8 für Python soll eingehalten werden.
- Klassen und nicht-triviale Methoden sollen im Code dokumentiert werden (Doc-Strings)

Modifizierbarkeit Es soll bei der Entwicklung darauf geachtet werden, zukünftige Erweiterungen nicht durch Design-Entscheide auszuschliessen.

Übertragbarkeit

Server: Die Anwendung muss mithilfe eines Django kompatiblen Webservers zum laufen gebracht werden. Zusätzlich zu den Django Anforderungen muss strongSwan mit der VICI-Schnittstelle installiert sein.

Client: Auf dem Client muss ein aktueller Browser mit HTML5 und Javascript Unterstützung installiert sein. Die Applikation wird mit Google Chrome 49 und Firefox 45 getestet.

Internationalization Die Applikation ist in der Englischen Sprache verfügbar.

Testing Die Hauptfunktionen im Projekt sollten mittels Unit Tests getestet werden. Die Haupt Use Cases müssen mit Integration Tests überprüft werden. Es sind keine Stresstests mit Loadtesting Frameworks wie Gatling vorgesehen.

1.3 Mockups

Um allen Projektbeteiligten ein Bild der zukünftigen Applikation zu vermitteln, sind für die wichtigsten Seiten Mockups entstanden. In diesem Abschnitt wird auf zwei Ansichten auf die Connection Verwaltung eingegangen. Alle anderen Mockups sind im Anhang auf der Seite 75 ersichtlich.
Dank der Mockups konnten diverse nicht angedachte Use Cases und Features ermittelt werden.

Connection Übersicht

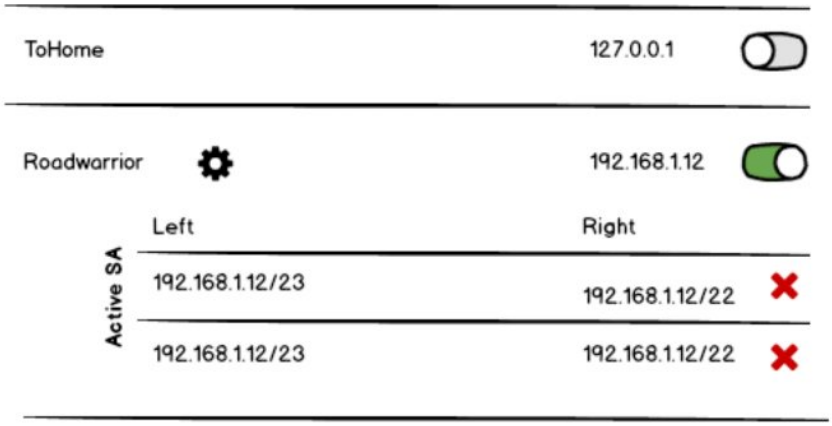


Abbildung 1.5: Ausschnitt aus der Connection Übersicht, UC01: VPN Verbindung CRUD

Die erstellten Verbindungen werden in Zeilenform untereinander angeordnet. Es gibt einen zentralen Toggle Button, der die Verbindung aktiviert/deaktiviert und gleichzeitig den Verbindungsstatus anzeigt. Nachdem eine Verbindung gestartet und diese erfolgreich aufgebaut wurde, erweitert sich die Ansicht und es werden Statusinformationen angezeigt.

Connection hinzufügen

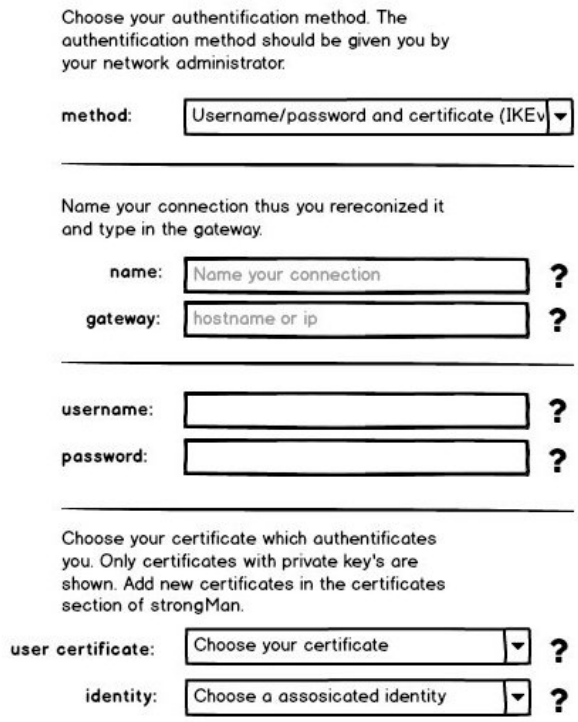


Abbildung 1.6: Ausschnitt aus dem Connection hinzufügen, UC01: VPN Verbindung CRUD

Nachdem der Benutzer die Authentifizierungsmethod ausgewählt hat (hier nicht ersichtlich), erscheinen unterhalb die entsprechenden Eingabefelder.
Das Konfigurieren einer Verbindung wird so einfach wie möglich gestaltet. Der Benutzer soll kurz vor jeder Eingabe informiert werden, welchen Nutzen das entsprechende Feld hat und sollte sich bei Schwierigkeiten an den Hilfetexten orientieren können (klick auf die Fragezeichen).

1.4 Evaluation Usermanagement

Da mithilfe von stongSwan VPN Verbindungen zu schützenswerten Daten ermöglicht werden, sollte auch die Applikation strongMan mit einer Authentifizierung versehen werden.

Anforderungen

- Benutzer hat ohne sich einzuloggen keinen Zugriff auf die Applikation
- Benutzer hat die Möglichkeit sein Passwort zu ändern

Varianten

- Ein vorkonfigurierter Benutzer
- Mehrere Benutzer
- Benutzer des Betriebssystems verwenden

Auswertung

Variante	Implementierung	Framework Unterstützung
Ein vorkonfigurierter Benutzer	leicht	gegeben
Mehrere Benutzer	mittel	gegeben
Benutzer des Betriebssystems verwenden	schwer	nicht gegeben

Tabelle 1.1: Evaluation Usermanagement

Entscheid 1. Evaluation Usermanagement

Durch das Vergleichen der verschieden Varianten und in Absprache mit den Betreuern hat sich ”Ein vorkonfigurierter Benutzer” als Favorit herausgestellt. Dabei wir ein Standard Benutzer während der Installation der Applikation erstellt und dessen Passwort kann im Anschluss geändert werden. Die Optionen mit mehreren Benutzern wäre vor allem im Hinblick auf den Serverteil vorteilhaft, bedeuten aber einen grösseren Implemenationsaufwand.

1.5 Evaluation Zertifikatsbibliothek

Um VPN Verbindungen mit Zertifikaten abzusichern, muss das strongMan mit Zertifikaten umgehen können. Dabei müssen verschiedene Formate unterstützt, ausgelesen und entsprechend ihrem Inhalt an die VICI-Schnittstelle geliefert werden.

ASN.1 Schemas

In der Welt der asymmetrischen Verschlüsslung werden alle nötigen Daten in einer ASN.1 Struktur abgelegt. Der ASN.1 Standard stellt Datentypen und Datenstrukturen zur Verfügung, um Informationen mittels Schemas abzulegen. Über die Zeit sind verschiedene RFC's entstanden, die ASN.1 Schemas definieren. Unten aufgelistet sind die Standards, welche wir in unserem Projekt berücksichtigen.

Standard	Inhalt	Forderung
PKCS1	Private Schlüssel der verschiedene Algorithmen RSA, DSA und ECDSA. Können mit einem Passwort verschlüsselt sein.	Muss
PKCS8	Gleich wie PKCS1, zusätzlich mit einem Feld, der den Algorithmus beschreibt.	Muss
X.509	Zertifikatsstandard mit allen Informationen zum Eigner (Unverschlüsselt).	Muss
PKCS12	Container mit Zertifikaten und privaten Schlüsseln. Kann verschlüsselt sein.	Kann

Tabelle 1.2: ASN.1 Schemas

Encoding

Die ASN.1 Strukturen werden nicht direkt in die Dateien abgespeichert, sondern zuerst in einem speziellen Format codiert. Dabei sind zwei verschiedene Standards üblich.

Standard	Inhalt	Forderung
DER	Binär kodiert und somit nicht in einem Editor lesbar (Dateiendung .der).	Muss
PEM	Base64 encodet in ASCII Zeichen (Dateiendung .pem). Openssl nutzt PEM als Standardencoding.	Muss

Tabelle 1.3: Encoding

Verschlüsslung

Die Dateien mit Private Keys (PKCS1, PKCS8 und PKCS12) können verschlüsselt sein. Dabei kommen verschiedene symmetrische Verschlüsslungen zum Einsatz, die mit einem Passwort entschlüsselt werden.

Anforderungen

In dieser Evaluation soll eine Python Crypto Bibliothek gefunden werden, mit der man grundlegende Informationen wie CNAME und ASN.1 Schema auslesen kann. Mithilfe der Anforderungen im Anschluss soll eine passende Bibliothek gefunden werden.

Muss Anforderungen

- Unterscheidung Private Key / Zertifikat
- Erkennung von CA Zertifikaten
- Unterstützt alle obigen ASN.1 Schemas
- Unterstützt PEM und DER Encoding
- Aus X.509 können Informationen wie der CNAME ausgelesen werden

Kann Anforderungen

- Private Keys können entschlüsselt werden
- Bibliothek ist pure Python (Keine Kompilation bei Installation)
- PKCS12 wird unterstützt
- Matching zwischen Public und Private Key ist möglich
- Einfaches API

Kandidaten

Pyasn1 [Eti16]

Anforderungen	Muss / Kann	Erfüllt
Unterscheidung Private Key / Zertifikat	Muss	x
Erkennung von CA Zertifikaten	Muss	x
Unterstützt alle obigen ASN.1 Schemas	Muss	x
Unterstützt PEM und DER Encoding	Muss	x
Aus X.509 können Infos wie der CNAME ausgelesen werden	Muss	x
Private Keys können entschlüsselt werden	Kann	
Bibliothek ist pure Python (Keine Kompilation bei Installation)	Kann	x
PKCS12 wird unterstützt	Kann	
Matching zwischen Public und Private Key ist möglich	Kann	
Einfaches API	Kann	

Tabelle 1.4: Pyasn1

Pyasn1 stellt einzelnen ASN.1 Schemas für die Dekodierung von ASN.1 Dateien zur Verfügung. Einzelne Standards wie X.509 sind schon implementiert, jedoch ist man selbst verantwortlich, das richtige Schema zu der entsprechenden Datei zu finden. Das API ist sehr kompliziert und der Programmieraufwand für die Unterstützung aller nötigen Formate ist gross.

Cryptography [cd16]

Anforderungen	Muss / Kann	Erfüllt
Unterscheidung Private Key / Zertifikat	Muss	
Erkennung von CA Zertifikaten	Muss	x
Unterstützt alle obigen ASN.1 Schemas	Muss	
Unterstützt PEM und DER Encoding	Muss	x
Aus X.509 können Infos wie der CNAME ausgelesen werden	Muss	x
Private Keys können entschlüsselt werden	Kann	
Bibliothek ist pure Python (Keine Kompilation bei Installation)	Kann	
PKCS12 wird unterstützt	Kann	
Matching zwischen Public und Private Key ist möglich	Kann	
Einfaches API	Kann	x

Tabelle 1.5: Cryptography

Cryptography ist eine Python Bibliothek mit dem Fokus auf komfortable Kryptografie in Python. Es hat eine tolle Unterstützung für X.509 Zertifikate. Man kann Private Keys generieren, ein CSR erstellen und sogar CSR signieren. Im Hintergrund verwendet es OpenSSL. Nichts desto trotz bietet es keinen Support ausserhalb vom X.509 und ist somit unbrauchbar für unsere Zwecke.

Pycrypto [Lit16]

Anforderungen	Muss / Kann	Erfüllt
Unterscheidung Private Key / Zertifikat	Muss	x
Erkennung von CA Zertifikaten	Muss	x
Unterstützt alle obigen ASN.1 Schemas	Muss	x
Unterstützt PEM und DER Encoding	Muss	x
Aus X.509 können Infos wie der CNAME ausgelesen werden	Muss	x
Private Keys können entschlüsselt werden	Kann	
Bibliothek ist pure Python (Keine Kompilation bei Installation)	Kann	
PKCS12 wird unterstützt	Kann	
Matching zwischen Public und Private Key ist möglich	Kann	x
Einfaches API	Kann	

Tabelle 1.6: Pycrypto

Pycrypto unterstützt alle geforderten Formate inklusive Verschlüsslung. Es basiert auf der OpenSSL Bibliothek, welche davor zuerst installiert werden muss. Die Anwendung von Pycrypto erfordert jedoch einiges an Code, um jedes Format durchzutesten und um die entsprechenden Formate zu erkennen.

Oscrypto [wbo16]

Anforderungen	Muss / Kann	Erfüllt
Unterscheidung Private Key / Zertifikat	Muss	x
Erkennung von CA Zertifikaten	Muss	x
Unterstützt alle obigen ASN.1 Schemas	Muss	x
Unterstützt PEM und DER Encoding	Muss	x
Aus X.509 können Infos wie der CNAME ausgelesen werden	Muss	x
Private Keys können entschlüsselt werden	Kann	x
Bibliothek ist pure Python (Keine Kompilation bei Installation)	Kann	
PKCS12 wird unterstützt	Kann	x
Matching zwischen Public und Private Key ist möglich	Kann	
Einfaches API	Kann	x

Tabelle 1.7: Oscrypto

Oscrypto ist eine neu entstandene Crypto Bibliothek für Python. Es unterstützt alle nötigen Formate inklusive Verschlüsslung. Die meisten Funktionen können ohne die OpenSSL Bibliothek im Hintergrund verwendet werden. Mit wenigen Befehlen kann oscrypto alle nötigen Formate lesen.

Entscheid 2. Evaluation Zertifikatsbibliothek

Der Entscheid fiel klar auf oscrypto. Oscrypto ist eine pure Python Bibliothek, welche verschiedene Kryptografiefunktionen anbietet. Für das Parsen und Lesen von Keycontainern wrappt oscrypto die asn1crypto Bibliothek (auch pure Python). Für andere Kryptografiefunktionen wrappt oscrypto die standartmässig installierte OpenSSL Library, welche wir in dieser Arbeit aber nicht benötigen. Mithilfe dieser Library ist nun ein komfortables Arbeiten mit allen oben aufgeführten Keycontainern möglich, ohne Abhängigkeiten ausserhalb von Python zu haben.

Kapitel 2

Technischer Bericht

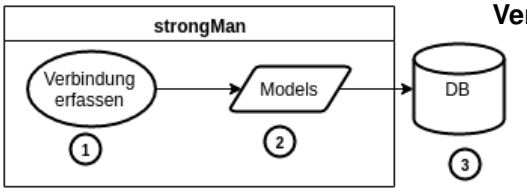
2.1 Einführung

Im Kapitel Technischer Bericht werden zuerst die von strongMan unterstützten Verbindungstypen mit ihren Implementationsdetails vorgestellt. Der zweite Teil dieses Kapitels beschreibt verschiedene Implementationshürden mit ihren Problematiken und Lösungen. Zu guter Letzt sind allgemeine Informationen von verwendeten Abhängigkeiten, Codestatistiken und die Resultate dokumentiert.

2.2 Verbindungskonfigurationen

Unter einer Verbindungskonfiguration werden alle Angaben zu einem IPsec-Tunnel verstanden, dazu zählen Authentiserungsmethode und die dazugehörigen Parameter, Zertifikate und private Schlüssel, Angaben zu Security Associations, sowie anfallende Secrets.

Ablauf Die folgenden beiden Grafiken demonstrieren das Zusammenspiel zwischen den Konfigurationen und den schlussendlich aufgebauten Verbindungen.



Verbindung erfassen

1. Eine Verbindung wird über das Webformular erfasst.
2. Die Eingaben werden in Django Models umgewandelt.
3. Die Models werden in der Datenbank persisitiert.

Abbildung 2.1: Verbindung erfassen

Verbindung starten

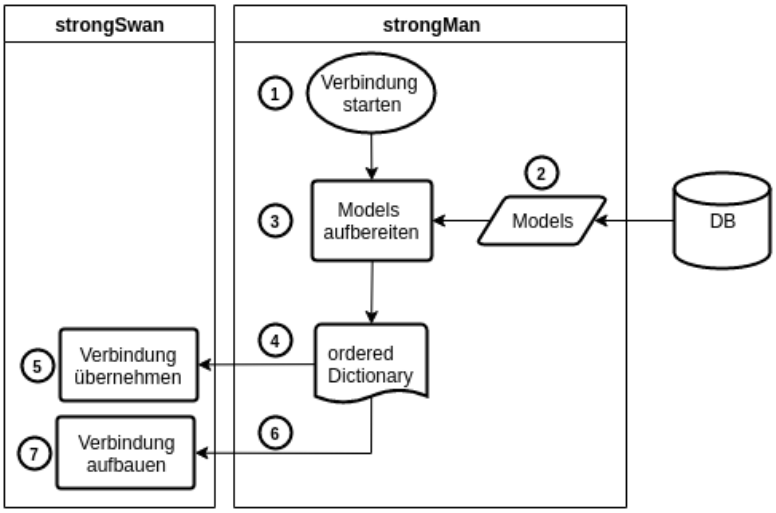


Abbildung 2.2: Verbindung starten

1. Eine erfasste Verbindung wird über die Management-Oberfläche gestartet.
2. Die Angaben zu der Verbindung werden aus der Datenbank gelesen und als Django Models instantiiert und initialisiert.
3. Aus den Attributen der Models wird ein Python ordered Dictionary generiert. (Python ordered Dictionaries sind das unterstütze Format der VICI-Schnittstelle, siehe Seite 39)
4. Das ordered Dictionary wird über die VICI-Schnittstelle dem strongSwan übergeben.
5. strongSwan übernimmt die Verbindung.
6. strongMan gibt den Befehl zum Etablieren der Verbindung.
7. strongSwan baut ein IPsec-Tunnel anhand der übernommen Verbindung auf.

2.2.1 Automatisch konfigurierte Parameter

Um die Anzahl der Parameter, welche der Benutzer eintragen muss möglichst gering zu halten, werden weitere Parameter durch die strongMan Applikation automatisch gesetzt.

Parameter	Wert	Erklärung
vips	0.0.0.0, ::	Virtuelle IP, welche dem Client vom Server zugewiesen wird. 0.0.0.0 und :: dienen als Wildcard-Adressen für IPv4 und IPv6, damit wird jede virtuelle IP akzeptiert.
version	2	IKE Version, es wird nur die Version 2 von strongMan verwendet.
proposals	default	Proposals sind Vorschläge von Authentisierungs- und Verschlüsselungsalgorithmen. Der default Wert führt dazu, dass automatisch als sicher geltende Vorschläge gemacht werden.
esp_proposals	aes128gcm128-modp2048	ESP ist für die Sicherung von Authentizität, Integrität und Vertraulichkeit der übertragenen IP-Pakete zuständig. Mit den esp_proposals werden die verwendeten Algorithmen vorgeschlagen.
auth	pubkey, eap, eap-tls	Definiert die verwendete Authentisierungsmethode.
round	1, 2	Der Verbindungstyp Certificate + EAP benötigt zwei Authentiserungsrunden. Dieser Parameter wird erst ab strongSwan Version 5.4.0 unterstützt.

Tabelle 2.1: Automatisch konfigurierte Parameter

2.2.2 Manuell konfigurierte Parameter

Wird eine neue Verbindung erfasst, kann zuerst zwischen den Verbindungstypen (Siehe Tabelle unten) ausgewählt werden, diese repräsentieren die verschiedenen Authentiserungsmethoden.

Nr	Typ	Authentiserungsmethode
1	IKEv2 Certificate	X.509 Zertifikat und privater RSA/ECDSA Schlüssel
2	IKEv2 EAP (Username/Password)	EAP mit Benutzername/Passwort
3	IKEv2 Certificate + EAP (Username/Password)	Zweirunden-Authentisierung mit Methode 1) gefolgt von Methode 2)
4	IKEv2 EAP-TLS	EAP-TLS mit X.509 Zertifikat und privatem RSA/ECDSA Schlüssel

Tabelle 2.2: Automatisch konfigurierte Parameter

Im Anschluss wird auf jeden Verbindungstypen und dessen Parameter, welche manuell über die Eingabefelder erfasst werden, eingegangen. Dazu ist jeweils die durch den strongMan generierte Konfiguration, wie auch eine Beispielkonfiguration für die Serverseite aufgeführt.

IKEv2 Certificate

Die Authentisierung zwischen Client und Server findet auf Basis eines X.509 Zertifikats und privater RSA/ECDSA Schlüssel statt.

Eingabefelder

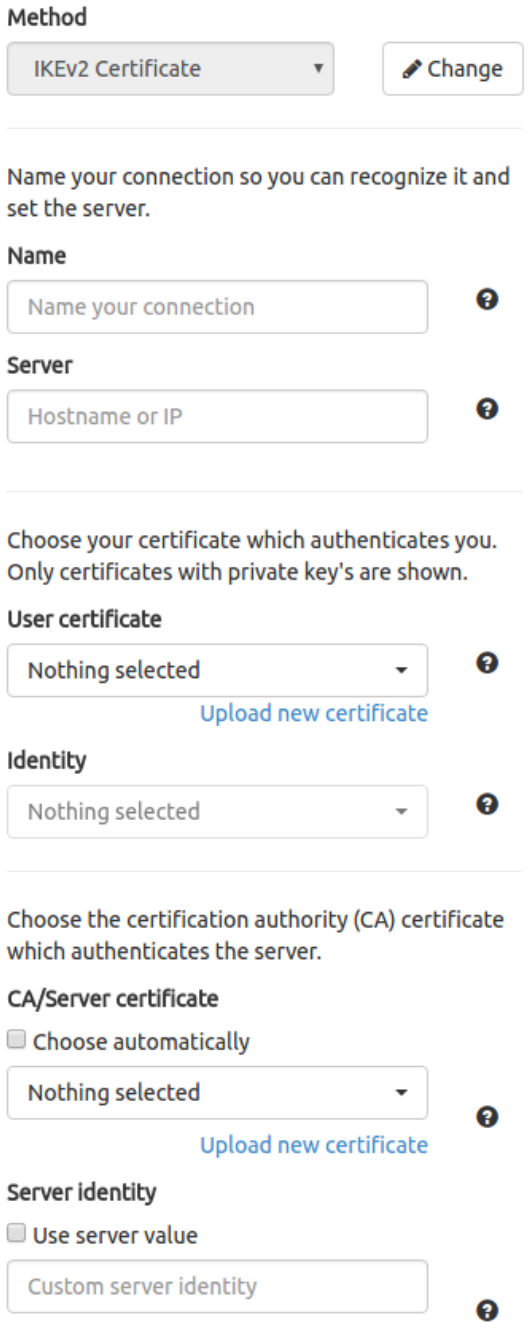


Abbildung 2.3: IKEv2 Certificate

Beschreibung

Name
Ordered Dictionary: 'Bezeichner'
Eindeutige Bezeichnung für die Verbindung

Server
Ordered Dictionary: remote_addrs
Hostname oder IP des Servers

User certificate
Ordered Dictionary: local.certs
Auswahl der Benutzerzertifikate. Im ordered Dictionary wird der komplette DER-Container in Binärer Form übermittelt.

Identitiy
Ordered Dictionary: local.id
Auswahl zwischen Subject Alternative Name und Distinguished Name. Standardmässig wird der Distinguished Name gesetzt, somit wird nur bei Auswahl eines Subject Alternative Name die local.id in das ordered Dictionary geschrieben.

CA/Server certificate
Ordered Dictionary: remote.cacerts
Die automatische Auswahl bewirkt, dass alle CA/-Server Zertifikate an den strongSwan übergeben werden und dieser selbständig entscheidet. Ansonsten wird das Passende manuell ausgewählt.

Server identity
Ordered Dictionary: remote.id
Bezeichner für die Serveridentifikation. Per Default wird der **Server** Eintrag übernommen. Ansonsten auch manuell setzbar.

IKEv2 Certificate ordered Dictionaries

Client

```
{
    "cert": {
        "remote_addrs": [
            "gateway"
        ],
        "vips": [
            "0.0.0.0",
            "::"
        ],
        "version": 2,
        "proposals": [
            "default"
        ],
        "children": {
            "cert": {
                "remote_ts": [
                    "::/0",
                    "0.0.0.0/0"
                ],
                "esp_proposals": [
                    ''aes128gcm128-
                    modp2048''
                ]
            }
        },
        "local": {
            "round": 1,
            "auth": "pubkey"
            "certs": [
                "b'bytes_of_cert'"
            ]
        },
        "remote-cert": {
            "round": 1,
            "auth": "pubkey",
            "id": "moon.strongswan.org"
            "cacerts": [
                "b'bytes_of_cert'"
            ]
        }
    }
}
```

Server

```
{
    "server-cert": {
        "pools": [
            "server-pool"
        ],
        "local": {
            "auth": "pubkey",
            "id": "moon.strongswan.org",
            "certs": [
                "moonCert.pem"
            ]
        },
        "remote": {
            "auth": "pubkey"
        },
        "version": 2,
        "proposals": [
            "aes128-sha256-modp2048"
        ],
        "children": {
            "server-cert": {
                "esp_proposals": [
                    ''aes128gcm128-
                    modp2048''
                ]
            }
        }
    }
}
```

Pools

```
"server-pool": {
    "addrs": "10.6.0.0/24"
}
```

IKEv2 EAP (Username/Password)

Die Authentisierung zwischen Client und Server findet auf Basis von EAP mithilfe eines Benutzernamens und Passwortes statt.

Eingabefelder

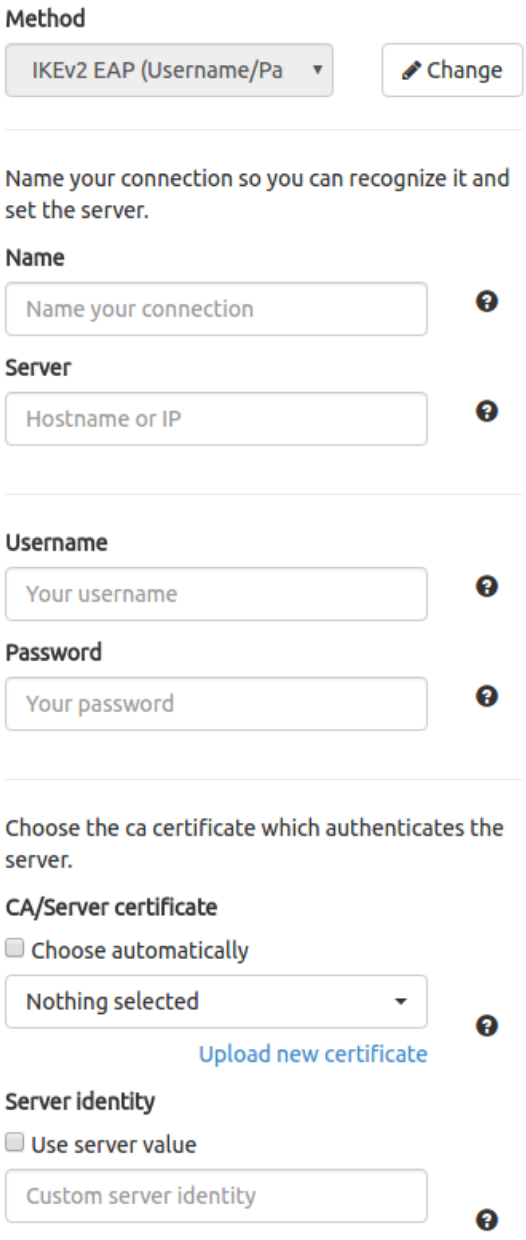


Abbildung 2.4: IKEv2 EAP

Beschreibung

Name
Siehe Seite 28

Server
Siehe Seite 28

Username
Ordered Dictionary: secrets.id, local.eap_id
Username wird als als Referenz zwischen den Secrets und der local Section verwendet.

Password
Ordered Dictionary: secrets.data
Das Passwort für die EAP Authentisierung, es wird verschlüsselt in der Datenbank gespeichert und muss sowohl auf Server-, wie auch Clientseite gesetzt sein.

CA/Server certificate
Siehe Seite 28

Server identity
Siehe Seite 28

IKEv2 EAP (Username/Password) ordered Dictionaries

Client

```
{
    "eap": {
        "remote_addrs": [
            "gateway"
        ],
        "vips": [
            "0.0.0.0",
            "::"
        ],
        "version": 2,
        "proposals": [
            "default"
        ],
        "children": {
            "eap": {
                "remote_ts": [
                    "::/0",
                    "0.0.0.0/0"
                ],
                "esp_proposals": [
                    ''aes128gcm128-
                    modp2048''
                ]
            }
        },
        "local-eap": {
            "round": 1,
            "auth": "eap",
            "eap_id": "eap-test"
        },
        "remote-cert": {
            "round": 1,
            "auth": "pubkey",
            "id": "moon.strongswan.org"
        }
    }
}
```

Secrets

```
{
    "data": "test",
    "id": "eap-test",
    "type": "EAP"
}
```

Server

```
{
    "server-eap": {
        "pools": [
            "server-pool"
        ],
        "local": {
            "auth": "pubkey",
            "id": "moon.strongswan.org",
            "certs": [
                "moonCert.pem"
            ]
        },
        "remote-eap": {
            "auth": "eap-md5"
        },
        "version": 2,
        "proposals": [
            "aes128-sha256-modp2048"
        ],
        "children": {
            "server-eap": {
                "esp_proposals": [
                    ''aes128gcm128-
                    modp2048''
                ]
            }
        }
    }
}
```

Secrets

```
{
    "data": "test",
    "id": "eap-test",
    "type": "EAP"
}
```

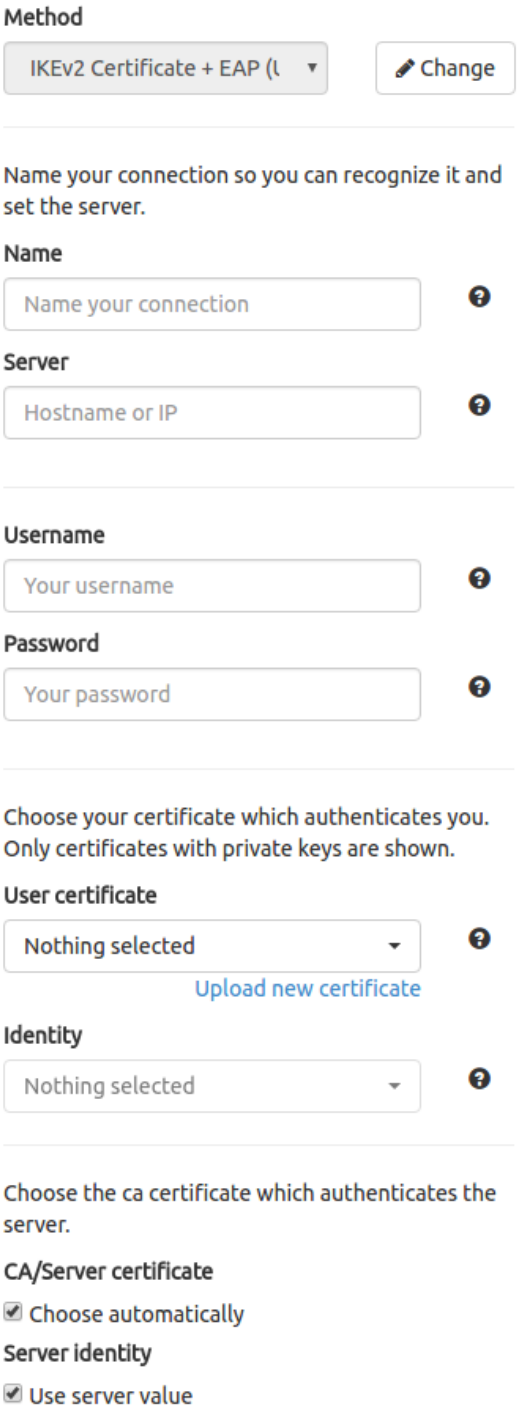
Pools

```
"server-pool": {
    "addrs": "10.6.0.0/24"
}
```

IKEv2 Certificate + EAP (Username/Password)

Zwei Runden Authentisierung, basierend auf der Kombination von EAP und Zertifikat.

Eingabefelder



Beschreibung

Name
Siehe Seite 28

Server
Siehe Seite 28

Username
Siehe Seite 30

Password
Siehe Seite 30

User certificate
Siehe Seite 28

Identitiy
Siehe Seite 28

CA/Server certificate
Siehe Seite 28

Server identity
Siehe Seite 28

Abbildung 2.5: IKEv2 Certificate + EAP

IKEv2 Certificate + EAP (Username/Password) ordered Dictionaries

Client

```
{
    "eap−cert": {
        "remote_addrs": [
            "gateway"
        ],
        "vips": [
            "0.0.0.0",
            "::"
        ],
        "version": 2,
        "proposals": [
            "aes128−sha256−modp2048"
        ],
        "children": {
            "eap−cert": {
                "remote_ts": [
                    "::/0",
                    "0.0.0.0/0"
                ],
                "esp_proposals": [
                    ''aes128gcm128−
                    modp2048''
                ]
            }
        },
        "local−cert": {
            "round": 1,
            "auth": "pubkey"
        },
        "local−eap": {
            "round": 2,
            "auth": "eap",
            "eap_id": "eap−test"
        },
        "remote−eap−cert": {
            "round": 1,
            "auth": "pubkey",
            "id": "moon.strongswan.org"
        }
    }
}
```

Secrets

```
{ "data": "test",
    "id": "eap−test",
    "type": "EAP" }
```

Server

```
{
    "server−cert−eap": {
        "pools": [
            "server−pool"
        ],
        "local": {
            "auth": "pubkey",
            "id": "moon.strongswan.org",
            "certs": [
                "moonCert.pem"
            ]
        },
        "remote": {
            "auth": "pubkey",
            "round": 1
        },
        "remote−eap": {
            "auth": "eap−md5",
            "eap_id": "eap−test",
            "round": 2
        },
        "version": 2,
        "proposals": [
            "aes128−sha256−modp2048"
        ],
        "children": {
            "server−cert−eap": {
                "esp_proposals": [
                    ''aes128gcm128−
                    modp2048''
                ]
            }
        }
    }
}
```

Secrets

```
{
    "data": "test",
    "id": "eap−test",
    "type": "EAP"
}
```

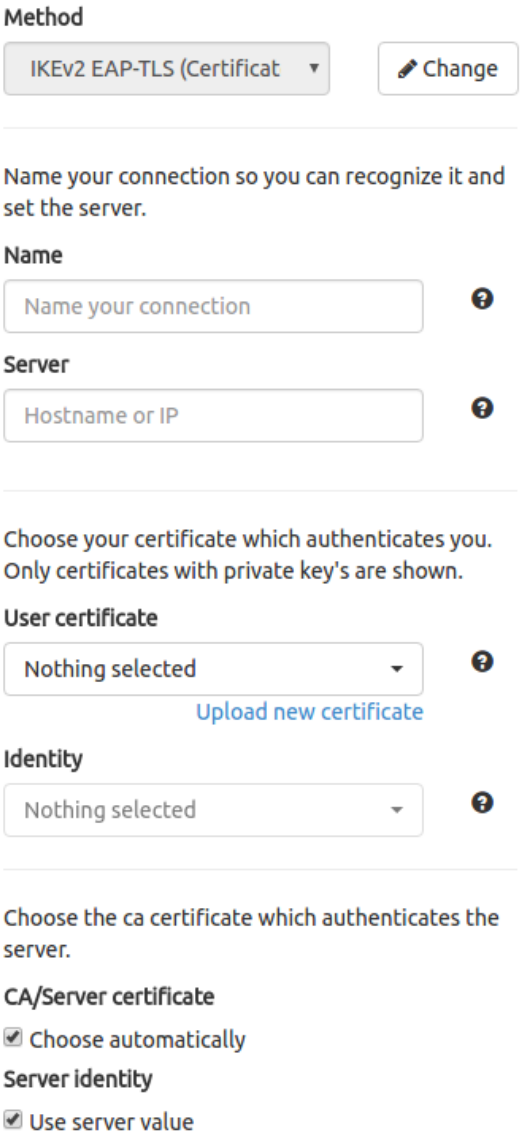
Pools

```
"server−pool": {
    "addrs": "10.6.0.0/24"
}
```

IKEv2 EAP-TLS

Mit der EAP-TLS Konfiguration wird ohne separate IKEv2 Authentifikation ein Verbindung aufgebaut, die das TLS Client- und Serverzertifikat verwendet.

Eingabefelder



Beschreibung

Name
Siehe Seite 28

Server
Siehe Seite 28

User certificate
Siehe Seite 28

Identitiy
Siehe Seite 28

CA/Server certificate
Siehe Seite 28

Server identity
Siehe Seite 28

Abbildung 2.6: IKEv2 EAP-TLS

IKEv2 EAP-TLS ordered Dictionaries

Client

```
{
    "eap−tls": {
        "remote_addrs": [
            "gateway"
        ],
        "vips": [
            "0.0.0.0",
            "::"
        ],
        "version": 2,
        "proposals": [
            "default"
        ],
        "children": {
            "eap−tls": {
                "remote_ts": [
                    "::/0",
                    "0.0.0.0/0"
                ],
                "esp_proposals": [
                    ''aes128gcm128
                    −modp2048''
                ]
            }
        },
        "local−eap−tls": {
            "round": 1,
            "auth": "eap−tls",
            "eap_id": "eap−test"
        },
        "remote−cert": {
            "round": 1,
            "auth": "pubkey",
            "id": "moon.strongswan.org"
        }
    }
}
```

Secrets

```
{
    "data": "test",
    "id": "eap−test",
    "type": "EAP"
}
```

Server

```
{
    "server−eap−tls": {
        "pools": [
            "server−pool"
        ],
        "local": {
            "auth": "pubkey",
            "id": "moon.strongswan.org",
            "certs": [
                "moonCert.pem"
            ]
        },
        "remote−eap": {
            "auth": "eap−dynamic"
            "eap_id": "eap−test"
        },
        "version": 2,
        "proposals": [
            "aes128−sha256−modp2048"
        ],
        "children": {
            "server−eap−tls": {
                "esp_proposals": [
                    ''aes128gcm128−
                    modp2048''
                ]
            }
        }
    }
}
```

Secrets

```
{
    "data": "test",
    "id": "eap−test",
    "type": "EAP"
}
```

Pools

```
"server−pool": {
    "addrs": "10.6.0.0/24"
}
```

2.3 Implementation

2.3.1 Einführung Django

Django [Fou16a] ist ein Python Framework zur Webentwicklung. Projekte werden zur Strukturierung in Apps eingeteilt, die hierarchisch aufeinander aufbauen. Apps sollten so gestaltet werden, dass man sie wiederverwenden kann.

MVC Konzept Django baut auf dem MVC Konzept auf, unterscheidet sich jedoch in der Namensgebung vom klassischen Aufbau.

klassisch	Django	Anmerkung
Model	Model	Models sind gleich wie allgemein üblich
View	Templates	Views sind in Django Templates, diese sind nicht mehr als Vorlagen
Controller	Views	Der Controller wird zur View und dient als Schnittstelle für den Client

Tabelle 2.3: MVC Konzept Django

Templates In Django ist ein Template im Grund eine einfach Textdatei, welche **variables** und **tags** enthält. Die **{{ variables }})** werden durch die entsprechenden Werte ersetzt und die **{% tags %}** beinhalten die Logik der Templates.

Beispiel Template

```
{% extends "base.html" %}
{% load vici_checker %}
{% block content %}
        {% vici_reachable as reachable %}
        {% if reachable %}
                Daemon: {{ daemon }} <br>
        <h2>Installed plugins</h2>
        {% for plugin in plugins %}
                {{ plugin }}
        {% endfor %}
        {% else %}
        <h3>Can not display information without the vici interface.</h3>
    {% endif %}
{% endblock %}
```

Views Die Views verarbeiten die Requests, sammeln Daten aus der Datenbank, bestücken die Templates und retournieren diese.

Beispiel views.py

```
from django.shortcuts import render
from django.contrib.auth.decorators import login_required

@login_required
def overview(request):
        connections = Connection.objects.all()
        return render(self.request, 'overview.html', {'connections': connections})
```

Models Django Models definieren die Daten. Sie beinhalten die entsprechenden Felder und Relation, regeln die Zugriffe auf die Datenbank mithilfe eines OR-Mappers(ORM) und beinhalten das Verhalten der Daten.

- Ein Model steht für eine Tabelle in der Datenbank
- Die Attribute der Models widerspiegeln die Spalten
- Und die Methoden repräsentieren das Verhalten

Beispiel models.py

```
from collections import OrderedDict
from django.db import models

class Secret(models.Model):
    type = models.TextField()
    data = fields.EncryptedCharField(max_length=50)
    authentication = models.ForeignKey(Authentication)

    def dict(self):
        eap_id = self.authentication.subclass().eap_id
        return OrderedDict(type=self.type, data=self.data, id=eap_id)
```

Routing Um Adressen auf Funktionen und schlussendlich Seiten zu mappen, verwendet Django URL-Konfigurationsdateien (urls.py). Diese enthalten reguläre Ausdrücke, welchen Funktionen aus Module zugeordnet werden.

Beispiel urls.py

```
from django.conf.urls import url

app_name = 'connections'
urlpatterns = [
    url(r'^$', views.overview, name='index'),
    url(r'^add/$', views.create, name='choose'),
    url(r'^(?P<id>\d+)/$', views.update, name='update')
]
```

2.3.2 Struktur strongMan

Die strongMan Projektstruktur ist im Anschluss aufgeführt, sie beinhaltet nur die strukturell wichtigsten Punkte.

```
strongMan
├── apps
│   ├── connections
│   ├── certificates
│   ├── vici
│   └── encryption
├── settings
├── fixtures
└── tests
```

apps

vici stellt den anderen Apps den Einstiegspunkt zur VICI-Schnittstelle zur Verfügung.

encryption ist für die Verschlüsselung der sensitiven Daten verantwortlich.

certificates verwaltet die Zertifikate, regelt somit das Erfassen, Löschen, Entschlüsseln und Persistieren. Weiter bestehen Abhängigkeiten zu der **vici** App, welche Zertifikate die von der strongSwan Applikation verwaltet werden auch in strongMan einbindet und zur **encryption**, die die privaten Schlüssel verschlüsselt in der Datenbank ablegt.

connections verwaltet die Konfiguration der Verbindungen, sowie das Aufbereiten dieser Daten in VICI/strongSwan konforme ordered Dictionaries. Es gibt also Abhängikeiten zu allen anderen Apps.

settings

Beinhaltet die Django spezifischen Einstellung, wie zum Beispiel die verwendeten Apps, Environement Variabeln, Modies etc. Besteht aus verschieden Settingsfiles, die je nach Verwendung in Einsatz kommen. Namentlich:

- base.py, dient als Basis für alle anderen
- local.py, für die Entwicklung
- deployment.py, gedacht für die produktiven Verwendung

fixtures

Wird genutzt um Initialdaten in Form einer JSON-Datei der Datenbank zur Verfügung zu stellen. In unserem Fall wird der vorkonfigurierte Benutzer und dessen Passwort gesetzt.

tests

Ordner der die Unit-Tests, sowie die Integrations-Tests beinhaltet. Diese können durch Strukturierung seperat ausgeführt werden.

2.3.3 Kapselung Versatile IKE Configuration Interface

Wie erwähnt bietet die strongSwan Open Source VPN Software das Versatile IKE Configuration Interface (VICI) [sP16] an, welche es erlaubt eine Management Anwendung über ein C, Ruby, Python oder Perl Binding an den Charon IKE Daemon anzubinden. Um die VICI-Schnittstelle in Python zu verwenden, kann das passende Package via PIP installiert werden.

```
1        pip install vici==5.4.1dev3
```

Kommunikation

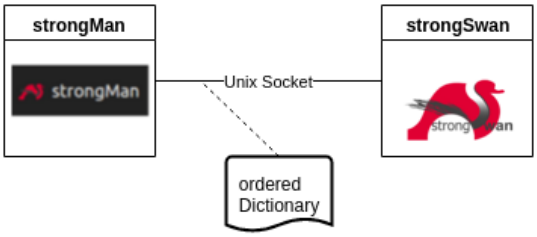


Abbildung 2.7: VICI Übersicht

Die Kommunikation, welche durch die Schnittstelle angeboten wird, basiert auf einem Unix Socket (AF_UNIX). Dabei wird in Python die OrderedDictionary-Klasse verwendet, die als verschachtelte Property List für die Konfigurationsparameter dient.

ordered Dictionaries sind folgendermassen strukturiert:

```
OrderedDict([
        ('cert', OrderedDict([
                ('remote_addrs', ['gateway'])
        )])
])
```

ViciWrapper

Um das VICI Plugin von unserem Code zu trennen, haben wir eine Wrapperklasse darum herum geschrieben. Dies verringert die Kopplung, ermöglicht uns eigene Exception zu werfen und gewisse Aufrufe zu kombinieren.
Die Hauptfunktionen des Wrappers sind:

- Auf- und Abbau der Sockets
- Übertragen der Verbindungskonfigurationen
- Initialisieren und Terminieren der Tunnels
- Auslesen von Statusinformationen
- Auslesen der strongSwan Konfiguration

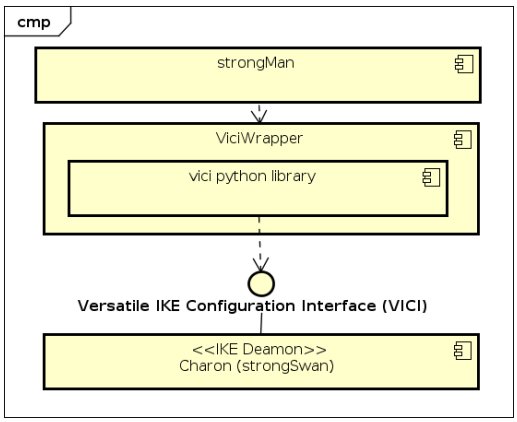


Abbildung 2.8: VICI Diagramm

2.3.4 Log Messages

Ausgangslage

Beim Klick auf den Toogle Button, der eine Verbindung zu starten versucht, löst der Webbrowser eine Ajax Request zum Server aus. Dieser blockierende Aufruf lädt die Verbindungskonfiguration in das VICI Interface, startet diese und wartet dann auf Log Messages. Dieser Vorgang, der Verbindungsaufbau des strongSwans, kann mehrere Minuten dauern.

Problematik

Der Webbrowser möchte möglichst in Echtzeit an diese Loginformationen herankommen. Eine mehrere Minuten dauernde Wartezeit sollte dem Benutzer nicht zugemutet werden.

Lösungsansatz

Die Lösung des Problems bietet die Aufsplittung in zwei verschiedene Ajax Requests.

Der erste Request startet die Verbindung und persistiert die empfangenen Log Messages in der Datenbank. Dieser Request beinhaltet den Verbindungsaufbau des strongSwans.

Der zweite Request dient der Aktualisierung der Oberfläche, respektive der Anzeige der Log Messages. Beim Aufruf der Connections Seite wird dieser HTTP Long Polling Request ausgelöst, der die neusten Log Messages zurück gibt. Dieser Request wird dauernd wiederholt, um die Logs auf dem neusten Stand zu halten. Log Messages älter als 5 Minuten werden aus der Datenbank gelöscht.

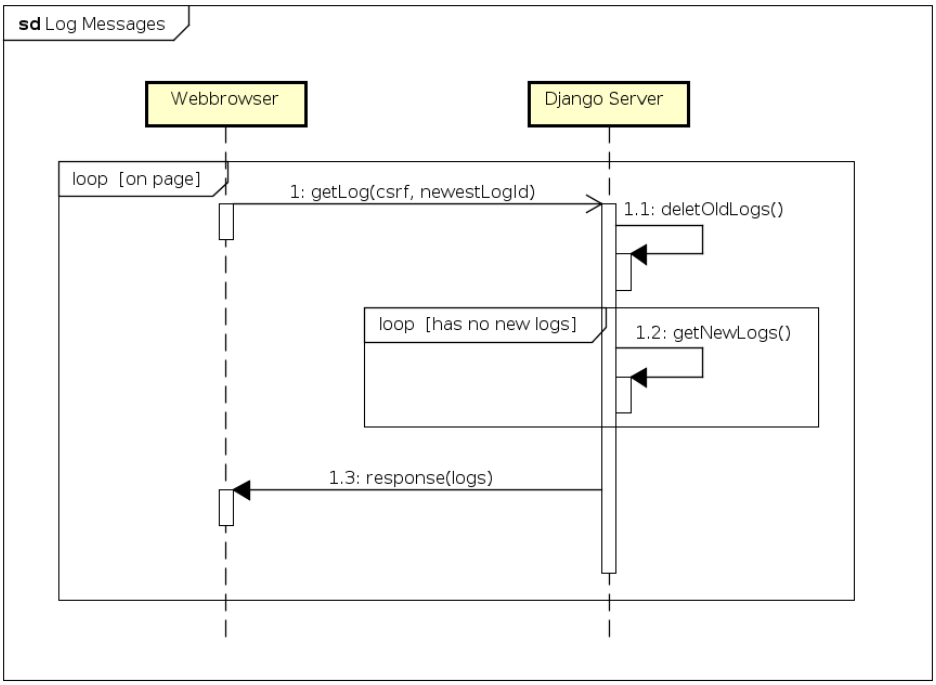


Abbildung 2.9: Log Messages Sequenzdiagramm

2.3.5 Statusinformationen

Ausgangslage

Die strongSwan Applikation bietet Statusinformation zu aktiven Verbindungen (Child SAs). Diese können über die VICI-Schnittstelle mithilfe der Funktion **list-sa** unter dem ordered Dictionary **child-sas** ausgelesen werden. Folgende Einträge sind dabei für uns relevant:

- remote-ts
- local-ts
- bytes-in
- bytes-out
- packets-in
- packets-out

Problematik

Es stellt sich die Frage, wie dem Webbrowser ständig die aktuellsten Statusinformationen zur Verfügung gestellt werden.

Lösungsansatz

Da die Statusinformationen nur informeller Natur sind, hat die Echtzeit dieser Daten keine hohe Priorität. Somit wurde eine eher triviale Lösung implementiert, die per Ajax Request alle 10 Sekunden die neusten Daten abfragt und darstellt.

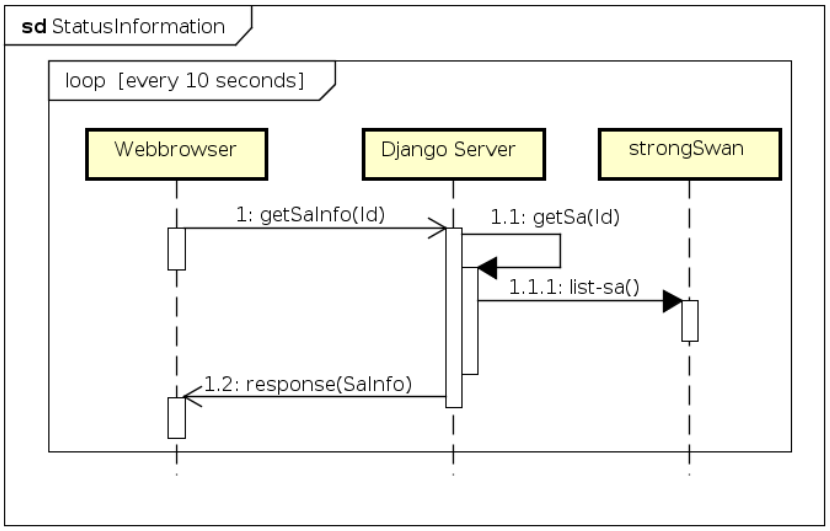


Abbildung 2.10: Statusinformationen Sequenzdiagramm

2.3.6 Subklassen

Ausgangslage

Um die diversen Authentiserungsmethoden sowohl in der Datenbank, wie auch in unseren Models widerzuspiegeln, setzen wir auf Vererbung. Wird in Django Vererbung bei Models eingesetzt, wird standardmässig **Multi-table inheritance** verwendet. Das heisst, es gibt für jede Subklasse eine weiter Tabelle in der Datenbank.

Beispiel Vererbung

Beispiel Code in Django um die Vererbung zu demonstrieren.

```
from django.db import models

class Authentication(models.Model):
    name = models.TextField()
    round = models.IntegerField(default=1)

class EapAuthentication(Authentication):
    eap_id = models.TextField()
```

Dies führt zu folgenden Tabellen in der Datenbank:

id	name	round
1	eap home	1
2	cert work	1

Tabelle 2.4: Authentication

authentication_ptr_id	eap_id
1	eap-home-id

Tabelle 2.5: EapAuthentication

Problematik

Ein Django Fremdschlüssel referenziert immer auf eine spezifische Klasse. Im Falle einer Klasse mit Subklassen referenziert der Fremdschlüssel somit immer auf die Basisklasse. Die Auflösung des Fremdschlüssels gibt immer die Basisklasse zurück, nicht direkt die Subklasse. Aus diesem Grund musste ein Workaround implementiert werden, der die Basisklasse wenn möglich in die Subklasse umwandelt.

Das gleiche Problem ergibt sich, wenn anhand der ID aus einem Formular auf das spezifische Objekt zugegriffen werden muss.

Dies könnte wie folgt aussehen. CaCertificateAuthentication ist eine Subklasse von Authentication.

```
def check_instance(auth):
    print("is instance of Authentication: " +
        str(isinstance(ca_auth, Authentication)))
    print("is instance of CaCertificateAuthentication: " +
        str(isinstance(ca_auth, CaCertificateAuthentication)))
```

Die Funktion **check_instance** macht entsprechende Instanzprüfungen.

Speichere Instanz von Subklasse

```
ca_auth = CaCertificateAuthentication()
ca_auth.save()
check_instance(ca_auth)
```

```
1 # Output
2 is instance of Authentication: True
3 is instance of CaCertificateAuthentication: True
```

Erstellt ein CaCertificateAuthentication Objekt und speichert es in der Datenbank ab. Wie an der Ausgabe ersichtlich ist, ist es vom Typ der Subklasse CaCertificateAuthentication.

Lade Objekt wieder

```
loaded_auth = Authentication.objects.get(id=ca_auth.id)
check_instance(loaded_auth)
```

```
1 # Output
2 is instance of Authentication: True
3 is instance of CaCertificateAuthentication: False
```

Das Objekt wird nun wieder aus der Datenbank gelesen. Die Ausgabe zeigt, dass es sich nun nur um die Basisklasse und nicht um die Subklasse handelt. Die Umwandlung in die Subklasse ist somit Aufgabe des Programmierers.
Das geladene Objekt ist von Typ Authentication. Falls hinter dieser ID jedoch noch eine Spezialisierung stecken würde, sieht man dies dem Objekt nicht an.

Lösungsansatz

Die Lösung für dieses Problem bietet Django selbst leider nicht an. Jedoch ist eine elegante Möglichkeit die Spezialisierung des Objektes über Reflection zu lösen. Dazu implementieren wir in den Basisklassen die Methode **subclass()**.
Diese ermöglicht es für jede Subklasse der Basisklasse zu testen, ob das aktuelle Objekt eine Spezialisierung ist oder nicht, falls ja, wird ein neue spezialisierte Instanz erstellt und zurück gegeben.

```
class Authentication(models.Model):
    @classmethod
    def get_types(cls):
        ```
        return: A list of all subclasses
        ```
        subclasses = [subclass() for subclass in cls.__subclasses__()]
        return [subclass.get_typ() for subclass in subclasses]

    def subclass(self):
        ```
        return: The specific subclass
        ```
        for cls in self.get_types():
            authentication_class = getattr(sys.modules[__name__], cls)
            authentication = authentication_class.objects.filter(id=self.id)
            if authentication:
                return authentication.first()
        return self
```

Mithilfe dieser Erweiterung sieht das verbesserte obige Beispiel nun folgendermassen aus.

```
def update(request, id):
    authentication = Authentication.objects.get(id=id).subclass()
    ...
```

Der Typ des **authentication** Objektes entspricht nun der gewünschten Subklasse.

2.3.7 Verschlüsselte Datenbankfelder

Während des Projekts ist die Anforderung an eine Verschlüsslung sensitiver Datenbankfelder aufgekommen. Die Datenbank verfügt über sensible Daten wie Private Keys und Passwörter, die nicht einfach offen gelesen werden dürfen.

Entscheid 3. Datenbankfelder werden mit einem Secret Key verschlüsselt

Nach einer Besprechung mit unserem Betreuer, haben wir uns auf die Verschlüsslung von einzelnen sensiblen Datenbankfelder mit einem Secret Key entschieden. Der Secret Key wird bei jeder Neuinstallation der Applikation zufällig generiert werden.
Das Projekt Django Fernet-Fields [ORC16] stellt ein angepasstes Datenbankfeld EncryptedField zur Verfügung, das als Django ORM Feld verwendet werden kann. Django kann Klartext in EncryptedField schreiben und wieder Klartext daraus lesen. Das EncryptedField übernimmt das ganze Verschlüsslungsmanagement. In der Datenbank sind nur kryptische Zeichen zu finden.

Anpassung des Fernet-Fields

Das Fernet-Field Projekt benötigt die Python Bibliothek Cryptography, welche wiederum C++ Code während der Installation kompilieren muss. Um dies zu vermeiden und um die Installation zu vereinfachen, haben wir die EncryptedField Klasse leicht angepasst, damit diese Pyaes nutzt und wir die Abhängigkeit zu Cryptography verlieren.

Beim ersten Start von strongMan generiert base.py einen neuen Secret Key[1] und schreibt diesen in die Datei db_key.txt. Das EncryptedField benutzt diesen Secret Key und verschlüsselt die Daten mit AES256.

[1] https://gist.github.com/ndarville/3452907

2.3.8 Zertifikate

Um IPsec Verbindungen mit asymmetrischen Schlüsseln zu erstellen, müssen Schlüssel verwaltet werden. strongMan persistiert geuploadeten Schlüsselcontainer in der Datenbank. Dabei werden folgende Schlüsselcontainer Formate unterstützt:

Schlüsselcontainer:	X.509, PKCS1, PKCS8, PKCS12
Encoding:	PEM, DER
Algorithmen:	RSA, ECDSA

Die Applikation hat ein Uploadfeld um Schlüsselcontainer hinzuzufügen. Dabei müssen die vier verschiedenen Containertypen unterschieden werden, um die Daten richtig auszulesen. Die Klasse ContainerDetector probiert die Datei mit allen vier ASN.1 Strukturen zu parsen und gibt bei Erfolg den entsprechenden Typ zurück.

Wichtige Implementationsmerkmale

Private Keys Wie im Domain Model zu erkennen ist, hängt der Private Key immer an einem User-Certificate. Das Zertifikat muss somit immer vor dem Private Key geuploadet werden. Dies verhindert Private Key Leichen in der Datenbank und vereinfacht das Schlüsselhandling.

Doppelte Einträge Der CNAME ist normalerweise ein einfacher Eintrag wie www.raiffeisen.ch. In Ausnahmefällen kann der CNAME, oder auch andere Felder, eine Liste wie ['raiffeisen.ch', 'www.raiffeisen.ch'] sein. Tritt dieser Fall ein, wird die Liste in einen String umgewandelt und so dargestellt.

Doppelte Zertifikate/Private Keys Werden bestehende Zertifikate oder Private Keys nochmals geuploadet, erkennt die Zertifikatsverwaltung dies und verweigert den Upload. Das Matching wird anhand des Public Keys vorgenommen. Bei Zertifikaten wird der Public Key und die Version gematcht.

Zertifikate in Gebrauch Ist ein Zertifikat in einer Verbindungskonfiguration ausgewählt, so kann das Zertifikat nicht gelöscht werden. Das Zertifikat muss zuerst deselektiert werden, um es aus der Zertifikatsverwaltung zu löschen.

2.3.9 Webserver

Django ist ein Webframework und muss somit mit einem Webserver betrieben werden. Im Entwicklungsmodus kann der Django interne Webserver verwendet werden, dieser wird folgendermassen gestartet:

```
1    python manage.py runserver −−settings=strongMan.settings.local
```

In der Produktionsumgebung rät das Django Team jedoch von der Verwendung des internen Webservers ab[2]. Der interne Server sei nicht auf Vulnerabilities getestet und ist somit unsicher.

Produktionsumgebung

Jedes Djangoprojekt enhält standardmässig eine WSGI [fou16b] Schnittstelle mit der man verschiedene Webserver anbinden kann. Die Schnittstelle in strongMan befindet sich unter strongMan/wsgi.py.

Entscheid 4. Gunicorn - Statische Dateien

In diesem Projekt verwenden wir gunicorn [Che16]. Gunicorn oder auch 'Green Unicorn' ist ein Python WSGI HTTP Server für Unix. Er verarbeitet die Webrequest die Django betreffen. Normalerweise verarbeitet Gunicorn aus Performancegründen keine statischen Daten wie Javascript, CSS, Bilder. Da es sich hier aber um eine Einzeluserapplikation handelt, haben wir uns trotzdem dazu entschieden, auch die statische Daten über Gunicorn auszuliefern. Um dies zu ermöglichen wird DJ-Static [Rei16] eingesetzt. Ansonsten müsste ein richtiger Webserver wie Apache oder Nginx installiert und konfiguriert werden.

Gunicorn wird einfach via PIP installiert und kann danach im Terminal verwendet werden.

Die Datei run.py im Hauptverzeichnis des Projekts startet den Gunicorn Server entsprechend in der Produktionsumgebung:

```
1    gunicorn −−workers 6 −−bind 0.0.0.0:1515 −−env
2    DJANGO_SETTINGS_MODULE=strongMan.settings.deployment
3    strongMan.wsgi:application
```

- –workers 6
  - Startet 6 Threads zur Behandlung von Webrequest.
- –bind 0.0.0.0:1515
  - Beschreibt denn Socket, auf den gehört werden soll.
- –env DJANGO_SETTINGS_MODULE
  - Zeigt auf die entsprechende Django Settings Datei, die für die Produktionsumgebung genutzt wird.
- strongMan.wsgi:application
  - Die WSGI Schnittstelle

[2] https://docs.djangoproject.com/en/1.9/ref/django-admin/#runserver

Betreibung von strongMan hinter einem Nginx Server

strongMan kann auch direkt hinter einem Nginx Webserver betrieben werden. Besonders wenn in strongMan der Serverteil implementiert ist und mehrere User auf die Applikation zugreifen, kann es aus Performanzgründen Sinn machen, den Nginx Server vor Gunicorn heranzustellen. Der Nginx Server liefert in diesem Fall die statischen Daten wie Javascript, CSS, Bild Dateien aus und Gunicorn kümmert sich um die dynamischen Daten in der Django Applikation.

Eine Anleitung zur Konfiguration von Nginx, Gunicorn und Django findet man unter [Ell16]

2.4 Testing

In der Python Standard Library gibt es das Unit Testing Framework **unittest**, das erlaubt Unit Tests zu implementieren. Django's Unit Tests basieren auf dieser Library und erweitern diese, beispielsweise erbt **django.test.TestCase** von **unittest.TestCase**. Weiter werden auch Integrationtests ermöglicht.

Beispiel Integrationtests

```
from django.test import Client, TestCase
from django.contrib.auth.models import User

class AboutViewsTests(TestCase):

    def setUp(self):
        self.user = User.objects.create(username='testuser')
        self.user.set_password('12345')
        self.user.save()
        self.client = Client()
        self.client.login(username='testuser', password='12345')

    def test_about_get(self):
        response = self.client.get('/about/')
        self.assertEquals(response.status_code, 200)
```

2.4.1 Continuous Integration (CI)

Der Aufbau der Continuous Integration Umgebung des strongMans ist nicht trivial und stellt folgede Anforderungen:

- Aufbau von IPsec Tunnels zwischen mindesten zwei Rechnern
- automatisierter Ablauf
- geringe Einarbeitungszeit in Technologien
- möglichst kostenfreie Dienste nutzen

Um den Anforderungen gerecht zu werden setzten wir auf ein Zusammenspiel folgender Tools:

GitHub wird von uns als online Repository verwendet und wird mit dem Versionsverwaltungssystem Git eingesetzt. Weiter bietet es für andere Dienste WebHooks an.

Travis CI wird als eigentlicher Continuous Integration Anbieter verwendet. Es ermöglicht automatisierte Builds und Tests. Travis CI ist für Projekte, welche auf GitHub gehostet werden, ausgelegt.

Docker Um die strongSwan Konfiguration direkt zu testen, wird docker-compose eingesetzt. Docker-compose gewährleistet mehrere Docker Container zu builden, ein Netzwerk untereinander und somit einen IPsec Tunnel aufzubauen.

Ablauf

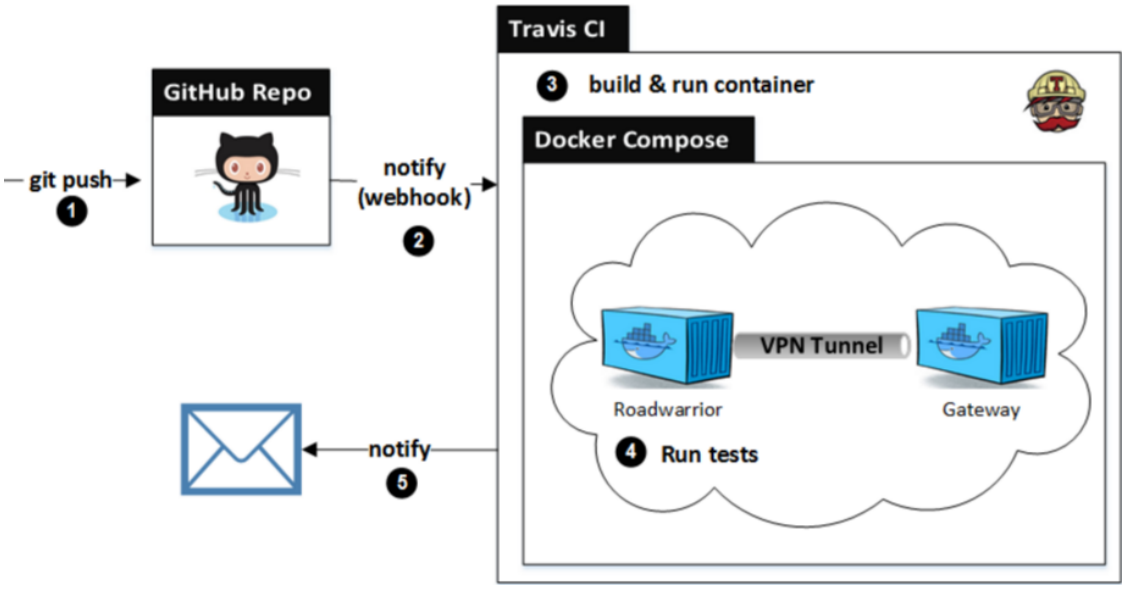


Abbildung 2.11: Schematische Darstellung der Testumgebung

1. git push, lokaler Code wird auf GitHub publiziert.
2. Travis CI wird durch WebHook notifiziert.
3. Travis CI buildet zwei Docker Container, welche auf der Basis des offiziellen Django Container aufbaut.
   (a) StrongSwan mit den nötigen Plugins wird installiert und gestartet.
   (b) Auf dem Roadwarrior wird die strongMan Applikation vom GitHub Repository installiert und der aktuelle Branch wird eingecheckt.
4. Der Roadwarrior started die Unit- und die Integrationstests. Dabei werden zwischen den Docker Container diverse IPsec Tunnels auf- und abgebauen.
5. Über Erfolg und Misserfolg wird notifiziert.

2.4.2 Usability Tests

Die Usability Tests sollen Ablauffehler aufzeigen und die Benutzbarkeit der Applikation insgesamt steigern. Dazu sind in einer ersten Phase zwei Personas erstellt und anschliessend mit entsprechenden Personen Tests durchgeführt worden.

Entscheid 5. Usability Tests vs. Server Features

Am Ende der zweit letzten Construction Phase, nach einer Besprechung mit Herrn Steffen, entschieden wir uns Usability Tests durchzuführen und den Server Teil der Applikation in diesem Projekt nicht zu implementieren. Damit werden die Use Cases UC03: Child_SA beenden und UC06: Admin Mode wechseln hinfällig.

Primäre Persona User

Name:	Markus Egger
Alter:	40 Jahre
Beruf:	Verkäufer Aussendienst
Wohnort:	Uster ZH
Zivilstand:	Verheiratet, 2 Kinder
Hobbies:	Gleitschirm, Bowling
Sprachen:	Deutsch / Englisch

Abbildung 2.12: Primäre Persona User

Markus Egger arbeitet in einem mittelgrossen Industrieunternehmen und verkauft dort Isolationsmatten aus Steinwolle. Da er viel auf Reisen bei seinen Kunden ist und dadurch längere Zeit keinen physischen Zugang zum Firmennetzwerk hat, benutzt er eine VPN Software um auf firmeninterne Daten zuzugreifen. Seine Informatikfähigkeiten beziehen sich dabei auf die in seiner KV Lehre erworbenen Kenntnisse mit Email, Word und Excel, sowie dem oberflächlichen Umgang mit ERP/CRM Systemen. Markus beherrscht die englische Sprache auf Verhandlungsniveau.

Zusätzlich zu seinen Reisen arbeitet Markus gerne auch mal Zuhause, um mehr Zeit bei seinen erst kürzlich eingeschulten Kindern zu verbringen und seine Frau zu entlasten.

Test Vorlage

Sie sind Angestellter in der Personalabteilung der Industriefirma Techtronic AG und arbeiten heute zum ersten Mal Zuhause an ihrem Arbeitslaptop. Um Zugriff auf firmeninterne Dokumente zur erhalten, müssen Sie eine sichere Verbindung zur Tectronic AG aufbauen. Dazu hat Ihnen die Informatikabteilung ein USB-Stick mit Anleitungen, Username, Passwort und Zertifikatsdateien zur Verfügung gestellt. Ihre Aufgabe ist es nun, die unten aufgeführten Tasks zu lösen.

Passwort ändern Loggen Sie sich ein und ändern Sie ihr Passwort.

Preconditions:	Applikation ist gestartet. User befindet sich auf der Login Seite.
Hilfsmittel:	Username und aktuelles Passwort
Artefakte:	User konnte sich einloggen. Passwort ändern Feld gefunden. Passwort konnte geändert werden.

Verbindung erstellen Erstellen Sie eine neue Verbindung zur HSR mit den gegebenen Daten.

Preconditions:	User befindet sich eingeloggt auf der Startseite von strongMan. Connections und Certificates sind leer. Der System Administrator hat eine Anleitung zum Erstellen der Verbindung zur Verfügung gestellt.
Hilfsmittel:	Schritt für Schritt Anleitung mit Server, Username, Passwort, CA-Zertifikat, Identity
Artefakte:	User hat den Connections Bereich gefunden. User hat den Add Button gefunden. Der richtige Authentifizierungstyp ist ausgewählt. Verbindung richtig ausgefüllt und erstellt.

Verbindung starten / **stoppen** Starten Sie die Verbindung und stoppen Sie diese wieder.

Preconditions:	User befindet sich auf der Connections Seite. Die in Aufgabe 2 erstellte Connection ist abgespeichert.
Hilfsmittel:	-
Artefakte:	User erkennt den On/Off Button. User kann die Verbindung starten. User ist sich bewusst, dass die Verbindung jetzt steht (Fragen!). Verbindung ist wieder gestoppt.

Sekundäre Persona Administrator

Name:	Remo Stoop
Alter:	28 Jahre
Beruf:	System Administrator
Wohnort:	Greifensee ZH
Zivilstand:	Konkubinat, 1 Kind
Hobbies:	Raspberry Pi, Gamen
Sprachen:	Deutsch / Englisch

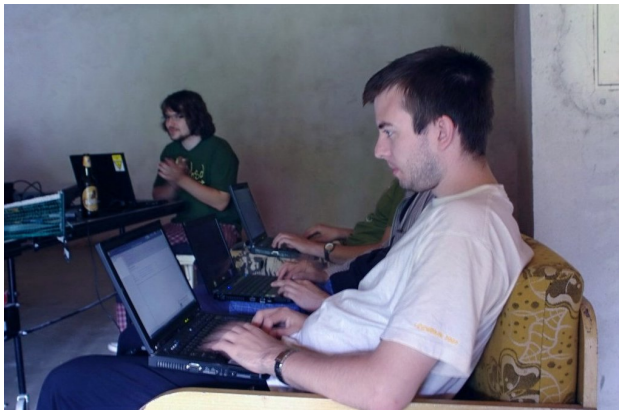


Abbildung 2.13: Sekundäre Persona Administrator

Remo Stoop arbeitet wie Markus Egger im gleichen mittelgrossen Industrieunternehmen, welche Isolationsmatten herstellt. Er ist dort als System Administrator angestellt und verantwortlich für die Netzwerkinfrastruktur und der Netzwerksicherheit, inklusive dem Netzwerkzugriff von aussen. Remo hat sich bewusst in Richtig Netzwerk weitergebildet. Nach seiner Lehre als Informatiker Systemtechnik hat er zwei fünf tägige Cisco Netzwerk Kurse besucht und ist seit dem netzwerktechnisch auf dem neusten Stand. Die symmetrischen und asymmetrischen Verschlüsslungstechniken kennt er oberflächlich aus einem Modul aus seiner Lehre.

Auch Remo arbeitet gerne einen Tag in der Woche zuhause, sofern es seine Arbeit erlaubt. Neben seiner Familie ist seine Hauptbeschäftigung das Gamen mit Freunden sowie Raspberry Pi Bastelprojekte.

Test Vorlage (erweitert User Tests)

Sie sind Systemadministrator der Industriefirma Techtronic AG und haben zuvor einen strongSwan VPN Server eingerichtet. Nun soll das erste Mal eine Verbindung von ausserhalb (genauer von Ihnen zuhause) in die Firma aufgebaut werden. Dazu stehen Ihnen Username, Passwort und Zertifikatsdateien zur Verfügung. Ihre Aufgabe ist es nun, die unten aufgeführten Tasks zu lösen.

Key Container hinzufügen

1. Fügen Sie in der Zertifikatsverwaltung einen verschlüsselten PKCS12 Container hinzu.
2. Fügen Sie in der Zertifikatsverwaltung ein neues Zertifikat mit Private Key hinzu.
3. Welche Email Adresse ist beim Zertifkat roadwarrior.hsr.ch eingetragen?
4. Zertifkat roadwarrior.hsr.ch löschen.

Preconditions:	User befindet sich auf der Startseite von strongMan.
Hilfsmittel:	Roadwarrior PKCS12 Container mit Passwort, Zertifikat mit Private Key
Artefakte:	User hat den Add Button gefunden. User konnte den verschlüsselten PKCS12 Container hinzufügen. User konnte das Zertifikat hinzufügen. User konnte den Private Key hinzufügen. User findet Email-Adresse. User konnte Zertifikat löschen.

Fazit

Wir haben die Usability Tests mit vier Personen durchgeführt. Die Administrator Tests haben zwei unbeteiligte HSR Studenten durchgeführt und die User Tests zwei Personen aus unserem Bekanntenkreis mit Büroerfahrung.

Die wichtigsten Erkenntnisse aus den Tests sind:

- Es braucht einen Text, der beschreibt, dass zuerst das Zertifikat und erst danach der Private Key geuploaded werden kann.
- Die State Spalte in der Connection Table ist nicht sortierbar.
- Der Verbindungstyp muss änderbar sein nach der Auswahl.
- Zertifikate müssen in der Verbindungskonfiguration hinzugefügt werden können.
- Beschriftung des Remove Buttons muss eindeutig sein. Private Key löscht keine Zertifikate.
- Verbindung darf nicht löschbar sein, wenn sie aktiv ist.
- Diverese Verbesserungen an der Englischen Sprache sind nötig.
- Die Anordnung der Buttons im Add-Certificate Formular verwirrt.
- About Pages zeigt keine Infos.

2.5 Abhängigkeiten

2.5.1 Python

Ein Ziel dieses Projektes war es, möglichst keine kompilierende Abhängigkeiten zu verwenden. Die Bibliotheken, die mit PIP installiert werden, dürfen also keinen komplierenden Code wie C++ enthalten. In anderen Worten sollte das Projekt Pure-Python sein.

django [Fou16a] Bibliothek zur Verwendung des Django Webframeworks.

oscrypto [wbo16] Zertifikatsparser.

asn1crypto Zertifikatsparser, Abhängigkeit von oscrypto.

pyaes [Moo16] Python AES Implementation. Gebraucht für die Datenbankverschlüsslung.

django-tables2 [Aye16] Plugin zur Anzeige von HTML Tabellen. Unterstützt Pagination, Sortierung.

vici [sP16] VICI Plugin für strongSwan.

gunicorn [Che16] Webserver für die Produktionsumgebung.

djstatic [Rei16] Plugin zur Auslieferung von den statischen Dateien (css, js) in der Produktionsumgebung.

2.5.2 Javascript/CSS

Das Userinterface verwendet einige Javascript und CSS Bibliotheken. Alle Bibliotheken wurden direkt in das Projekt importiert und werden nicht aus dem Internet nachgeladen. Dies dient der Verwendung der Applikation in einer Umgebung mit keiner/beschränkter Internetverbindung.

bootstrap [Twi16] CSS Design Bibliothek.

jquery [jFoc16] Helfer zur HTML Dom Manipulation.

bootstrap-fileinput [Vis16] HTML5 Datei Upload Element.

bootstrap-select [Mor16] Auswahlfeld, gebraucht um Zertifikate in der Connection Config auszuwählen.

bootstrap-toggle [Hur16] Toggle Button, der die Connection startet/stoppt.

callout [Pra16] CSS Callout Bibliothek.

font-awesome [Gan16] Icon Bibliothek.

2.6 Codestatistik

Test Coverage

Test Coverage wurde mithilfe von **Pycharm** gemessen.

App	Coverage
connections	88%
certificates	88%
vici	88%
encryption	94%
Durchschnitt	90%

Tabelle 2.6: Test Coverage

Codezeilen

Die Codezeilen wurden mithilfe von **CLOC** [AlD16] gezählt. Die Anzahl Zeilen beinhalten nur selbst geschriebenen Code. Externe Bibliotheken wurden ausgeschlossen.

Sprache	Dateien	Zeilen
Python	70	5508
HTML	41	1857
JavaScript	4	397

Tabelle 2.7: Codezeilen

2.7 Resultate

Abbildung 2.14: Connection Overview

2.7.1 Management-Oberfläche für VPN Clients

Die Management-Oberfläche der strongMan Applikation gibt eine Übersicht über alle erfassten Verbindungen. Diese können per Klick auf den Namen geändert oder gelöscht werden.
Zum Starten oder Stoppen der konfigurierten VPN Verbindungen ist der Toggle Button gedacht.
Wurde ein VPN-Tunnel erfolgreich aufgebaut, werden Statusinformationen angezeigt.

2.7.2 Zertifikatsverwaltung

Unter dem Reiter Certificates finde sich ein Zusammenstellung aller selbst erfassten Zertifikate und derer die von strongSwan verwaltet werden. Diese werden über die VICI-Schnittstelle geladen und auch in der Datenbank persisitiert. Es ist möglich nach Zertifikaten zu suche, diese zu sortieren und durch das Auswählen weitere Information anzuzeigen.

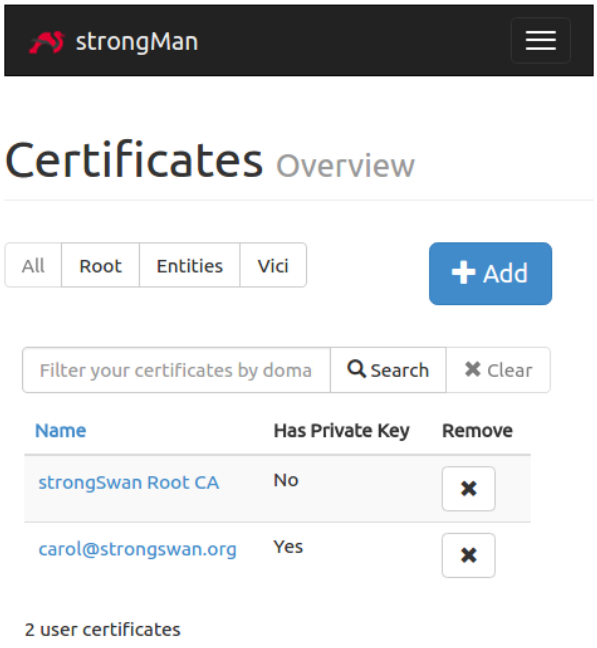


Abbildung 2.15: Certificate Overview

2.7.3 Ausblick

strongMan implementiert nach Abschluss dieses Projekts Clientfunktionalitäten wie oben beschrieben. Ausbaumöglichkeiten gibt es in der nicht realisierten Kann Anforderung der Serverfunktionalität. Mit dieser Erweiterung soll es möglich sein, einen VPN Server zu konfigurieren und auch zu verwalten. Gerade die Verwaltung eines Servers zieht aufwändigere Funktionalität bezüglich Security Associations Management, Visualiserung der Verbindungen, sowie Statistikinformationen mit sich.

Wenn gewünscht kann das Benutzermanagement von derzeit einem Standardbenutzer auf eine belibige Anzahl ausgebaut werden. Diese Änderung ermöglicht die getrennte Verwaltung von Verbindungen in einem Multiusersystem.

Weiter ist derzeit strongMan auf wenigen Unix Distributionen lauffähig. Es können weitere Distributionen getestet und eine Portierung auf Windows erwogen werden.

2.7.4 Auswertung

Ziel	Bemerkung
Implementation einer grafischen Management-Oberfläche mit Django für strongSwan zur Konfiguration eines VPN Clients für folgende vier IKEv2 Authentisierungsmethoden: 1. X.509 Zertifikat und privater RSA/ECDSA Schlüssel 2. EAP mit Benutzername/Passwort 3. Zweirunden-Authentisierung mit Methode 1) gefolgt von Methode 2) 4. EAP-TLS mit X.509 Zertifikat und privatem RSA/ECDSA Schlüssel	Die definierten Authentisierungsmethoden werden unterstützt und durch die Architektur der strongMan Applikation sind Erweiterungen möglich.
Oberfläche zur Verwaltung von X.509 End Entity und CA Zertifikaten, sowie privater RSA/ECDSA Schlüssel.	strongMan ermöglicht das Hinzufügen und Entfernen von Zertifikaten, weiter ist das Filtern und Suchen möglich. Als zusätzliches Feature kann einem Zertifikat ein Nickname gesetzt werden, welcher bei der Identifikation hilft.
Persistierung der Konfigurationsdaten in einer Datenbank.	Alle Konfigurationsdaten, sowie Zertifikate werden in der Datenbank gespeichert. Dabei wird standardmässig eine SQLite Datenbank verwendet, welche es ermöglicht ohne zusätzliche Konfiguration und Installation die Applikation zu verwenden.
Verschlüsselte Ablage der RSA/ECDSA Authentisierungsschlüssel in der Datenbank.	Die schützenswerten Daten wie Private Keys und strongSwan Secrets (Bsp. EAP Passwort) werden durch eine AES256 Verschlüsselung in der Datenbank persistiert.
Starten und Stoppen von konfigurierten VPN Verbindungen	Korrekt erfasste Verbindungen können über die VICI-Schnittstelle gestartet und gestopppt werden.
Darstellung von Statusinformation über aktive VPN Verbindungen	Informationen zu Local- und Remoteselectors, Traffic sowie Logs werden dargestellt.
Optional: Oberfläche zur Konfiguration eines VPN Gateways	Die Erweiterung zur Gateway/Server Konfiguration wurde nicht implementiert. Gegen Ende des Projektes wurde entschieden, dass der Fokus auf die Clientseite gesetzt wird und dieser noch verbessert werden soll.

Tabelle 2.8: Auswertung

Kapitel 3

Projektmanagement

3.1 Rollen und Verantwortlichkeiten

Prof. Steffen Andreas

Professor, Institutsleiter ITA

Abbildung 3.1: Prof. Dr. Andreas Steffen

Tobias Brunner

Institutsmitarbeiter ITA

Bühler Severin

Severin Bühler, Student an der HSR, ist Entwickler des Projekts.

Abbildung 3.2: Bühler Severin

Kurath Samuel

Samuel Kurath, Student an der HSR, ist Entwickler des Projekts.

Abbildung 3.3: Kurath Samuel

3.2 Entwicklungsumgebung und Infrastruktur

IDE (Integrated Development Environment)

Entscheid 6. PyCharm

Beiden Projektmitgliedern ist JetBrains Intellij bekannt und PyCharm ist im Umgang nahe zu identisch. Für Studenten sind die Entwicklungsumgebungen kostenlos verfügbar.

SCM (Source Control Management)

Entscheid 7. GitHub

Der Umgang mit Git ist beiden Projektmitglieder bestens bekannt. GitHub ist ohne Unkosten von überall verfügbar.

CI (Continuous Integration)

Entscheid 8. Travis CI

Mit Travis CI wurde ein Anbieter gefunden, der die komplexeren Anforderungen an das automatisierte Testing des strongMan Projektes erfüllt, sich nathlos in Github integrieren lässt, kostenfrei ist, sowie den Projektmitgliedern schon bekannt ist.

Projektmanagement Tool

Entscheid 9. Jira

Jira ist den Projektmitgliedern schon aus der Studienarbeit bekannt und hat sich sehr bewährt. Das Dashboard ist übersichtlich gestaltet. Es ermöglicht eine Übersicht über die aktuellen Tasks auf einen Blick. Alle Mitglieder haben jederzeit Zugriff auf die Plattform, was die Transparenz erhöht. Weiter bietet Jira diverse Reports um Auswertungen über das Projekt zu fahren.

3.3 Planung

Das Projekt strongMan wird in einer Mischung aus RUP und Agile durchgeführt. Der Projektzeitraum wird zuerst in die RUP Phasen Inception, Elaboration, Construction und Transition aufgeteilt, wobei in der Construction agile vorgegangen wird.

Phasen

1. Inception
    (a) Einarbeitung in das Projekt
2. Elaboration
    (a) Evaluation und Einarbeitung der Techniken (Django, Crypto-Library, VICI-Schnittstelle)
    (b) Erstellen der Requirement-Analyse
    (c) Erstellen eines lauffähigen Prototyps
3. Construction1
    (a) VPN-Tunnel CRUD
    (b) VPN-Tunnel starten/stoppen
    (c) Login
    (d) Zertifikate CRUD
4. Construction2
    (a) Bestehende Verbindung terminieren
    (b) User Passwort ändern
    (c) Verbindungselemente zwischen VPN-Tunnel und Zertifikate
    (d) Verschlüsslung Private Keys und User Password für Tunnels
5. Construction3
    (a) Erstellen einer Informationsseite
    (b) Optional: Administratorsmodus um Serverkonfigurationen vor zu nehmen
6. Construction4
    (a) Applikation finalisieren
    (b) Deployment
7. Transition
    (a) Dokumentationsabschluss

Entscheid 10. Construction3

Durch den Entscheid Usability Tests vs. Server auf Seite 51 wurden die Serverfunktionalitäten (Adminstratormodus) in der Construction3 durch Usablitiy Tests ersetzt.

Meilensteine

1. MS1 Techniken evaluiert (Django, Crypto-Library, VICI-Schnittstelle)
2. MS2 Lauffähiger Prototyp
3. MS3 VPN-Tunnel erfass- und änderbar
4. MS4 VPN-Tunnel initialisier- und terminerbar
5. MS5 Implementation abgeschlossen

Zeitplanung

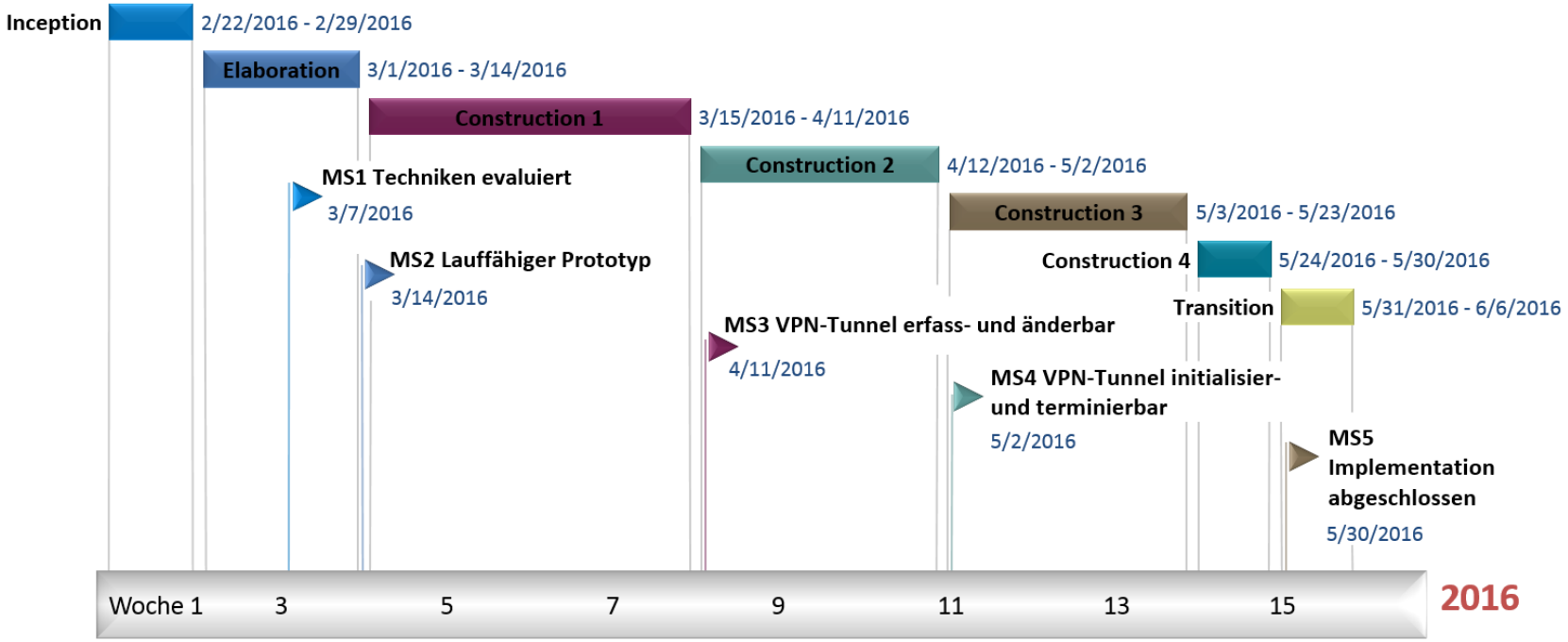


Abbildung 3.4: Gantt Chart

3.4 Risiken

Um den Problemen, die während des Projekts auftreten können, entgegenzuwirken, haben wir zu Beginn des Projekts eine Risiko Analyse durchgeführt. Diese konnte dann bei der Planung eingesetzt werden.

Technische Risiken

Nr	Titel	Beschreibung	maximaler Schaden	Eintrittswahrscheinlichkeit	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten
R1	Python Crypto Library	Die evaluierte Crypto Library für die Zertifikaterkennung ist fehlerhaft oder ungenügend.	50h	20%	10h	Evaluation Library	Zertifikate werden nicht mehr ausgelesen, sondern nur noch simpel importiert.
R2	Komplexität VICI-Schnittstelle	Die Python VICI-Schnittstelle deckt sich nicht mit der swanControl Notation. Dadurch kann massiv mehr Aufwand während der Implemenation entstehen.	50h	40%	20h	Die VICI-Schnittstelle muss mit dem Prototypen gut durchgetestet werden, um Fehler möglichst früh zu finden.	Kontakt mit Tobias Brunner aufnehmen
R3	Einrichten automatisierte Testumgebung	Aufbau und Einrichten einer Integrationtestumgebung, welche es erlaubt IPsec Tunnels zwischen mehreren Rechnern aufzubauen.	60h	60%	36h	Informationen zu CI Anbieter sammeln	Eigene Infrastruktur verwenden.

Tabelle 3.1: Risiken - Die technischen Risiken wurden zu Beginn des Projektes, wie in der Tabelle ersichtlich, definiert.

Auswertung

R1 Python Crypto Library Mithilfe der Evaluation der Crypto Library konnte dieses Risiko aus dem Weg geräumt werden. Der Aufwand blieb im erwarteten Rahmen und mit **oscrypto** wurde eine passende pure Python Bibliothek gefunden werden.

R2 Komplexität VICI-Schnittstelle Das Zusammenspiel zwischen Python/Django und strongSwan funktionierte nahtlos und wie in der Dokumentation beschrieben. Problematischer und zeitintensiver war jedoch das korrekte Konfigurieren von strongSwan. Erst dank dem Einsatz von docker-compose und der Hilfe durch Tobias Brunner konnte dieses Problem aus der Welt geschafft werden.

R3 Einrichten automatisierte Testumgebung Docker-compose legte den Grundstein, welcher uns ermöglichte mehrere Rechner für die Testumgebung zu nutzen. Da der uns wohl bekannte CI Anbieter Travis CI dies zudem noch unterstützte, wurden automatische Integrationtest möglich. Das Finden einer passenden Lösung war ohne grössere Komplikationen möglich. Mehr Aufwand als erwartet, gab jedoch die Konfiguration der Docker-Container. Womit dieses Risiko teils eintrat.

Nr	Titel	Schaden
R1	Python Crypto Library	0 Stunden
R2	Komplexität VICI-Schnittstelle	18 Stunden
R3	Einrichten automatisierte Testumgebung	32 Stunden
Total		50 Stunden

Tabelle 3.2: Risikoauswertung

3.5 Arbeitsaufwand

Soll

Phase	Aufwand
Inception	1 Woche
Elaboration	2 Wochen
Construction1	3 Wochen
Construction2	3 Wochen
Construction3	3 Wochen
Construction4	3 Wochen
Transition	1 Woche
Total	14 Wochen

Tabelle 3.3: Zeitplanung

Um den Aufwand über die ganze Projektdauer ausgeglichen zu verteilen rechneten wir mit:

- Aufwand pro Woche: 52 Stunden
- Aufwand geplant Total: 52 Stunden * 14 Wochen = **728 Stunden**
- Aufwand pro Student: 728 Stunden / 2 = **364 Stunden**

Das Frühlingssemester 2016 an der HSR hat offiziell 15 Wochen (inklusive Ostern, Auffahrt und Pfingsten). Bis zur Abgabe der Bachelorarbeit, dem 17.06.2016, sind zusätzlich zwei weitere Wochen gegeben. Damit am Ende des Projektes der Aufwand nicht exponentiell steigt, planten wir unsere Stunden über das offizielle Frühlingssemester und hielten uns die restliche Zeit als Reserve frei.

Ist

Während des ganzen Projektes haben wir in Jira Phasen geplant, dazu Tasks erstellt und entsprechen Zeit verbucht, was im folgendem Aufwand resultiert.

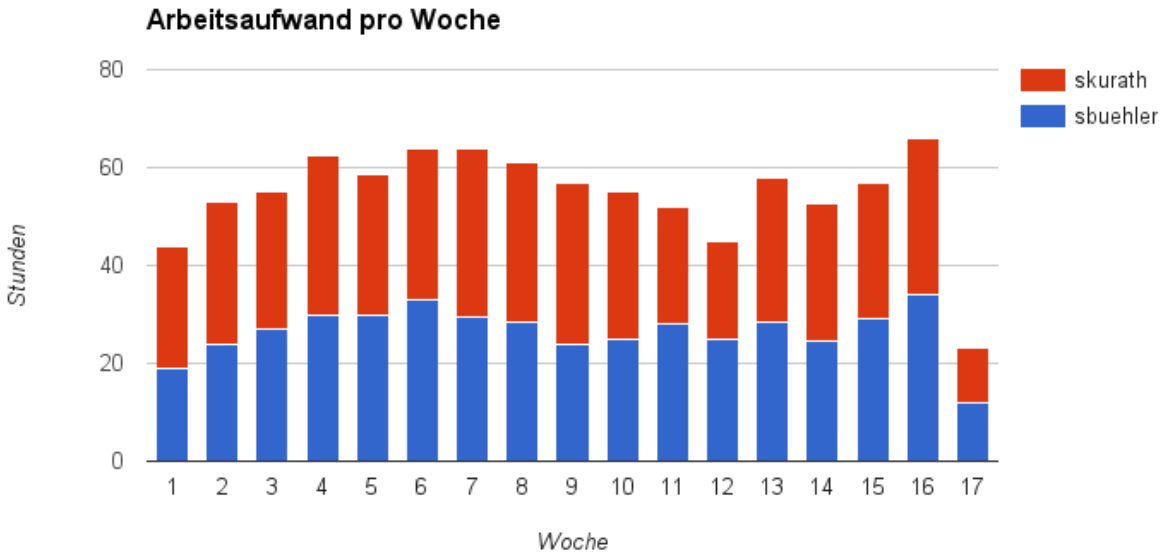


Abbildung 3.5: Arbeitsaufwand pro Woche

Student	Aufwand
Severin Bühler	451 Stunden
Samuel Kurath	476.5 Stunden
Total	927.5 Stunden

Tabelle 3.4: Arbeitsaufwand

Fazit

Generell konnten wir uns an die geplanten Phasen halten, dies konnte teils jedoch nur durch ein höheres Arbeitspensum erreicht werden. Was sich in mehr geleisteten Stunden widerspiegelt. Aufwändige Aufgaben waren das Aufbauen einer geeigneten CI Lösung, die in der Implementierung aufgezählten Hürden, die Erarbeitung eines passenden Domain Models und das Verfassen der Dokumentation.

3.6 Erfahrungsbericht

Samuel Kurath

Das Projekt strongMan war einen ausgezeichnete Gelegenheit, die während dem Studium erlangten Software Engineering Kenntnisse umzusetzen. Mit der Verwendung einer automatisierten Testumgebung und natürlich den implementierten Unit- wie auch Integrationtests konnten auch grössere Änderungen am Code vorgenommen werden ohne später auf unerwünschte Fehler zu stossen.
Weiter hatten wir die Gelegenheit unser Wissen in diversen Technologien auszubauen, dazu zähle ich Python, Schlüsselcontainer und Docker. Mit dem Einsatz von Django, einem Python basiertem Webframework und der Docker Erweiterung docker-compose konnte mein persönlicher Toolstack bereichert werden.
Die Arbeit an strongMan und die Zusammenarbeit im Team, sowie die Betreuung durch den Experten war immer sehr harmonisch und produktiv, was sich aus meiner Sicht auf das positive Gelingen des Projektes ausgewirkt hat.

Severin Bühler

Während dieses Projekts kamen sehr viel neue oder nur in der Theorie bekannte Technologien wie Django, VirtualEnv, asymmetrische Schlüsselcontainer, SystemD, Webserver auf mich zu. Die Einarbeit in diese war sehr spannend und ich haben einiges gelernt. Auch die vertiefte Arbeit mit Python selbst hat mich noch einige weitere sprachspezifische Eigenheiten lernen lassen.

Nicht desto trotz hatte die Verwendung von Django auch seine negativen Seiten. Das Django ORM hat einen schwerwiegenden Nachteil bezüglich Vererbung (siehe Seite 42) und wir hatten einige Mühe diesen zu umgehen. Auch das Django Templating hat seine mühsamen Seiten.

Die Zusammenarbeit mit Herrn Steffen und auch Herr Brunner war zu jedem Zeitpunkt zielorientiert und positiv. Bei Fragen konnten wir immer auf die Unterstützung von Ihnen zählen und wir standen nie unter überzogenem Leistungsdruck.

Kapitel 4

Softwaredokumentation

4.1 Installation script setup.py

The 'setup.py' file in the main directory of the project is the installation script for strongMan. Additionally to the setup commands, the script also provides a developer command to migrate the database automatically.

```
./setup.py <command> [options]
```

- -v | –verbose
  - Sets the output of setup.py to verbose.

install

```
./setup.py install [−p %python−interpreter%]
```

Makes strongMan ready to run. Sets up a virtual environment, installs the requirements, migrates the database and copies the static files to the static-files folder.

- -p | –python
  - Select a specific python interpreter to run strongMan
  - python3 is used as default value

uninstall

```
./setup.py uninstall
```

Undoes the installation. Puts the application to the same status as before the installation. Removes the virtual environment and deletes all static files.

add-service

```
sudo ./setup.py add−service
```

Adds strongMan as a systemd service. The service will be enabled and started. Additionally to the service, the script creates a desktop icon in the launcher which opens a webbrowser with the url http://localhost:1515.

You need root permissions to make changes in systemd.

remove-service

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

Removes the systemd service and the launcher icon.

migrate

```
./setup.py migrate [−dm]
```

Migrates the database if changes are available. Have a look on Django Migrations.

- -dm | –delete-migrations
  - Deletes all old migration scripts and the sqlite database. All data are lost!

4.2 Benutzerhandbuch

4.2.1 Installation

Requirements

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

Installation

```
1        git clone https://github.com/Sebubu/strongMan.git
2        cd strongMan
3        ./setup.py install
```

Start

```
1        ./run.py
```

Add a systemd service (optional)

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

```
1    sudo ./setup.py add-service # Adds the service
```

4.2.2 Uninstallation

Systemd Service entfernen (Optional)

Removes the systemd service and the launcher icon.

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

Remove application folder

To remove strongMan completely, just remove the application directory.

```
1    rm -rf strongMan/
```

Anhang A

Mockups

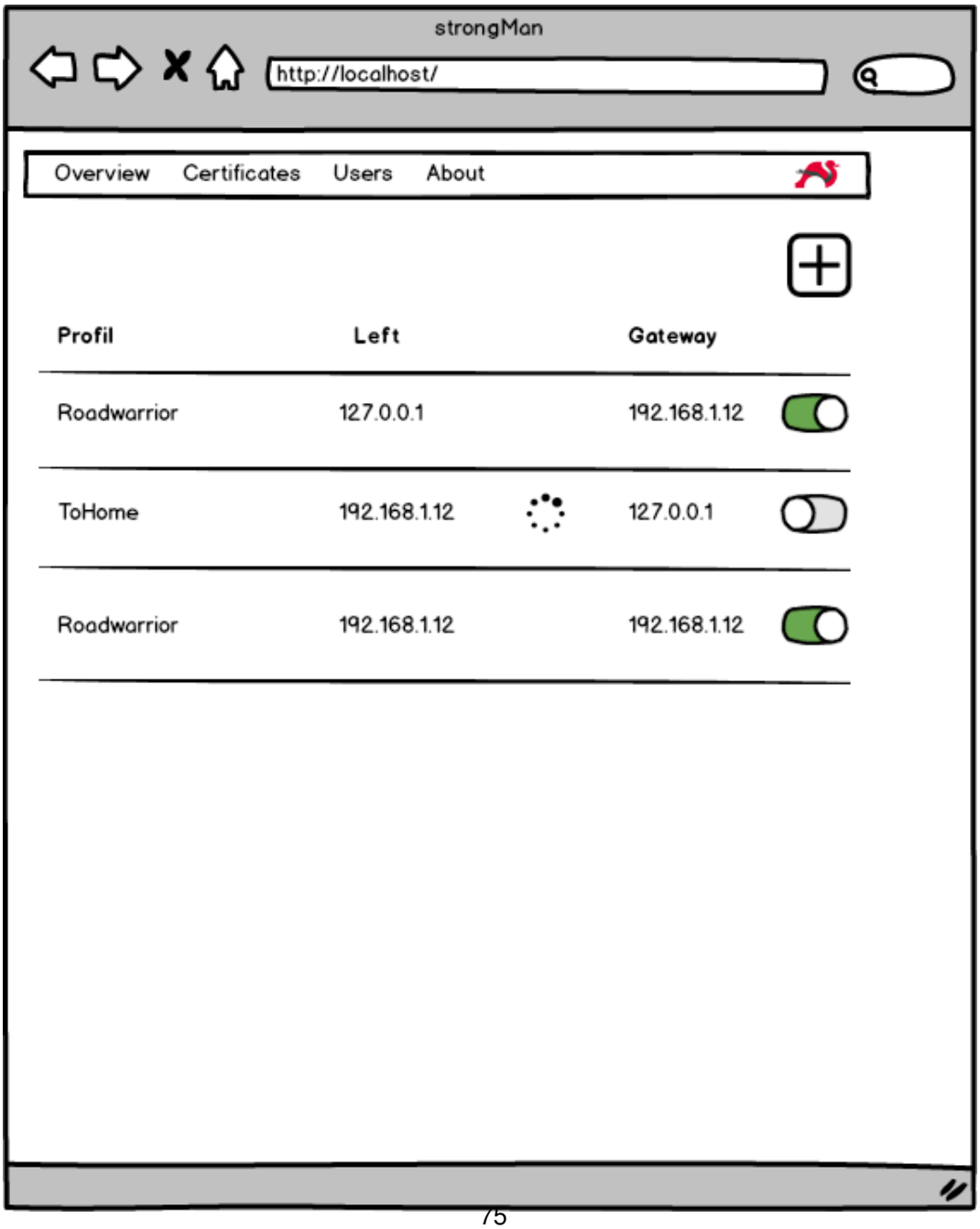


Abbildung A.1: Connection overview

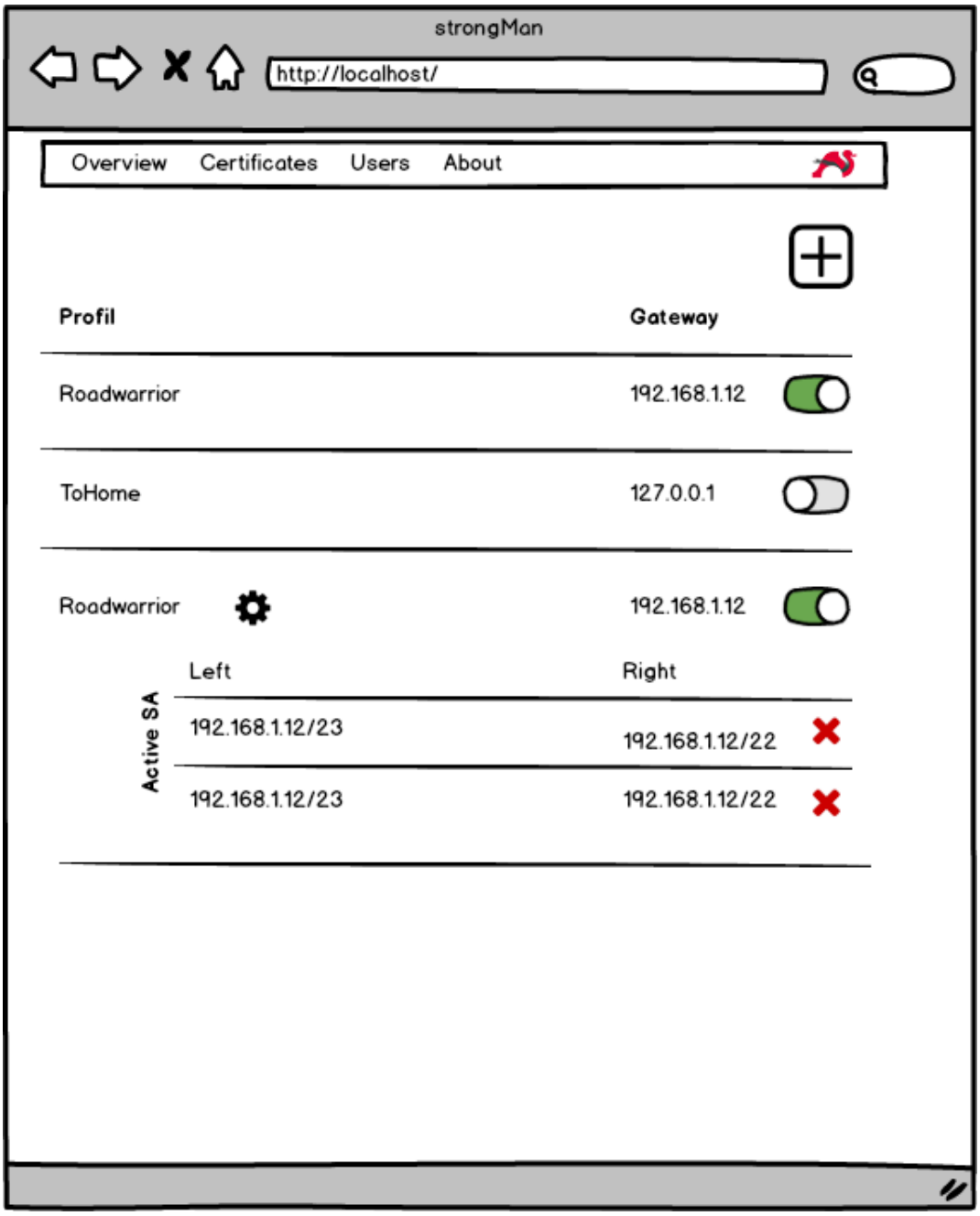


Abbildung A.2: Connection overview - connected and expanded

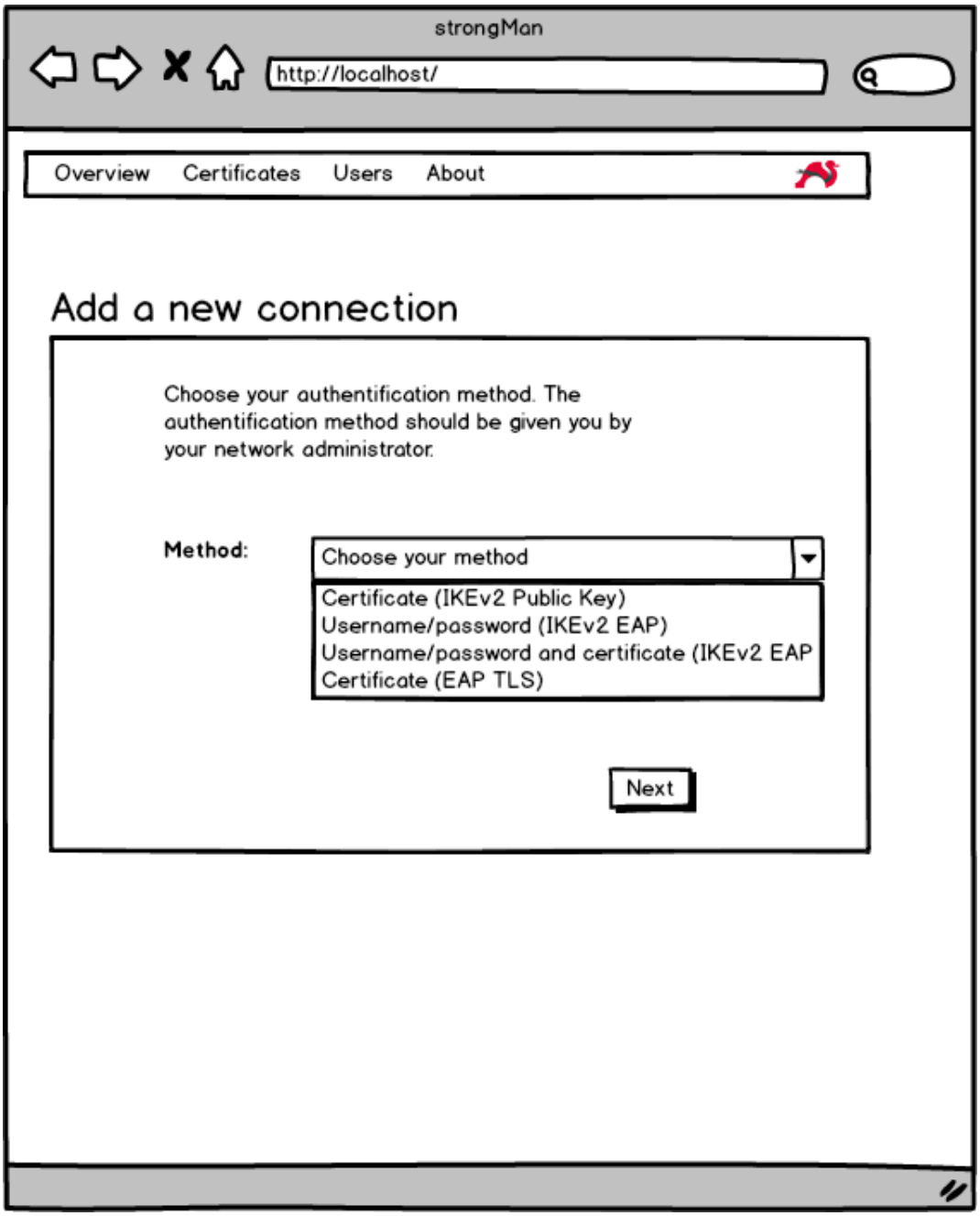


Abbildung A.3: Connection add - select method

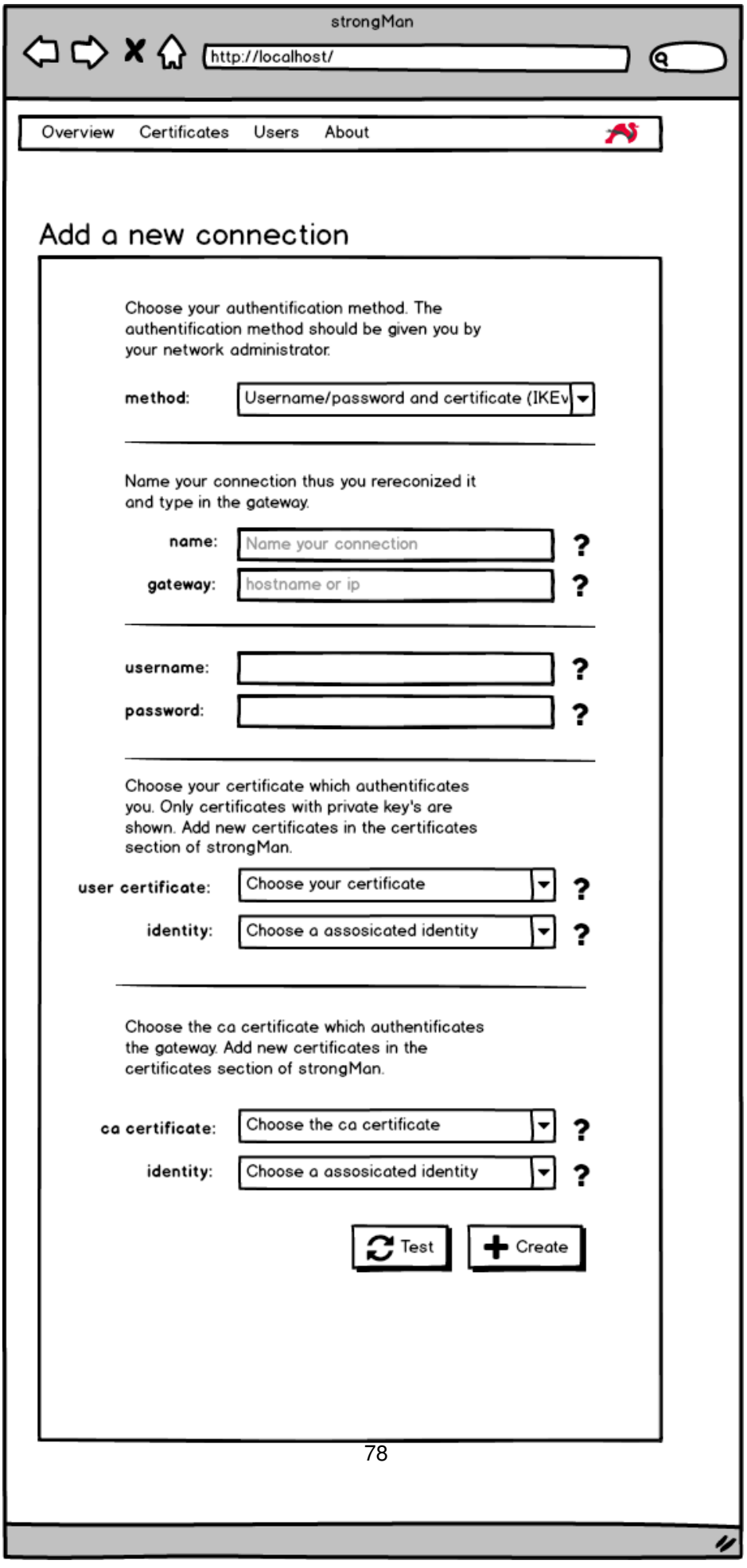


Abbildung A.4: Connection add - fill fields

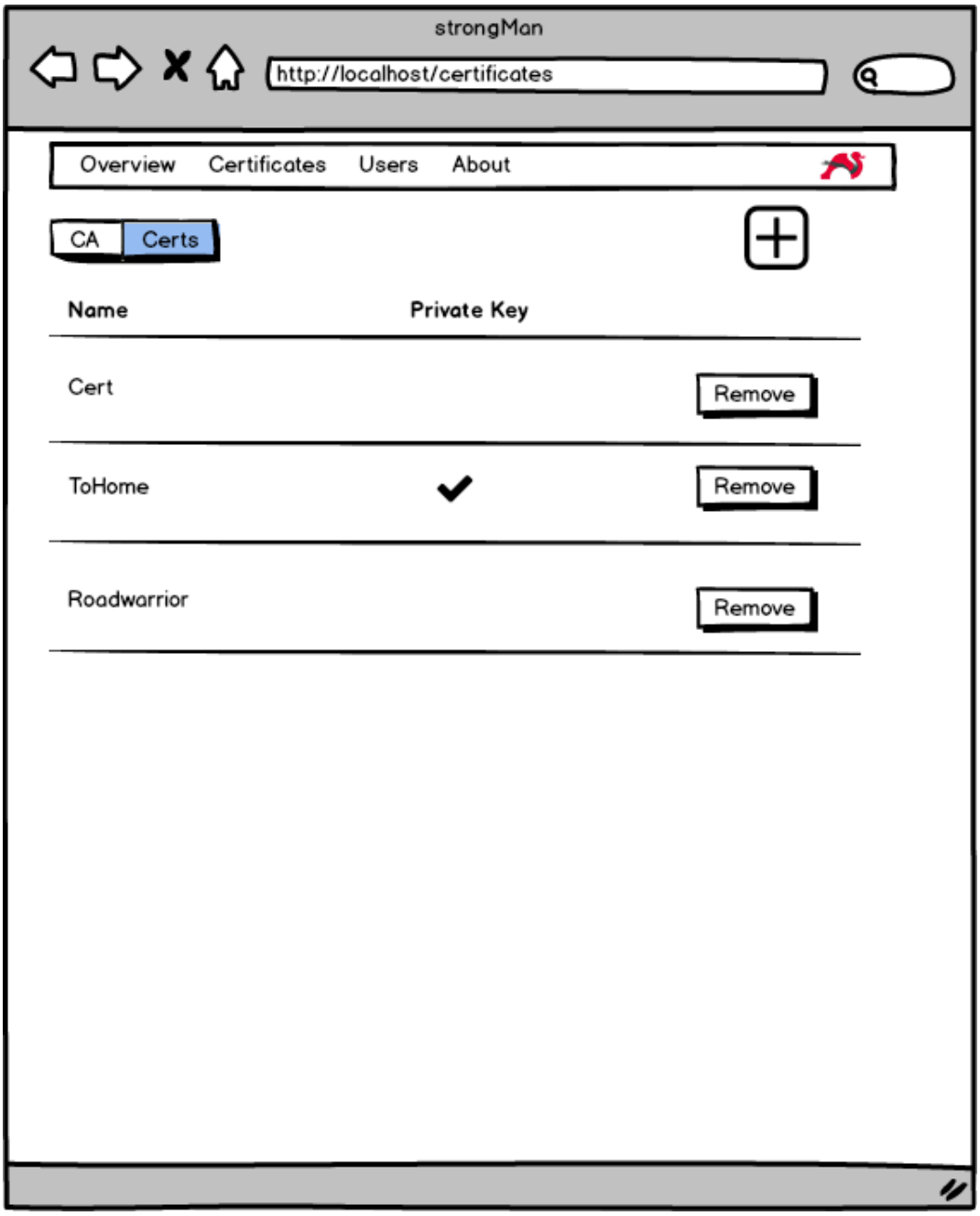


Abbildung A.5: Certificate overview

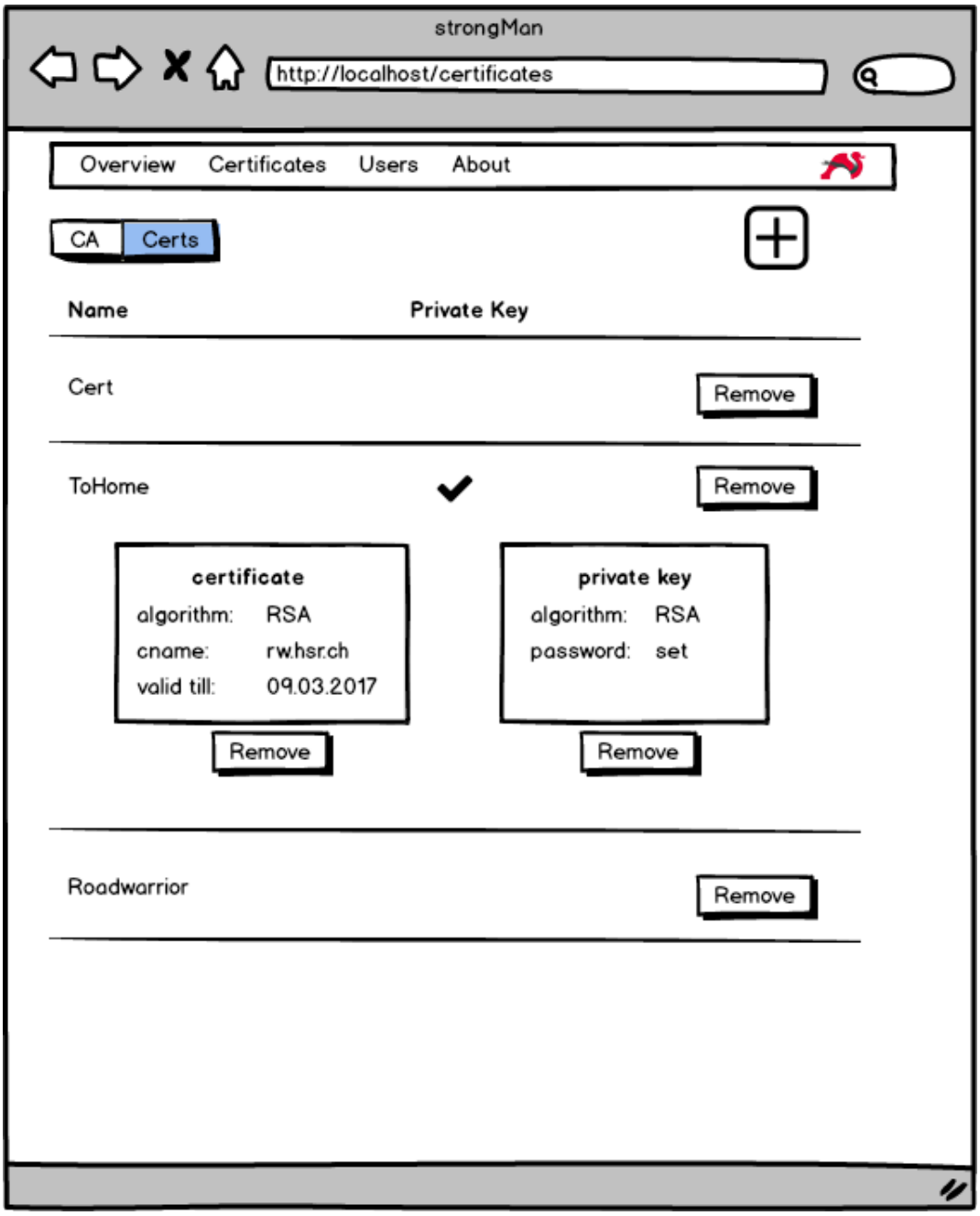


Abbildung A.6: Certificate overview - expanded

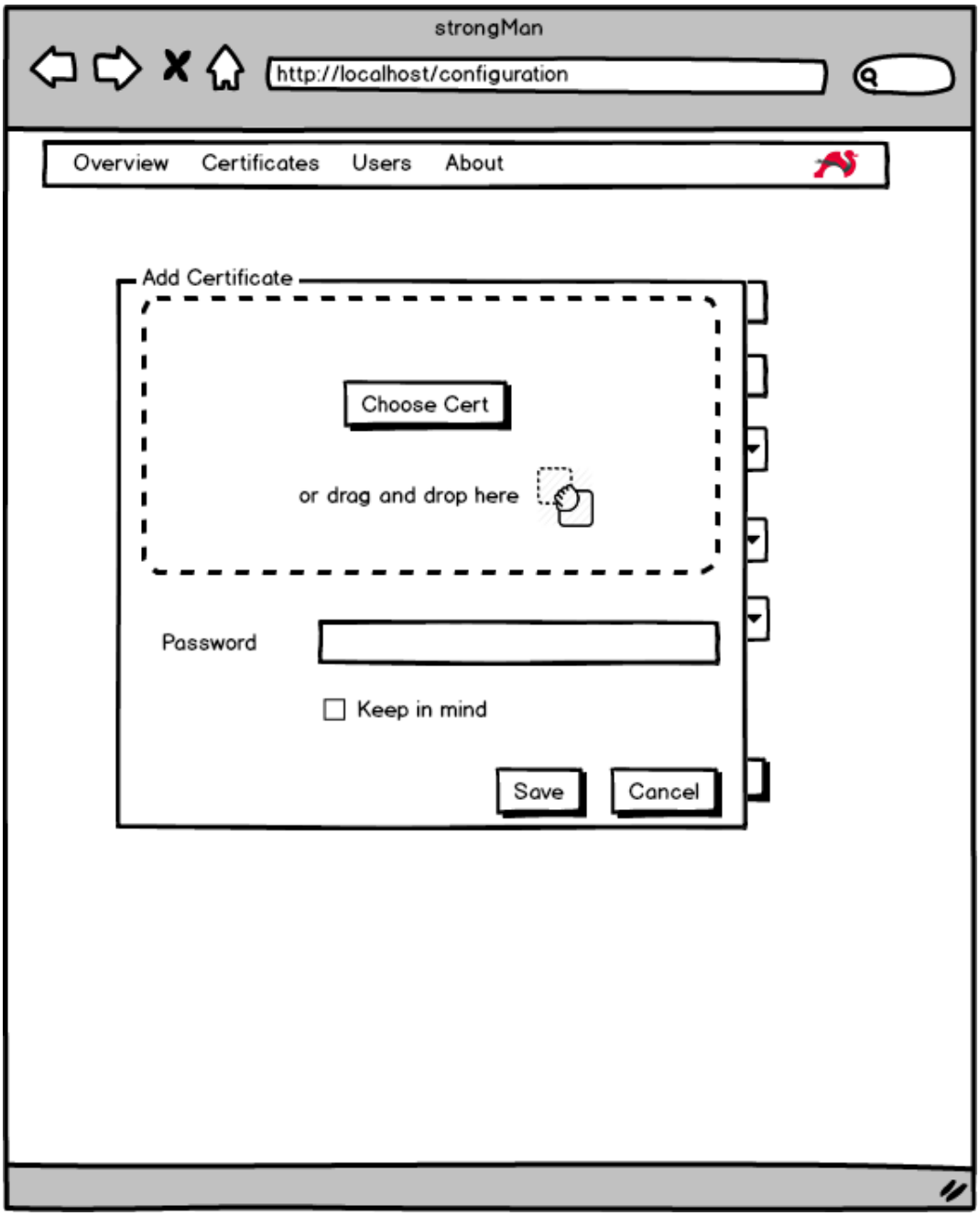


Abbildung A.7: Add certificate

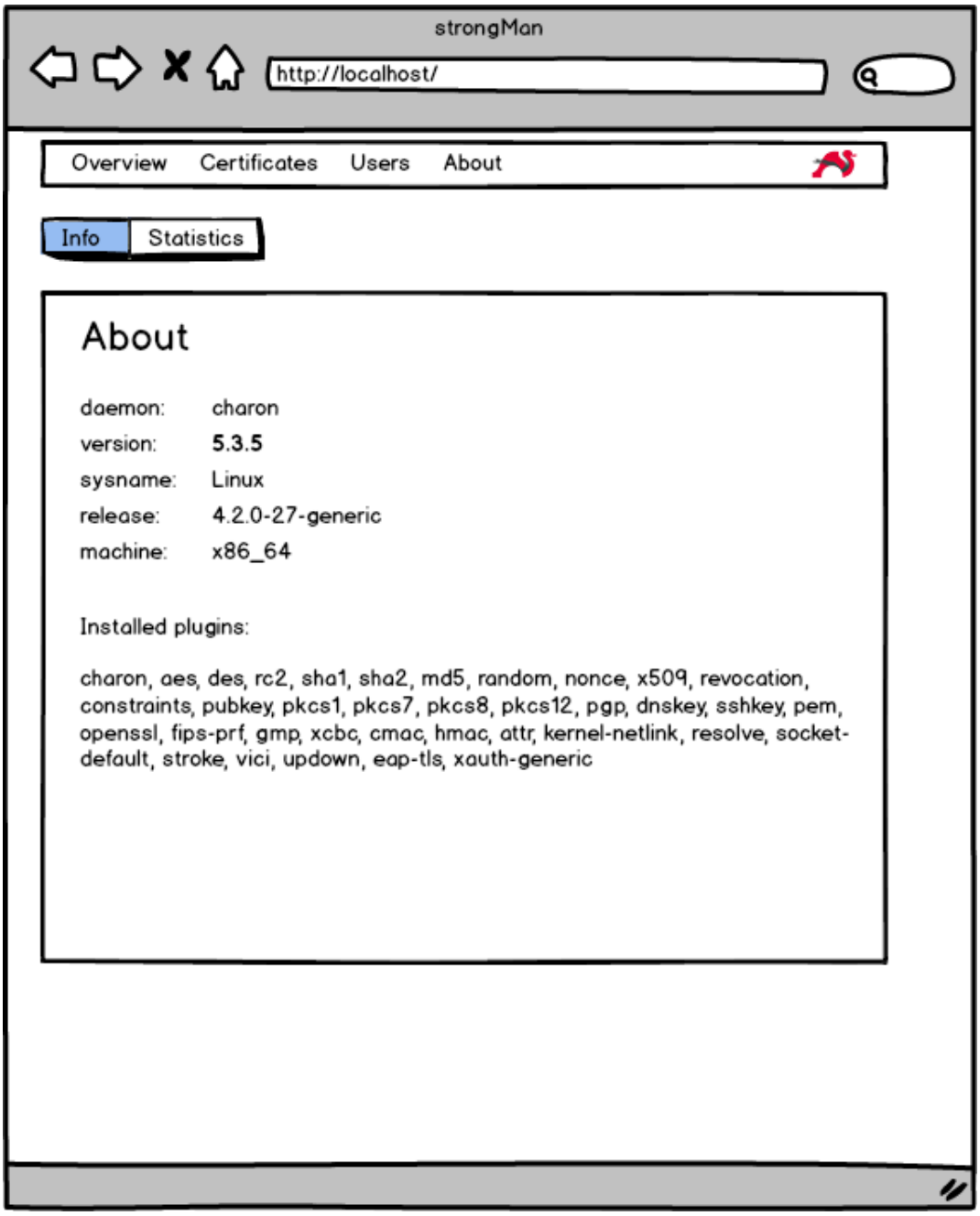


Abbildung A.8: About

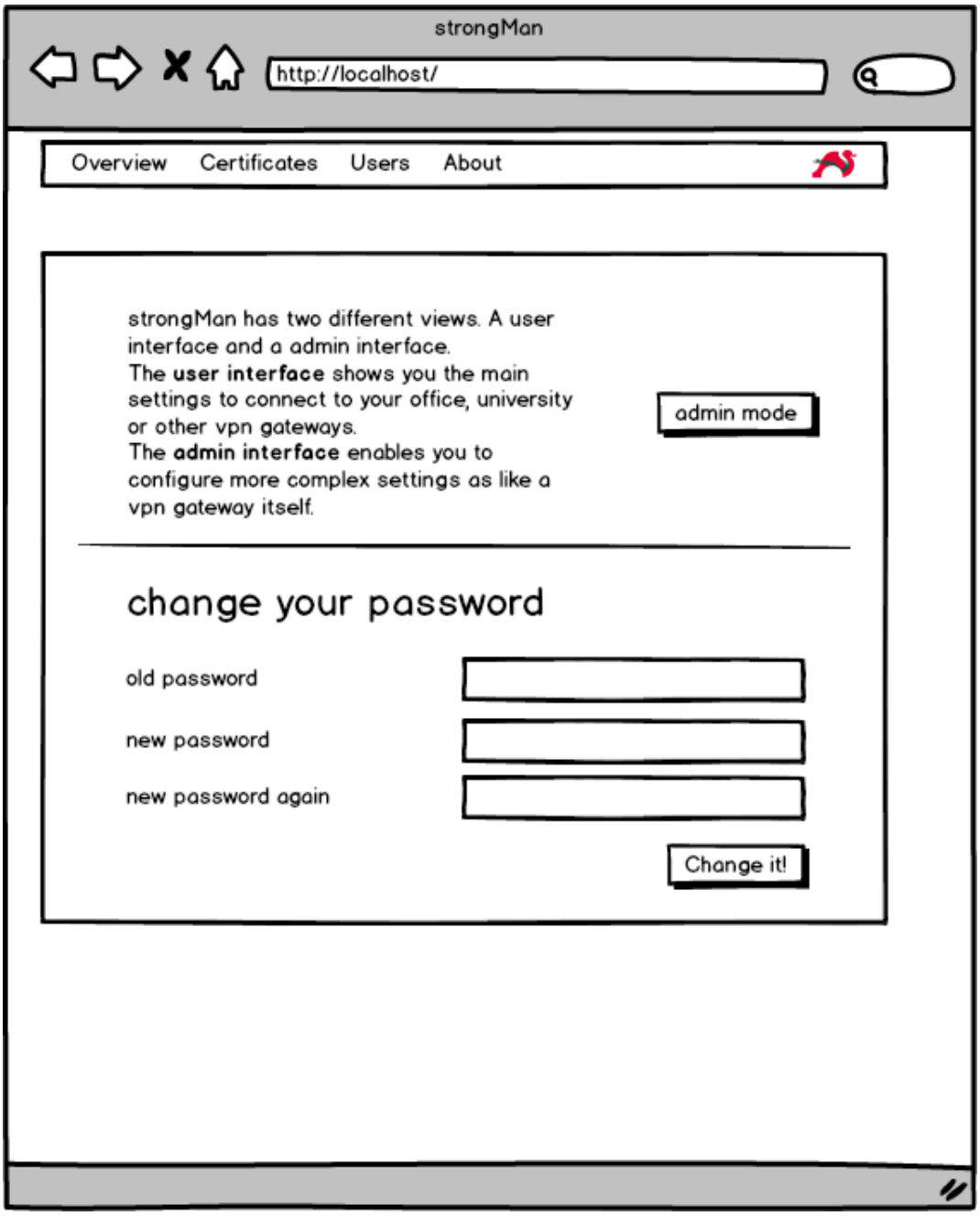


Abbildung A.9: User page

Anhang B

Inhalt der CD

Der Inhalt der CD gliedert sich folgendermassen:

```
CD
├── 0_strongMan_Bachelorarbeit.pdf
├── 1_Aufgabenstellung.pdf
├── 2_CODE
│   └── GitHub Repository
└── 3_Anhang
    ├── 0_Sequenzdiagramme
    │   ├── 0_ViciWrapper.asta
    │   ├── 1_deployment.asta
    │   └── 2_StatusInformation.asta
    ├── 1_Poster_deutsch.odp
    ├── 2_Poster_deutsch.pdf
    ├── 3_Zwischenpraesentation.pptx
    ├── 4_DomainModel.asta
    └── 5_UseCase.asta
```

Anhang C

Eigenständigkeitserklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt sind oder mit dem Betreuer schriftlich vereinbart wurden.
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Ort, Datum:

Rapperswil, 15. Juni 2016

Namen, Unterschriften:

Severin Bühler

Samuel Kurath

Anhang D

Glossar

ASN.1 Abstract Syntax Notation One ist eine Beschreibungssprache zur Definition von Datenstrukturen. 20, 46

Docker Leichtgewichtige Virtualisierung. 49

EAP Das Extensible Authentication Protocol wurde entwickelt um eine generische Schnittstelle für unterschiedliche Authentisierungsverfahren zu bieten. 15, 30, 59

ESP Encapsulating Security Payload ist für die Sicherung von Authentizität, Integrität und Vertraulichkeit der übertragenen IP-Pakete zuständig. 27

Git Git ist ein Versionsverwaltungssystem. 62

GitHub GitHub ist ein webbasierter Dienst für Software-Entwicklungsprojekte. 49, 62

IKE Internet Key Exchange wird verwendet um Security Association für IPsec aufzubauen. 2, 27, 59

IPsec IPsec erlaubt es sichere Verbindung über eine potentiell unsicheres Netz auf dem Internet Layer zu verschlüsseln. 8, 12, 26, 46

Jira Jira ist eine webbasierte Anwendung für Projektmanagement. 62

ORM Object-relational mapping, Software zur Ablegung von Objekten in relationalen Datenbanken. 14, 37, 45, 69

PIP PIP ist ein Paketverwaltungsprogramm für Python. 39

PKCS1 Container für asymmetrische Schlüssel. 20, 46

PKCS12 Container für asymmetrische Schlüssel. Kann mehrere Schlüssel beinhalten. 20, 21, 46

PKCS8 Container für asymmetrische Schlüssel. 20, 46

PyCharm PyCharm ist eine IDE von JetBrains für Python. 62, 93

RFC Requests for Comments, technische und organisatorische Dokumenten die einem Standard gleichen. 20

RUP Rational Unified Process, ist eine iteratives Vorgehensmodell zur Umsetzung von Softwareprojekten. 63

SA Eine Security Association beschreibt die gemeinsamen Sicherheitsattribute zwischen zwei kommunizierenden Endpunkten. 2, 11

Travis CI Travis CI ist ein Continuous Integration Anbieter. 50, 62, 66

VICI Versatile IKE Configuration Interface ist eine JSON-artige Schnittstelle von strongSwan.. 2, 8, 11, 12, 14, 26, 38, 39, 59, 64–66

X.509 Zertifikatsstandard. 20, 22, 28, 46, 59

Anhang E

Literaturverzeichnis

[AlD16] AlDanial. Count Lines of Code. https://github.com/AlDanial/cloc, Aufgerufen Juni 2016.

[Aye16] Bradley Ayers. django-tables2. https://github.com/bradleyayers/django-tables2, Aufgerufen Juni 2016.

[cd16] The cryptography developers. Cryptography. https://cryptography.io/en/latest/, Aufgerufen Juni 2016.

[Che16] Benoit Chesneau. Gunicorn. http://gunicorn.org/, Aufgerufen Juni 2016.

[Ell16] Justin Ellingwood. Howto set up django with nginx and gunicorn. https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-14-04, Aufgerufen Juni 2016.

[Eti16] Ilya Etingof. pyasn1. http://pyasn1.sourceforge.net/, Aufgerufen Juni 2016.

[Fou16a] Django Software Foundation. Django project. https://www.djangoproject.com/, Aufgerufen Juni 2016.

[fou16b] Wikipedia foundation. wsgi. https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface, Aufgerufen Juni 2016.

[Gan16] Dave Gandy. fontawesome. http://fontawesome.io/, Aufgerufen Juni 2016.

[Hur16] Min Hur. bootstrap-toggle. https://github.com/minhur/bootstrap-toggle/, Aufgerufen Juni 2016.

[jFoc16] jQuery Foundation and other contributors. jquery. https://jquery.com/, Aufgerufen Juni 2016.

[Lit16] Dwayne C. Litzenberger. PyCrypto. https://pypi.python.org/pypi/pycrypto, Aufgerufen Juni 2016.

[Moo16] Richard Moore. Pyaes. https://github.com/ricmoo/pyaes, Aufgerufen Juni 2016.

[Mor16] Silvio Moreto. bootstrap-select. https://silviomoreto.github.io/bootstrap-select/, Aufgerufen Juni 2016.

[ORC16] Inc ORCAS. django-fernet-fields. https://github.com/orcasgit/django-fernet-fields, Aufgerufen Juni 2016.

[Pra16] Chris Pratt. callout. http://cpratt.co/twitter-bootstrap-callout-css-styles/, Aufgerufen Juni 2016.

[Rei16] Kenneth Reitz. Dj-static. https://github.com/kennethreitz/dj-static, Aufgerufen Juni 2016.

[sP16] strongSwan Project. Versatile IKE Configuration Interface. https://wiki.strongswan.org/projects/strongswan/wiki/VICI, Aufgerufen Juni 2016.

[Twi16] Inc. Twitter. bootstrap. http://getbootstrap.com/, Aufgerufen Juni 2016.

[Vis16] Kartik Visweswaran. bootstrap-fileinput. https://github.com/kartik-v/bootstrap-fileinput, Aufgerufen Juni 2016.

[wbo16] wbond. Oscrypto. https://github.com/wbond/oscrypto, Aufgerufen Juni 2016.

Anhang F

Abbildungsverzeichnis

1 Übersicht 2

1.1 strongMan Verbindungsübersicht 8
1.2 Use Case Diagramm 10
1.3 Deployment 12
1.4 Domain Model 13
1.5 Ausschnitt aus der Connection Übersicht, UC01: VPN Verbindung CRUD 17
1.6 Ausschnitt aus dem Connection hinzufügen, UC01: VPN Verbindung CRUD 18

2.1 Verbindung erfassen 26
2.2 Verbindung starten 26
2.3 IKEv2 Certificate 28
2.4 IKEv2 EAP 30
2.5 IKEv2 Certificate + EAP 32
2.6 IKEv2 EAP-TLS 34
2.7 VICI Übersicht 39
2.8 VICI Diagramm 39
2.9 Log Messages Sequenzdiagramm 40
2.10 Statusinformationen Sequenzdiagramm 41
2.11 Schematische Darstellung der Testumgebung 50
2.12 Primäre Persona User 51
2.13 Sekundäre Persona Administrator 53
2.14 Connection Overview 57
2.15 Certificate Overview 57

3.1 Prof. Dr. Andreas Steffen 61
3.2 Bühler Severin 61
3.3 Kurath Samuel 61
3.4 Gantt Chart 64
3.5 Arbeitsaufwand pro Woche 68

A.1 Connection overview 75
A.2 Connection overview - connected and expanded 76
A.3 Connection add - select method 77

A.4 Connection add - fill fields 78
A.5 Certificate overview 79
A.6 Certificate overview - expanded 80
A.7 Add certificate 81
A.8 About 82
A.9 User page 83

Anhang G

Tabellenverzeichnis

1.1 Evaluation Usermanagement 19
1.2 ASN.1 Schemas 20
1.3 Encoding 20
1.4 Pyasn1 21
1.5 Cryptography 22
1.6 Pycrypto 22
1.7 Oscrypto 23

2.1 Automatisch konfigurierte Parameter 27
2.2 Automatisch konfigurierte Parameter 27
2.3 MVC Konzept Django 36
2.4 Authentication 42
2.5 EapAuthentication 42
2.6 Test Coverage 56
2.7 Codezeilen 56
2.8 Auswertung 59

3.1 Risiken 65
3.2 Risikoauswertung 66
3.3 Zeitplanung 67
3.4 Arbeitsaufwand 68

Anhang H

Verzeichnis der Entscheide

1.1 Evaluation Usermanagement 19
1.2 Evaluation Zertifikatsbibliothek 23
2.3 Datenbankfelder werden mit einem Secret Key verschlüsselt 45
2.4 Gunicorn - Statische Dateien 47
2.5 Usability Tests vs. Server Features 51
3.6 PyCharm 62
3.7 GitHub 62
3.8 Travis CI 62
3.9 Jira 62
3.10 Construction3 63