HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Konzeption und Umsetzung eines IPv6-VPN für die Abteilung Informatik

Jan Philipp Timme

Masterarbeit im Studiengang „Angewandte Informatik“

6. November 2018

Autor Jan Philipp Timme
Matrikelnummer 1433117
Abteilung Informatik, Fakultät IV
Hochschule Hannover
jan.philipp@timme.it

Erstprüfer Prof. Dr. Stefan Wohlfeil
Abteilung Informatik, Fakultät IV
Hochschule Hannover
stefan.wohlfeil@hs-hannover.de

Zweitprüfer Prof. Dr. Matthias Hovestadt
Abteilung Informatik, Fakultät IV
Hochschule Hannover
matthias.hovestadt@hs-hannover.de

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die eingereichte Masterarbeit selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Hannover, den 6. November 2018 Unterschrift

Inhaltsverzeichnis

1 Einleitung

Die Menge der noch verfügbaren IPv4-Adressen neigt sich dem Ende zu. Laut Angaben des RIPE NCC[1] vom Juni 2018 sind noch ungefähr 8,63 Millionen IPv4-Adressen verfügbar[2]. Das entspricht etwa der Hälfte der nutzbaren Host-Adressen eines `/8`-Blocks. Betrachtet man die Vergabegeschwindigkeit von IPv4-Adressen aus den letzten drei Jahren, so könnte man den Zeitpunkt der Erschöpfung von IPv4-Adressen zwischen 2019 und 2021 vermuten[3].

Vor diesem Hintergrund wird IPv6 als Nachfolger von IPv4 zunehmend verwendet: Immer mehr Internetdienste können über IPv6 erreicht werden, und auch die Internetanbieter stellen ihren Kunden IPv6-fähige Internetanschlüsse zur Verfügung. Der Anteil von Suchanfragen, die über IPv6 an Google gestellt wurden, hat von 5,84% am 1. Januar 2015 auf 21,11% am 1. Juni 2018 zugenommen[4]. Am AMS-IX[5], dem Internet-Austauschpunkt in Amsterdam, hat sich der Durchfluss von IPv6-Verkehr in den letzten 12 Monaten im Durchschnitt von etwa 55 Gbit/s im August 2017 auf etwa 85 Gbit/s im Mai 2018 gesteigert[6].

An der Hochschule Hannover ist die Abteilung Informatik Vorreiter in der Erprobung von IPv6. Seit Anfang 2018 ist das Netz der Abteilung über IPv6 direkt an das Internet angebunden. Damit ist die Voraussetzung gegeben, um Netzwerkgeräte für IPv6 zu konfigurieren und bestehende Netzwerkdienste auch über IPv6 anzubieten.

Beschäftigten und Studierenden der Abteilung Informatik steht ein VPN-Dienst zur Verfügung, um Zugang in das Netz der Abteilung aus dem Internet heraus zu erhalten. Bisher ist dieser Dienst nur über IPv4 erreichbar und ermöglicht den Zugang in das Abteilungsnetz ausschließlich über IPv4.

[1] RIPE Network Coordination Centre
[2] `https://www.ripe.net/publications/ipv6-info-centre/about-ipv6/ipv4-exhaustion/ipv4-available-pool-graph`, abgerufen am 03.06.2018
[3] `https://ipv4.potaroo.net/`, abgerufen am 03.06.2018
[4] `https://www.google.com/intl/en/ipv6/statistics.html`, abgerufen am 03.06.2018
[5] Amsterdam Internet Exchange
[6] `https://ams-ix.net/technical/statistics/sflow-stats/ipv6-traffic`, abgerufen am 03.06.2018

In dieser Masterarbeit soll ein neuer, IPv6-fähiger VPN-Dienst konzipiert werden, der die Idee des bisherigen IPv4-VPN-Dienst aufgreift. Der genaue Arbeitsauftrag dahinter wird in Kapitel 2 erläutert. Dort werden alle Anforderungen und Rahmenbedingungen erfasst, die bei der Konzeption des neuen VPN-Dienst berücksichtigt werden müssen. Im Anschluss wird die zukünftige Umgebung des VPN-Dienstes in Kapitel 3 vorgestellt: Die Netzarchitektur der Abteilung Informatik inklusive Firewallkonzept. Auf Basis der daraus gewonnenen Informationen werden dann in Kapitel 4 Konzepte für den VPN-Dienst und dessen Benutzerverwaltung abgeleitet. Anhand der Konzepte wird in Kapitel 5 dann nach Software zur Realisierung des VPN-Dienstes gesucht, mögliche VPN-Software präsentiert und die Auswahl begründet. Anhand der ausgewählten Software wird das in Kapitel 4 vorgestellte VPN-Konzept in Kapitel 6 auf die gewählte VPN-Software zugeschnitten. Im Anschluss wird in Kapitel 7 der VPN-Dienst installiert und konfiguriert. In Kapitel 8 wird die Benutzerverwaltung umgesetzt. In Kapitel 9 werden die Ergebnisse und gewonnen Erkenntnisse dieser Arbeit bewertet und Ideen für weiterführende Arbeiten aufgeführt.

Die während der Durchführung dieser Arbeit erzeugten Dokumente sind dem Anhang beigefügt.

2 Arbeitsauftrag

Die Abteilung Informatik betreibt einen VPN-Dienst auf Basis von OpenVPN, der von Beschäftigten und Studierenden benutzt werden kann, um über das Internet auf das Netz der Abteilung Informatik zuzugreifen. Dieser Dienst ist bisher nur über IPv4 erreichbar, und ermöglicht auch nur über IPv4 den Zugriff auf das Netz der Abteilung Informatik durch das VPN.

VPN: Hinter der Abkürzung „VPN“ verbirgt sich der Begriff *Virtual Private Network*. Ein VPN ist ein *virtuelles* Netz zwischen mindestens zwei Teilnehmern eines physischen Netzes. Es ist virtuell, weil es nicht für alle Teilnehmer des physischen Netzes existiert, sondern nur für die VPN-Teilnehmer logisch existiert.

Ein VPN ist aus mehreren Gründen ein *privates* Netz: Zunächst muss man von dem VPN Kenntnis haben, um daran teilnehmen zu können. Zusätzlich kann die Authentisierung gegenüber einem oder mehreren VPN-Teilnehmern als Zugangsvoraussetzung bestehen. Die Verschlüsselung des VPN-Datenverkehrs vor dessen Übertragung über ein physisches Netz ist ebenfalls ein Grund, weshalb ein VPN ein privates Netz ist.

Verschickt ein VPN-Teilnehmer ein Datenpaket über das VPN an einen anderen VPN-Teilnehmer, so wird das Paket vom Sender in ein neues Datenpaket gekapselt und über das physische Netz an den anderen VPN-Teilnehmer verschickt. Der Empfänger erhält das gekapselte Datenpaket über das physische Netz und kann das über das VPN übertragene Datenpaket daraus herausholen. Das VPN schafft somit eine direkte, logische Verbindung zwischen beiden Teilnehmern, während die eigentliche Datenübertragung über das physische Netz geleitet wird.

Auftrag: Der Auftrag dieser Arbeit ist die Konzeption und Umsetzung eines neuen VPN-Dienstes, der sowohl über IPv4 und IPv6 erreichbar ist und den Zugriff auf das Abteilungsnetz über diese beiden Protokolle ermöglicht. Dieser neue VPN-Dienst soll den alten VPN-Dienst ablösen. Dafür wird vorausgesetzt, dass der neue VPN-Dienst den Zugang zu allen IPv4-Netzen der Abteilung Informatik ermöglicht, die bisher über den alten VPN-Dienst erreichbar sind. Zusätzlich soll der neue VPN-Dienst den Zugang zu allen IPv6-Netzen der Abteilung Informatik ermöglichen, die logisch zu den zuvor genannten IPv4-Netzen gehören.

Teil des Auftrags ist die Erstellung eine Konzepts für die Verwaltung der VPN-Benutzer, sowie die Erstellung eines Konzepts für den Betrieb des neuen VPN-Dienstes. Die zur Umsetzung verwendete Software ist nicht vorgegeben und soll anhand der aufgestellten Konzepte, Anforderungen und Rahmenbedingungen ausgewählt werden.

Ermittelte Anforderungen: In diesem Abschnitt werden alle Anforderungen und Rahmenbedingungen vorgestellt, die bei der Konzeption des neuen VPN-Dienst berücksichtigt werden müssen. Es handelt sich hier um Vorgaben, die im persönlichen Gespräch mit dem Auftraggeber und Erstprüfer dieser Arbeit ermittelt wurden.

Anf1 **Dual-Stack-Betrieb:** Der VPN-Dienst soll aus dem Internet über IPv4 und IPv6 erreichbar sein und auch innerhalb des VPN diese beiden Protokolle anbieten.

Anf2 **VPN-interner Datenverkehr:** Nur die internen Netzbereiche der Abteilung Informatik sollen für Benutzer über das VPN erreichbar sein. Das betrifft alle Sicherheitszonen außer dem Internet. Neue Verbindungen aus allen Sicherheitszonen in das VPN-Netz sind verboten. VPN-Clients dürfen nicht im VPN untereinander kommunizieren. VPN-Clients dürfen durch das VPN nicht auf NFS[1]-Dienste zugreifen.

Anf3 **VPN-externer Datenverkehr:** Die Kommunikation zwischen VPN-Client und VPN-Server soll authentisiert und vertraulich stattfinden.

Anf4 **Benutzer:** Der VPN-Dienst soll nur von autorisierten Beschäftigten und Studierenden aus der Abteilung Informatik benutzt werden können. Die Benutzer des VPN-Dienstes sollen durch die Administratoren des VPN-Dienstes einfach verwaltet werden können. Der Dienst soll 50-500 VPN-Benutzer bedienen können.

Anf5 **Betrieb des VPN-Servers:** Die Serverkomponente des VPN-Dienstes soll auf einer aktuellen Version von Debian (9 oder höher) betrieben werden.

Anf6 **Betrieb der VPN-Clients:** Die VPN-Clientsoftware soll für aktuelle Versionen gängiger Betriebssysteme zur Verfügung stehen. Darunter fallen Microsoft Windows 10 (Version 1709 oder höher), Apple MAC OS (ab Version 10.13) und Linux-Distributionen (ab Kernel Version 3.10).

Anf7 **Betriebsprotokoll:** In Bezug auf die *Datenschutzgrundverordnung* (DSGVO) soll der VPN-Dienst im Regelbetrieb keine personenbezogenen Daten protokollieren.

Anf8 **Finanzieller Rahmen:** Für die Umsetzung des VPN-Dienstes muss auf kostenfreie Software und die bestehende Infrastruktur (Server, Netze, ...) der Abteilung Informatik zurückgegriffen werden, da keine finanziellen Mittel für den Erwerb einer VPN-Lösung zur Verfügung stehen.

[1] *Network File System* (NFS) ist ein Dienst für netzwerkübergreifenden Dateiaustausch

3 Netzarchitektur der Abteilung Informatik

Das Netz der Abteilung Informatik wird durch eine Firewall von Netzen außerhalb der Abteilung getrennt. Zu den Netzen außerhalb der Abteilung gehören (1) das Netz der Hochschule Hannover und (2) das Internet.

An der Firewall sind zwei lokale Netze angeschlossen: (1) Die *Demilitarisierte Zone* (DMZ) und (2) das interne Abteilungsnetz, welches durch einen zentralen Switch in mehrere *virtuelle Netze* (VLANs) unterteilt wird. Zusätzlich sind die Netze des Netzwerklabors und des IT-Sicherheitslabors über je einen eigenen Router an den Switch angeschlossen. Eine Skizze der Netztopologie mit den für diese Arbeit relevanten Teilen zeigt Abbildung 3.1.

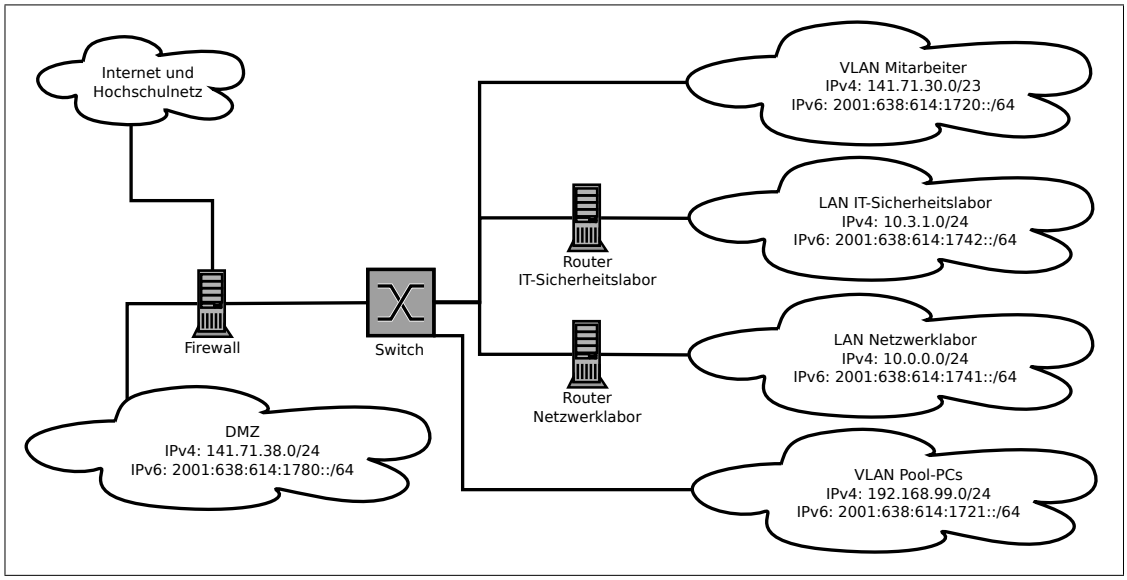

Abbildung 3.1: Skizze der Netztopologie der Abteilung Informatik

Die im Netz der Abteilung Informatik verwendeten Netze werden im Firewallkonzept als verschiedene Sicherheitszonen betrachtet. Für diese Arbeit sind die folgenden Zonen relevant:

Internet: Das „Internet“ bezeichnet den Bereich außerhalb des Netzes der Abteilung Informatik. Diese Zone umfasst neben dem Internet auch das Netz der Hochschule Hannover inklusive eduroam. Verbindungen in das Internet sind aus allen Zonen, außer der DMZ, erlaubt. Verbindungen aus dem Internet werden nur zu Diensten in der DMZ (wie zum Beispiel VPN), sowie zu dem SSH-Dienst im Mitarbeiter-Netz zugelassen.

DMZ: Von der Abteilung Informatik betriebenen Server stellen in diesem Netz Dienste zur Verfügung, die sowohl innerhalb der Abteilung als auch über das Internet erreichbar sind. Verbindungen in die DMZ zu Diensten wie VPN sind aus allen anderen Zonen heraus erlaubt. Verbindungen aus der DMZ in alle anderen Zonen sind verboten, um im Fall eines Sicherheitsvorfalls Angriffe auf alle anderen Zonen zu verhindern. Eine Ausnahme für dieses Verbot sind Verbindungen vom VPN-Dienst, die in das Mitarbeiter-Netz aufgebaut werden dürfen.

Mitarbeiter-Netz: An dieses Netz sind die Rechner aller Beschäftigten der Abteilung Informatik angeschlossen. Auch die internen Server der Abteilung, wie zum Beispiel Dateiserver oder der SSH-Server, sind an dieses Netz angeschlossen. Verbindungen in das Mitarbeiter-Netz aus dem Pool-PC-Netz und den Labor-Netzen sind erlaubt. Außerdem sind Verbindungen von dem VPN-Dienst aus der DMZ in das Mitarbeiter-Netz erlaubt. Zusätzlich ist der SSH-Server für SSH-Zugriffe aus dem Internet erreichbar. Verbindungen aus dem Mitarbeiter-Netz sind in alle anderen Zonen erlaubt.

Pool-PC-Netz: Enthält die Rechner in allen Poolräumen, die von Studierenden und Mitarbeitern der Abteilung Informatik benutzt werden. Verbindungen in das Pool-PC-Netz sind aus dem Mitarbeiter-Netz und den Labor-Netzen erlaubt. Verbindungen aus dem Pool-PC-Netz sind in alle anderen Zonen erlaubt.

Labor-Netze: Das Netzwerklabor und das Labor für IT-Sicherheit werden für diese Arbeit unter der Zone „Labor-Netze“ zusammengefasst. Verbindungen aus den Labornetzen heraus sind in alle anderen Zonen erlaubt. Verbindungen in die Labornetze sind aus dem Mitarbeiter-Netz und aus dem Pool-PC-Netz heraus erlaubt.

Ein Überblick der erlaubten Verbindungen zwischen den Sicherheitszonen ist in Tabelle 3.1 skizziert.

Tabelle 3.1: Überblick über erlaubte Verbindungen zwischen Sicherheitszonen

	... in die Zone ...				
Aus der Zone ...	Internet	DMZ	Mitarbeiter	Pool-PC-Netz	Labor-Netze
Internet	—	erlaubt	verboten	verboten	verboten
DMZ	verboten	—	verboten	verboten	verboten
Mitarbeiter	erlaubt	erlaubt	—	erlaubt	erlaubt
Pool-PC-Netz	erlaubt	erlaubt	erlaubt	—	erlaubt
Labor-Netze	erlaubt	erlaubt	erlaubt	erlaubt	—

4 Konzeptphase

In diesem Abschnitt soll aus dem Arbeitsauftrag aus Kapitel 2 ein konkretes Vorhaben entstehen. Dafür wird in diesem Kapitel das Konzept für den VPN-Dienst und das Konzept für die Benutzerverwaltung anhand der in Kapitel 2 ermittelten Anforderungen und Rahmenbedingungen erstellt.

4.1 Konzept des VPNs

Um VPN-Benutzern einen Zugang zum Netz der Abteilung Informatik zu ermöglichen, muss ein Zugangspunkt im Netz der Abteilung geschaffen werden, der für die VPN-Benutzer über das Internet erreichbar ist. Deshalb wird das VPN in Client-Server-Architektur aufgebaut, wobei der VPN-Server zum Zugangspunkt wird. Damit der VPN-Dienst auf dem VPN-Server über das Internet erreichbar ist, muss eine entsprechende Freigabe auf der Firewall der Abteilung Informatik eingereichtet werden.

Das Betriebssystem für den VPN-Server soll Debian 9 sein (Anf5). Der Server wird an das DMZ-Netz der Abteilung Informatik angeschlossen, weil Server in diesem Netz Dienste anbieten können, die im Internet erreichbar sind. Um den Dual-Stack-Betrieb zu ermöglichen, werden dem Server jeweils eine IPv4- und IPv6-Adresse zugewiesen (Anf1).

Damit nur Benutzer den VPN-Zugang benutzen können, die als Beschäftigte oder Studierende zur Abteilung Informatik gehören (Anf4), müssen sich VPN-Benutzer gegenüber dem VPN-Server authentisieren. Die Details zur Authentisierung von Benutzern und der Verwaltung autorisierter Benutzer werden in Kapitel 4.2 behandelt.

Vor der Benutherauthentisierung muss sich der VPN-Server gegenüber dem VPN-Client authentisieren, damit bei der Benutzerauthentisierung übertragene Zugangsdaten nur durch den VPN-Server der Abteilung Informatik empfangen und verarbeitet werden. Das Ausspähen von VPN-Zugangsdaten durch einen Angreifer wird somit verhindert, weil sich ein Angreifer gegenüber VPN-Clients nicht mehr als VPN-Server der Abteilung Informatik ausgeben kann, ohne sich vorher erfolgreich als VPN-Server zu authentisieren. Zusätzlich verhindert diese Maßnahme, dass VPN-Clients eine VPN-Sitzung zu dem VPN-Server eines Angreifers aufbauen können - Angriffe gegen Clientrechner durch den VPN-Tunnel stellen in diesem Kontext keine Bedrohung dar.

Die Authentisierung des VPN-Servers könnte über ein zuvor geteiltes, gemeinsames Geheimnis durchgeführt werden. Diese Vorgehensweise ist jedoch nicht sinnvoll, da der VPN-Dienst für mindestens 50-500 Benutzer ausgelegt wird. Unter diesem Umstand ist ein allen Benutzern bekanntes, gemeinsames Geheimnis für einen Angreifer leichter in Erfahrung zu bringen, je mehr Benutzer dieses Geheimnis kennen. Deshalb soll die Authentisierung des VPN-Servers gegenüber den VPN-Clients mit X.509-Public-Key-Zertifikaten durchgeführt werden.

Die Kommunikation innerhalb des VPNs soll auf OSI-Schicht 3 stattfinden, da lediglich IPv4- und IPv6-Datenverkehr durch das VPN übertragen werden soll (Anf2). Sonstige Protokolle auf OSI-Schicht 2 und 3 werden nicht benötigt. Die Übertragung von Ethernet-Frames durch den VPN-Tunnel ist nicht notwendig, und würde durch unnötige Datenübertragung nur zusätzliche Netzwerklast verursachen.

Für die IP-Netze der Abteilung Informatik, die über das VPN erreichbar sein sollen (Anf2), werden für die Dauer der VPN-Sitzung Einträge in der Routingtabelle des Clients erzeugt. IP-Pakete, die der Client an Computer im Abteilungsnetz schickt, sollen so durch den VPN-Tunnel geroutet werden. Damit die Pakete ihr Ziel auch erreichen, wird der VPN-Server als Router konfiguriert, der Pakete zwischen VPN-Tunnel und Abteilungsnetz weiterleitet.

Damit IP-Pakete auch aus dem Abteilungsnetz über den VPN-Tunnel zu den VPN-Clients geschickt werden können, müssen für die VPN-internen IP-Adressen der Clients entsprechende Routen zum VPN-Server auf den Routern im Abteilungsnetz konfiguriert sein. Da diese Konfiguration nur durch die Administratoren der Router vorgenommen wird, muss eine Lösung gewählt werden, die einheitlich für alle VPN-Clients funktioniert. Deshalb werden in Absprache mit dem IT-Team der Abteilung Informatik ein IPv4- und ein IPv6-Netz gewählt, aus denen die Clients VPN-interne IP-Adressen vom Server erhalten. Die Konfiguration von Routen für die beiden IP-Netze ist dadurch unabhängig von individuellen VPN-Sitzungen und kann einmalig vorgenommen werden.

Da keine öffentlichen IPv4-Netze für diesen Zweck verfügbar sind, wird ein Netz aus dem privaten Adressbereich `10.0.0.0/8` gewählt. Weil IP-Adressen aus diesem privaten Bereich nicht geroutet werden sollen, führt der VPN-Server *Network Address Translation* (NAT) durch. IPv4-Datenverkehr von VPN-Clients ins Abteilungsnetz trägt somit die öffentliche IPv4-Adresse des VPN-Servers. Dazugehörige Antwortpakete können dadurch zurück zum VPN-Server geroutet werden, welcher diese nach der NAT-Übersetzung an den richtigen VPN-Client weiterleitet.

Für IPv6 wurde ein `/64`-Netz aus dem Bereich `2001:638:614:1700::/56` gewählt. Die Router im Abteilungsnetz wurden konfiguriert, Pakete an Hosts in diesem Netz an die öffentliche IPv6-Adresse des VPN-Servers weiterleiten.

Damit keine Verbindungen aus allen Sicherheitszonen zu VPN-Clients möglich sind, die VPN-Clients über das VPN nicht untereinander kommunizieren können, und der NFS-Dienst über das VPN nicht benutzt werden kann (Anf2), soll eine lokale Firewall auf dem VPN-Server eingerichtet werden, die alle Regeln aus Anf2 umsetzt.

Damit die Kommunikation zwischen VPN-Clients und VPN-Server vertraulich bleibt (Anf3), soll der Datenverkehr zwischen Client und Server mit modernen Chiffren verschlüsselt werden. Um *Perfect Forward Secrecy* (PFS) zu errreichen, soll beim Aufbau der VPN-Sitzung ein entsprechend geeignetes Verfahren zum Schlüsselaustausch verwendet werden, sodass die ausgehandelten Sitzungsschlüssel im Nachhinein nicht rekonstruiert werden können.

Die Protokolleinstellungen des VPN-Servers sollen vor der Inbetriebnahme so angepasst werden, dass nach DSGVO keine personenbezogenen Daten protokolliert werden (Anf7).

Um die Wartbarkeit des VPN-Dienstes zu erhöhen, sollen bei der Installation und Konfiguration des VPN-Servers (und gegebenenfalls zusätzlicher Server) die existierenden Konzepte des IT-Teams zum Betrieb von Servern berücksichtigt werden.

Um das in diesem Abschnitt beschriebene Konzept umzusetzen, soll eine passende VPN-Software gewählt werden, deren Serverkomponente auf Debian 9 läuft (Anf5), und für die kompatible Clientkomponenten auf allen gängigen Betriebssystemen (Anf6) verfügbar sind.

4.2 Konzept der Benutzerverwaltung

Neben dem Konzept des VPNs muss auch ein Konzept zur Verwaltung der VPN-Benutzer definiert werden. Laut Anf4 aus Kapitel 2 gehören Beschäftigte und Studierende der Abteilung Informatik zur Zielgruppe des Dienstes. Der VPN-Dienst wird für mindestens 50-500 gleichzeitig aktive Benutzer ausgelegt. Beschäftigte und Studierende dürfen während ihrer Zugehörigkeit zur Abteilung Informatik den VPN-Zugang benutzen.

Methoden zur Benutzerauthentisierung: Die Authentisierung von VPN-Benutzern ist mit den folgenden Methoden möglich:

- X.509-Public-Key-Zertifikate
- Angabe von Benutzername und Passwort

Somit muss entschieden werden, ob die Authentisierung von VPN-Benutzern über Zertifikate oder über Zugangsdaten in Form von Benutzername und Passwort durchgeführt werden soll.

Authentisierung mit Zugangsdaten: Die Verwendung von Zugangsdaten zur Authentisierung bietet dem Benutzer einen hohen Komfort, da keine individuelle Konfiguration des VPN-Clients erforderlich ist, und der Benutzer sich seine Zugangsdaten leicht merken kann.

Zusätzlich ist denkbar, dass bereits existierende Zugangsdaten - wie zum Beispiel das Hochschulkonto des Benutzers - zur Authentisierung verwendet werden könnten, indem der VPN-Server an den Verzeichnisdienst der Hochschule Hannover angebunden wird. In diesem Fall erfolgt die Aktualisierung der Benutzerkonten durch die Hochschul-IT, sodass nur die Benutzerkonten aktiv sind, deren Benutzer gerade als Beschäftigte oder Studierende zur Hochschule gehören. Der Administrator des VPN-Servers könnte die Menge der erlaubte VPN-Benutzer durch Gruppenmitgliedschaften im Verzeichnisdienst der Hochschule, oder durch eine vom Verzeichnisdienst getrennte geführte Liste steuern.

Die genannten Vorteile ziehen allerdings auch Nachteile mit sich: Ein VPN-Dienst, der Benutzer über eingegebene Zugangsdaten authentisiert, kann zu einem Ziel für einen Brute-Force-Angriff werden. Kompromittierte Zugangsdaten können in diesem Fall für den Missbrauch des VPN-Dienstes verwendet werden.

Zusätzlich ist denkbar, dass kompromittierte Zugangsdaten auch für den Missbrauch anderer Systeme verwendet werden können, für die diese Zugangsdaten gültig sind. Das ist der Fall, wenn Benutzer ihr VPN-Passwort für weitere Dienste verwenden, und ein Angreifer sich mit diesen kompromittierten Zugagnsdaten Zugang zu diesen Diensten verschaffen kann. Das Schadenspotential umfasst in diesem Fall zusätzlich zu über den VPN-Zugang begangene Straftaten auch das Ausspähen von persönlichen Daten, die über die kompromittierten Zugangsdaten zugänglich sind.

Ein solches Schadenspotential besteht auch dann, wenn der VPN-Dienst an den Verzeichnisdienst der Hochschule Hannover angebunden würde. Dann könnten die kompromittierten Zugangsdaten verwendet werden, um beispielsweise die E-Mails des betroffenen Benutzers zu lesen, Daten von dessen Homelaufwerk zu stehlen, oder - im Fall eines Studierenden - über das Prüfungsverwaltungssystem Ergebnisse und weitere persönliche Daten auszuspähen. Eine Anbindung des VPN-Dienstes an den Verzeichnisdienst der Hochschule Hannover kommt aufgrund des damit verbundenen Risikos nicht in Frage.

Ein weiterer Nachteil ergibt sich dadurch, dass Zugangsdaten zum Zweck der Benutzerauthentisierung durch VPN-Client und -Server, sowie zusätzlich durch auf dem Server eingebundene Authentisierungsprogramme verarbeitet und übertragen werden müssen. Die involvierten Programme erhöhen die Menge des Quellcodes, der aufgrund seiner Position als Angriffsfläche keine Schwachstellen enthalten sollte. Da die Abwesenheit von Schwachstellen in Quellcode nicht garantiert werden kann, ergibt sich bei der Verwendung von Zugangsdaten zur Benutzerauthentisierung ein erhöhtes Bedrohungsrisiko.

Auch auf den Administrator des VPN-Servers kommt ein erheblicher Mehraufwand hinzu, wenn ein eigener Verzeichnisdienst (wie zum Beispiel ein OpenLDAP-Server) installiert werden soll, um die Zugangsdaten der VPN-Benutzer zu speichern und zu verwalten. Dieser zusätzliche Aufwand umfasst neben der Installation und dem Betrieb auch die Bereitstellung von Infrastruktur, über die Benutzer ihr VPN-Passwort gegebenenfalls ändern oder zurücksetzen können. Auch die Absicherung des Verzeichnisdienst ist notwendig, um Manipulation oder Auslesen der gespeicherten Daten durch einen Angreifer zu verhindern.

Authentisierung mit Zertifikaten: Die Authentisierung von Benutzern anhand von Zertifikaten erfordert den Aufbau und die Verwaltung einer *Public-Key-Infrastructure* (PKI) und bringt somit erst einmal eine Reihe Nachteile mit sich: Für die Benutzer ergibt sich ein zusätzlicher Aufwand durch das regelmäßige Beantragen von neuen Zertifikaten, sowie die Notwendigkeit der Anpassung der VPN-Clientkonfiguration, um die neuen Zertifikate zu benutzen. Auch für den Systemadministrator ergibt sich dieser zusätzliche Aufwand in Bezug auf die regelmäßige Erneuerung des Serverzertifikats und der *Certificate Revocation List* (CRL). Zusätzlich ist der Systemadministrator mit der Betreuung der *Zertifizierungsstelle* (CA) beschäftigt, indem er die Zertifikatsanträge der Benutzer entgegennimmt, prüft, und Benutzerzertifikate ausstellt. Der daraus resultierende Aufwand ist proportional zur Anzahl der VPN-Benutzer.

Dennoch gibt es Vorteile, die diesen zusätzlichen Aufwand rechtfertigen: Zertifikate haben eine klar definierte Laufzeit, die bei Ausstellung des Zertifikates festgelegt wird. Aufgrund der Länge der verwendeten Schlüssel ist es für einen Angreifer nicht möglich, diese in akzeptabler Zeit durch Brute-Force-Angriffe zu finden. Auch das Stehlen von privaten Schlüsseln kann einem Angreifer weniger attraktiv gemacht werden, wenn man die privaten Schlüssel nur verschlüsselt abspeichert.

Sollte ein privater Schlüssel dennoch kompromittiert werden, besteht jederzeit die Möglichkeit, diesen durch eine Ergänzung der CRL seitens der CA unbrauchbar zu machen. Zusätzlich gilt, dass ein kompromittierter Schlüssel lediglich zur Benutzung des VPN-Dienst berechtigt. Dadurch reduziert sich der potentielle Schaden, den ein Angreifer mit einem kompromittierten Schlüssel anrichten könnte, da der Angreifer lediglich den VPN-Dienst im Namen des Besitzers des kompromittierten Schlüssels benutzen könnte. Würde man VPN-Benutzer über ihr Hochschulkonto authentisieren, so hätte ein Angreifer mit einem kompromittierten Hochschulkonto beispielsweise Zugriff auf E-Mails, das Homelaufwerk oder Prüfungsergebnisse des Kontobesitzers.

Wahl der Authentisierungsmethode: Insgesamt bringt die Authentisierung von Benutzern mit Zertifikaten im Vergleich zur Authentisierung auf Basis von Zugangsdaten einen geringeren Arbeitsaufwand mit sich, da der Betrieb einer PKI weniger komplex ist, als der Betrieb eines eigenen Verzeichnisdienstes. Gleichzeitig ist die Angriffsfläche für Brute-Force-Angriffe bei einer Authentisierung auf Basis von Zugangsdaten größer als bei Zertifikaten, weil Passwörter im Vergleich zu RSA-Schlüsseln ab 1024 Bit Länge deutlich geringere Komplexität aufweisen, und durch einen Angreifer leichter ausprobiert werden können.

Zusätzlich besteht ein reduziertes Bedrohungsrisiko bei kompromittierten privaten Schlüsseln im Vergleich zu kompromittierten Zugangsdaten, welche gegebenenfalls für weitere Dienste gültig sein könnten, oder deren Passwort für mehrere Dienste durch den Benutzer wiederverwendet wurde.

Bei der Authentisierung von Zugangsdaten besteht die Möglichkeit, dass zusätzlicher Code in Form von Programmen oder Skripten in die VPN-Software integriert werden muss, womit die Zugangsdaten mit Hilfe eines Verzeichnisdienstes geprüft werden. Dadurch erhöht sich nicht nur die potentielle Angriffsfläche, sondern auch die Anzahl an potentiellen Fehlerquellen bei der Übertragung und Verarbeitung der Zugangsdaten.

Unter Abwägung dieser Vor- und Nachteile werden Zertifikate zur Authentisierung von Benutzern verwendet.

Installationskonzept für die PKI: Die PKI, die zur Ausstellung von Zertifikaten für die VPN-Benutzer und den VPN-Server benutzt werden soll, muss auf einem Computer installiert werden. Damit die PKI später von den zuständigen Administratoren aus dem IT-Team der Abteilung Informatik bedient werden kann, empfiehlt sich die Einrichtung der PKI auf einem Server, zu dem nur die Administratoren über SSH Zugang haben.

Da bereits ein Server für den Betrieb der VPN-Serverkomponente verwendet werden soll, liegt der Gedanke nahe, die PKI ebenfalls auf diesem Server zu installieren. Davon sollte jedoch abgesehen werden, weil die VPN-Serverkomponente über das Internet erreichbar ist, und somit das Risiko besteht, dass ein entfernter, nicht authentisierter Angreifer unter Ausnutzung von Sicherheitslücken in der VPN-Serverkomponente die Kontrolle über den VPN-Server erlangen kann. In diesem Fall ist die Vertraulichkeit des privaten Schlüssels vom Wurzelzertifikat der PKI gefährdet. Wird dieser private Schlüssel einem Angreifer bekannt, kann sich der Angreifer gültige Client- und Serverzertifikate ausstellen, und diese beispielsweise für Man-in-the-Middle-Angriffe verwenden.

Die Vertraulichkeit, die Authentizität und die Integrität des VPN-Dienstes wären ab diesem Zeitpunkt verletzt. VPN-Clients könnten nicht mehr darauf vertrauen, dass sie eine Sitzung mit dem VPN-Server der Abteilung Informatik aufgebaut haben. Ebenso könnte der VPN-Server nicht mehr darauf vertrauen, dass VPN-Clients mit gültigem Benutzerzertifikat wirklich im Auftrag legitimer VPN-Benutzer agieren. Insgesamt stellt das Szenario des kompromittierten, privaten Schlüssels des Wurzelzertifikats der PKI einen Totalschaden dar. Die einzige Abhilfe ist ein vollständiger Neuaufbau der PKI.

Um dieses Risiko im Vorfeld zu reduzieren, soll die PKI auf einem eigenen Server eingerichtet werden, der nur für diesen Zweck installiert wird. Dieser Server soll nur von den Administratoren aus dem IT-Team bedient werden können, und bietet keine Dienste an, die über das Internet erreichbar sind. Eine Platzierung des Servers in dem Mitarbeiter-Netz der Abteilung Informatik ist deshalb sinnvoll. Das Mitarbeiter-Netz wird für diesen Zweck als vertrauenswürdig eingestuft, weil nur Mitarbeiter und Studierende, die zur Abteilung Informatik gehören, Zugriffe in dieses Netzwerk vornehmen dürfen.

Die öffentlichen Daten der PKI sollen via HTTP über einen Webserver zur Verfügung gestellt werden. Zu den öffentlichen Daten der PKI gehören beispielsweise das Wurzelzertifikat oder die *Certificate Revocation List* (CRL).

Dieser Webserver soll auf dem PKI-Server installiert werden und soll die PKI-Daten im Mitarbeiter-Netz der Abteilung anbieten. Weitere Daten, wie zum Beispiel Anleitungen oder Konfigurationsdateien zur Installation und Einrichtung von VPN-Clients für den VPN-Dienst sollen ebenfalls über diesen Webserver den VPN-Benutzern zur Verfügung gestellt werden. Damit die VPN-Serverkomponente die aktuelle CRL der PKI über HTTP abrufen kann, soll in der Firewall der Abteilung Informatik eine entsprechende Freigabe eingerichtet werden.

4.3 Gesamtkonzept des VPN-Dienstes

Die geplante Konfiguration des VPN-Servers und dessen Platzierung im DMZ-Netz der Abteilung Informatik wurde in Kapitel 4.1 bereits dargelegt. Ein Konzept für die Installation der PKI auf einem separaten Server im Mitarbeiter-Netz wurde in Kapitel 4.2 erläutert. Abbildung 4.1 zeigt das Konzept des VPN-Dienstes logisch zusammengefasst.

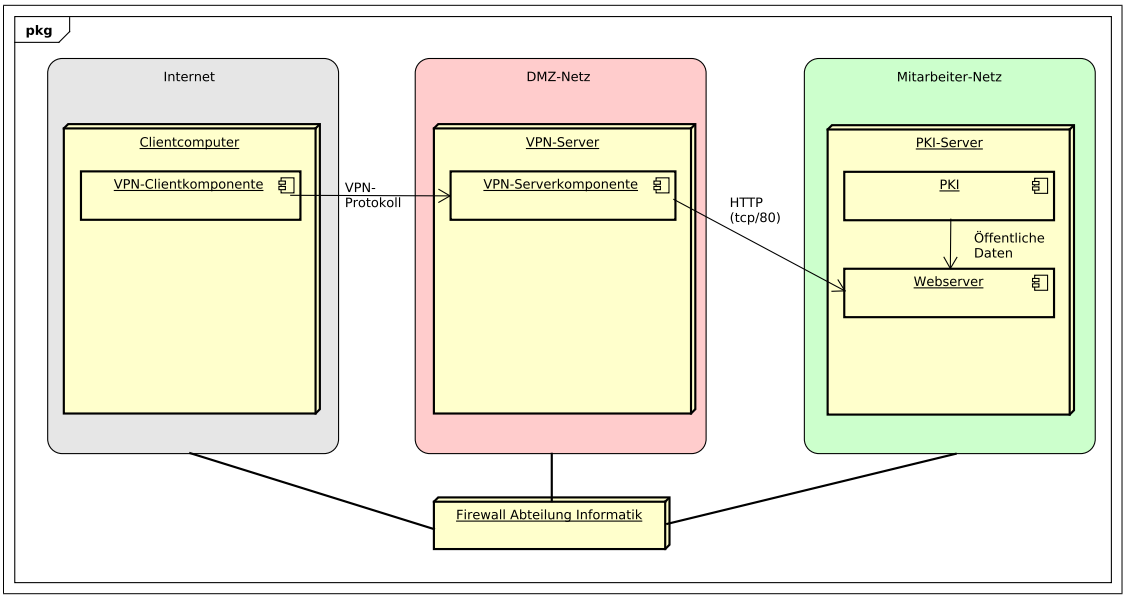

Abbildung 4.1: Installationskonzept für den VPN-Dienst

In der Abbildung sind zunächst die drei Netzwerk-Sicherheitszonen zu sehen, die in Kapitel 3 bereits vorgestellt wurden: Das Internet, das DMZ-Netz und das Mitarbeiter-Netz. Die Firewall der Abteilung Informatik agiert als Router zwischen diesen Netzen und lässt lediglich die Kommunikation zwischen den Netzen zu, die laut Regelwerk der Firewall erlaubt ist.

Im Mitarbeiter-Netz soll der CA-Server platziert werden, auf dem die PKI eingerichtet wird. Damit die öffentlichen Daten der PKI, so wie Anleitungen und Konfigurationsdateien, abrufbar sind, wird auf dem CA-Server zusätzlich eine Webserver-Komponente eingeplant.

Im DMZ-Netz soll der VPN-Server platziert werden, auf dem die VPN-Serverkomponente installiert wird. Da die VPN-Serverkomponente eine aktuelle CRL von der PKI benötigt, muss der HTTP-Zugriff vom VPN-Server auf den CA-Server in der Firewall der Abteilung Informatik erlaubt werden.

Im Internet steht beispielhaft der Clientcomputer eines VPN-Benutzers. Die auf dem Clientcomputer installierte VPN-Clientkomponente baut über ein VPN-Protokoll eine VPN-Sitzung mit der VPN-Serverkomponente auf. Diese Kommunikation zwischen Clientcomputern im Internet und dem VPN-Server im DMZ-Netz muss ebenfalls in der Abteilungsfirewall erlaubt werden.

5 Softwareauswahl

Das Umfeld, in dem der VPN-Dienst errichtet werden soll, wurde in Kapitel 3 bereits vorgestellt. In diesem Kapitel wird die Software ausgesucht, mit welcher der VPN-Dienst umgesetzt werden soll. Die bereits in Kapitel 2 ermittelten Anforderungen und Rahmenbedingungen sollen dabei berücksichtigt werden. Dann werden passende Softwarekandidaten vorgestellt, und für die Auswahl der in dieser Arbeit zu verwendenden Software gegenübergestellt. Im Anschluss soll auf Basis der gewählten VPN-Software gegebenenfalls noch benötigte Software zur Umsetzung der Benutzerverwaltung ausgewählt werden.

Anforderungsprofil: Aufgrund des finanziellen Rahmens (Anf8) kommt nur kostenfreie Software in Frage, deren Serverkomponente mit aktuellem Debian (Anf5) kompatibel ist. Die Clientkomponenten der gesuchten Software müssen unter den aktuellen Betriebssystemen laufen (Anf6).

Die Vorgabe von vertraulicher und authentisierter Kommunikation zwischen VPN-Client und VPN-Server (Anf3) impliziert, dass in der gesuchten Software Algorithmen zum Verschlüsseln und Signieren von Daten verwendet werden. Damit die Implementation dieser kryptografischen Algorithmen von jedermann nachvollzogen werden kann, soll ausschließlich quelloffene Software für den VPN-Dienst verwendet werden. Dadurch sind unabhängige Untersuchungen der Software auf mögliche Sicherheitslücken von jedermann jederzeit möglich. Dadurch erhöht sich die Wahrscheinlichkeit bestehende Sicherheitslücken zu finden, und die Implementation der kryptografischen Algorithmen kann auf Korrektheit überprüft werden.

Außerdem kann vermutet werden, dass gefundene und behobene Sicherheitslücken besser kommuniziert werden, da alle Änderungen am Quellcode der Software öffentlich sichtbar sind. Das wirkt sich auch auf Reaktionszeiten der Software-Distributoren aus: Entsprechend aktualisierte Softwarepakete stehen in der Regel zeitnah bereit und können sofort installiert werden.

Weiterhin soll die gesuchte Software IPv4 und IPv6 unterstützen (Anf1), die Routingtabellen der VPN-Clients (Anf2) anpassen können und in Bezug auf Protokollierung (Anf7) konfigurierbar sein.

5.1 Kandidaten für VPN-Serversoftware

Ausgangspunkt für die Suche nach passender VPN-Software ist die Wahl der Serverkomponente: Sie soll quelloffen sein und auf einem Server mit Debian 9 eingesetzt werden können. Deshalb sind die Debian-Paketquellen die erste Anlaufstelle für die Suche. Durch die Nutzung der Paketquellen ist das Installieren von Sicherheitsaktualisierungen über den Debian-Paketmanager möglich. Arbeitsschritte, wie das Anpassen und Kompilieren des Quellcodes, sowie Tests und das Packen der Software, werden von den Verwaltern der Debian-Pakete ausgeführt. Die Authentizität der Pakete wird anhand von GPG-Signaturen durch den Paketmanager vor der Installation überprüft [HM15, Kapitel 6.5].

Um den Wartungsaufwand des VPN-Servers zu reduzieren, kann die Installation von Updates durch den Debian-Paketmanager automatisiert werden [HM15, Kapitel 6.7 und 6.8]. Somit muss der Systemadministrator lediglich Upgrades zur nächsthöheren Debian-Version durchführen, da dabei Anpassungen an der Systemkonfiguration notwendig werden.

Im Folgenden werden mögliche Software-Kandidaten aus den Debian-Paketquellen vorgestellt.

5.1.1 Strongswan

Strongswan[1] ist eine quelloffene, modular aufgebaute Software, die unter den in Anf5 und Anf6 genannten Betriebssystemen läuft. Sie kann in Kombination mit IPsec-fähigen Betriebssystem-Kerneln geschützte Verbindungen zwischen zwei oder mehr Computern einrichten. Strongswan wird unter der GPLv2-Lizenz verbreitet[2].

IPsec ist ein Internetstandard, der kryptografische Sicherheit für IPv4 und IPv6 (sowie darüber übertragenen Daten) bieten soll. Das beinhaltet unter anderem Vertraulichkeit übertragener Daten durch den Einsatz von Verschlüsselung, Authentisierung von Paketen durch Prüfung von Prüfsummen, und Schutz vor Replay-Angriffen [Vergleich KS05, Kapitel 2.1]. Mit IPsec können Richtlinien definiert werden, ob und wie Datenverkehr von einem Host zu einem anderen Host geschützt werden soll. Zum Schutz des Datenverkehrs können die Protokolle AH und ESP benutzt werden, die in den folgenden Absätzen kurz vorgestellt werden.

[1] `https://wiki.strongswan.org/projects/strongswan/wiki/IntroductionTostrongSwan`, zuletzt abgerufen am 18.07.2018

[2] `https://wiki.strongswan.org/projects/strongswan/wiki/Contributions`, zuletzt abgerufen am 04.09.2018

Das Protokoll *IP Authentication Header* (AH) ist in [Ken05a] definiert und ermöglicht den Versand von authentisierbaren Paketen an eine Gegenstelle. Vor dem Versand wird über den Inhalt der beim Transport unveränderlichen Felder des IP-Pakets eine Prüfsumme gebildet. Die Gegenstelle kann die Prüfsumme des empfangenen Pakets berechnen und mit der im Paket enthaltenen Prüfsumme abgleichen [Siehe Ken05a, Kapitel 3.3.3]. Die Funktion zur Berechnung der Prüfsumme wird nicht explizit definiert und kann daher anhand der zur Zeit aktuellen Vorgaben [definiert in WMM+17] gewählt werden. Abhängig von der gewählten Funktion fließen bereits im Vorfeld ausgehandelte, gemeinsame Geheimnisse oder Signaturalgorithmen in die Berechnung der Prüfsumme ein, sodass eine korrekte Prüfsumme ein Paket authentisiert. Eine Verschlüsselung der Paketinhalte ist im AH-Protokoll nicht vorgesehen.

Das Protokoll *IP Encapsulating Security Payload* (ESP) ist in [Ken05b] definiert und ermöglicht den Versand von Paketen mit vertraulichen Inhalten an eine Gegenstelle. Ähnlich wie bei dem AH-Protokoll ist auch im ESP-Protokoll die Authentisierung von Paketen mit einer Prüfsumme vorgesehen [Siehe Ken05b, Kapitel 2.8]. Aktuell empfohlene Algorithmen zum Berechnen der Prüfsumme oder zum Verschlüsseln der Paketinhalte sind in [WMM+17] aufgeführt. Durch die Verschlüsselung der Paketinhalte vor dem Transport wird die Vertraulichkeit der übertragenen Inhalte gewährleistet. Als Schlüssel wird ein zwischen Sender und Empfänger im Vorfeld ausgehandeltes gemeinsames Geheimnis verwendet. Auch die verwendeten Verschlüsselungsalgorithmen müssen zwischen Sender und Empfänger ausgehandelt werden.

IPsec kann im Transportmodus und im Tunnelmodus betrieben werden. Im Transportmodus werden die Inhalte von IP-Paketen in AH- beziehungsweise ESP-Pakete gekapselt. Da die Sender- und Empfängeradressen der IP-Pakete hierbei nicht verändert werden, kann dieser Modus nur für direkte Ende-zu-Ende-Kommunikation der beteiligten Sender und Empfänger verwendet werden. Der Schutz von zu routendem Datenverkehr zwischen zwei Routern ist damit nicht möglich.

Im Tunnelmodus werden die IP-Paketen selbst in AH- beziehungsweise ESP-Pakete gekapselt. Im Anschluss werden die AH- beziehungsweise ESP-Pakete dann in neue IP-Pakete gekapselt, deren Sender- und Empfängeradressen sich von denen des inneren IP-Paketes unterscheiden dürfen. Somit ist der Tunnelmodus im Prinzip für die Umsetzung eines VPN geeignet.

Die Protokolle AH und ESP definieren selbst kein Verfahren zum Aushandeln von verwendeten Prüfsummenfunktionen, kryptografischen Algorithmen oder allgemeiner Konfigurationsparameter. Auch der Austausch gemeinsamer Geheimnisse beziehungsweise Schlüsselmaterial wird nicht definiert. Dafür wird das *Internet Key Exchange Protocol* (IKE) in Version 2 (IKEv2) verwendet. An dieser Stelle kommt Strongswan als IKEv2[3]-Dienst zum Einsatz.

[3] Internet Key Exchange Protokoll Version 2, definiert in [KHN+14]

Strongswan kann über IKEv2 authentisiert und verschlüsselt mit IKEv2-Gegenstellen kommunizieren. Dabei werden mit der Gegenstelle Schlüssel- und Konfigurationsparameter ausgehandelt beziehungsweise ausgetauscht, anhand derer Strongswan IPsec-Verbindungen im Kernel des Host-Betriebssystems konfigurieren kann. Die Verarbeitung des durch IPsec geschützten Datenverkehrs über die Protokolle AH oder ESP wird jedoch direkt im IPsec-Stack des Kernels abgewickelt. Dadurch ist der Einsatz von IPsec für lokal ausgeführte Programme transparent.

5.1.2 OpenVPN

OpenVPN ist eine quelloffene Software zur Einrichtung von VPNs in Peer-to-Peer- oder Client-Server-Architektur [Yon18, Abschnitt „Server Mode“]. Sie läuft unter den in Anf5 und Anf6 genannten Betriebssystemen und wird unter der GPLv2-Lizenz verbreitet. OpenVPN verwendet die in der Bibliothek OpenSSL enthaltenen Implementationen von kryptografischen Algorithmen [Yon18, Abschnitt „Introduction“].

OpenVPN unterstützt IPv4 und IPv6 sowohl als Transportprotokoll innerhalb eines VPN, als auch als Hüllenprotokoll zur Kommunikation zwischen OpenVPN-Prozessen. Die Kommunikation zwischen zwei OpenVPN-Prozessen erfolgt durch den Versand von UDP-Paketen. Wird UDP durch eine Firewall blockiert, kann man auf TCP ausweichen[Yon18, Option `-proto`]. OpenVPN läuft im Benutzerkontext und unterstützt nach dem Programmstart den Wechsel in einen nicht-privilegierten Benutzerkontext [Siehe Yon18, Option `-user`], um im Fall eines erfolgreichen Angriffs den potentiellen Schaden zu begrenzen.

Für die Bereitstellung einer virtuellen Netzwerkkarte als Schnittstelle zum VPN wird ein TUN/TAP-Treiber verwendet. Über die Routingtabellen auf Client und Server wird bestimmt, welcher Datenverkehr durch das VPN geleitet wird. Für lokal ausgeführte Programme entspricht der Einsatz von OpenVPN der Installation einer zusätzlichen Netzwerkkarte im lokalen Rechner.

Die Kommunikation zwischen OpenVPN-Client und -Server enthält zwei Kanäle: Einen Datenkanal und einen Kontrollkanal [Yon18, Abschnitt „TLS Mode Options“]. Der Kontrollkanal wird zur Kommunikation zwischen zwei OpenVPN-Prozessen verwendet. Über ihn werden Konfigurationsparameter vom Server an den Client übertragen [Yon18, Option `-pull`]. Gleichzeitig senden sich Client und Server „keepalive“-Nachrichten durch den Kontrollkanal, um die VPN-Sitzung bei Inaktivität auf dem Datenkanal aufrecht zu halten [Yon18, Option `-keepalive`].

Im „TLS Mode“ wird über den Kontrollkanal eine TLS-Sitzung aufgebaut, in der Chiffren und Schlüssel ausgetauscht werden, mit denen der Datenkanal geschützt werden soll. Mit dem Aufbau dieser TLS-Sitzung ist die Authentisierung von Client und Server mit X.509-Public-Key-Zertifikaten[4] möglich.

[4] X.509-Public-Key-Zertifikate werden oft als „SSL-Zertifikate“ bezeichnet

Zusätzlich kann *Perfect Forward Secrecy* (PFS) durch Einsatz des Diffie-Hellman-Verfahren für den Schlüsselaustausch erreicht werden [Vergleich Yon18, Abschnitt „TLS Mode Options“]. Außerdem können die zur Verschlüsselung des Datenkanals ausgehandelten Schlüssel während der Sitzung mehrfach erneuert werden.

Im „Static Key Mode“ wird beiden Prozessen beim Start ein zuvor geteiltes gemeinsames Geheimnis als Parameter gegeben, mit dem der Datenkanal zwischen den beiden Prozessen symmetrisch verschlüsselt wird [Yon18, Option `-secret`].

5.1.3 Wireguard

Wireguard ist ein Softwareprojekt, mit dem ein geschützter Netzwerktunnel zwischen zwei Netzwerkteilnehmern aufgebaut werden kann. Aktuell (am 12.10.2018) befindet sich Wireguard noch in Entwicklung, der Quellcode wird noch als experimentell eingestuft[5]. Wireguard wurde als Kernel-Modul für Linux entwickelt, welches einen Umfang von weniger als 4000 Zeilen Code hat [Don17, Abschnitt VII], wodurch Audits und Reviews erleichtert werden [Don17, Abschnitt VII]. Im Vergleich zu OpenVPN und IPsec arbeitet Wireguard als Kernel-Modul schneller und effizienter [Don17, Abschnitt VIII]. Für MacOS, FreeBSD und Windows (Unterstützung für Windows ist noch nicht fertiggestellt) wird mit `wireguard-go` eine in Go geschriebene Wireguard-Implementation entwickelt. Eine auf Rust aufbauende Implementation wird unter der Bezeichnung `wireguard-rs` entwickelt[6].

Wireguard arbeitet nur auf OSI-Schicht 3 und unterstützt IPv4 und IPv6 sowohl zur Kommunikation zwischen zwei Netzwerkteilnehmern, als auch für den Transport durch den Netzwerktunnel [Don17, Abschnitt I]. Die durch Wireguard geschützten IP-Pakete werden in UDP-Paketen gekapselt zwischen den Netzwerkteilnehmern übertragen [Don17, Abschnitt III]. Durch das periodische Senden von „Keepalive“-Nachrichten [Don17, Abschnitt VI, Absatz E] können Wireguard-Sitzungen, auch hinter *Network Address Translation* (NAT) [Don17, Abschnitt II], aufrecht gehalten werden.

Während OpenVPN und IPsec die Konfiguration der zu verwendenden kryptografischen Algorithmen und Parameter erlauben, gibt Wireguard diese fest vor. Sollten Schwachstellen in der verwendeten Kryptografie vorliegen, so müssen alle Wireguard-Endpunkte mit Sicherheitsaktualisierungen versorgt werden [Don17, Abschnitt I]. Durch diesen Schritt wird Wireguard weniger komplex; Verwundbarkeiten, wie sie bei SSL/TLS häufig aufgetreten sind, werden vermieden [Don17, Abschnitt I].

Ein Netzwerkteilnehmer wird durch seinen öffentlichen Schlüssel eindeutig identifiziert. Dieser öffentliche Schlüssel ist ein Punkt auf der elliptischen Kurve `Curve25519`, welcher mit 32 Bytes beschrieben wird [Don17, Abschnitt I].

[5] Vergleich `https://www.wireguard.com/install/`
[6] Siehe `https://www.wireguard.com/xplatform/`

Der Austausch von Sitzungsschlüsseln wird durch das Diffie-Hellman-Verfahren auf elliptischen Kurven durchgeführt, dessen Ergebnisse mit einer Schlüsselableitungsfunktion auf Basis eines HMAC (HKDF) gestreckt werden [Don17, Abschnitt I]. Für die Verschlüsselung des VPN-Verkehrs wird die in [NL15] konstruierte *Authenticated Encryption with Associated Data* (AEAD)-Chiffre aus ChaCha20 und Poly1205 verwendet [Don17, Abschnitt I]. Als Hashfunktion kommt BLAKE2s [ANWOW13] zum Einsatz [Don17, Abschnitt I].

Bevor eine Wireguard-Sitzung zwischen zwei Teilnehmern konfiguriert werden kann, müssen die beiden Teilnehmer ihren öffentlichen Schlüssel austauschen. Dieser Austausch ist mit Absicht nicht durch Wireguard spezifiziert und wird, wie auch bei OpenSSH, den Teilnehmern selbst überlassen [Don17, Abschnitt I]. Um eine Sitzung mit einem anderen Teilnehmer zu konfigurieren, muss zunächst die eigene Wireguard-Netzwerkschnittstelle konfiguriert werden [Vergleich Don17, Abschnitt II]. Dafür wird das Schlüsselpaar des Teilnehmers benötigt, sowie die Angabe, über welchen UDP-Port Wireguard Pakete empfangen und verschicken soll. Als nächstes kann die Liste der bekannten Sitzungsteilnehmer mit einem neuen Eintrag versehen werden. Der andere Sitzungsteilnehmer wird durch seinen öffentlichen Schlüssel identifiziert. In einer Liste werden IP-Adressen hinterlegt, die der anderen Sitzungsteilnehmer innerhalb des Netzwerktunnels verwenden darf. Optional ist eine Angabe von IP-Adresse und UDP-Port des anderen Sitzungsteilnehmers möglich, damit eine Sitzung direkt aufgebaut werden kann. Wird diese Angabe weggelassen, lernt Wireguard diese Informationen aus den empfangenen Paketen des anderen Teilnehmers automatisch. Durch diese Flexibilität können Teilnehmer ihre IP-Adresse wechseln, ohne dass die Wireguard-Sitzung dadurch abgebrochen wird. Mehr Details zur Benutzung von Wireguard unter Linux können anhand eines Beispiels in [Don17, Abschnitt IV] eingesehen werden.

Seit Veröffentlichung des Wireguard-Papers [Don17] im Frühling 2017 ist im Juli 2018 eine kryptografische Analyse des Wireguard-Protokolls [DP18] erschienen. In der Analyse wird auf den Schlüsselaustausch des Protokolls via *One Roundtrip Handshake* (1-RTT) eingegangen. Aufgrund einer direkten Abhängigkeit zwischen dem Schlüsselaustausch-Protokoll und der ersten Nachricht des darauf folgenden Datentransport-Protokolls, welche den Sitzungsinitiator gegenüber dem anderen Sitzungsteilnehmer authentisiert, sei die Sicherheit des Schlüsselaustauschs nicht beweisbar. Technisch betrachtet sei die Phase des Schlüsselaustauschs dadurch kein 1-RTT, da der andere Teilnehmer erst dann mit der Datenübertragung beginnen könne, nachdem der Sitzungsinitiator sich mit der ersten Datentransport-Nachricht authentisiert habe [DP18, Abschnitt 1, „Security of WireGuard“].

Es wird eine Analysemethode gewählt, die eine Trennung von Handshake-Protokoll und Datentransport-Protokolls verlangt. Diese Trennung wird durch minimale Veränderungen am Wireguard-Protokoll herbeigeführt [DP18, Abschnitt 1, „Our Contributions“]. Im Ergebnis sind durch die Analyse keine Schwachstellen am Wireguard-Protokoll gefunden worden. Allerdings wird gezeigt, dass eine saubere Trennung zwischen Schlüsselaustausch und Datentransport im Wireguard-Protokoll benötigt wird.

Zusätzlich werden Aspekte des Wireguard-Protokolls aufgezeigt, die nicht Teil der Analyse waren, und es wird empfohlen, Wireguard weiteren Analysen zu unterziehen [DP18, Abschnitt 6].

5.2 Vergleich der Softwarekandidaten

Die zuvor vorgestellten VPN-Softwarelösungen OpenVPN und Strongswan werden nun in den folgenden Kategorien miteinander verglichen, um im Anschluss die Software zu ermitteln, mit der das Vorhaben dieser Arbeit umgesetzt wird. Aufgrund des experimentellen Status von Wireguard wird es für den Aufbau eines produktiv eingesetzten VPN-Servers nicht berücksichtigt.

Kommunikationsprotokolle: OpenVPN kommuniziert mit einem eigenen Protokoll über UDP auf Port 1194. Eine Trennung zwischen Kontrollnachrichten und Datenübertragung erfolgt innerhalb der Software.

Strongswan kommuniziert mit kompatiblen Gegenstellen über das IKEv2-Protokoll, welches über UDP auf Port 500 (bei stattfindender *Network Address Translation* (NAT) auf Port 4500) übertragen wird. Der durch IPsec geschützte Datenverkehr lässt sich daran erkennen, dass in den übertragenen IPv4- beziehungsweise IPv6-Paketen das Protokoll AH oder ESP eingetragen ist.

Für die Freigabe von IPsec-Datenverkehr in einer Firewall sind somit mehrere Regeln notwendig, während die Freigabe von OpenVPN-Verkehr über UDP-Port 1194 übersichtlicher ausfällt.

Benutzerfreundlichkeit: OpenVPN steht für Linux/Unix und Windows bereits kompiliert zur Verfügung. Für Mac OS wird OpenVPN durch Tunnelblick kompiliert zur Verfügung gestellt. Da OpenVPN auf allen Clientbetriebssystemen verfügbar ist, gelten die selben Prinzipien für die Clientkonfiguration auf allen Betriebssystemen. Wird OpenVPN mit X.509-Public-Key-Zertifikaten verwendet, so reicht es aus, diese als Datei im PEM-Format bereitzustellen. Je nach Plattform stehen unterschiedliche grafische Oberflächen zur Verfügung, welche die Benutzung von OpenVPN zusätzlich erleichtern können: Für Linux kann NetworkManager verwendet werden, unter Windows kann OpenVPN-Gui benutzt werden, und unter Mac OS steht Tunnelblick als GUI zur Verfügung.

Strongswan hingegen muss für die Benutzung unter Windows zuvor vom Anwender mit Hilfe einer MinGW[7]-W64-Umgebung kompiliert werden [str18b]. Die Mac-Version kann über `brew`[8] bezogen werden [str18a]. Für Linux/Unix stehen kompilierte Versionen zur Verfügung.

[7] Minimalist GNU for Windows, siehe `http://www.mingw.org`
[8] „The missing package manager for macOS“, siehe `https://brew.sh`

Wurde Strongswan auf den jeweiligen Clientbetriebssystemen erfolgreich installiert, so läuft die Konfiguration wie bei OpenVPN auf allen Betriebssystemen nach den selben Prinzipien ab. Da unter Mac OS und Windows bereits IPsec-Funktionalität durch das Betriebssystem zur Verfügung gestellt wird, wird seitens Strongswan keine grafische Oberfläche entwickelt. Unter Linux kann NetworkManager als grafische Oberfläche für Strongswan verwendet werden.

In der Kategorie Benutzerfreundlichkeit schneidet OpenVPN durch die Bereitstellung von vorkompilierter Software und die Verfügbarkeit grafischer Oberflächen besser als Strongswan ab.

Verfügbarkeit des Quellcode: Der Quellcode von OpenVPN ist inklusive der durch OpenVPN verwendeten Bibliotheken, wie zum Beispiel OpenSSL, öffentlich verfügbar. Auch der Quellcode von Strongswan ist einschließlich des Quellcode aller durch Strongswan eingesetzten Bibliotheken öffentlich verfügbar.

Die Umsetzung eines VPN auf Basis von IPsec impliziert, dass Code aus dem verwendeten Betriebssystemkernel den durch IPsec geschützten Datenverkehr kryptografisch verarbeitet. Der Kernel des eingesetzten Betriebssystems ist damit aktiv an der Umsetzung des VPNs beteiligt und sollte somit ebenfalls die an die VPN-Software gestellten Kriterien erfüllen. Die Verfügbarkeit des Quellcodes des Kernels hängt vom verwendeten Betriebssystem ab und ist deshalb nicht in jedem Fall garantiert. Damit wird die Forderung nach ausschließlich quelloffener VPN-Software von Strongswan nicht erfüllt.

Plattformabhängigkeit: Sowohl OpenVPN als auch Strongswan sind für alle in Anf6 und Anf5 genannten Betriebssysteme verfügbar. Sollten Sicherheitslücken innerhalb von beiden Softwareprojekten bekannt werden, können diese in vergleichbarer Zeit geschlossen werden und auf Client- und Serverrechnern gleichermaßen installiert werden.

Bei einem auf OpenVPN gestützten VPN ist es möglich, TLS-Chiffren, Hashfunktionen und Verschlüsselungsalgorithmen vorzugeben, die unabhängig vom eingesetzten Betriebssystem durch OpenVPN verwendet werden.

Für ein VPN mit Strongswan auf Basis von IPsec sieht die Situation anders aus: Wie bereits in Kapitel 5.1.1 erläutert wurde, kann Strongswan nur in Kombination mit dem Betriebssystem-Kernel ein IPsec-VPN aufbauen. Die in Anf6 und Anf5 genannten Betriebssysteme enthalten unterschiedliche Kernel, die jeweils eine individuelle Implementierung von IPsec enthalten. Aus diesem Grund kann nicht garantiert werden, dass die verschiedenen Kernel die aktuellen Empfehlungen [Aktuell aus WMM+17] bezüglich kryptografischer Algorithmen zeitnah gemeinsam unterstützen. Ebenso sind unterschiedliche Reaktionszeiten auf auftretende Sicherheitslücken zu erwarten, die einen oder mehrere Kernel betreffen.

Insgesamt kann für VPNs auf Basis von IPsec nicht garantiert werden, dass ein einheitliches Sicherheitsniveau über die verschiedenen Betriebssysteme aufrecht erhalten werden kann. Erschwerend kommt hinzu, dass Strongswan auf Windows keine Unterstützung für die Einrichtung virtueller IP-Adressen bietet, welche bei IPsec-VPNs im Tunnelmodus zum Einsatz kommen [str18b, Abschnitt „Important notes“].

Komplexität: Im Betrieb besteht OpenVPN aus einem Prozess, der über eine Konfigurationsdatei alle für den Betrieb notwendigen Informationen erhält. Eine Konfigurationsdatei für OpenVPN besteht aus einer Liste von Optionen, die das OpenVPN-Programm als Argumente akzeptiert. Alle Optionen werden in der Manpage von OpenVPN (nach meiner Meinung) ausführlich und leicht verständlich erklärt, und sollten für Einsteiger mit Grundkenntnissen in Bezug auf Netzwerke und Linux kein Hindernis darstellen.

Strongswan ist modular aus einer Sammlung von Programmen aufgebaut, die abhängig von einer Sammlung von Konfigurationsdateien (vorgegebene Beispielszenarien enthalten typischerweise Inhalte für vier verschiedene Konfigurationsdateien) zum Einsatz kommen. Zusätzlich ist beim Einsatz von Strongswan immer der Kernel des Betriebssystems involviert, da die Verarbeitung des IPsec-Datenverkehrs im Kernel durchgeführt wird. Die Konfigurationsdateien bestehen aus Zeilen mit Parametern. Zusätzlich enthalten die Konfigurationsdateien durch Schweifklammern abgegrenzte Abschnitte, in denen Konfigurationen für bestimmte Kontexte hinterlegt werden. Durch diese Konstrukte ist es beispielsweise möglich, Standardwerte für eine logische IPsec-Verbindung zu definieren, diese später als Grundlage für eine andere IPsec-Verbindung zu verwenden und bei Bedarf Teile der zuvor definierten Parameter zu überschreiben. Die verschiedenen Konfigurationsdateien werden in ihren eigenen Manpages detailliert und technisch erläutert, wobei das zusätzlich notwendige Fachwissen für IPsec für einen Einsteiger (nach meiner Ansicht) ein unbequemes Hindernis darstellen kann.

5.3 Zusammenfassung und Auswahl der VPN-Software

In diesem Abschnitt wurden OpenVPN und Strongswan als Lösungen für die Umsetzung eines VPN gegenübergestellt. Dabei hat sich gezeigt, dass OpenVPN in den Kategorien Plattformabhängigkeit, Verfügbarkeit des Quellcode, Komplexität und Benutzerfreundlichkeit im Vergleich zu Strongswan besser abschneidet. Dadurch ist OpenVPN besser für die Umsetzung eines VPN-Dienst geeignet, der die in Kapitel 2 genannten Anforderungen erfüllt. Somit wird OpenVPN als VPN-Software zur Umsetzung des VPN-Dienst ausgewählt.

5.4 Auswahl von Software zur Benutzerverwaltung

Nach der Wahl der VPN-Software soll in diesem Abschnitt diskutiert werden, mit welcher Software eine PKI umgesetzt werden soll, über welche die VPN-Benutzer mit Zertifikaten ausgestattet werden.

OpenVPN benötigt zur Laufzeit die Bibliothek OpenSSL. OpenSSL stellt seine Funktionen auch als Kommandozeilenwerkzeug zur Verfügung, mit denen alle Basisfunktionen einer Zertifizierungsstelle, wie zum Beispiel die Erzeugung von Schlüsselpaaren und Zertifikatsanträgen, sowie das Ausstellen von Zertifikaten auf Basis von Zertifikatsanträgen, möglich ist. Man kann mit OpenSSL eine *Zertifizierungsstelle* (CA) betreiben. Diese Vorgehensweise verlangt jedoch Fachwissen und Sorgfalt von den Betreibern der CA und eignet sich aufgrund des hohen Aufwands nur für die Verwaltung weniger Benutzer oder zu Zwecken der Lehre.

Mit der Installation von OpenVPN wird die Installation der Software „EasyRSA“ durch den Debian-Paketmanager empfohlen, welches ebenfalls von den OpenVPN-Entwicklern geschrieben wurde. Dieses Paket enthält eine Sammlung von Shellskripten, welche die Funktionalität von OpenSSL kapseln, um damit einen erleichterten Betrieb einer CA zu ermöglichen. Die Skripte abstrahieren allgemeine Aufgaben für den Betrieb einer CA. EasyRSA enthält Funktionen zur Erzeugung eines Wurzelzertifikats, zum Erzeugen von Zertifikatsanträgen, für das Ausstellen von Client- oder Serverzertifikaten auf Basis von Zertifikatsanträgen und für das Erzeugen einer *Certificate Revocation List*(CRL). Dabei ist es durch Anpassen von Konfigurationsdateien möglich, eine CA ganz nach den eigenen Bedürfnissen zu betreiben und Einfluss auf Parameter wie zum Beispiel Schlüssellängen, Laufzeiten von Zertifikaten oder verwendete kryptografische Algorithmen zu nehmen.

Aufgrund der Vorteile von EasyRSA in Bezug auf einfachere Konfiguration und erleichterte Bedienung für CA-Betreiber und Benutzer im Vergleich zur manuellen Umsetzung einer CA mit OpenSSL alleine, soll EasyRSA für den Aufbau der Zertifizierungsstelle verwendet werden. Diese Entscheidung wird untermauert von der Tatsache, dass sowohl EasyRSA als auch OpenVPN von den selben Entwicklern weiterentwickelt wird, sodass Kompatibilität zwischen den beiden Softwareprojekten zu erwarten ist. Die Notwendigkeit zum selbstständigen Entwickeln von Skripten zum Erreichen eines ähnlichen Funktionsumfangs ergibt sich somit nicht.

Das über Debian 9 beziehbare Paket enthält EasyRSA in Version 2.3.x. Die heute (01.10.2018) über GitHub[9] beziehbare Version von EasyRSA trägt die Nummer 3.0.5. EasyRSA wurde in Version 3 von Grund auf neu geschrieben und verfügt über ein im Vergleich zu EasyRSA Version 2 vereinfachtes Benutzerinterface, welches nun von einem einzigen Kommandozeilenbefehl zur Verfügung gestellt wird.

[9] `https://github.com/OpenVPN/easy-rsa/releases/tag/v3.0.5`

Zusätzlich hinzugekommen sind neue Features, wie etwa die Unterstützung des *Elliptische-Kurven-Kryptosystems* (EKK), Unterstützung von UTF-8, oder die Verwendung von AES256 zum verschlüsselten Speichern von privaten Schlüsseln[10].

Die Installation von EasyRSA wird durch das Kopieren sämtlicher Dateien von EasyRSA in ein neues Verzeichnis durchgeführt. Eine auf diese Weise eingerichtete CA kann aus diesem Grund nicht durch den Debian-Paketmanager mit Updates versorgt werden. Aufgrund des Installationsprozess und den in EasyRSA Version 3.0.5 enthaltenen Vorteilen wird entschieden, dass EasyRSA Version 3.0.5 zum Aufbau der CA für den VPN-Dienst eingesetzt wird.

Um die öffentlichen Daten der CA über HTTP zur Verfügung zu stellen, kommt der Apache httpd als Webserver zum Einsatz, weil dieser unter Debian 9 über das Paket `apache2` leicht installiert werden kann.

[10]Vergleich `https://github.com/OpenVPN/easy-rsa/blob/v3.0.5/ChangeLog`

6 Planung der Installation

Nach Abschluss der Konzeptphase kann die Installation des VPN-Dienstes und der dazugehörigen PKI konkret geplant werden. Abbildung 6.1 zeigt eine Gesamtübersicht über die geplante Installation des VPN-Dienstes und der dazugehörigen CA.

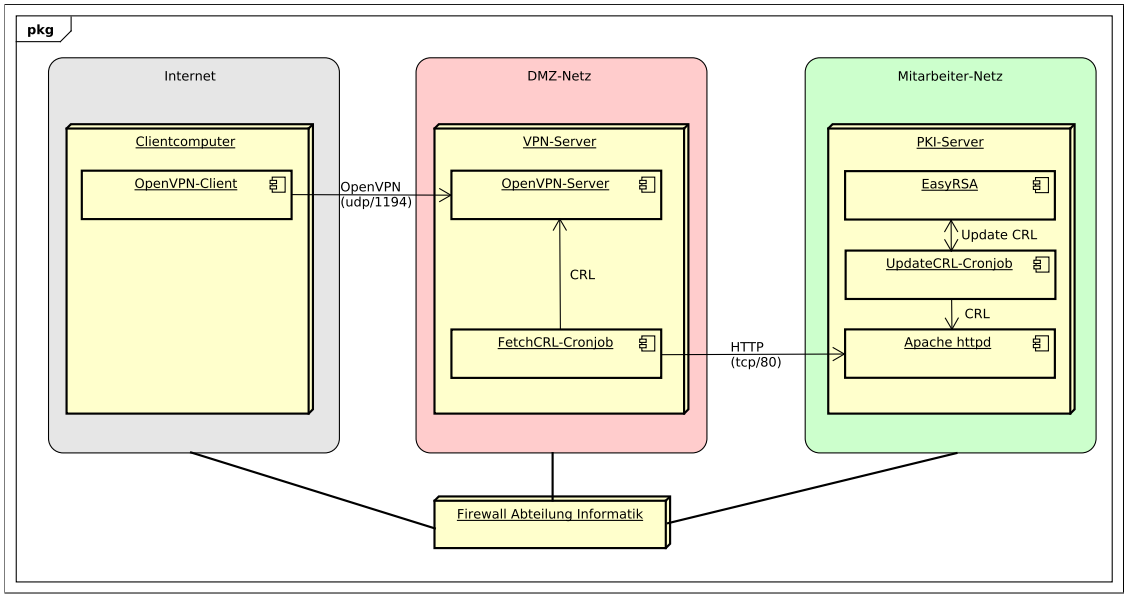

Abbildung 6.1: Gesamtübersicht der geplanten VPN-Dienst-Installation

Im Vergleich mit Abbildung 4.1 aus Kapitel 4.3 wurde OpenVPN als Software für die Client- und Serverkomponente des VPN gewählt.

Die PKI wird mit Hilfe der Software EasyRSA eingerichtet werden. Für die Bereitstellung der öffentlichen Daten der PKI wird ein Apache httpd als Webserver verwendet. Damit die CRL der PKI aktualisiert wird, wird ein Cronjob auf dem CA-Server eingerichtet, der diese Aufgabe übernimmt, und die aktuelle CRL anschließend über den Webserver veröffentlicht. Auf dem VPN-Server kommt ein weiterer Cronjob zum Einsatz, um die aktuelle CRL vom Webserver auf dem CA-Server zu beschaffen und in die Konfiguration des OpenVPN-Servers zu integrieren.

6.1 OpenVPN-Server

In diesem Abschnitt werden die in Kapitel 4 bereits beschlossenen Konzepte für den VPN-Dienst auf die gewählte VPN-Software OpenVPN zugeschnitten. Die durch OpenVPN gegebenen Funktionen und Einschränkungen werden hierbei berücksichtigt, um später eine funktionsfähige Konfiguration für OpenVPN aus dieser Planung abzuleiten.

Manuelles Failover: Um die Verfügbarkeit des VPN-Dienstes im Fall eines Hardwareausfalls zu erhöhen, soll ein manuelles Failover auf einen zweiten, identisch konfigurierten Server möglich sein. Das IT-Team vergibt für diesen Zweck verschiedene IPv4- und IPv6-Adressen für den Dienst und den Host. Dadurch kann ein manuelles Failover durch die Übertragung der IP-Adresse des Dienstes von dem defekten Server auf einen weiteren, identisch konfigurierten Server durchgeführt werden. Im Beispiel sieht ein Failover von Server A auf Server B wie folgt aus: Bei einem Defekt von Server A werden zunächst die IP-Dienstadressen auf A deaktiviert. Anschließend werden die IP-Dienstadressen auf B aktiviert.

Sollte dieses Verfügbarkeitsniveau nicht mehr ausreichen, so ist ein Parallelbetrieb von zwei VPN-Servern möglich. Dafür muss lediglich ein weiteres IPv6-Netz für VPN-Clients bereitgestellt werden und neue IP-Dienstadressen für einen weiteren VPN-Dienst vergeben werden. In der OpenVPN-Clientkonfiguration können beide VPN-Server eingetragen werden, sodass die Clients bei Ausfall eines Servers automatisch eine Sitzung mit dem zweiten Server aufbauen können. Damit ist ein Parallelbetrieb von zwei OpenVPN-Diensten möglich, die sich in der Konfiguration nur durch ihre IP-Dienstadresse und das verwendete IPv6-Netz für VPN-Clients unterscheiden.

VPN-Transportprotokoll: OpenVPN bietet UDP und TCP als mögliche Transportprotokolle für den VPN-Datenverkehr an. Für diesen VPN-Dienst wird das UDP-Protokoll gewählt, weil das TCP-Protokoll in Kombination mit stark ausgelasteten Netzwerken oder beim Transport von TCP-Paketen durch den VPN-Tunnel Performanceeinbußen verursachen kann [Yon18, Option `-proto`]. Bei Interesse kann eine detaillierte Analyse der Probleme der TCP-in-TCP-Situation in [HOI+05] nachgelesen werden.

OSI-Schicht des VPN-Tunnels: OpenVPN unterstützt den Aufbau eines VPN-Tunnels auf OSI-Schicht 2 und OSI-Schicht 3. Im Prinzip könnten beide Tunnelvarianten für den Anwendungsfall dieser Arbeit verwendet werden. Da im Rahmen dieser Arbeit nur die Erreichbarkeit über IPv4 und IPv6 relevant ist, wird ein VPN-Tunnel auf OSI-Schicht 2 nicht benötigt. Zusätzlich würde ein VPN-Tunnel auf OSI-Schicht 2 mehr Bandbreite benötigen: IP-Pakete würden inklusive der dazugehörigen Ethernet-Frames übermittelt. Aus diesen Gründen fällt die Wahl auf VPN-Tunnel auf OSI-Schicht 3.

Kompression des VPN-Verkehrs: Auf die Kompression des VPN-Datenverkehrs wird aufgrund der in diesem Absatz aufgeführten Punkte verzichtet. Am 03.06.2018 wurden Hinweise in der Manpage von OpenVPN eingefügt[1], in denen vor der Verwendung von Kompression gewarnt wird. Diese Hinweise sind in der Manpage von OpenVPN in Version 2.4.6 noch nicht enthalten, da diese Version von den Entwicklern bereits am 19.04.2018[2] freigegeben wurde. Zusätzlich wird in der „Best Current Practice“ RFC 7525 zur Deaktivierung der Kompression von TLS geraten [Vergleich SHSA15, Kapitel 3.3].

Auf der DEFCON 26 wurde mit „VORACLE“ ein in diesem Kontext relevanter Angriff[3] auf OpenVPN vorgestellt[4], der auf aktivierter Kompression aufbaut.

Kryptografische Parameter: Wie in Kapitel 5.1.2 bereits erläutert wurde, unterscheidet OpenVPN bei der Kommunikation zwischen zwei OpenVPN-Prozessen zwischen Datenkanal für Nutzdaten und Kontrollkanal für Steuernachrichten. Während OpenVPN im TLS-Modus über den Kontrollkanal den TLS-Sitzungsaufbau durchführt und den Kontrollkanal anschließend über TLS absichert, wird für den Datenkanal eine separat konfigurierbare, symmetrische Chiffre verwendet. Das gemeinsame Geheimnis für den Schutz des Datenkanals wird dabei über den durch TLS geschützten Kontrollkanal ausgetauscht. Mit den in diesem Absatz definierten Parametern soll die Kommunikation zwischen Clients und Server unabhängig von den verwendeten Betriebssystemen einheitlich geschützt werden, da die Vertraulichkeit und Authentizität der übertragenen Daten nach Anforderung Anf3 eine Kernfunktion dieses VPN-Dienstes darstellt.

Die TLS-Kommunikation wird nach Empfehlung des BSI über TLS-Version 1.2 oder höher durchgeführt [BSI18b, Abschnitt 2.3]. Zur Absicherung des Kontrollkanals wird die TLS-Chiffre TLS-DHE-RSA-WITH-AES-256-GCM-SHA384 ausgewählt.

Diese Chiffre verwendet *Ephemeral Diffie Hellman* (DHE) für den Schlüsselaustausch [DR08, Abschnitt A.5], RSA als Signaturalgorithmus[5] zur Authentisierung [DR08, Abschnitt A.5], AES-256-GCM für die Verschlüsselung und SHA384 als Funktion für Pseudozufallszahlen für den *Keyed-Hash Message Authentication Code* (HMAC) [DR08, Kapitel 5]. Durch die Verwendung von DHE kann *Perfect Forward Secrecy* (PFS) gewährleistet werden [DR08, Abschnitt F.1.1.2].

Die gewählte TLS-Chiffre stammt aus der Liste der empfohlenen Chiffren in RFC 7525 [SHSA15, Abschnitt 4.2]. Laut BSI kann diese Chiffre durch Dienstanbieter optional unterstützt werden [BSI18b, Kapitel 3].

[1] `https://github.com/OpenVPN/openvpn/commit/6795a5f3d55f658fc1a28eb9f3b11d1217e3329c`
[2] `https://github.com/OpenVPN/openvpn/releases/tag/v2.4.6`
[3] Proof of Concept: `https://github.com/skepticfx/voracle`
[4] `https://media.defcon.org/DEF%20CON%2026/DEF%20CON%2026%20presentations/Nafeez/DEFCON-26-Nafeez-Compression-Oracle-attacks-on-VPN-Networks.pdf`
[5] RSA wird von der PKI des VPN-Dienstes für Zertifikate eingesetzt.

Die eng verwandte Chiffre TLS-ECDHE-RSA-WITH-AES-256-GCM-SHA384 setzt PFS mit ECDHE anstelle von DHE um, und wird laut BSI für Dienstanbieter empfohlen [BSI18b, Kapitel 3], wodurch sich unter der Voraussetzung, dass ECDHE und DHE einen vergleichbar sicheren Schlüsselaustausch ermöglichen, auch für die gewählte TLS-Chiffre eine Eignung für die langfristige Verwendung ableiten lässt. Diese Voraussetzung wird durch einen aktuellen Bericht[6] der ECRYPT-CSA[7] vom Februar 2018 untermauert [EC18, Kapitel 8.1].

Anmerkung: TLS-Chiffren mit Ephemeral-Diffie-Hellman-Verfahren auf Basis von elliptischen Kurven (ECDHE) können im Rahmen dieser Arbeit nicht verwendet werden. Grund dafür sind Kompatibilitätsprobleme zwischen OpenVPN und OpenSSL 1.1.x auf der Clientseite[8].

OpenVPN verwendet einen HMAC mit einem zuvor ausgetauschtem, gemeinsamen Geheimnis, um für Daten- und Kontrollkanal eingehende Datenpakete vor ihrer Verarbeitung zu authentisieren. Die im HMAC als Quelle für Pseudozufallszahlen verwendete Hashfunktion wird auf SHA-256 festgelegt, weil SHA-1, trotz seiner weiterhin bestehenden, theoretischen Eignung zur Verwendung in einem HMAC, laut BSI nicht mehr verwendet werden sollte [BSI18a, Abschnitt 1.4, Punkt 3].

Für die Verschlüsselung des Datenkanals wird auf Basis der zuvor gewählten TLS-Chiffre die Chiffre AES-256-GCM ausgewählt. Diese Chiffre ist eine *Authenticated Encryption with Associated Data* (AEAD)-Chiffre. Deshalb wird die Authentisierungsfunktion der Chiffre anstelle der zuvor beschriebenen HMAC-Funktion für den Datenkanal benutzt [Yon18, Option `-auth`].

Durch die Verwendung der selben Chiffre zur Verschlüsselung von Daten- und Kontrollkanal muss ein Angreifer für einen erfolgreichen Angriff - egal auf welchen Kanal - diese Chiffre brechen. Gelingt es einem Angreifer den Kontrollkanal zu dechiffrieren, so erhält er Kenntnis über die Schlüssel, mit denen der Datenkanal verschlüsselt wird. Gelingt es einem Angreifer den Datenkanal zu dechiffrieren, so erhält er Kenntnis über alle übertragenen Daten. Die Vertraulichkeit der über den VPN-Dienst übertragenen Daten hängt demnach gleichermaßen von der Verschlüsselung beider Kanäle ab. Der logische Schluss daraus ist die Verwendung der selben Chiffre zur Verschlüsselung beider Kanäle.

[6] Dieser Bericht löst einen vorangegangenen Bericht der ENISA [ENI14] von 2014 ab [EC18, Kapitel 1, „Executive Summary“].
[7] Siehe `http://www.ecrypt.eu.org/csa/`
[8] Siehe `https://community.openvpn.net/openvpn/ticket/963`

Statische Konfiguration kryptografischer Parameter: Damit alle VPN-Sitzungen die zuvor erläuterten, kryptografischen Parametern einhalten, werden diese Parameter in Client- und Serverkonfiguration hinterlegt. OpenVPN unterstützt die Aushandlung der verwendeten TLS-Chiffre für den Kontrollkanal, sowie der Chiffre für den Datenkanal. Die für den HMAC verwendete Pseudozufallsfunktion kann zum aktuellen Zeitpunkt (04.11.2018) jedoch nicht ausgehandelt werden. Müssen die kryptografischen Parameter ausgetauscht werden, so besteht die Gefahr, dass die in der Clientkonfiguration festgelegte Funktion für den HMAC vergessen wird.

Die flexible Aushandlung der Chiffren kann unsichere VPN-Sitzungen durch Fehlkonfigurationen auf Client und Server begünstigen, weshalb auf die Aushandlung von Chiffren explizit verzichtet wird. Eine sichere VPN-Sitzung kommt somit nur bei korrekter Konfiguration von Client und Server zustande. Potentielle Konfigurationsfehler bei Client oder Server verhindern den Sitzungsaufbau und werden dadurch sichtbar.

Sollten die hier gewählten, kryptografischen Parameter (einzeln oder insgesamt) nicht mehr als sicher gelten, so sollen **alle** kryptografischen Parameter nach aktuellem Stand der Technik und unter Berücksichtigung der verschiedenen Clientbetriebssysteme neu ausgewählt werden. Dabei müssen alle VPN-Benutzer über die Änderung der Parameter benachrichtigt werden. Die Aktualisierung der Vorlage für die OpenVPN-Clientkonfiguration ist ebenfalls notwendig. Der damit verbundene Arbeitsaufwand ist akzeptabel, weil alle kryptografischen Parameter für eine Laufzeit von 20 Jahren gewählt wurden, und die Wahrscheinlichkeit, dass die Parameter ausgetauscht werden müssen, als gering eingeschätzt wird.

Eine alternativ mögliche, kontinuierliche Aktualisierung der kryptografischen Parameter unter Vorgabe der zur Aushandlung verfügbaren Chiffren in der Serverkonfiguration erscheint zunächst trivial: Eine neue, moderne Chiffre kann in der Serverkonfiguration zur Liste der erlaubten Chiffren hinzugefügt werden und wird nach einem Neustart des VPN-Servers von allen kompatiblen VPN-Clients sofort verwendet.

Es müssen jedoch auch nicht mehr als modern geltende Chiffren aus der Liste der erlaubten Chiffren entfernt werden. Das kann dazu führen, dass einzelne VPN-Benutzer mit älterer VPN-Clientsoftware nach der Deaktivierung einer alten Chiffre plötzlich keine VPN-Sitzung mehr aufbauen können.

Da die Aktualisierung der VPN-Clientsoftware im Einzelfall nicht unmittelbar erfolgen kann, erfordert die Deaktivierung einer Chiffre vorher eine sorgfältige Koordination mit den VPN-Benutzern. Aufgrund des damit verbundenen Arbeitsaufwands wird erwartet, dass insgesamt mehr neue Chiffren aktiviert werden, als alte Chiffren deaktiviert werden. Die Prüfung, ob alle erlaubten Chiffren noch ausreichend sicher sind, wird dadurch aufwändiger; die Wahrscheinlichkeit, dass eine unsichere Chiffre übersehen wird, steigt. Für einen Angreifer steigt die Angriffsfläche, da ältere VPN-Clients durch die Wahl einer älteren Chiffre von modernen Clients unterschieden werden könnten.

Gleichzeitig können in einer VPN-Sitzung unterschiedliche Chiffren für den Schutz von Kontroll- und Datenkanal ausgehandelt werden, sodass die effektive Vertraulichkeit von der jeweils schwächeren Chiffre bestimmt wird.

Insgesamt bringt der Ansatz zur flexiblen Aushandlung und Anpassung von Chiffren mehr kontinuierlichen Arbeitsaufwand mit sich, während die statische Konfiguration der Chiffren für Klarheit in Bezug auf die verwendeten Parameter sorgt und im schlimmsten Fall, dessen Eintrittswahrscheinlichkeit als hinreichend gering eingeschätzt wird, eine große Konfigurationsanpassung für alle VPN-Benutzer erfordert. Deshalb wird die statische Konfiguration aller kryptografischen Parameter, und der damit verbundene Verzicht auf jegliche dynamische Aushandlung beim Sitzungsaufbau, für den VPN-Dienst gewählt.

6.2 PKI-Server

In diesem Abschnitt wird das Konzept zur Benutzerverwaltung aus Kapitel 4.2 unter Berücksichtigung der gewählten PKI-Software EasyRSA konkretisiert.

Vorgaben: Die folgenden Vorgaben für die CA wurden in Absprache mit dem Auftraggeber und Erstprüfer dieser Arbeit festgelegt: Das Wurzelzertifikat soll für 20 Jahre gültig sein. Ausgestellte Clientzertifikate sollen für Beschäftigte 5 Jahre lang gültig sein, für Studierende 2 Jahre lang. Ausgestellte Serverzertifikate sollen 5 Jahre lang gültig sein. In den ausgestellten Benutzerzertifikaten soll der volle Name und die Hochschul-E-Mail-Adresse des Benutzers abgelegt werden. Die Serverzertifikate sollen den *vollqualifizierten Domainnamen* (FQDN) des Servers enthalten.

Auswahl des Kryptosystems: EasyRSA unterstützt sowohl RSA-Schlüsselpaare, als auch Schlüsselpaare auf Basis der *Elliptische-Kurven-Kryptografie* (EKK). Somit stehen RSA und EKK als Kryptosystems zur Auswahl, mit denen die CA aufgebaut werden kann.

OpenVPN unterstützt die Verwendung von Zertifikaten mit Schlüsseln auf Basis der EKK ab Version 2.4.0[9] vom Dezember 2016[10]. Die Unterstützung von Zertifikaten mit RSA-Schlüsseln ist jedoch schon in Version 1.2.0 vorhanden gewesen[11], und existiert somit spätestens seit Mai 2002. Zertifikate auf Basis des RSA-Kryptosystems wurden im OpenVPN-Umfeld also mindestens 14 Jahre länger erprobt als Zertifikate auf Basis von EKK. Die daraus resultierenden Erfahrungswerte sind für die Auswahl des Kryptosystems der VPN-CA ausschlaggebend, da die Zertifikate für den VPN-Dienst, abhängig von ihrem Verwendungszweck, für Zeiträume von 5 bis 20 Jahren gültig sein sollen.

[9] Vergleich `https://github.com/OpenVPN/openvpn/blob/v2.4.6/README.ec`
[10] Siehe `https://github.com/OpenVPN/openvpn/releases/tag/v2.4.0`
[11] Vergleich `https://github.com/OpenVPN/openvpn/blob/v2.4.6/ChangeLog`

Für das gewählte Kryptosystem müssen Parameter, wie die gewünschte Schlüssellänge und, im Fall von EKK, eine elliptische Kurve, gewählt werden. Sollten die Parameter des Kryptosystems, oder das Kryptosystem selbst, während des Betriebszeitraums nicht mehr als sicher gelten, so muss eine neue CA aufgebaut werden und die unsichere CA ersetzt werden. Tritt dieser Fall ein, so ist er mit einem hohen Arbeitsaufwand verbunden, da neue Betriebsparameter für die CA gewählt werden müssen und alle gültigen Zertifikate der alten CA entsprechend ersetzt werden müssen. Deshalb ist es geboten, das Risiko für diesen Fall zu minimieren.

Aus diesem Grund fällt die Wahl auf RSA, da mit RSA im Vergleich zu EKK mehr Erfahrungswerte vorliegen, die für die langfristige Stabilität von RSA sprechen. Ein weiterer Grund für die Wahl von RSA liegt darin, dass lediglich die Schlüssellänge als Parameter gewählt werden muss. Mögliche Fehler bei der Wahl einer elliptischen Kurve, wie sie bei EKK notwendig ist, entfallen bei RSA.

Die Vorteile von EKK sind bei dem Anwendungsfall der CA hingegen nicht relevant: Die im Vergleich zu RSA geringeren Schlüssellängen bei gleichem Sicherheitsniveau werden nicht zwingend benötigt, da die Schlüssel beider Kryptosysteme auf modernen Computern in mehr als ausreichender Anzahl abgespeichert werden können. Auch auf die im Vergleich zu RSA effizienteren Verfahren in EKK zum Signieren und Verschlüsseln von Daten kann verzichtet werden, da alle diese Operationen auf modernen Computern in wenigen Sekunden berechnet werden können.

Laut BSI kann RSA über das Jahr 2023 hinaus eingesetzt werden [BSI18a, Kapitel 3.5, Absatz „Schlüssellänge“ (S.38)].

Festlegen der Schlüssellänge: Im nächsten Schritt wird die Länge der RSA-Schlüssel festgelegt, die in allen durch die CA ausgestellten Zertifikaten zum Einsatz kommen soll. OpenVPN empfiehlt eine Schlüssellänge von mindestens 2048 Bit: *„OpenVPN will migrate to 'preferred' as default in the future. Please ensure that your keys already comply.“*. Das Profil „preferred“ ist dabei wie folgt definiert: *„SHA2 and newer, RSA 2048-bit+, any elliptic curve.“* [Aus Yon18, Option `-tls-cert-profile`]. Das BSI empfiehlt Schlüssellängen von mindestens 3000 Bit für Verwendungen über das Jahr 2023 hinaus [BSI18a, Kapitel 3.5, Absatz „Schlüssellänge“ (S.38)]. Damit die CA für 20 Jahre sicher betrieben werden kann, wird auf Basis dieser Empfehlungen die Schlüssellänge auf 4096 Bit festgelegt.

Metadaten: EasyRSA unterstützt zwei Varianten, um den Inhalt des `Subject`-Felds eines X.509-Zertifikat zu füllen. Im Modus „cn_only“ wird nur der `Common Name` in das `Subject`-Feld gesetzt. Im Modus „org“ wird ein `Distinguished Name` in dem `Subject`-Feld abgelegt, der die Felder `Country`, `Province`, `City`, `Org`, `OU`, `email` und `CN` beinhaltet.

Laut Vorgaben soll der volle Name und die Hochschul-E-Mail-Adresse der Benutzer in den Clientzertifikaten abgelegt werden. Somit muss EasyRSA auf den Modus „org“ eingestellt werden. Für Clientzertifikate wird festgelegt, dass der volle Name im Feld `Common Name` abgelegt wird, und die E-Mail-Adresse im Feld `Email Address`. Für Serverzertifikate wird festgelegt, dass der *vollqualifizierte Domainname* (FQDN) im Feld `Common Name` abgelegt wird. Das Feld `Email Address` wird mit der Hochschul-E-Mail-Adresse der für den Server zuständigen Administratoren gefüllt. Für alle weiteren Felder werden folgende Vorgaben festgelegt: `Country`: „DE“, `Province`: „Niedersachsen“, `City`: „Hannover“, `Org`: „Hochschule Hannover“, `OU`: „Abteilung Informatik“.

Gültigkeitsdauer der CRL: OpenVPN kann die von der CA ausgestellte *Certificate Revocation List* (CRL) benutzen, um die Gültigkeit von Clientzertifikaten zu überprüfen. Da dieses Feature benutzt werden soll, sind Einstellungen in Bezug auf die CRL für diese Arbeit relevant.

Die CRL enthält Informationen über alle von der CA zurückgerufenen Zertifikate und ist nur für einen begrenzten Zeitraum gültig, der anhand der Felder „Last Update“ und „Next Update“ begrenzt wird [CSF+08, Kapitel 5.1.2]. Dabei sollte eine aktualisierte CRL noch vor Erreichen des „Next Update“-Zeitpunkt der vorherigen CRL durch die CA veröffentlicht werden [CSF+08, Kapitel 5.1.2.5].

Für EasyRSA kann die Gültigkeitsdauer der CRL in Tagen konfiguriert werden. Da OpenVPN bei einer abgelaufenen CRL den Dienst verweigert, ist für einen unterbrechungsfreien Betrieb wichtig, dass immer eine gültige CRL zur Verfügung steht. Grundsätzlich stellt dies kein Problem dar: Sowohl das Ausstellen, als auch das Installieren der CRL auf dem OpenVPN-Server können automatisiert werden. Um auch bei manueller Installation der CRL einen unterbrechungsfreien Betrieb des VPN-Dienst zu gewährleisten, wird für die CRL eine Gültigkeitsdauer von 180 Tagen festgelegt.

7 Konfiguration des OpenVPN-Servers

In diesem Kapitel wird gezeigt, wie der Server für den VPN-Dienst installiert und konfiguriert wird. Wie in Kapitel 2 bereits geklärt, wird laut Anforderung Anf5 Debian 9 als Betriebssystem verwendet. Die Konfiguration des Betriebssystems erfolgt dabei nach den Vorgaben des IT-Teams, damit der Dienst ohne zusätzliche Arbeiten als Produktivsystem übernommen werden kann.

7.1 Konfiguration des Serverbetriebssystems

Netzwerkkonfiguration: Da der OpenVPN-Dienst aus dem Internet heraus erreichbar sein soll, wird der Server an das DMZ-Netz angeschlossen. Das IT-Team hat dafür insgesamt vier IPv4- und IPv6-Adressen vergeben, um den Server, und den darauf existierenden Dienst, logisch zu trennen. Über die vergebenen IPv4- und IPv6-Hostadressen kann der physische Server angesprochen werden, um zum Beispiel für administrative Aufgaben eine SSH-Sitzung zu dem Server aufzubauen. Über ein weiteres Paar von IP-Dienstadressen wird der eigentliche OpenVPN-Dienst zur Verfügung gestellt.

Um Datenverkehr zwischen den VPN-Clients und dem Netz der Abteilung Informatik routen zu können, wird für IPv4 und IPv6 jeweils ein IP-Adressbereich benötigt, aus dem die VPN-Clients ihre Clientadressen zugewiesen bekommen können. Für IPv4 wurde das private Netz `10.2.0.0/16` durch das IT-Team vergeben. Für IPv6 wurde das Netz `2001:638:614:1750::/64` vergeben, welches durch die Firewall der Abteilung Informatik an die zuvor vergebene IPv6-Dienstadresse geroutet wird. Damit VPN-Clients über IPv4 mit dem Abteilungsnetz kommunizieren können, wird der private IPv4-Adressbereich für die VPN-Clients durch den VPN-Server via *Network Address Translation* (NAT) auf die IPv4-Dienstadresse übersetzt.

Lokale Firewall: In Absprache mit dem Erstprüfer dieser Arbeit wurde die folgende Firewall-Richtlinie für den VPN-Server geplant, und wird unter Nutzung von `iptables` umgesetzt. Diese lokale Richtlinie wird durch die Firewall der Abteilung Informatik ergänzt.

Zuerst werden allgemeine Regeln definiert: Als Standardverhalten wird festgelegt, dass alle Pakete verworfen werden. Datenverkehr aus oder in die lokale Loopback-Schnittstelle wird zugelassen. Datenverkehr mit den Protokollen ICMP und ICMPv6 wird zugelassen.

SSH-Zugriffe auf den Server über TCP-Port 22 werden zugelassen. UDP-Pakete an den OpenVPN-Dienst auf Port 1194 werden zugelassen. Vom Server ausgehender Datenverkehr wird grundsätzlich zugelassen, um die Grundfunktionen (beispielsweise: Installieren von Updates, DNS, NTP, ...) des Betriebssystems zu ermöglichen.

Im Anschluss werden Regeln für die Behandlung des VPN-Datenverkehrs definiert: Bei IPv4-Verkehr vom VPN-Netz zum Netz der Abteilung Informatik wird via NAT die Absenderadresse auf die IPv4-Dienstadresse übersetzt. Datenverkehr, der aus dem VPN-Netz wieder in das VPN-Netz geroutet wird, soll verworfen werden. Datenverkehr aus dem VPN-Netz über TCP oder UDP auf Port 2049 (NFS) wird verworfen. Jeglicher weiterer Datenverkehr aus dem VPN-Netz heraus ist gestattet.

Alle Details zu Installation und Betrieb des VPN-Servers sind in dem Dokument „IPv6-VPN Serverdokumentation“ im Anhang dieser Arbeit zu finden.

7.2 Konfiguration von OpenVPN

Nach Einrichtung des Servers wird in diesem Schritt die Konfiguration des OpenVPN-Servers im Detail erläutert. Nähere Informationen zu den verwendeten Optionen können in [Yon18] nachgeschlagen werden.

Erreichbarkeit des Dienstes: Der VPN-Server soll über IPv4 und IPv6 aus dem Internet erreichbar sein (Anf1). In der **Serverkonfiguration** reichen die ersten drei Zeilen aus, um über IPv4 und IPv6 auf UDP-Port 1194 den VPN-Dienst anzubieten:

```
port 1194
proto udp
proto udp6
multihome
```

OpenVPN beantwortet damit Anfragen für alle auf dem Server konfigurierten IP-Adressen. Da auf dem Server neben den Dienstadressen auch die Hostadressen existieren, wird OpenVPN mit `multihome` angewiesen, eingehende Pakete mit der IP-Adresse als Absender zu beantworten, an die diese Pakete gerichtet waren.

In der **Clientkonfiguration** wird OpenVPN mit `nobind` angewiesen, den Clientsocket nicht an eine lokale Adresse zu binden. Die Angaben `port` und `proto` legen fest, wie der VPN-Server zu erreichen ist. Über `remote` wird dann der zu verwendende VPN-Server explizit genannt. Die Erreichbarkeit über IPv4 und IPv6 ergibt sich aus den im DNS unter `vpn-test.inform.hs-hannover.de` eingetragenen IP-Adressen.

```
nobind
port 1194
proto udp
remote vpn-test.inform.hs-hannover.de 1194
```

OSI-Schicht des VPN-Tunnels: In **Client- und Serverkonfiguration** wird mit angegeben, dass eine virtuelle Netzwerkkarte auf OSI-Schicht 3 als Endpunkt für das VPN verwendet wird:

```
dev tun
```

IP-Adressen im VPN-Tunnel: Die vom IT-Team vergebenen IP-Adressbereiche für VPN-Clients werden durch den OpenVPN-Server an die Clients vergeben. Für das vergebene IPv4-Netz wurde berücksichtigt, dass 50-500 VPN-Clients problemlos mit IP-Adressen versorgt werden können, indem ein `/16`-Block vergeben wurde. In der Konfiguration des Clients sind keine Anweisungen notwendig. Die **Serverkonfiguration** enthält diese Anweisungen:

```
topology subnet
server 10.2.0.0 255.255.0.0
server-ipv6 2001:638:614:1750::/64
```

Mit `topology subnet` wird festgelegt, dass die IPv4-Adressvergabe so durchgeführt werden soll, als wären alle VPN-Clients mit dem VPN-Server in einem Subnetz. Anschließend werden mit `server` und `server-ipv6` die IP-Netze festgelegt, die an die VPN-Clients vergeben werden sollen.

Erreichbare Netze im VPN: In der **Serverkonfiguration** werden `push`-Anweisungen eingetragen, über die IPv4- und IPv6-Routen an die VPN-Clients bekannt gegeben werden können. Ein VPN-Client ist durch die Anweisung `client` implizit auch mit dem Parameter `pull` konfiguriert, sodass die vom Server über `push` bekanntgegebenen Optionen übernommen werden.

Damit die einzurichtenden Routen den OpenVPN-Datenverkehr nicht durch das VPN leiten, sodass der VPN-Client den VPN-Server nicht mehr direkt erreichen kann, wird eine IPv4-Route zum VPN-Server über das Standardgateway des VPN-Clients eingerichtet:

```
push "route remote_host 255.255.255.255 net_gateway"
```

Für IPv6 ist dieser Schritt nicht notwendig, da OpenVPN diesen Fall für IPv6 automatisch erkennen kann, und der VPN-Client selbstständig eine direkte IPv6-Route zum VPN-Server einrichtet.

Dann werden alle IPv4-Routen bekanntgegeben:

```
push "route 141.71.38.0 255.255.255.0 vpn_gateway" # DMZ
push "route 141.71.30.0 255.255.254.0 vpn_gateway" # Inform
push "route 192.168.99.0 255.255.255.0 vpn_gateway" # Edu
push "route 192.168.90.0 255.255.255.0 vpn_gateway" # NAO
push "route 192.168.70.0 255.255.255.0 vpn_gateway" # iDrac
push "route 10.0.20.0 255.255.255.0 vpn_gateway" # Cluster
push "route 10.0.30.0 255.255.255.0 vpn_gateway" # educloud
```

```
push "route 10.0.40.0 255.255.255.0 vpn_gateway" # experimental ipv6
   network
push "route 141.71.2.0 255.255.255.0 vpn_gateway" # server network
   from H-IT for KMS
```

Der Platzhalter `vpn_gateway` wird bei IPv4-Routen durch die IPv4-Adresse des VPN-Servers im VPN ersetzt. Anschließend werden alle IPv6-Routen an die VPN-Clients bekanntgegeben:

```
push "route-ipv6 2001:638:614:1780::/64 2001:638:614:1750::1" # DMZ
push "route-ipv6 2001:638:614:1720::/64 2001:638:614:1750::1" # Inform
push "route-ipv6 2001:638:614:1721::/64 2001:638:614:1750::1" # Edu
push "route-ipv6 2001:638:614:1722::/64 2001:638:614:1750::1" # NAO
push "route-ipv6 2001:638:614:1743::/64 2001:638:614:1750::1" #
   Cluster
push "route-ipv6 2001:638:614:1744::/64 2001:638:614:1750::1" #
   experimental ipv6 network
```

Da der Platzhalter `vpn_gateway` für IPv6-Routen leider nicht funktioniert, muss hier die IPv6-Adresse des VPN-Servers im VPN direkt als Ziel der Route angegeben werden. Aufgrund dieser einfachen Konfiguration besteht jederzeit die Möglichkeit der Anpassung und Erweiterbarkeit.

Zertifikatgestützte Authentisierung: Um die in Kapitel 4.2 beschriebene Zertifizierungsstelle zur Benutzerauthentisierung zu verwenden, und die VPN-Kommunikation nach Anforderung Anf3 aus Kapitel 2 abzusichern, wird OpenVPN im TLS-Modus betrieben.

In der **Clientkonfiguration** wird dafür der Parameter `tls-client` gesetzt, in der **Serverkonfiguration** wird analog dazu der Parameter `tls-server` verwendet.

Anschließend werden in der **Client- und Serverkonfiguration** über die Parameter `ca`, `cert` und `key` Dateipfade konfiguriert, unter denen das Wurzelzertifikat, das Client- beziehungsweise Serverzertifikat und der dazugehörige private Schlüssel abgelegt sind. In der **Serverkonfiguration** werden zusätzlich über die Parameter `dh` und `crl-verify` Dateipfade angegeben, unter denen die im Voraus berechneten Parameter für den Diffie-Hellman-Schlüsselaustausch, und die aktuelle Kopie der von der CA herausgegebenen CRL, abgespeichert sind.

```
ca inform/ca.crt
cert inform/aither.inform.hs-hannover.de.crt
key inform/aither.inform.hs-hannover.de.key
dh inform/dh.pem
crl-verify inform/crl.pem
```

Um zu unterbinden, dass Clientzertifikate zum Betrieb eines OpenVPN-Servers verwendet werden, wird mit dem Parameter `remote-cert-tls` in der **Clientkonfiguration** angegeben, dass von der Gegenseite ein Zertifikat mit „Serverrolle“ präsentiert werden muss:

```
remote-cert-tls server
```

Analog dazu wird in der **Serverkonfiguration** mit dem selben Parameter verlangt, dass VPN-Clients immer ein Zertifikat mit „Clientrolle“ besitzen.

```
remote-cert-tls client
```

Kryptografische Parameter: Wie in Kapitel 6.1 bereits erläutert, werden die kryptografischen Parameter wie folgt in **Client- und Serverkonfiguration** eingetragen.

Die TLS-Kommunikation wird über TLS Version 1.2 oder höher durchgeführt:

```
tls-version-min "1.2"
```

Die für den Kontrollkanal verwendete TLS-Chiffre wird konfiguriert:

```
tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384
```

Als Quelle für Pseudozufallszahlen im HMAC für Kontroll- und Datenkanal kommt SHA-256 zum Einsatz:

```
auth SHA256
```

Zur Verschlüsselung des Datenkanals kommt die Chiffre AES-256-GCM zum Einsatz.

```
cipher AES-256-GCM
```

Um zu verhindern, dass fehlerhaft konfigurierte VPN-Clients eine weniger sichere Verschlüsselung des Datenkanals aushandeln, wird in der **Serverkonfiguration** die Aushandlung der Chiffre zur Verschlüsselung des Datenkanals abgeschaltet.

```
ncp-disable
```

Sitzungsparameter: VPN-Client und VPN-Server sollen zeitnah reagieren, wenn eine Sitzung durch Verlust der Internetverbindung abbricht. Mit `keepalive` kann in der **Serverkonfiguration** angegeben werden, in welchem Intervall eine Ping-Nachricht an die Gegenseite geschickt werden soll. Ein zweiter Parameter erlaubt die Definition eines Zeitlimits, nachdem dessen Ablauf eine Sitzung für abgebrochen erklärt wird.

```
keepalive 10 30
```

Diese Anweisung sorgt dafür, dass der Server alle 10 Sekunden eine Ping-Nachricht an den Client schickt, und nach einem Zeitlimit von 60 Sekunden ohne Erhalt einer Antwort die Sitzung beendet, und damit verbundene Ressourcen wieder freigibt. Gleichzeitig wird die Anweisung über `push` an den Client übermittelt, sodass dieser ebenfalls alle 10 Sekunden eine Ping-Nachricht an den Server schickt. Erhält der Client innerhalb von 30 Sekunden keine Antwort auf seine Ping-Nachricht, so startet er sich neu um einen erneuten Sitzungsaufbau zu versuchen.

Um Verbindungsprobleme auf dem Client frühzeitig erkennen zu können, wird die Option `connect-timeout` in der **Clientkonfiguration** verwendet, um nach einem Zeitlimit von 20 Sekunden den Sitzungsaufbau abzubrechen.

```
connect-timeout 20
```

Damit Client und Server sich bei einem gewollten Sitzungsabbau gegenseitig benachrichtigen, wird die Option `explicit-exit-notify` verwendet. In der **Clientkonfiguration** wird als zusätzlicher Parameter angegeben, wie oft der Versand der Benachrichtigung an den Server probiert werden soll.

```
explicit-exit-notify 3
```

In der **Serverkonfiguration** gibt der zusätzliche Parameter an, ob der Client einen neuen Sitzungsaufbau bei dem selben VPN-Server oder einem anderen VPN-Server probieren soll. In diesem Fall soll der selbe Server erneut kontaktiert werden.

```
explicit-exit-notify 1
```

Zuletzt wird in der **Serverkonfiguration** eingestellt, dass ein Clientzertifikat für mehr als eine VPN-Sitzung gleichzeitig verwendet werden darf. Dadurch kann ein Benutzer den VPN-Zugang auf mehreren Geräten gleichzeitig benutzen, falls das notwendig sein sollte.

```
duplicate-cn
```

Lokale Sicherheit: Auf den Betriebssystemen Linux und Mac OS kann OpenVPN nach seinem Start nicht mehr benötigte Berechtigungen abgeben und im Kontext eines lokalen Benutzers weiterlaufen. Dies kann in **Client- und Serverkonfiguration** bewerkstelligt werden. OpenVPN kann unter diesen Bedingungen auf bestimmte Ressourcen nicht erneut zugreifen. Dazu gehören beispielsweise private Schlüssel für Zertifikate, oder die virtuelle Netzwerkkarte. Damit ein Neustart von OpenVPN auch ohne Privilegien funktionieren kann, ist es möglich mit `persist-tun` und `persist-key` das Handle auf die virtuelle Netzwerkkarte, sowie bereits gelesene Schlüssel im Speicher zu behalten.

```
persist-tun
persist-key
```

Dann kann über die Parameter `user` und `group` gesteuert werden, in welchem Kontext OpenVPN mit reduzierten Privilegien betrieben werden soll.

```
user nobody
group nogroup
```

Für die **Clientkonfiguration** kann es sinnvoll sein, dass OpenVPN das Passwort, mit dem ein privater Schlüssel geschützt wurde, nicht im Speicher behält. Mit der folgenden Option kann OpenVPN gezwungen werden, ein solches Passwort nicht im Speicher zu halten.

```
auth-nocache
```

Protokollierung: Um den Detailgrad der Protokolle von OpenVPN zu steuern, können in der **Client- und Serverkonfiguration** die Parameter `verb` und `mute` verwendet werden. Der Parameter `verb` steuert dabei den Detailgrad der protokollierten Meldungen. Der Bereich reicht von 0 (keine Meldungen) über 1-4 (normale Meldungen), 5 (Für jedes verarbeitete Paket wird ein Buchstabe in das Log geschrieben) bis hin zu 6-11 (Für Fehlersuche in der Entwicklung).

Für DSGVO-konforme Protokolle wird der Wert 0 in der **Serverkonfiguration** verwendet; der Wert 3 wird für übersichtliche Protokolle mit mehr Informationen in der **Clientkonfiguration** empfohlen. Der Parameter `mute` gibt an, wie viele aufeinander folgende Nachrichten aus der selben Kategorie protokolliert werden sollen. Damit können aufeinander folgende, ähnliche Nachrichten unterdrückt werden, die sich wiederholen. Um einen guten Überblick im Testbetrieb zu erhalten, werden vorerst die folgenden Parameter gewählt:

```
verb 3
mute 5
```

In der **Serverkonfiguration** kann mit `status` dafür gesorgt werden, dass OpenVPN regelmäßig eine Liste aller aktiven Sitzungen in die als Parameter angegebene Datei schreibt. Eine solche Liste ist nützlich, um die Auslastung des Servers zu überwachen.

```
status inform/status.log
```

Damit ist die Erzeugung der OpenVPN-Konfiguration für Client und Server abgeschlossen. Die resultierenden Konfigurationdateien befinden sich im Anhang dieser Arbeit.

8 Implementieren der Benutzerverwaltung

Nachdem die Installation des OpenVPN-Servers in Kapitel 7 beschrieben wurde, wird in diesem Kapitel die Einrichtung der Zertifizierungsstelle zur Verwaltung der VPN-Benutzer auf Basis von EasyRSA gezeigt.

Für den Betrieb der Zertifizierungsstelle wird eine virtuelle Maschine exklusiv für diesen Zweck erzeugt und im Mitarbeiter-Netz platziert. Unter Berücksichtigung der Anforderung Anf5 aus Kapitel 2 wird Debian 9 nach den Vorgaben des IT-Teams auf der virtuellen Maschine installiert. Das EasyRSA-Paket wird unterhalb von `/root` ausgepackt und für den Einsatz als CA für den VPN-Dienst konfiguriert.

Berechtigungen: Durch die Platzierung der CA unterhalb von `/root` kann nur der Benutzer `root` die Zertifizierungsstelle benutzen und auf den privaten Schlüssel des Wurzelzertifikats zugreifen. Lokale Benutzer können über `sudo` für die Benutzung der CA berechtigt werden. Zusätzlich können direkte Zugriffsrechte durch die Verwendung von SSH-Schlüsseln für den `root`-Login verteilt werden.

Gleichzeitig beinhaltet der `root`-Benutzer eine Warnfunktion: Die Berechtigung für Zugriffe auf die CA impliziert viel Verantwortung und verlangt sorgfältiges Vorgehen bei der Benutzung der CA. Auch die regelmäßige Kontrolle aller erteilten Berechtigungen und die überlegte Erteilung von Berechtigungen soll dadurch motiviert werden.

Öffentliche Daten: VPN-Benutzer benötigen bestimmte Dateien, um die CA zu verwenden. Neben dem Wurzelzertifikat der CA, und der aktuellen CRL, benötigen Benutzer eine vorkonfigurierte Version des EasyRSA-Pakets zur Erzeugung von Zertifikatsanträgen. Um diese Daten zur Verfügung zu stellen, wird auf der virtuellen Maschine der Webserver über das Debian-Paket `apache2` installiert, welcher ausschließlich das für diesen Zweck erzeugte Verzeichnis `/public` über HTTP zur Verfügung stellen soll.

Alle in `/public` platzierten Dateien und Verzeichnisse gehören dem Benutzer `root` und der Gruppe `root`. Alle Dateien werden mit den Dateirechten `444` versehen, Verzeichnisse erhalten die Dateirechte `555`. Dadurch werden Manipulationen durch nicht privilegierte, lokale Benutzer verhindert. Gleichzeitig signalisieren die Dateirechte, dass die Inhalte unter `/public` nur durch `root` verändert werden dürfen und für alle anderen lediglich lesbar zur Verfügung stehen sollen.

Die Veröffentlichung von Konfigurationsdateien und Anleitungen für VPN-Benutzer kann über diesen Webserver ebenfalls stattfinden.

Aktualisierung der CRL: Damit der OpenVPN-Server immer Zugriff auf eine aktuelle CRL hat, wird ein Cronjob via `crontab -e` unter dem Benutzer `root` eingerichtet, der über die EasyRSA-Skripte eine neue CRL erzeugt und diese anschließend mit den zuvor erläuterten Dateirechten in `/public` platziert.

Auf dem OpenVPN-Server wird ebenfalls ein Cronjob eingerichtet, der die aktuelle CRL von dem Webserver der CA herunterlädt und in das Konfigurationsverzeichnis von OpenVPN integriert.

Details zur Benutzung der CA sind in dem Dokument „Dokumentation der Zertifizierungsstelle für den IPv6-VPN-Dienst“ im Anhang dieser Arbeit zu finden.

9 Fazit

In dieser Arbeit wurde ein IPv6-VPN-Dienst für die Abteilung Informatik anhand von gegebenen Anforderungen mit OpenVPN konzipiert und umgesetzt. Alle für die Installation und den Betrieb notwendigen Anleitungen befinden sich im Anhang dieser Arbeit:

- Benutzeranleitung für das IPv6-VPN der Abteilung Informatik
- Dokumentation der Zertifizierungsstelle für den IPv6-VPN-Dienst
- IPv6-VPN Serverdokumentation

Während des Vergleichs und der Auswahl der VPN-Softwarekandidaten hat sich gezeigt, dass mit OpenVPN eine flexible und - im Vergleich zu IPsec - unkomplizierte und vollständig quelloffene IPv4- und IPv6-VPN-Lösung geboten wird, die auf allen gängigen Client-Betriebssystemen läuft. Fortgeschrittene Anwender können zusätzlich die kryptografischen Parameter plattformunabhängig festlegen, mit denen die Kommunikation abgesichert wird. Die Verwaltung von VPN-Benutzern kann dabei unter anderem über X.509-Public-Key-Zertifikate geschehen, und bietet somit einen Kompromiss aus Flexibilität und Sicherheit.

Die Installation und Konfiguration des VPN-Dienst konnte ohne Komplikationen durchgeführt werden, und erfüllt alle gegebenen Anforderungen. Da die Konzeption in enger Absprache mit dem IT-Team der Abteilung Informatik stattgefunden hat, und alle im IT-Team gängigen Vorgehensweisen berücksichtigt wurden, steht dem langfristigen Betrieb des in dieser Arbeit erschaffenen VPN-Servers und der dazugehörigen VPN-CA nichts im Weg.

Mit der in dieser Arbeit erschaffenen VPN-Lösung ist der Bedarf der Abteilung Informatik an einem IPv6-fähigen VPN-Dienst gedeckt worden. Allerdings bedeutet die erfolgreiche Inbetriebnahme dieses neuen Produktivsystems keinesfalls, dass nun nichts mehr zu tun sei. Es lohnt sich jederzeit, die Aktualisierung und Weiterentwicklung dieses Dienstes im Auge zu behalten. Als Motivation dafür seien mögliche Gewinne in den Punkten IT-Sicherheit, Effizienz, Benutzerfreundlichkeit oder schlicht ein reduzierter Aufwand bei Wartung und Pflege des Dienstes genannt.

Aktualisierung der kryptografischen Parameter Sollten die in dieser Arbeit gewählten, kryptografischen Parameter nicht mehr als sicher gelten, so sollen alle Parameter nach aktuellem Stand der Technik neu gewählt werden. Dafür ist neben einer Anpassung der Serverkonfiguration auch die Anpassung sämtlicher Clientkonfiguration notwendig. Dies kann beispielsweise über eine veröffentlichte Beispielkonfiguration angestoßen werden, und muss allen VPN-Benutzer mitgeteilt werden. Die mit dieser Methode verbundenen Vor- und Nachteile wurden in Kapitel 6.1 im Detail diskutiert.

Unterstützung von TLS 1.3 Heute (04.11.2018) unterstützt OpenVPN in der aktuellen Version 2.4.6 TLS 1.3 noch nicht. Laut einem Ticket im OpenVPN Issue-Tracker[1] kann vermutet werden, dass OpenVPN ab Version 2.4.7 TLS 1.3 bereits unterstützen könnte. Um TLS 1.3 benutzen zu können, müssen die auf dem VPN-Server installierten Pakete `openssl` und `openvpn` TLS 1.3 unterstützen - da diese aus Debian-Paketquellen installiert werden, ist denkbar, dass TLS 1.3 erst mit Debian 10 unterstützt werden könnte. Besteht die Unterstützung für TLS 1.3 seitens des VPN-Servers, so können VPN-Clients die Verwendung von TLS 1.3 erproben, indem in der Clientkonfiguration der Parameter `tls-version-min` auf 1.3 gestellt wird. Sobald hinreichende Unterstützung für TLS 1.3 für alle VPN-Clients besteht, kann die minimal erlaubte TLS-Version in die Serverkonfiguration auf TLS 1.3 gestellt werden, um alle VPN-Sitzungen nur über TLS 1.3 aufzubauen. Die Anpassung aller Clientkonfigurationen ist in diesem Fall nicht zwingend. Jedoch sollten alle VPN-Benutzer auf die Umstellung hingewiesen werden und die Vorlage für die Clientkonfiguration aktualisiert werden.

Nachfolger OpenVPN 3: Während für diesen Dienst OpenVPN ab Version 2.4.0 zum Einsatz kommt, befindet sich mit OpenVPN 3[2] ein Nachfolger in Entwicklung. Der OpenVPN 3 Linux-Client[3] verfolgt einen modularen Ansatz, bei dem die verschiedenen Module nur mit unbedingt notwendigen Privilegien ausgestattet werden sollen. Die Kommunikation zwischen den einzelnen Modulen soll über D-Bus ablaufen[4]. Heute (13.10.2018) ist OpenVPN 3 noch nicht bereit für den produktiven Einsatz. Sollte sich das jedoch ändern, könnte im Rahmen einer neuen Arbeit untersucht werden, welche Verbesserungen OpenVPN 3 mit sich bringt, und ob ein Umstieg von OpenVPN 2.4.0 auf OpenVPN 3 lohnenswert ist.

[1] Siehe `https://community.openvpn.net/openvpn/ticket/1080`
[2] `https://github.com/OpenVPN/openvpn3`
[3] `https://github.com/OpenVPN/openvpn3-linux`
[4] Siehe `https://github.com/OpenVPN/openvpn3-linux/blob/master/README.md`

Alternative Wireguard: Wie in Kapitel 5.1.3 dieser Arbeit bereits erwähnt wurde, ist Wireguard heute (13.10.2018) noch in einem experimentellen Zustand. Sobald Wireguard für den produktiven Einsatz geeignet ist und Clientsoftware für alle benötigten Betriebssysteme zur Verfügung steht, kann in einer neuen Arbeit untersucht werden, ob die sich die Ablösung des VPN-Dienstes aus dieser Arbeit mit einer neuen Lösung auf Basis von Wireguard lohnt. Die bisherigen Angaben in Bezug auf Effizienz, Wahl der kryptografischen Parameter und Benutzerfreundlichkeit sprechen meiner Meinung nach stark dafür, dass ein Umstieg von OpenVPN auf Wireguard vorteilhaft ist.

Zusätzlich wäre ein weiteres Einsatzszenario denkbar, das durch die hohe Effizienz und die einfachen Konzepte von Wireguard möglich gemacht wird. Studierenden der Abteilung Informatik ist es nur nach Genehmigung des IT-Teams erlaubt, ihre Privatgeräte - wie zum Beispiel mitgebrachte Laptops - über Ethernet mit dem Netz der Abteilung Informatik zu verbinden. Bei erteilter Erlaubnis wird die MAC-Adresse des Privatgeräts auf dem DHCP-Server über eine Whitelist freigeschaltet, damit das Privatgerät über DHCP IP-Adressen beziehen kann, und sich dadurch mit dem Netz der Abteilung Informatik verbinden kann.

An dieser Stelle könnte eine Lösung auf Basis von Wireguard ansetzen: Anstatt Zugriffsfreigaben auf Basis von MAC-Adressen zu erteilen, könnte ein mit Wireguard ausgestattetes Gateway so konfiguriert werden, dass nur durch Wireguard authentisierter Datenverkehr in das Netz der Abteilung Informatik weitergeleitet wird. Als positiver Nebeneffekt werden „Lauschangriffe“ auf OSI-Schicht 2 durch andere Privatgeräte dank der Verschlüsselung des Netzwerkverkehrs durch Wireguard effektiv verhindert. Da Wireguard-Teilnehmer durch ihren öffentlichen Schlüssel durch das Gateway eindeutig identifiziert werden können, könnte man zusätzlich untersuchen, ob die Umsetzung unterschiedlicher Zugriffsrechte für verschiedene Wireguard-Teilnehmer möglich und sinnvoll ist.

Literaturverzeichnis

[ANWOW13] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. Blake2: Simpler, smaller, fast as md5. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*, pages 119–135, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[BSI18a] BSI. Bsi technische richtlinie tr-02102-1: Kryptographische verfahren: Empfehlungen und schlüssellängen, 2018.

[BSI18b] BSI. Tls nach tr-03116-4 checkliste für diensteanbieter, 2018.

[CSF+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, Internet Engineering Task Force, May 2008. `https://tools.ietf.org/html/rfc5280.txt`.

[Don17] Jason A. Donenfeld. Wireguard: Next generation kernel network tunnel. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*, 2017.

[DP18] Benjamin Dowling and Kenneth G. Paterson. A cryptographic analysis of the wireguard protocol. In *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, pages 3–21, 2018.

[DR08] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. RFC 5246, Internet Engineering Task Force, August 2008. `https://tools.ietf.org/html/rfc5246.txt`.

[EC18] ECRYPT-CSA. Algorithms, key size and protocols report (2018), 2018. `http://www.ecrypt.eu.org/csa/publications.html`.

[ENI14] ENISA. Algorithms, key size and parameters report – 2014, 2014. `https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014`.

[HM15] Raphaël Hertzog and Roland Mas. *The Debian Administrator's Handbook*. 2015.

[HOI+05] Osamu Honda, Hiroyuki Ohsaki, Makoto Imase, Mika Ishizuka, and Junichi Murayama. Understanding tcp over tcp: effects of tcp tunneling on end-to-end throughput and latency, 2005.

[Ken05a] S. Kent. Ip authentication header. RFC 4302, Internet Engineering Task Force, December 2005. `https://tools.ietf.org/html/rfc4302.txt`.

[Ken05b] S. Kent. Ip encapsulating security payload (esp). RFC 4303, Internet Engineering Task Force, December 2005. `https://tools.ietf.org/html/rfc4303.txt`.

[KHN+14] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. Internet key exchange protocol version 2 (ikev2). RFC 7296, Internet Engineering Task Force, October 2014. `https://tools.ietf.org/html/rfc7296.txt`.

[KS05] S. Kent and K. Seo. Security architecture for the internet protocol. RFC 4301, Internet Engineering Task Force, December 2005. `https://tools.ietf.org/html/rfc4301.txt`.

[NL15] Y. Nir and A. Langley. Chacha20 and poly1305 for ietf protocols. RFC 7539, Internet Engineering Task Force, May 2015. `https://tools.ietf.org/html/rfc7539.txt`.

[SHSA15] Y. Sheffer, R. Holz, and P. Saint-Andre. Recommendations for secure use of transport layer security (tls) and datagram transport layer security (dtls). BCP 195, Internet Engineering Task Force, May 2015. `https://tools.ietf.org/html/rfc7525.txt`.

[str18a] strongswan on mac os x, 2018. `https://wiki.strongswan.org/projects/strongswan/wiki/MacOSX`, zuletzt abgerufen am 27.09.2018.

[str18b] strongswan on windows, 2018. `https://wiki.strongswan.org/projects/strongswan/wiki/Windows`, zuletzt abgerufen am 07.08.2018.

[WMM+17] P. Wouters, D. Migault, J. Mattsson, Y. Nir, and T. Kivinen. Cryptographic algorithm implementation requirements and usage guidance for encapsulating security payload (esp) and authentication header (ah). RFC 8221, Internet Engineering Task Force, October 2017. `https://tools.ietf.org/html/rfc8221.txt`.

[Yon18] James Yonan. *openvpn(8) - System Manager's Manual*, openvpn version 2.4 edition, 2018. `https://manpages.debian.org/stretch/openvpn/openvpn.8.en.html`.

Anhang

OpenVPN Clientkonfiguration

```
# This is the client configuration
client

# No need to bind on specific interfaces, just send packets to the
   openvpn server
nobind

# Send udp packets to port 1194
port 1194
proto udp

# We're using the virtual network interface on layer 3
dev tun

# Specify vpn server
remote vpn-test.inform.hs-hannover.de 1194

# Certificates to use. EDIT THIS SECTION to reflect your situation
ca /etc/openvpn/vpnclient/ca.crt
cert /etc/openvpn/vpnclient/jan-philipp.timme@stud.hs-hannover.de.crt
key /etc/openvpn/vpnclient/jan-philipp.timme@stud.hs-hannover.de.key

# Prevent OpenVPN from caching the password of your private key in
   memory.
# Depending on your use case for OpenVPN, enabling this option can
   provide more protection
# for your private key (and more password prompts during an OpenVPN
   session)
#auth-nocache

# Assume client role in tls handshake
tls-client

# Make sure the server presents a certificate with "server role"
# This way people with proper client certificates are unable to
   impersonate the server
remote-cert-tls server

### START BLOCK CRYPTOGRAPHY
# Specific settings regarding TLS, chiphers and hash algorithms
```

```
# DO NOT CHANGE THIS unless you receive explicit instructions to do so
# These settings need to be identical in client and server
    configuration!

# Protect data channel with this cipher
cipher AES-256-GCM

# Authenticate packets in data and control channel using HMAC with
    this
# message digest algorithm
auth SHA256

# Use this specific cipher to secure the control channel
tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384

# Only allow TLS version 1.2 and higher
tls-version-min "1.2"
### END BLOCK CRYPTOGRAPHY

# Reduce connection timeout so connection problems are visible sooner
connect-timeout 20

# Notify server on client shutdown/restart events, so old sessions
    get terminated immediately
# Try to send notification 3 times (because we're using UDP)
explicit-exit-notify 3

# Enable these if you plan to enable running on reduced privileges
# These options allow to keep the private key and the virtual network
    device handle in memory
#persist-key
#persist-tun

# Reduce privileges after launch (uncomment and adapt on unix/linux
    system)
# Note: On some systems, the group is called "nobody" instead of
    "nogroup"
#user nobody
#group nogroup

# Logging settings
verb 3
mute 5
```

OpenVPN Serverkonfiguration

```
# Listen on 1194 for both IPv4 and IPv6
port 1194
proto udp
```

```
proto udp6

# Since we have more than one ip address, this makes openvpn respond
# with the right sender address
multihome

# We're using the virtual network interface on layer 3
dev tun

# Certificates to use. Paths are relative to config file location.
ca inform/ca.crt
cert inform/aither.inform.hs-hannover.de.crt
key inform/aither.inform.hs-hannover.de.key

# Assume server role in tls handshake
tls-server

# Diffie-Hellman parameter file
# (not needed for TLS cipher with ECDHE instead of DHE)
dh inform/dh.pem

# Certificate revocation list location
# Make sure this file is always valid, otherwise OpenVPN refuses to
   (re)start!
crl-verify inform/crl.pem

# Make sure the client presents a certificate with "client role"
remote-cert-tls client

# Allow multiple connections using the same certificate?
# There is no reason to not allow this, so it is allowed.
duplicate-cn

# We're using subnet topology for IPv4 tunnel connectivity
topology subnet

# Use this IPv4 range for clients (/16, so we can cope with potential
   500 clients)
server 10.2.0.0 255.255.0.0

# Use this IPv6 network for clients
server-ipv6 2001:638:614:1750::/64

# Make sure the client can still reach the OpenVPN server via its
   IPv4 default gateway
# This is needed because the IPv4 route for DMZ is pushed below,
# which overlaps the OpenVPN server IPv4 address.
push "route remote_host 255.255.255.255 net_gateway"

# Push routes for local IPv4 networks
#
# DMZ
```

```
push "route 141.71.38.0 255.255.255.0 vpn_gateway"
# Inform
push "route 141.71.30.0 255.255.254.0 vpn_gateway"
# Edu
push "route 192.168.99.0 255.255.255.0 vpn_gateway"
# NAO
push "route 192.168.90.0 255.255.255.0 vpn_gateway"
# iDrac
push "route 192.168.70.0 255.255.255.0 vpn_gateway"
# Cluster
push "route 10.0.20.0 255.255.255.0 vpn_gateway"
# educloud
push "route 10.0.30.0 255.255.255.0 vpn_gateway"
# experimental ipv6 network
push "route 10.0.40.0 255.255.255.0 vpn_gateway"
# server network from H-IT for KMS
push "route 141.71.2.0 255.255.255.0 vpn_gateway"

# Push routes for local IPv6 networks
# (The vpn_gateway placeholder does not work here.)
# Note: IPv6 routes that overlap the IPv6 address of the OpenVPN
   server will
# automatically trigger creating a direct route to the OpenVPN server
   on the client.
#
# DMZ
push "route-ipv6 2001:638:614:1780::/64 2001:638:614:1750::1"
# Inform
push "route-ipv6 2001:638:614:1720::/64 2001:638:614:1750::1"
# Edu
push "route-ipv6 2001:638:614:1721::/64 2001:638:614:1750::1"
# NAO
push "route-ipv6 2001:638:614:1722::/64 2001:638:614:1750::1"
# Cluster
push "route-ipv6 2001:638:614:1743::/64 2001:638:614:1750::1"
# experimental ipv6 network
push "route-ipv6 2001:638:614:1744::/64 2001:638:614:1750::1"

### START BLOCK CRYPTOGRAPHY
# Specific settings regarding TLS, chiphers and hash algorithms
# DO NOT CHANGE THIS unless you receive explicit instructions to do so
# These settings need to be identical in client and server
   configuration!

# Protect data channel with this cipher
cipher AES-256-GCM

# Authenticate packets in data and control channel using HMAC with
   this
# message digest algorithm
auth SHA256
```

```
# Use this specific cipher to secure the control channel
tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384

# Only allow TLS version 1.2 and higher
tls-version-min "1.2"
### END BLOCK CRYPTOGRAPHY

# Disable cipher negotiation on server side
ncp-disable

# Send ping message every ten seconds, expect session loss after 60
   seconds of no response
keepalive 10 30

# Notify clients when the server restarts or shuts down.
# Default behaviour: Tell clients to try to connect to the same
   server again.
explicit-exit-notify 1

# Enable these if you plan to enable running on reduced privileges
# These options allow to keep the private key and the virtual network
   device handle in memory
persist-key
persist-tun

# Reduce privileges after launch (uncomment and adapt on unix/linux
   system)
user nobody
group nogroup

# Logging settings
# - ‘verb 3‘ is enough for debugging most issues
# - ‘verb 0‘ is recommended for regular operation
verb 3
mute 5

# Print a list of active sessions into this file
# This might be helpful if you plan big maintenance
status inform/status.log
```

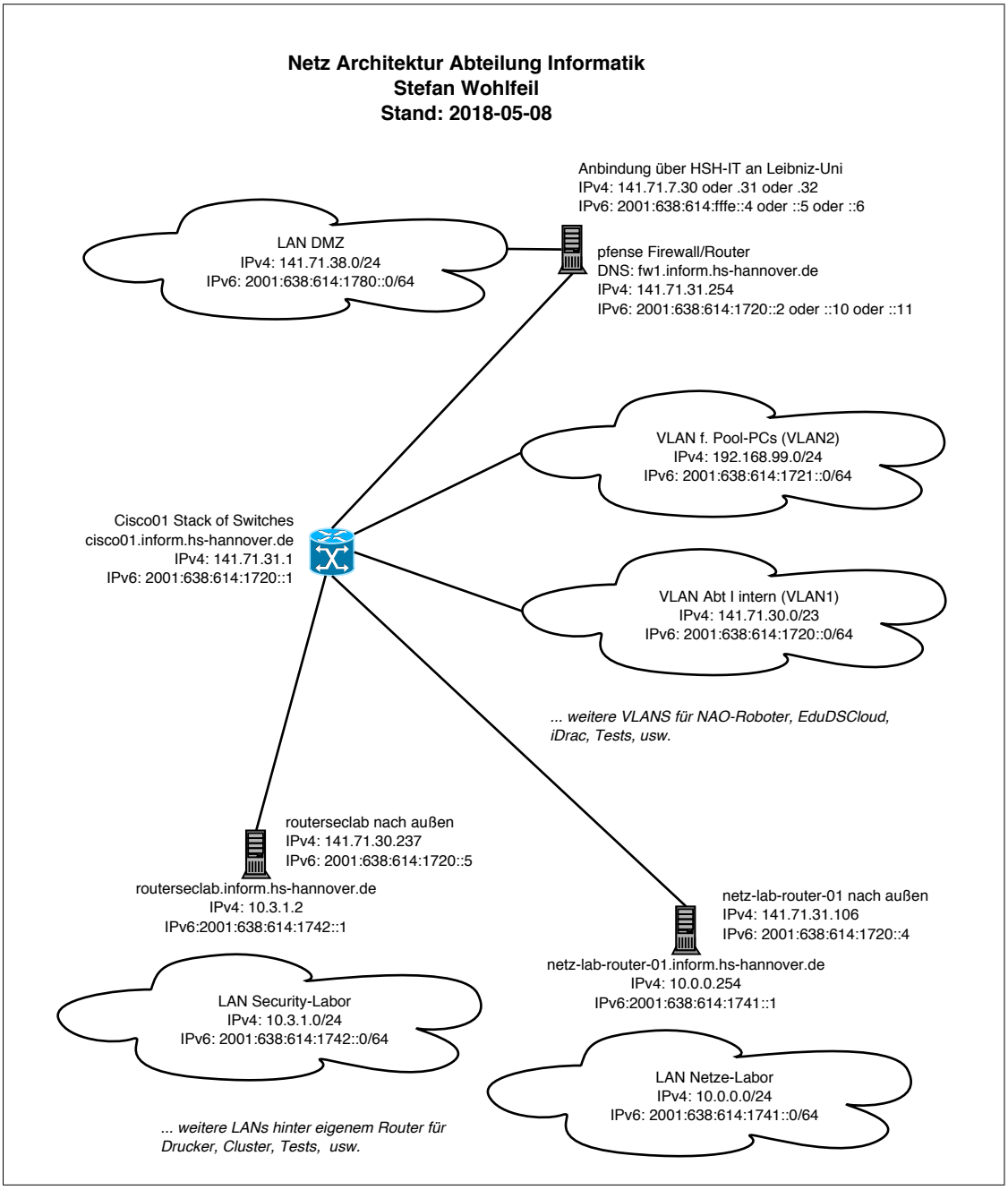

Abbildung 9.1: Dokumentation Netzarchitektur der Abteilung Informatik

Benutzeranleitung für das IPv6-VPN der Abteilung Informatik

Diese Anleitung enthält Informationen zur schnellen Einrichtung des IPv6-VPN-Clients mit OpenVPN ab Version 2.4.0. Die Anweisungen in dieser Anleitung sind für Debian 9 konzipiert und müssen je nach verwendetem Betriebsystem angepasst werden.

OpenVPN oder eine kompatible Alternative stehen für viele gängige Betriebsysteme zur Verfügung. Für **Linux**/**Unix** und **Windows** finden sich unter `https://community.openvpn.net/openvpn#GettingOpenVPN` weiterführende Links.

Zusätzlich kann unter **Windows** die Software „OpenVPN-GUI“ die Bedienung erleichtern: `https://community.openvpn.net/openvpn/wiki/OpenVPN-GUI-New`.

Für **Mac OS** sei auf die Software Tunnelblick (`https://tunnelblick.net/downloads.html`) verwiesen.

Benutzerzertifikat beantragen

Hinweis: Details zu diesem Vorgang können in Kapitel 3 des Dokuments „Dokumentation der Zertifizierungsstelle für den IPv6-VPN-Dienst“ nachgelesen werden.

Um ein Benutzerzertifikat zu beantragen, muss zunächst das von der VPN-CA gestellte EasyRSA-Paket heruntergeladen, entpackt und initialisiert werden.

```
wget http://vpnca.inform.hs-hannover.de/VPN-EasyRSA.zip; unzip VPN-EasyRSA.zip
cd EasyRSA-3.0.5; ./easyrsa init-pki
```

Im Anschluss wird der private Schlüssel und der dazugehörige Zertifikatsantrag erzeugt. Der Platzhalter `entityName` muss durch Ihre Hochschul-E-Mail-Adresse ersetzt werden.

```
./easyrsa gen-req entityName [nopass]
```

Der Parameter `nopass` ist optional und kann angegeben werden, um auf den Passwortschutz für den erzeugten privaten Schlüssel zu verzichten. In diesem Rahmen geben Sie bei der Abfrage des `Common Name` Ihren **vollen Namen** an. Bei der Abfrage der `Email Address` geben Sie Ihre **Hochschul-E-Mail-Adresse** an. Für alle weiteren Abfragen werden die Vorgaben unverändert übernommen.

Der erzeugte Zertifikatsantrag wird dann per E-Mail an das IT-Team (`F4-I-IT-Team@hs-hannover.de`) verschickt. Hat alles geklappt, erhalten Sie Ihr neues Benutzerzertifikat vom IT-Team.

OpenVPN-Client einrichten

In diesem Schritt wird die OpenVPN-Clientkonfiguration und das Wurzelzertifikat der CA benötigt.

```
wget http://vpnca.inform.hs-hannover.de/client.conf
wget http://vpnca.inform.hs-hannover.de/ca.crt
```

Die bereitgestellte Konfigurationsdatei `client.conf` enthält bereits das vorgefertigtes Gerüst und wird zusammen mit dem Wurzelzertifikat, Ihrem privaten Schlüssel und Ihrem Benutzerzertifikat in `/etc/openvpn/client/` platziert. Die Parameter `ca`, `cert` und `key` in der `client.conf` müssen entsprechend angepasst werden. Die Dateipfade können relativ oder absolut angegeben werden. Ist der private Schlüssel mit einem Passwort geschützt kann die Option `auth-nocache` eingefügt werden, um das Vorhalten dieses Passworts im Arbeitsspeicher zu verhindern. Details dazu können in der Manpage[1] von OpenVPN nachgelesen werden.

Achtung: Auch unter **Windows** müssen Schrägstriche `/` anstelle von Gegenschrägstrichen `\` verwendet werden.

Unter Debian kann der OpenVPN-Client als Systemdienst aktiviert und gestartet werden:

```
systemctl enable openvpn-client@client.service; systemctl start openvpn-client@client.service
```

Alternativ kann OpenVPN unter Angabe der Konfigurationsdatei direkt gestartet werden:

```
openvpn --config /path/to/client.conf
```

[1] `https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage`

HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Dokumentation der Zertifizierungsstelle für den IPv6-VPN-Dienst

Jan Philipp Timme

6. November 2018

Inhaltsverzeichnis

1 Vorwort

Dieses Dokument wurde im Rahmen der Masterarbeit „Konzeption und Umsetzung eines IPv6-VPN für die Abteilung Informatik“ erstellt. Im Rahmen der Masterarbeit wird beschrieben, wie ein IP6-VPN-Dienst auf Basis von OpenVPN konzipiert und eingerichtet wird. Für den Betrieb dieses Dienst wird eine Zertifizierungsstelle (*Certificate Authority*, kurz CA) benötigt, die für Benutzer und den Serverbetreiber SSL-Zertifikate ausstellen kann.

Diese CA wird auf einer virtuellen Maschine installiert, welche vom IT-Team den Hostnamen `vpnca.inform.hs-hannover.de` zugeteilt bekommen hat, und nur im Netz der Abteilung Informatik erreichbar ist.

In diesem Dokument wird die Einrichtung und der Betrieb der CA beschrieben. Weiterhin sind Anleitungen für Benutzer und Serveradministratoren des VPN-Dienstes enthalten, in denen die Beantragung von Client- und Server-Zertifikaten bei der CA beschrieben wird.

Die in diesem Dokument beschriebene Konfiguration der CA wurde im Rahmen der Masterarbeit konzipiert. Entscheidungen, die zu der hier beschriebenen Konfiguration geführt haben können in der Masterarbeit nachvollzogen werden.

Die Kapitel in diesem Dokument sind je nach Zielgruppe unterschiedlich interessant. In der folgenden Liste werden sie zur Vereinfachung der Navigation aufgelistet.

- **Benutzer**: Beantragen von Zertifikaten
- **Serveradministratoren**: Beantragen von Zertifikaten, Erzeugen der DH-Parameter
- **CA-Betreiber**: Aufbau der Zertifizierungsstelle, Ausstellen von Zertifikaten, Erzeugen der CRL

2 Aufbau der Zertifizierungsstelle

Bevor Zertifikate durch die CA ausgestellt werden könnnen, muss die CA erst eingerichtet werden. Dies geschieht auf Basis von EasyRSA, welches von den OpenVPN-Entwicklern zur Verfügung gestellt wird. EasyRSA ist eine Sammlung von Skripten, die unter Verwendung von OpenSSL den Funktionsumfang einer Zertifizierungsstelle bieten. Durch die Kapselung von OpenSSL-Befehlen und die Verwendung einer übersichtlichen Konfigurationsdatei ermöglicht EasyRSA das komfortable und sichere Verwalten einer CA.

Der Aufbau der Zertifizierungsstelle wurde für die Umsetzung unter Debian 9 konzipiert. Für die Durchführung aller Schritte dieser Anleitung werden die folgenden Programme benötigt.

- `openssl`
- `gnupg`
- `wget`
- `zip`

2.1 Installation der CA

Aktuelle Versionen von EasyRSA sind auf GitHub unter „Releases“[1] zu finden. Neben den auf GitHub zur Verfügung stehenden Releases werden auch GPG-Signaturen angeboten, über welche die Releases authentisiert werden können. Im oben erwähnten Repository sind unterhalb von `./release-keys/README.md`[2] passende GPG-Schlüssel aufgeführt. Zum Zeitpunkt der Erstellung dieses Dokuments (17.09.2018) ist Version 3.0.5 aktuell - alle in diesem Dokument aufgeführten Befehle beziehen sich auf diese Version. Für die Verwendung dieser Dokumentation mit einer höheren Version von EasyRSA sind gegebenenfalls Anpassungen an diesem Dokument erforderlich.

In dieser Anleitung wird davon ausgegangen, dass die CA als Benutzer `root` unterhalb von `/root/VPN-CA/` eingerichtet wird.

Im ersten Schritt wird die aktuelle Version von EasyRSA beschafft und authentisiert.

[1] `https://github.com/OpenVPN/easy-rsa/releases`
[2] `https://github.com/OpenVPN/easy-rsa/tree/v3.0.5/release-keys`

```
cd /root
# Aktuelles Release v3.0.5 beschaffen
wget https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.5/ \
 EasyRSA-nix-3.0.5.tgz
wget https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.5/ \
 EasyRSA-nix-3.0.5.tgz.sig

# Aktuell gültige(n) GPG-Schlüssel für die Signaturprüfung beschaffen
gpg --recv-keys 6F4056821152F03B6B24F2FCF8489F839D7367F3

# Signatur der heruntergeladenen Datei überprüfen
gpg --verify EasyRSA-nix-3.0.5.tgz.sig EasyRSA-nix-3.0.5.tgz
```

Nach der erfolgreichen Authentisierung des heruntergeladenen Archivs kann dieses nun entpackt und verwendet werden.

```
tar xzf EasyRSA-nix-3.0.5.tgv; mv EasyRSA-3.0.5 VPN-CA; cd VPN-CA
```

2.2 Konfiguration der CA

Im nächsten Schritt wird auf Basis der Datei `vars.example` die Konfigurationsdatei `vars` erzeugt und anschließend bearbeitet.

```
cp vars.example vars
vim vars
```

In der so erzeugten Konfigurationsdatei werden nun die folgenden Einträge auskommentiert und entsprechend angepasst.

Zertifikate verwenden das volle Organisationsschema als *Distinguished Name* (DN). Dadurch können Zertifikate bei Betrachtung leicht der Abteilung Informatik zugeordnet werden.

```
set_var EASYRSA_DN "org"
```

Die folgenden Werte sind Standardvorgaben des Organisationsschema für neue Zertifikatsanträge. Lediglich der *Common Name* (CN) wird für alle Zertifikate individuell gewählt.

```
set_var EASYRSA_REQ_COUNTRY "DE"
set_var EASYRSA_REQ_PROVINCE "Niedersachsen"
set_var EASYRSA_REQ_CITY "Hannover"
set_var EASYRSA_REQ_ORG "Hochschule Hannover"
set_var EASYRSA_REQ_EMAIL "F4-I-IT-Team@hs-hannover.de"
set_var EASYRSA_REQ_OU "Abteilung Informatik"
```

Alle erzeugten Zertifikate basieren auf RSA-Schlüsselpaaren mit 4096 Bit Schlüssellänge.

```
set_var EASYRSA_KEY_SIZE 4096
set_var EASYRSA_ALGO  rsa
```

Das Wurzelzertifikat der CA ist für 20 Jahre gültig.

```
set_var EASYRSA_CA_EXPIRE 7300
```

Ausgestellte Zertifikate sind für fünf Jahre gültig[3].

```
set_var EASYRSA_CERT_EXPIRE 1825
```

Die von der CA erzeugte *Certificate Revocation List* (CRL) ist ein halbes Jahr lang gültig.

```
set_var EASYRSA_CRL_DAYS 180
```

Mit der Durchführung der aufgeführten Änderungen ist die Konfiguration der CA abgeschlossen.

2.3 Initialisierung der CA

Als nächstes muss die Verzeichnisstruktur der CA initialisiert werden.

```
./easyrsa init-pki
```

Im Anschluss kann das Wurzelzertifikat der CA erzeugt werden.

```
# ./easyrsa build-ca nopass
./easyrsa build-ca
```

Hinweis 1: Besteht der begründete Wunsch, den privaten Schlüssel der CA **nicht** mit einem Passwort zu schützen, so kann das Argument `nopass` an den Befehl `build-ca` angehängt werden. Dies kann nützlich sein um die regelmäßige Ausstellung einer CRL zu automatisieren. Sollte der private Schlüssel in die Hände eines Angreifers gelangen, so kann dieser beliebige Zertifikate durch die CA ausstellen. Ein Passwortschutz wird ausdrücklich empfohlen sofern keine sonstigen Maßnahmen zum Schutz des privaten Schlüssels der CA vor unbefugtem Zugriff getroffen werden!

Hinweis 2: Im Rahmen der Erzeugung des Wurzelzertifikats werden Angaben zum Zertifikat abgefragt. Hier muss ein passender CN angegeben werden, wie zum Beispiel „IPv6-VPN CA“. Alle weiteren Vorgaben werden unverändert übernommen.

[3] Dieser Standardwert wird für Zertifikate für Mitarbeiter und Server angewandt. Für Studenten ist eine maximale Laufzeit von 730 Tage (zwei Jahren) festgelegt.

2.4 Bereitstellung der CA-Konfiguration für Benutzer

Damit Benutzer korrekte Zertifikatsanträge erzeugen können, muss ihnen die Datei `vars` zur Verfügung gestellt werden. Um Kompatibilitätsprobleme durch die Verwendung von verschiedenen EasyRSA-Versionen auszuschließen, wird empfohlen für Benutzer ein passendes Paket zusammenzustellen.

Als Basis wird jeweils das `*.zip`-Release der aktuellen Version von EasyRSA empfohlen, da dieses für Windows-Benutzer notwendige Abhängigkeiten wie `awk` und `sed` bereits enthält.

Zunächst wird das aktuelle `*.zip`-Release von EasyRSA beschafft und authentisiert.

```
wget https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.5/ \
 EasyRSA-Windows-3.0.5.zip
wget https://github.com/OpenVPN/easy-rsa/releases/download/v3.0.5/ \
 EasyRSA-Windows-3.0.5.zip.sig

gpg --recv-keys 6F4056821152F03B6B24F2FCF8489F839D7367F3

gpg --verify EasyRSA-Windows-3.0.5.zip.sig EasyRSA-Windows-3.0.5.zip
```

Nun kann die konfigurierte `vars`-Datei der CA zu dem beschafften `*.zip`-Archiv hinzugefügt werden.

```
# Temporäres Verzeichnis erzeugen
TD=‘mktemp -d‘

# EasyRSA in temporäres Verzeichnis entpacken
unzip EasyRSA-Windows-3.0.5.zip -d $TD

# Vars-Datei hinzufügen
cp /root/VPN-CA/vars $TD/EasyRSA-3.0.5/

# Ergebnis in neue Zip-Datei einpacken
cd $TD
zip -r VPN-EasyRSA.zip EasyRSA-3.0.5

# Neue Zip-Datei an sicheren Ort verschieben
cd /root
cp $TD/VPN-EasyRSA.zip .

# Temporäres Verzeichnis entfernen
rm -rf $TD
```

Anschließend kann das angefertigte `*.zip`-Archiv den Benutzern zum Beispiel durch einen Webserver zur Verfügung gestellt werden.

2.5 Einrichtung eines Webservers

Um öffentliche Dateien wie das Wurzelzertifikat, die CRL und das konfigurierte EasyRSA-Paket den VPN-Benutzern zugänglich zu machen, wird ein Webserver auf der CA-Maschine installiert. Der Hostname `vpnca.inform.hs-hannover.de` wurde für die CA-Maschine vom IT-Team vorgegeben.

Zuerst wird die Webserversoftware über den Debian-Paketmanager installiert und die standardmäßig aktiven `VirtualHost`-Konfigurationen deaktiviert.

```
apt-get install apache2
rm /etc/apache2/sites-enabled/*
```

Im nächsten Schritt wird ein öffentliches, nur lesbares Verzeichnis erzeugt, welches später alle öffentlich verfügbaren Daten der CA enthalten soll und vom Webserver ausgeliefert wird.

```
mkdir /public; chown nobody:nogroup /public; chmod 555 /public;
```

Nun wird eine neue Konfigurationsdatei mit dem folgenden Inhalt in die neue Datei `/etc/apache2/sites-available/vpnca.inform.hs-hannover.de.conf` abgelegt.

```
<VirtualHost *:80>
 ServerName vpnca.inform.hs-hannover.de
 ServerAdmin F4-I-IT-Team@hs-hannover.de
 DocumentRoot /public

 <Directory /public>
  Options +Indexes
  Require all granted
 </Directory>

 ErrorLog ${APACHE_LOG_DIR}/error.log
 CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Nun kann der Webserver-Dienst aktiviert und gestartet werden.

```
systemctl enable apache2.service; systemctl start apache2.service
```

Jetzt werden öffentliche Daten der CA als Kopie in `/public` abgelegt. Es wird davon ausgegangen, dass das Wurzelzertifikat als `ca.crt`, die CRL als `crl.pem` und die konfigurierten EasyRSA-Skripte als `VPN-EasyRSA.zip` abgelegt werden. Auch die OpenVPN-Clientkonfiguration wird als `client.conf` hier abgelegt. Zuletzt müssen die Dateirechte angepasst werden, damit alle Dateien öffentlich lesbar sind.

```
chmod 444 /public/*
```

3 Beantragen von Zertifikaten

Um ein Zertifikat von der CA zu beantragen, wird das bereits vorkonfigurierte EasyRSA-Paket benötigt, welches durch das IT-Team der Abteilung Informatik bereitgestellt wird. Es kann von `http://vpnca.inform.hs-hannover.de/VPN-EasyRSA.zip` heruntergeladen und in einem lokalen Ordner entpackt werden. Eine lauffähige Installation von OpenSSL, welches auch als Abhängigkeit von OpenVPN benötigt wird, wird ebenfalls vorausgesetzt.

Hinweis für Windows-Benutzer: Der Pfad zum OpenSSL-Programm muss in der Datei `vars` von EasyRSA hinterlegt werden. Dafür kann auf das von OpenVPN installierte OpenSSL zurückgegriffen werden[1]. Die folgenden Zeile zeigt beispielhaft die Verwendung des durch OpenVPN installierten OpenSSL-Programms:

```
set_var EASYRSA_OPENSSL "C:/Program Files/OpenVPN/bin/openssl.exe"
```

Nun öffnet man ein Terminal, wechselt in das eben entpackte Verzeichnis von EasyRSA und kann den Befehl `./easyrsa` verwenden, um einen Zertifikatsantrag zu erzeugen. Unter Windows kann eine passende Konsole über die Datei `EasyRSA-Start.bat` aufgerufen werden.

Zuerst muss die benötigte Verzeichnisstruktur erzeugt werden.

```
./easyrsa init-pki
```

Im nächsten Schritt wird ein neuer Zertifikatsantrag inklusive neuem Schlüsselpaar erzeugt.

```
# ./easyrsa gen-req entityName nopass
./easyrsa gen-req entityName
```

Für die Beantragung eines **Clientzertifikats** muss der Platzhalter `entityName` durch den E-Mail-Adresse des Benutzers an der Hochschule Hannover ersetzt werden.

Für die Beantragung eines **Serverzertifikats** muss der Platzhalter `entityName` durch den vollqualifizierten Domainnamen des Servers ersetzt werden. Dieser kann beispielsweise `aither.inform.hs-hannover.de` lauten.

[1] Mehr dazu unter
`https://github.com/OpenVPN/easy-rsa/blob/v3.0.5/distro/windows/README-Windows.txt`

Im Rahmen der Erzeugung des Zertifikatsantrags werden zusätzliche Angaben abgefragt. Im Feld `Common Name` muss der volle Name des Antragsstellers (Clientzertifikat) oder der vollqualifizierte Domainname des Servers (Serverzertifikat) angegeben werden. Im Feld `Email Address` muss die Hochschul-E-Mail-Adresse des Antragsstellers angegeben werden. Alle weiteren Vorgaben werden unverändert übernommen.

Hinweis 1: Besteht der begründete Wunsch, den erzeugten, privaten Schlüssel **nicht** mit einem Passwort zu schützen, so kann das Argument `nopass` an den Befehl `gen-req` angehängt werden. Dies kann nützlich sein um den privaten Schlüssel ohne zusätzliche Passworteingabe zu benutzen - zum Beispiel zum Betrieb eines OpenVPN-Servers oder zum automatischen Verbindungsaufbau mit einem OpenVPN-Client. Sollte der private Schlüssel in die Hände eines Angreifers gelangen, so kann dieser ebenfalls das dazugehörige Zertifikat missbrauchen. Ein Passwortschutz wird ausdrücklich empfohlen sofern keine sonstigen Maßnahmen zum Schutz des privaten Schlüssels vor unbefugtem Zugriff getroffen werden!

Hinweis 2: Wurde bereits zuvor ein Zertifikatsantrag mit dem selben `entityName` erzeugt, so kann dieser inklusive dem dazugehörigen, privaten Schlüssel **überschrieben** werden. Dafür ist eine Bestätigung der Sicherheitsabfrage durch die Eingabe von `yes` erforderlich.

In diesem Schritt wurden nun zwei neue Dateien erzeugt:

- `*.key`: Der private Schlüssel. Nur mit ihm kann das später ausgestellte Zertifikat benutzt werden. Diese Datei sollte immer an einem sicheren Ort gespeichert werden, um Missbrauch durch Dritte zu verhindern. Bei Verlust dieser Datei wird das dazugehörige Zertifikat **unbrauchbar**! Aus diesem Grund wird die Durchführung von Backups empfohlen.
- `*.req`: Der Zertifikatsantrag, der zu dem neuen privaten Schlüssel gehört. Zum Ausstellen eines gültigen Zertifikats muss er an die CA übergeben werden. Nachdem ein gültiges Zertifikat aus dem Antrag erzeugt wurde, wird diese Datei nicht länger benötigt.

4 Ausstellen von Zertifikaten

Hat ein Benutzer einen Zertifikatsantrag erzeugt, so kann auf Basis dieses Antrags ein gültiges Zertifikat durch die CA ausgestellt werden. Hierfür muss der Zertifikatsantrag zunächst aus der vom Benutzer eingereichten `*.req`-Datei importiert werden. Dafür wird der Befehl `import-req` verwendet:

```
./easyrsa import-req /tmp/example.req entityName
```

Abhängig vom Typ des beantragten Zertifikats muss der Platzhalter `entityName` durch die CA-Betreiber gewählt werden. Der `entityName` wird auch zum Widerrufen bereits ausgestellter Zertifikate verwendet.

Für die Beantragung eines **Clientzertifikats** muss der Platzhalter `entityName` durch die E-Mail-Adresse des Benutzers an der Hochschule Hannover ersetzt werden, der das Zertifikat beantragt.

Für die Beantragung eines **Serverzertifikats** muss der Platzhalter `entityName` durch den vollqualifizierten Domainnamen des Servers ersetzt werden, für den das Zertifikat beantragt wird. Dieser kann zum Beispiel `aither.inform.hs-hannover.de` lauten.

Der importierte Zertifikatsantrag kann im Anschluss betrachtet werden.

```
./easyrsa show-req entityName
```

Hinweis: Falls bereits ein noch gültiges Zertifikat mit dem selben `CN` existiert, muss dieses vorher widerrufen werden.

```
./easyrsa revoke entityName
```

Ist der private Schlüssel der CA mit einem Passwort geschützt, so wird bei diesem Schritt danach verlangt.

Nun kann der importierte Zertifikatsantrag als Client- oder Serverzertifikat signiert werden.

```
# Clientzertifikat für Studenten ausstellen
# (reduzierte Laufzeit von 2 Jahren)
EASYRSA_CERT_EXPIRE=730 ./easyrsa sign-req client entityName

# Clientzertifikat für Mitarbeiter ausstellen
./easyrsa sign-req client entityName

# Serverzertifikat ausstellen
./easyrsa sign-req server entityName
```

Ist der private Schlüssel der CA mit einem Passwort geschützt, so wird bei diesem Schritt danach verlangt.

Hinweis: Die Gültigkeitsdauer kann nach Ermessen der CA-Betreiber in begründeten Einzelfällen frei gewählt werden, indem die Umgebungsvariable `EASYRSA_CERT_EXPIRE` auf die gewünschte Gültigkeitsdauer in Tagen gesetzt wird. Bei der Wahl einer erhöhten Gültigkeitsdauer wird empfohlen, eine maximale Gültigkeitsdauer von 1825 Tagen (fünf Jahre) nicht zu überschreiten. In diesem Beispiel wird ein Serverzertifikat mit einer Laufzeit von 730 Tagen ausgestellt.

```
EASYRSA_CERT_EXPIRE=730 ./easyrsa sign-req server entityName
```

5 Erzeugen der CRL

Der OpenVPN-Server verwendet die von der CA ausgestellte *Certificate Revocation List* (CRL) zur Überprüfung der Gültigkeit von Clientzertifikaten. In der Konfigurationsdatei des OpenVPN-Servers wird die CRL-Datei durch die Option `verify-crl /path/to/crl.pem` angegeben.

Somit können durch die CA widerrufene Zertifikate nicht mehr zur Anmeldung am VPN-Dienst benutzt werden. Mit dem folgenden Befehl wird die CRL erzeugt.

```
./easyrsa gen-crl
```

Ist der private Schlüssel der CA mit einem Passwort geschützt, so wird bei diesem Schritt danach verlangt.

Achtung: Der OpenVPN-Server lehnt eine abgelaufene CRL-Datei ab und verweigert dann den Dienst. Deshalb sollte die CRL durch die CA regelmäßig erneuert werden und die Kopie der CRL auf dem VPN-Server regelmäßig aktualisiert werden. Anschließend sollte der VPN-Dienst via `systemctl reload openvpn@inform.service` neu geladen werden. Eine Automatisierung dieses Vorgangs ist möglich und wurde auf der CA-Maschine entsprechend umgesetzt.

Um die CRL automatisiert zu erneuern und über den Webserver öffentlich zur Verfügung zu stellen, werden als Benutzer `root` Cronjobs angelegt. Dazu wird das folgende Kommando verwendet.

```
crontab -e
```

In dem nun offenen Editor werden die beiden folgenden Cronjobs eingetragen, welche sich um die Erzeugung der CRL sowie die Platzierung der neuen CRL im durch den Webserver ausgelieferten Verzeichnis kümmern.

```
# Aktualisiere CRL täglich
00 1    * * *   bash -c 'cd /root/VPN-CA; ./easyrsa gen-crl'

# Platziere die frische CRL in öffentlichem Ordner /public
10 1    * * *   cp -f /root/VPN-CA/pki/crl.pem /public/; chmod 444
   /public/crl.pem
```

6 Erzeugen der DH-Parameter

Um Chiffren mit *Perfect Forward Secrecy* zu verwenden benötigt der OpenVPN-Server eine Datei mit Diffie-Hellman-Parametern. Da die Berechnung dieser Parameter sehr lange dauert, ist es nicht sinnvoll sie bei jedem Sitzungsaufbau zwischen OpenVPN-Client und -Server zu berechnen. Deshalb werden diese Parameter vorab generiert und in einer Datei abgelegt. Der folgende Befehl erledigt diese Aufgabe. Die Berechnung von 4096 Bit-Parametern sollte je nach Hardware höchstens 25 Minuten in Anspruch nehmen[1].

```
./easyrsa gen-dh
```

Anschließen kann die erzeugte Datei in das Konfigurationsverzeichnis des OpenVPN-Servers hinzugefügt werden und über den Parameter `dh /path/to/dh.pem` in der Serverkonfiguration bekannt gemacht werden.

[1] 23 Minuten wurden mit der CPU „Intel(R) Core(TM) i5 CPU M 450 @ 2.40GHz“ gemessen.

HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

IPv6-VPN Serverdokumentation

Jan Philipp Timme

6. November 2018

Inhaltsverzeichnis

1 Vorwort

Diese Dokumentation beschreibt die Installation und den Betrieb des IPv6-VPN-Dienst auf Basis von OpenVPN. Die Hintergründe zu der hier vorgestellten Konfiguration können in der dazugehörigen Masterarbeit „Konzeption und Umsetzung eines IPv6-VPN für die Abteilung Informatik“ nachgelesen werden.

1.1 Konfigurationsvorgaben

Folgende Parameter wurden für die Konfiguration des Servers in Absprache mit dem IT-Team festgelegt:

- Hostname der Maschine: `aither.inform.hs-hannover.de`
- Hostname des OpenVPN-Dienstes: `vpn-test.inform.hs-hannover.de`
- IP-Adressen der Maschine
  - `141.71.38.70/24`
  - `2001:638:614:1780::131/64`
- IP-Adressen der zu benutzenden Gateways
  - `141.71.38.254`
  - `2001:638:614:1780::1`
- IP-Adressen des OpenVPN-Dienstes
  - `141.71.38.7`
  - `2001:638:614:1780::7`
- IP-Adressbereiche für VPN-Clients
  - `10.2.0.0/16`
  - `2001:638:614:1750::/64`
- Zu verwendender DNS-Server: `141.71.38.1`

2 Installation

Mit den abgesprochenen Parametern kann nun die Installation des VPN-Servers erfolgen. Eine bereits vorhandene Zertifizierungsstelle mit Webserver unter dem Hostnamen `vpnca.inform.hs-hannover.de` wird vorausgesetzt.

2.1 Konfiguration des Grundsystems

s

Hostname: Sofern der Hostname bei der Installation von Debian nicht schon gesetzt wurde, so muss dies in den Dateien `/etc/hostname`, `/etc/mailname` und `/etc/hosts` nachgeholt werden.

```
echo "aither" > /etc/hostname
echo "aither.inform.hs-hannover.de" > /etc/mailname
```

In `/etc/hosts` muss der Eintrag für `127.0.1.1` angepasst werden:

```
127.0.1.1 aither.inform.hs-hannover.de aither
```

OpenSSH: In der Datei `/etc/ssh/sshd_config` muss folgende Option auskommentiert und angepasst werden:

```
PermitRootLogin yes
```

Anschließend wird der OpenSSH-Dienst aktiviert und gestartet.

```
systemctl enable ssh.service
systemctl start ssh.service
```

sudo: Das IT-Team arbeitet mit passwortbasierten SSH-Sitzungen unter dem Benutzer `root`. Damit in der Übergangsphase auf dem Server Anpassungen auch ohne Kenntnis des `root`-Passworts durchgeführt werden können, wird ein lokaler Benutzer eingerichtet.

```
apt-get install sudo
adduser jpt
gpasswd -a jpt sudo
```

Nach erfolgreicher Übergabe des Servers an das IT-Team kann dieser Benutzer wieder entfernt werden.

apt: Um in der DMZ weiterhin Updates einspielen zu können, wird der vom IT-Team zur Verfügung gestellte Proxyserver in die Konfiguration von `apt` eingetragen.

```
echo 'Acquire::http::Proxy "http://proxy.inform.hs-hannover.de:3128";'
 > /etc/apt/apt.conf.d/80proxy
```

Das IT-Team stellt Debian-Pakete zur Verfügung, mit die Grundkonfiguration des Servers an die Vorgaben des IT-Teams angepasst werden kann. Um diese Pakete zu installieren, werden die Paketquellen des IT-Teams konfiguriert:

```
echo "deb http://http.edu.inform.hs-hannover.de/depot/debian/stretch/
   Packages/"
 > /etc/apt/sources.list.d/inform.list
```

Als nächstes wird der GPG-Key importiert, mit dem die Pakete signiert sind:

```
wget -O repositoryKeyFile
   http://http.edu.inform.hs-hannover.de/repository/repositoryKeyFile
apt-key add repositoryKeyFile
```

Anschließend können die Pakete über `apt-get` installiert werden

```
apt-get update
apt-get install f4-i-srv-config-all-* f4-i-srv-config-dmz-adminscripts
```

Systemzeit über Zeitserver beziehen: Für Vorgänge wie zum Beispiel die Gültigkeitsprüfung von Zertifikaten ist eine korrekt gestellte Systemuhr wichtig. Dafür wird dem bereits installierten Dienst `systemd-timesyncd` der Zeitserver der Abteilung Informatik bekannt gemacht. Die Datei `/etc/systemd/timesyncd.conf` enthält dafür folgenden Abschnitt:

```
[Time]
NTP=time.inform.hs-hannover.de
```

Netzwerkkonfiguration: Als nächstes werden die IP-Adressen der Maschine und des Dienstes in `/etc/network/interfaces` konfiguriert. Die IP-Adressen der Maschine werden direkt für die Netzwerkkarte des Servers konfiguriert.

Die IP-Adressen für den VPN-Dienst werden als zusätzliche IP-Adressen konfiguriert. Durch die Verwendung von separaten Dienst-Adressen ist es möglich, einen Server im laufenden Betrieb durch einen zweiten, identisch konfigurierten Server zu ersetzen. Dazu müssen lediglich die zusätzlichen IP-Adressen von dem einen Server entfernt werden, und anschließend auf dem zweiten Server hinzugefügt werden. VPN-Benutzer bekommen davon nur eine kurze Unterbrechung der VPN-Sitzung mit.

Die resultierende Konfiguration für die IP-Adressen der Maschine sieht so aus:

```
auto eno1
allow-hotplug eno1

#-primary network interface
iface eno1 inet static
        address 141.71.38.70/24
        gateway 141.71.38.254
        post-up /sbin/ip addr add 141.71.38.7/24 dev eno1
        pre-down /sbin/ip addr del 141.71.38.7/24 dev eno1

iface eno1 inet6 static
        address 2001:638:614:1780::0131/64
        gateway 2001:638:614:1780::1
        post-up /sbin/ip addr add 2001:638:614:1780::0007/64 dev eno1
        pre-down /sbin/ip addr del 2001:638:614:1780::0007/64 dev eno1
```

DNS: In der DMZ soll der DNS-Resolver mit der IP-Adresse `141.71.38.1` verwendet werden.

```
echo "nameserver 141.71.38.1" > /etc/resolv.conf
```

Paketweiterleitung einschalten: Mit dem VPN-Server verbundene Clients befinden sich in einem eigenen IPv4- beziehungsweise IPv6-Netz. Damit die VPN-Clients trotzdem über die Grenze ihres eigenen Netzes hinaus Kontakt zum Netz der Abteilung Informatik aufnehmen können, ist es notwendig für IPv4 und IPv6 die Paketweiterleitung einzuschalten.

```
echo "net.ipv4.conf.all.forwarding = 1" >
   /etc/sysctl.d/04-enable-ipv4-forwarding.conf
echo "net.ipv6.conf.all.forwarding = 1" >
   /etc/sysctl.d/06-enable-ipv6-forwarding.conf
```

Anschließend werden die vorgenommenen Einstellungen aktiviert.

```
sysctl --system
```

2.2 Firewallregeln

Welcher Datenverkehr für VPN-Clients erlaubt oder verboten ist, wurde im Rahmen der Masterarbeit bereits festgelegt. Diese Vorgaben werden jetzt über Filterregeln mit `iptables` und `ip6tables` umgesetzt.

Als Standardpolicy wird `DROP` gewählt.

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
ip6tables -P INPUT DROP
ip6tables -P OUTPUT DROP
ip6tables -P FORWARD DROP
```

Datenverkehr über das Loopback-Interface ist immer erlaubt.

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
ip6tables -A INPUT -i lo -j ACCEPT
ip6tables -A OUTPUT -o lo -j ACCEPT
```

Die Protokolle ICMP und ICMPv6 sind immer erlaubt.

```
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A FORWARD -p icmp -j ACCEPT
ip6tables -A INPUT -p icmpv6 -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 -j ACCEPT
ip6tables -A FORWARD -p icmpv6 -j ACCEPT
```

Zugriffe auf den VPN-Server sind für die Dienste SSH und OpenVPN erlaubt.

```
iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED
   -j ACCEPT
ip6tables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED
   -j ACCEPT

iptables -A INPUT -p udp --dport 1194 -j ACCEPT
ip6tables -A INPUT -p udp --dport 1194 -j ACCEPT
```

Antwortpakete für eingehende Pakete auf SSH und OpenVPN-Dienst sind erlaubt.

```
iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j
   ACCEPT
ip6tables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j
   ACCEPT

iptables -A OUTPUT -p udp --sport 1194 -j ACCEPT
ip6tables -A OUTPUT -p udp --sport 1194 -j ACCEPT
```

Vom VPN-Server ausgehende Pakete sind grundsätzlich erlaubt.

```
iptables -A OUTPUT -m state --state NEW,ESTABLISHED -j ACCEPT
ip6tables -A OUTPUT -m state --state NEW,ESTABLISHED -j ACCEPT
```

Zum VPN-Server eingehende Pakete sind als Antwort auf ausgehende Pakete erlaubt.

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Für IPv4-Verkehr aus dem VPN in andere Netze soll NAT auf die `141.71.38.7` durchgeführt werden.

```
iptables -t nat -A POSTROUTING -s 10.2.0.0/16 -d ! 10.2.0.0/16 -j
   SNAT --to 141.71.38.7
```

Datenverkehr zwischen VPN-Clients ist verboten und wird verworfen.

```
iptables -A FORWARD -s 10.2.0.0/16 -d 10.2.0.0/16 -j DROP
ip6tables -A FORWARD -s 2001:638:614:1750::/64 -d
   2001:683:614:1750::/64 -j DROP
```

Jeglicher weiterer Datenverkehr aus dem VPN ist erlaubt.

```
iptables -A FORWARD -s 10.2.0.0/16 -m state --state NEW,ESTABLISHED
   -j ACCEPT
ip6tables -A FORWARD -s 2001:638:614:1750::/64 -m state --state
   NEW,ESTABLISHED -j ACCEPT
```

In das VPN eingehender Verkehr ist nur als Antwort auf ausgehende Pakete erlaubt.

```
iptables -A FORWARD -d 10.2.0.0/16 -m state --state
   ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -d 2001:638:614:1750::/64 -m state --state
   ESTABLISHED,RELATED -j ACCEPT
```

Persistente Firewallregeln: Um die mit `iptables` und `ip6tables` umgesetzten Regeln auch über Neustarts hinweg zu behalten, wird das Paket `iptables-persistent` installiert.

```
apt-get install iptables-persistent
```

Anschließend können die aktuell aktiven Regeln in Dateien abgelegt werden, aus denen sie nach einem Neustart durch die von `iptables-persistent` installierten Skripte wieder geladen werden.

```
iptables-save > /etc/iptables/rules.v4
ip6tables-save > /etc/iptables/rules.v6
```

2.3 OpenVPN

Nun wird OpenVPN installiert und konfiguriert.

```
apt-get install openvpn
```

Anhand des separaten Dokuments „Dokumentation der Zertifizierungsstelle für den IPv6-VPN-Dienst“ wird ein neues Serverzertifikat für den VPN-Server beantragt und anschließend durch die Zertifizierungsstelle ausgestellt. Zusätzlich werden auf dem lokalen Server die Diffie-Hellman-Parameter generiert.

Außerdem wird das Wurzelzertifikat der Zertifizierungsstelle, sowie eine aktuelle *Certificate Revocation List* (CRL) benötigt. In diesem Szenario wird davon ausgegangen, dass beide Dateien in aktueller Version über einen HTTP-Server von der CA angeboten werden:

```
wget http://vpnca.inform.hs-hannover.de/ca.crt
wget http://vpnca.inform.hs-hannover.de/crl.pem
```

Die beschafften Dateien werden nun wie folgt auf dem VPN-Server abgelegt:

- Wurzelzertifikat: `/etc/openvpn/inform/ca.crt`
- Serverzertifikat: `/etc/openvpn/inform/aither.inform.hs-hannover.de.crt`
- Privater Schlüssel: `/etc/openvpn/inform/aither.inform.hs-hannover.de.key`
- Diffie-Hellman-Parameter: `/etc/openvpn/inform/dh.pem`
- Certificate Revocation List: `/etc/openvpn/inform/crl.pem`

Die folgende Konfiguration wird als Datei unter `/etc/openvpn/inform.conf` abgelegt.

```
# Listen on 1194 for both IPv4 and IPv6
port 1194
proto udp
proto udp6

# Since we have more than one ip address, this makes openvpn respond
# with the right sender address
multihome

# We're using the virtual network interface on layer 3
dev tun

# Certificates to use. Paths are relative to config file location.
ca inform/ca.crt
cert inform/aither.inform.hs-hannover.de.crt
key inform/aither.inform.hs-hannover.de.key

# Assume server role in tls handshake
tls-server
```

```
# Diffie-Hellman parameter file
# (not needed for TLS cipher with ECDHE instead of DHE)
dh inform/dh.pem

# Certificate revocation list location
# Make sure this file is always valid, otherwise OpenVPN refuses to
   (re)start!
crl-verify inform/crl.pem

# Make sure the client presents a certificate with "client role"
remote-cert-tls client

# Allow multiple connections using the same certificate?
# There is no reason to not allow this, so it is allowed.
duplicate-cn

# We're using subnet topology for IPv4 tunnel connectivity
topology subnet

# Use this IPv4 range for clients (/16, so we can cope with potential
   500 clients)
server 10.2.0.0 255.255.0.0

# Use this IPv6 network for clients
server-ipv6 2001:638:614:1750::/64

# Make sure the client can still reach the OpenVPN server via its
   IPv4 default gateway
# This is needed because the IPv4 route for DMZ is pushed below,
# which overlaps the OpenVPN server IPv4 address.
push "route remote_host 255.255.255.255 net_gateway"

# Push routes for local IPv4 networks
#
# DMZ
push "route 141.71.38.0 255.255.255.0 vpn_gateway"
# Inform
push "route 141.71.30.0 255.255.254.0 vpn_gateway"
# Edu
push "route 192.168.99.0 255.255.255.0 vpn_gateway"
# NAO
push "route 192.168.90.0 255.255.255.0 vpn_gateway"
# iDrac
push "route 192.168.70.0 255.255.255.0 vpn_gateway"
# Cluster
push "route 10.0.20.0 255.255.255.0 vpn_gateway"
# educloud
push "route 10.0.30.0 255.255.255.0 vpn_gateway"
# experimental ipv6 network
push "route 10.0.40.0 255.255.255.0 vpn_gateway"
# server network from H-IT for KMS
```

```
push "route 141.71.2.0 255.255.255.0 vpn_gateway"

# Push routes for local IPv6 networks
# (The vpn_gateway placeholder does not work here.)
# Note: IPv6 routes that overlap the IPv6 address of the OpenVPN
   server will
# automatically trigger creating a direct route to the OpenVPN server
   on the client.
#
# DMZ
push "route-ipv6 2001:638:614:1780::/64 2001:638:614:1750::1"
# Inform
push "route-ipv6 2001:638:614:1720::/64 2001:638:614:1750::1"
# Edu
push "route-ipv6 2001:638:614:1721::/64 2001:638:614:1750::1"
# NAO
push "route-ipv6 2001:638:614:1722::/64 2001:638:614:1750::1"
# Cluster
push "route-ipv6 2001:638:614:1743::/64 2001:638:614:1750::1"
# experimental ipv6 network
push "route-ipv6 2001:638:614:1744::/64 2001:638:614:1750::1"

### START BLOCK CRYPTOGRAPHY
# Specific settings regarding TLS, chiphers and hash algorithms
# DO NOT CHANGE THIS unless you receive explicit instructions to do so
# These settings need to be identical in client and server
   configuration!

# Protect data channel with this cipher
cipher AES-256-GCM

# Authenticate packets in data and control channel using HMAC with
   this
# message digest algorithm
auth SHA256

# Use this specific cipher to secure the control channel
tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384

# Only allow TLS version 1.2 and higher
tls-version-min "1.2"
### END BLOCK CRYPTOGRAPHY

# Disable cipher negotiation on server side
ncp-disable

# Send ping message every ten seconds, expect session loss after 60
   seconds of no response
keepalive 10 30

# Notify clients when the server restarts or shuts down.
# Default behaviour: Tell clients to try to connect to the same
```

```
   server again.
explicit-exit-notify 1

# Enable these if you plan to enable running on reduced privileges
# These options allow to keep the private key and the virtual network
   device handle in memory
persist-key
persist-tun

# Reduce privileges after launch (uncomment and adapt on unix/linux
   system)
user nobody
group nogroup

# Logging settings
# - ‘verb 3‘ is enough for debugging most issues
# - ‘verb 0‘ is recommended for regular operation
verb 3
mute 5

# Print a list of active sessions into this file
# This might be helpful if you plan big maintenance
status inform/status.log
```

Da die CRL durch die Zertifizierungsstelle jederzeit aktualisiert werden kann, ist es sinnvoll diese täglich mit einem Cronjob zu erneuern. Dafür wird dieser Eintrag als `root` mittels `crontab -e` angelegt:

```
15 2 * * * bash -c 'curl http://vpnca.inform.hs-hannover.de/crl.pem >
   /etc/openvpn/inform/crl.pem; systemctl restart
   openvpn@inform.service'
```

Anmerkung: Ein `systemctl reload` ist aufgrund der Abgabe von Berechtigungen nach dem Start nicht möglich. Deshalb muss auf `systemctl restart` zurückgegriffen werden.

Nun wird der OpenVPN-Dienst aktiviert und gestartet:

```
systemctl enable openvpn@inform.service
systemctl start openvpn@inform.service
```

3 Administrative Aufgaben

Dienst aktivieren/deaktivieren: Mit dem folgenden Befehl kann der VPN-Dienst aktiviert werden. Dadurch ist es möglich, den Dienst zu starten. Außerdem startet der aktivierte Dienst nach dem Neustart des Servers automatisch.

```
systemctl enable openvpn@inform.service
```

Mit dem folgenden Befehl kann der Dienst wieder deaktiviert werden.

```
systemctl disable openvpn@inform.service
```

Dienststatus abfragen: Ob der OpenVPN-Dienst gerade läuft oder nicht läuft, lässt sich mit diesem Befehl herausfinden.

```
systemctl status openvpn@inform.service
```

Dienst starten/stoppen/neustarten: Wurde der OpenVPN-Dienst aktiviert, kann er mit den folgenden Befehlen gestartet, gestoppt oder neu gestartet werden.

```
systemctl start openvpn@inform.service
systemctl stop openvpn@inform.service
systemctl restart openvpn@inform.service
```

Details zur Handhabung von Systemdiensten mit `systemctl` können im „Debian Administrator's Handbook“ Kapitel 9.1.1 nachgeschlagen werden[1].

Manuelles Failover: Sollte der Bedarf bestehen, dass ein identisch konfigurierter Server den aktuell aktiven VPN-Server ablöst, so müssen lediglich die IP-Adressen des VPN-Dienstes auf den zweiten Server umgezogen werden.

Hinweis: Gegebenenfalls muss der Name des verwendeten Netzwerkinterfaces - hier `eno1` - den aktuellen Umständen angepasst werden.

Zunächst müssen die Dienst-IPs auf dem aktuell aktiven Server deaktiviert werden und im Anschluss gegen versehentliche beziehungsweise automatische Reaktivierung nach einem Neustart gesichert werden. Deaktivieren der Dienst-IPs:

[1] Siehe `https://debian-handbook.info/browse/stable/unix-services.html#sect.systemd`

```
ip addr del 141.71.38.7 dev eno1
ip addr del 2001:638:614:1780::7 dev eno1
```

Datei `/etc/network/interfaces` angepasst, um die Reaktivierung der Dienst-IPs zu verhindern. In dieser müssen die Befehle `post-up` und `pre-down` auskommentiert werden, wie hier gezeigt wird:

```
iface eno1 inet static
        address 141.71.38.70/24
        gateway 141.71.38.254
        #post-up /sbin/ip addr add 141.71.38.7/24 dev eno1
        #pre-down /sbin/ip addr del 141.71.38.7/24 dev eno1

iface eno1 inet6 static
        address 2001:638:614:1780::0131/64
        gateway 2001:638:614:1780::1
        #post-up /sbin/ip addr add 2001:638:614:1780::0007/64 dev eno1
        #pre-down /sbin/ip addr del 2001:638:614:1780::0007/64 dev
           eno1
```

Auf dem zweiten Server können nun analog die (wie oben gezeigt) einkommentierten Einträge in der `/etc/network/interfaces` wieder auskommentiert werden. Anschließend werden die folgenden Befehle verwendet, um die Dienst-IPs auf dem zweiten Server zu konfigurieren:

```
ip addr add 141.71.38.7 dev eno1
ip addr add 2001:638:614:1780::7 dev eno1
```

Im Anschluss muss sichergestellt werden, dass der OpenVPN-Dienst auf dem zweiten Server aktiviert ist und läuft.

Backups: Da der VPN-Dienst sich anhand dieser Installationsanleitung von Grund auf neu einrichten lässt, sind im Prinzip keine Sicherheitskopien notwendig. Um eine Neueinrichtung des Dienstes zu erleichtern können bei Bedarf die Inhalte von `/etc`, sowie die Ausgabe von `crontab -l` als Benutzer `root` gesichert werden.

Einspielen von Updates: Das Einspielen von Updates erfolgt in zwei Schritten. Im ersten Schritt werden die Paketquellen aktualisiert, damit der Paketmanager die Versionen der aktuell verfügbaren Pakete kennt.

```
apt-get update
```

Anschließend können alle verfügbaren Aktualisierungen mit dem folgenden Befehl eingespielt werden.

```
apt-get dist-upgrade
```

Es folgt eine Auflistung aller zu aktualisierenden Pakete mit der Abfrage, ob die Aktion durchgeführt werden soll. Bestätigt man die Abfrage, so werden die Updates installiert. Details zur Handhabung des Debian-Paketmanagers können im „Debian Administrator's Handbook“ in Kapitel 6 nachgelesen werden[2].

[2]Siehe `https://debian-handbook.info/browse/stable/apt.html`