

IT Sicherachitekturen

**Einrichtung einer internen Firewall von  
Firma-a und Firma-b, sowie  
LAN-to-LAN-VPN von Firma-a  
(Server) nach Firma-b(Client)**

Paul Drautzburg      Georg Mohr

HTWG Konstanz, Sommersemester 2018

---

# Inhaltsverzeichnis

<b>1</b>	<b>Motivation</b>	<b>1</b>
<b>2</b>	<b>Problemstellung</b>	<b>2</b>
2.1	Laborumgebung und Versuchsaufbau . . . . .	2
2.2	Aufgabenstellung . . . . .	3
<b>3</b>	<b>Theoretische Grundlagen</b>	<b>4</b>
3.1	Netzwerkarchitekturen . . . . .	4
3.1.1	Die Begriffe Intranet, DMZ und Internet . . . . .	4
3.2	Netzwerkprotokolle/Kommunikationsprotokolle . . . . .	6
3.2.1	Aufbau eines Datenpakets . . . . .	7
3.3	Virtuelles Betriebssystem (Debian GNU/Linux 7.11) . . . . .	8
3.4	IP-Tables . . . . .	9
3.4.1	Allgemein . . . . .	10
3.4.2	Funktionsweise . . . . .	10
3.4.3	Tabellen . . . . .	10
3.4.4	Chains . . . . .	11
3.4.5	Filterregeln . . . . .	11
3.4.6	Policies . . . . .	12
3.5	Public Key Infrastructure . . . . .	12
3.5.1	Aufbau einer PKI . . . . .	12
3.5.2	Erstellung eines Server-Zertifikats . . . . .	13
3.6	OpenVPN . . . . .	16
3.6.1	Grundlagen . . . . .	16
3.6.2	Authentifizierung . . . . .	16
3.6.3	Kommunikation . . . . .	17
<b>4</b>	<b>Lösungsskizze</b>	<b>19</b>
4.1	Firewall . . . . .	19
4.2	VPN . . . . .	23
4.2.1	Serverzertifikate . . . . .	23
4.2.2	Server Firma A . . . . .	24
4.2.3	Client Firma B . . . . .	28
<b>5</b>	<b>Auswertung</b>	<b>30</b>



---

## Abbildungsverzeichnis

1	Schematischer Aufbau der Laborumgebung im IT-Sicherheitslabor [ND18]	3
2	Aufbau eines Datenpakets [Hea] . . . . .	8
3	Zertifikatserstellung im Labor[ND18] . . . . .	15
4	Verbindungsaufbau eines VPN's[Rij18] . . . . .	30
5	Log Ausgabe des gestarteten VPN-Client . . . . .	31
6	Log Ausgabe des gestarteten VPN-Servers . . . . .	32

---

## Tabellenverzeichnis

1	OSI-Schichtenmodell . . . . .	6
2	IP-Tables: Tabellen . . . . .	11
3	IP-Tables: Chains . . . . .	11
4	Bridging-Mode vs. Routing-Mode . . . . .	18
5	IPTABLES Optionen und Beschreibung . . . . .	21
6	Befehle zur Erstellung eines Privaten Schlüssels mittels OpenSSL . . . . .	24

---

# 1 Motivation

Im Zeitalter der Digitalisierung steigt der Grad der Vernetzung immer weiter an. Unternehmen, welche ihre Standorte über die ganze Welt verteilt haben, wollen Wege und Möglichkeiten haben ihre Arbeit sicher und Problemlos zu verrichten. Dafür ist es unerlässlich, dass verschiedene Standorte auf Ressourcen des anderen zugreifen können. Für solche Szenarien gibt es in der Netzwerktechnik und Netzwerkarchitektur verschiedene Lösungsansätze. Jedoch darf dabei ein wichtiger Aspekt nicht vernachlässigt werden, denn jedes Unternehmen hat Betriebsmittel und Informationen, welche es keinen Gefahren von außen aussetzen will. An diesem Punkt kommt der Begriff IT-Sicherheit ins Spiel und genau dieses Umfeld um den Begriff „IT-Sicherheit“ wird im Rahmen des Praktikums zur Lehrveranstaltung „IT-Sicherheitsarchitekturen“ untersucht. In den folgenden Kapiteln wird eine Problemstellung unter Laborbedingungen skizziert, welche ein Szenario widerspiegelt, mit dem sich Unternehmen täglich konfrontiert sehen. Anschließend wird werden die Laborbedienungen und der Versuchsaufbau beschrieben. Aus der Problemstellung und dem Versuchsaufbau wird eine Lösung auf Grundlage des theoretischen Hintergrunds erarbeitet. Diese Lösung wird abschließend in der Laborumgebung implementiert und auf ihre Akzeptanz gegenüber der Problemstellung untersucht und in einem Fazit bewertet.

---

## 2 Problemstellung

In diesem Kapitel wird die Laborumgebung eingeführt und beschrieben, sowie die damit verbundene Aufgaben-/Problemstellung skizziert.

### 2.1 Laborumgebung und Versuchsaufbau

Im Labor wird die IT-Landschaft zweier Unternehmen „A“ und „B“ simuliert. Jedes Unternehmen besitzt ein lokales LAN, an denen die pot. Mitarbeiter angeschlossen sind. Zudem gibt es mehrere Server mit Hilfe derer verschiedene Dienste angeboten werden sollen. Diese Dienste werden nachfolgend aufgelistet:

- Internetzugang von den Mitarbeiter-Computern,
- Zugang über HTTPS der Webseiten beider Unternehmen,
- Eine virtuelle Kopplung der beiden Unternehmen mit Hilfe einer sog. Site-to-Site Verbindung,
- Jeder Mitarbeiter der Unternehmen soll die Möglichkeit haben, E-Mails über einen sicheren Mailserver zu verschicken.

Eine weitere wichtige Komponente in dieser IT-Landschaft sind die externen und internen Firewalls, da diese den Grundstein für eine sichere Umgebung legen. In dieser IT-Landschaft gibt es insgesamt vier Firewalls, jedes Unternehmen hat jeweils eine interne und externe Firewall. Hinter den externen Firewalls der Unternehmen liegt die sog. DMZ – Demilitarisierte Zone. In dieser Zone stehen üblicherweise die Unternehmensserver, diese werden nach außen hin von der externen Firewall und nach innen von der internen Firewall geschützt. Jede Firewall läuft auf einem Server, somit besteht die IT-Landschaft insgesamt aus vier Servern auf welchen die Firewalls laufen, zwei Servern, welche die jeweiligen Unternehmensserver simulieren und einem Server welcher außerhalb der externen Firewall das Internet simulieren soll (vgl. Abb. 1).

Jede dieser angesprochenen Komponenten muss entsprechen konfiguriert werden, damit die angesprochenen Dienste fehlerfrei funktionieren können. Technisch gesehen wird jeder dieser Server auf Grundlage einer virtuellen Maschine abgebildet. Auf jeder dieser Maschinen ist Linux als Betriebssystem installiert und vorkonfiguriert, jedoch ohne zusätzliche Pakete, nur die reine Grundinstallation. Da die Implementierung und Installation aller dieser Dienste und der damit verbundenen Komponenten den Rahmen dieses

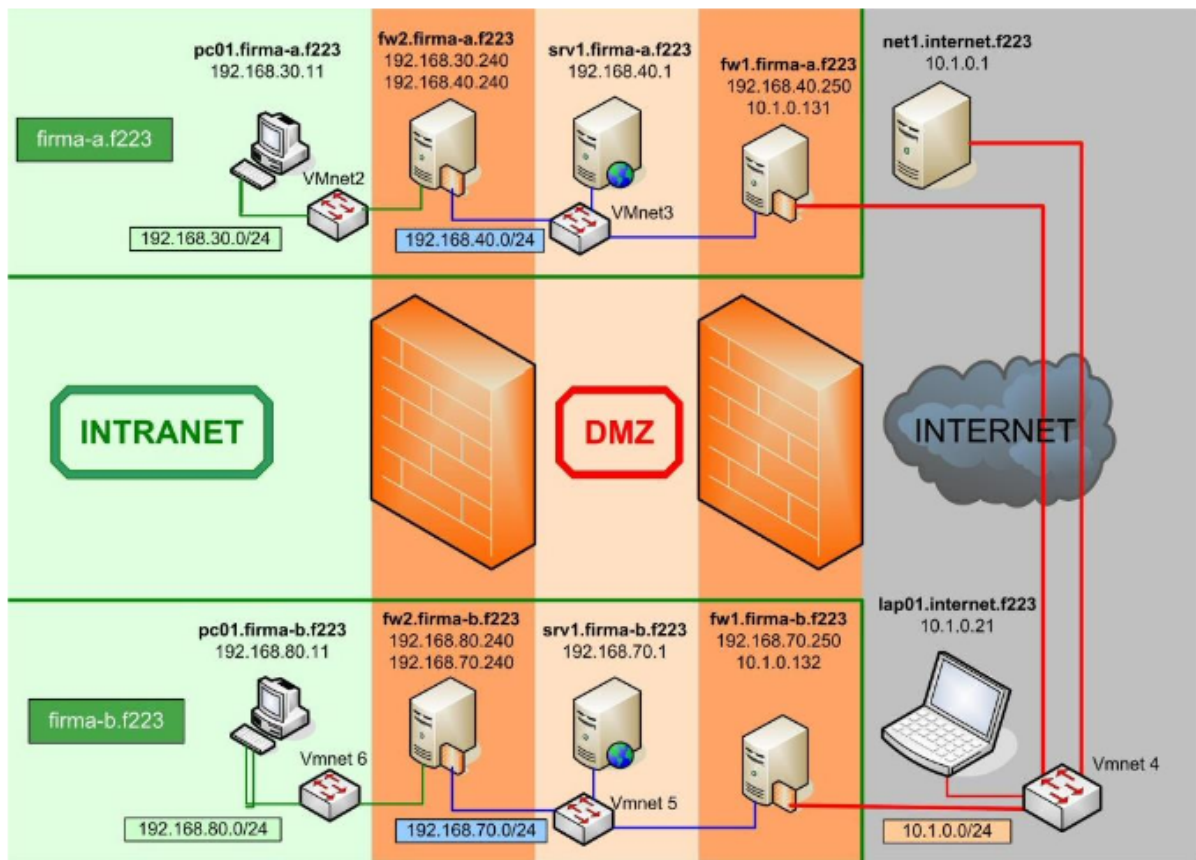


Abbildung 1: Schematischer Aufbau der Laborumgebung im IT-Sicherheitslabor [ND18]

Praktikums sprengen würde, werden die Aufgabenstellungen zuvor runter gebrochen. Die detaillierte Aufgabenstellung wird im nächsten Abschnitt eingeführt.

## 2.2 Aufgabenstellung

Wie schon im vorhergehenden Abschnitt beschrieben, gibt es mehrere Dienste, welche in der IT-Landschaft angeboten werden sollen. In diesem Bericht, wird die Virtuelle Kopp- lung der Unternehmen „A“ und „B“ durch eine „Site-to-Site-“ oder auch „LAN-to-LAN-“ genannt -Verbindung implementiert. Die Problemstellung gilt als gelöst, wenn es möglich ist, dass Unternehmen A eine gesicherte Verbindung zu Unternehmen B aufbauen kann und die jeweiligen internen Firewalls entsprechend konfiguriert sind. Die Konfiguration



---

der externen Firewalls kann in dieser Aufgabe vernachlässigt werden, hierfür sollen die bereitgestellten Firewalls verwendet werden.

## **3 Theoretische Grundlagen**

In diesem Kapitel werden die theoretischen Grundlagen eingeführt, welche im Kontext der Aufgabestellung benötigt werden. Dies beschränkt sich auf fünf Bereiche, die Netzwerkarchitekturen, Debian GNU Linux 7.11, IP-Tables, PKI und OpenVPN. Jede dieser genannten Bereiche wird innerhalb dieses Kapitels eingeführt und in den Aufgabenkontext eingeordnet.

### **3.1 Netzwerkarchitekturen**

Der Bereich Netzwerkarchitekturen ist ein sehr breites Gebiet, deshalb muss dieser zuvor anhand der Anforderungen der Aufgabenstellung abgegrenzt werden.

#### **3.1.1 Die Begriffe Intranet, DMZ und Internet**

Das Netzwerk des Labors besteht netzwerktechnisch aus drei Bereichen dem Intranet, der Demilitarisierten Zone und dem Internet. Diese Begriffe werden folgend für den Rahmen dieses Berichts definiert und in den Kontext der Problemstellung eingeordnet.

”Intranet” Nach dem Gabler Wirtschaftslexikon wird ein Intranet als ” ein unternehmensinternes Kommunikationsnetz, in dem Daten auf der Basis der Protokollfamilie TCP/IP übertragen werden” definiert [int] In dieser Arbeit wird der Aspekt der Protokollfamilie ”TCP/IP” von vorrangiger Bedeutung sein. Der Punkt des ”unternehmensinternen Kommunikationsnetzes”, kann für diese Arbeit in den Hintergrund gestellt werden, da sich das Laborpraktikum auf die technischen Umsetzung beschränkt. Ein weiterer wichtiger Aspekt des Intranets ist, dass es nur autorisierten Benutzern gestattet werden soll zuzugreifen.

---

”Demilitarisierte Zone” Der Begriff demilitarisierte Zone kommt ursprünglich aus dem militärischen Umfeld und beschreibt eine Zone oder Bereich in der sich keine militärischen Streitkräfte gegenüberstehen dürfen, quasi ein neutraler Bereich. In der Netzwerktechnik wird dieser Begriff benutzt, um eine Zone zu beschreiben welche sich zwischen zwei Schutzeinrichtungen befindet. Bei diesen Schutzeinrichtungen handelt es sich meistens um eine externe Firewall und eine interne Firewall. Der Hintergrund für die Einrichtung einer DMZ ist es die Sicherheit der Komponenten und Teilnehmer eines Intranets zu sichern, falls es einen Angriff auf die Komponenten innerhalb der DMZ gibt und diese korumpiert werden sollten. Die Komponenten innerhalb einer DMZ werden als potentielle Opfer oder ”Victims” bezeichnet. Es handelt sich hier meistens um Server welche nach einem Schadensfall einfach durch einen Reboot wiederhergestellt werden können. Als Alternative könnte statt der DMZ ein sog. Application Layer Gateway (APL) eingesetzt werden. Dieser zeichnet sich dadurch aus, dass er den Netzwerkverkehr komplett auftrennt und sich nach allen Seiten als Kommunikationspartner verhält. Zudem wäre auch eine Lösung aus DMZ und APL denkbar [int16]. Im Rahmen des Praktikums wird jedoch lediglich eine DMZ eingesetzt.

”Internet” Das Internet ist ”ein weltumspannendes, heterogenes Computernetzwerk, das auf dem Netzwerkprotokoll TCP/IP basiert. Über das Internet werden zahlreiche Dienste wie z.B. E-Mail, FTP, World Wide Web (WWW) oder IRC angeboten” [Gab]. Ein wichtiger Punkt im Rahmen dieses Praktikums ist, dass das Internet innerhalb der Laborumgebung nur simuliert wird, es besteht also kein richtiger Zugang zum Internet. Dieser Fakt stellt sicher, dass der Versuchsaufbau bei falscher Konfiguration von außen nicht beschädigt werden könnte. Zudem wird somit sichergestellt, dass bei der Konfiguration der virtuellen Maschinen nur die Versionen der zu Grunde liegenden Betriebssystem-Images verwendet werden können.

Nachdem die Begriffe Intranet, DMZ und Internet im Kontext der Laborumgebung und Problemstellung definiert und eingrenzt wurden müssen noch weitere Begriffe eingeführt werden. Im folgenden Abschnitt werden die Netzwerkprotokolle oder auch Kommunikationsprotokolle eingeführt, welche im Rahmen des Praktikums verwendet werden.

---

## 3.2 Netzwerkprotokolle/Kommunikationsprotokolle

Um eine Netzwerkkommunikation verschiedener Komponenten innerhalb eines Netzwerks zu gewährleisten müssen Netzwerkprotokolle eingesetzt werden.

Im Rahmen des Laborpraktikums werden verschiedene Netzwerkprotokolle eingesetzt, diese lassen sich am besten anhand der entsprechenden Layer im OSI-Schichtenmodell beschreiben. In der nachfolgenden Tabelle werden anhand der Schichten die einzelnen Protokolle eingeordnet, anschließend werden die für dieses Praktikum relevanten Protokolle gesondert beschrieben.

#	OSI-Schicht	Einordnung	Protokolle	Kopplungselemente
7	Anwendungen(Application)	Anwendungsorientiert	HTTP, FTP, HTTPS, SMTP, DNS, LDAP	Gateway
6	Darstellung(Presentation)			Content-Switch
5	Sitzung(Session)			Proxy
4	Transport(Transport)	Transportorientiert	TCP, UDP, SCTP, SPX	Layer-4-7-Switch
3	Vermittlung-/Paket(Network)		ICMP, IGMP, IP, IPsec, IPX	RouterLayer-3-Switch
2	Sicherung(Data Link)		Ethernet, Token Ring, FDDI	BridgeLayer-2-Switch
1	Bitübertragung(Physical)			Netzwerkkabel Repeater Hub

Tabelle 1: OSI-Schichtenmodell

Wie bereits in der Aufgabenstellung beschrieben (vgl. 2.2 Aufgabenstellung), soll eine site-to-site VPN eingerichtet werden. Da es sich um eine die Verbindung handelt, kann der Fokus auf die Transport-orientierten Schichten der OSI-Modells 1-4 gerichtet werden. Für die Umsetzung einer VPN-Verbindung und Konfiguration der internen Firewalls sind die folgenden Protokolle von vordergründiger Bedeutung.

”ICMP” Das Internet Control Message Protocol, wird dazu verwendet in Rechnernetzen einen Austausch von Fehlermeldungen durchzuführen.

”IP” Das Internet Protocol ist ein zustands- und verbindungsloses Protokoll, welches die Implementierung der Vermittlungsschicht (3) widerspiegelt.

”IPsec” Das Internet Protocol Security ist eine Erweiterung des IP-Protokolls und soll eine gesicherte Kommunikation über unsichere IP-Netze ermöglichen.

”TCP” Das Transmission Control Protocol wird von allen modernen Betriebssystemen genutzt um, zu definieren, wie die Daten zwischen verschiedenen Netzwerkkomponenten ausgetauscht werden sollen

---

”UDP” Das User Datagram Protocol zeichnet sich dadurch aus, dass es verbindungslos und ungeschützt ist. Es kann also nicht gesichert werden, ob ein gesendetes Datenpaket richtig (nicht verfälscht von Dritten) oder überhaupt ankommt. [KR08]

Diese beschriebenen Protokolle sind bei der Lösung der Problemstellung von großer Bedeutung. Sie werden bei der Erstellung der Lösungsskizze wieder aufgegriffen (vgl. Kapitel 4). Um die Beschreibung Netzwerkprotokolle zu komplettieren muss noch auf eine Ebene tiefer geschaut werden, auf die Datenpakete. Im nächsten Abschnitt wird der Aufbau und die wichtigsten Eigenschaften eines Datenpakets beschrieben.

### 3.2.1 Aufbau eins Datenpakets

Im vorhergehenden Abschnitt wurden die wichtigsten Netzwerkprotokolle für den Rahmen des Laborpraktikums beschrieben. In diesem Abschnitt, wird das Zusammenspiel eines Netzwerksprotokoll und eines Datenpakets gezeigt.

In einem Protokoll wird der Aufbau eines Datenpakets definiert, zudem enthält es wichtige Informationen über den Datenaustausch. Es definiert,

- wer der Absender und Empfänger eines Datenpaketes sein soll,
- von welchem Typ ein Datenpaket ist, ob es für den Verbindungsaufbau, Verbindungsabbau oder für Nutzdaten genutzt wird,
- die Größe des Datenpaket, welches beim Empfänger ankommen soll.

Wenn es sich um eine mehrteilige Kommunikation handelt, muss zusätzlich noch die laufende Nummer und die Anzahl der Pakte definiert werden. Zuletzt folgt noch eine Prüfsumme, mit dessen Hilfe der Empfänger prüfen kann, ob die Datenpakete fehlerfrei angekommen sind. Alle dieser beschriebenen Informationen werden dem ”Header” eines Datenpaketes vorne oder hinten angehängt, dieser angehängte Bereich wird auch ”Trailer” genannt. Anhand der folgenden Abbildung 2 wird der Aufbau eines Datenpakts und der einzelnen Bestandteile noch verdeutlicht.

Nach dem nun die wichtigsten Bestandteile eines auch Datenpakets eingeführt wurden, sind alle Grundlagen aus dem Bereich der Netzwerkarchitekturen und Netzwerktechnik beschrieben, welche im Rahmen des Praktikums benötigt werden. Es wurde die Begriffe DMZ, Intranet und Internet eingeführt, sowie die wichtigsten Netzwerkprotokolle und ihr Zusammenhang zu den Datenpaketen.

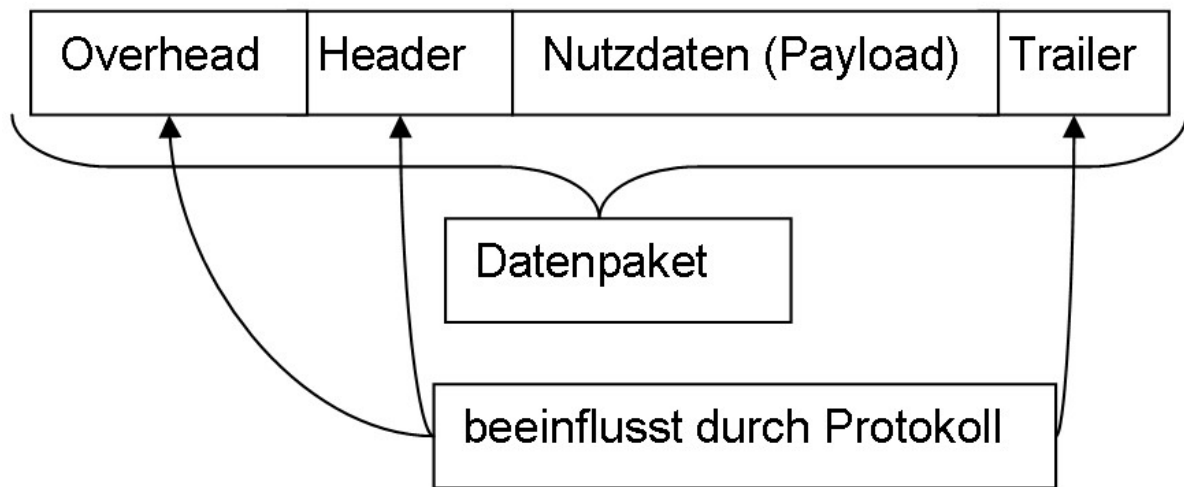


Abbildung 2: Aufbau eines Datenpakets [Hea]

Im folgenden Kapitel wird das zugrundeliegende Betriebssystem "Debian GNU Linux 7.11" und die Besonderheiten in der Laborumgebung in kürze beschrieben.

### 3.3 Virtuelles Betriebssystem (Debian GNU/Linux 7.11)

Debian GNU Linux 7.11 ist ein kostenfreies gemeinschaftliches entwickeltes Betriebssystem. Die Version 7.11 welche im Rahmen des Praktikums eingesetzt wird, gehört zu einer älteren Version und wird von Softwareentwicklern nicht mehr unterstützt. GNU Linux 7.11 unterstützt verschieden Rechnerarchitekturen wie z.B. x86(i386 & amd64).

Die Besonderheit im Laborpraktikum ist die, dass mit virtuellen Maschinen auf einem zugrundeliegenden Windows 10 Betriebssystems gearbeitet wird. Dies bringt einige Probleme mit sich, zum einen müssen die virtuellen Maschinen im Bereich der Netzwerkdapter einwandfrei konfiguriert sein.

Zum anderen kann mit den Maschinen nicht auf das Internet zugegriffen werden. Dies stellt ein wesentliches Problem da, da ohne Verbindung ins Internet Paketdienste wie "Aptitude" nicht funktionieren. Aus letzterem Grund muss zwingend bei der Installation von neuen Paketen das virtuelle Image in die Maschine "eingelegt" werden. Eine andere Möglichkeit wäre, die Pakete einzelnen mit Hilfe des zugrunde liegenden Systems zu Downloaden und einzeln einzuspielen. Diese Variante stellte sich aber als sehr zeitintensiv

---

und kompliziert heraus, da jedes Paket Abhängigkeiten mitbringt und diese zusätzlich an Versionen gebunden sind. So kann es passieren, dass dieser Prozess der Installation eine neue Problemkette im Betriebssystem schafft. Somit wurden alle Pakete, welche für die Lösung der Aufgabe benötigt wurden, mit Hilfe der bereitliegenden Linux-Images installiert.

Eine weitere Besonderheit ist die, dass es keine grafische Benutzeroberfläche gibt, dies setzt die Arbeit "auf Konsole" voraus. Somit muss auch mit den Linux internen Texteditoren wie z.B. dem "VI" oder "Nano" gearbeitet werden. Dies stellt insofern ein Problem da, da Linux in sich hochgradig "case sensitiv" reagiert. Dies bedeutet im Umkehrschluss, wenn ein Leerzeichen oder Komma falsch gesetzt wird, terminiert bspw. ein Skript nicht so wie der Nutzer es vorgesehen hat. Jedoch ist es zwingend empfohlen sämtliche Skripte auf dem virtuellen System zu schreiben, da sonst durch das Kopieren von bspw. einem Windows Betriebssystem illegale nicht sichtbare Steuerzeichen im Skript auftreten könnten[edi].

Ein wichtiger Punkt im Zusammenspiel der Skripte und dem Betriebssystem ist die Wahl der Shell mit welcher es ausgeführt werden soll. Linux bietet viele verschiedene Shells an, im Rahmen des Praktikums wurde ausschließlich mit der sog. "bash" gearbeitet, da sie unter Linux die "Standard-Login" Shell darstellt[bas]. In sämtlichen Skripten wurde in der ersten Zeile der sog. "Shebang", welcher aus "# !" besteht, eingefügt. Gefolgt von diesem wird der Ort des Interpreter angegeben mit welchem das Skript ausgeführt werden soll. In diesem Fall sieht diese erste Zeile eines Skripts immer wie folgt aus, "#!/bin/bash" [she].

Damit wurden alle wichtigen Besonderheiten des Betriebssystem im Rahmen des Praktikums eingeführt. Im nächsten Kapitel wird das "IP-Tables" Paket von Linux eingeführt, da dieses einen Grundstein für die Lösung der Aufgabenstellung im Bezug auf die internen Firewalls darstellt.

### 3.4 IP-Tables

In diesem Kapitel werden die Möglichkeiten des Linux Pakets "IP-Tables" zur Filterung von Paketen beschrieben und eingeführt. Mit Hilfe dieses Pakets werden die für die Lösung erforderlichen Firewall-Skripte erstellt.

---

### 3.4.1 Allgemein

Das Paket "IP-Tables" ist Standardmäßig in der Linux Installation enthalten. Es kann aber auch einfach über den Paketmanager mit Hilfe des Befehls "apt-get install iptables" installiert werden. In diesem Fall war das Paket schon installiert und musste nicht mehr manuell installiert werden.

Grundlegend stellt das IP-Tables einen Paketfilter dar, welches Pakete auf IP-Ebene auf Basis zuvor definierten Regeln filtert. Empfangene Pakete werden überprüft bevor sie weitergeleitet werden. Umgekehrt werden ausgehenden Pakete geprüft, bevor sie den Rechner verlassen. Es kann aber auch sein, dass der Rechner als Router fungiert, so müssen alle Pakete geprüft werden während diese weitergeleitet werden sollen.

### 3.4.2 Funktionsweise

Die Paketprüfung wird dreistufig in einer Hierarchie durchgeführt. Die folgenden drei Stufen gibt es

- Tabellen,
- Chains (Ketten),
- Filterregeln.

Wird eine definierte Regel in einer Chain oder Tabelle gefunden wird die hinterlegte Aktion ausgeführt. Wenn keine Regel definiert wurde, wird eine allgemein definierte sog. Policy ausgeführt.

### 3.4.3 Tabellen

In einer Tabellen werden die Filterregeln zu Gruppen zusammengefasst und nach der grundlegenden Aufgabe sortiert. Insgesamt gibt es vier Tabellen in denen eine Regel eingeordnet werden kann. In der folgenden Tabelle werden diese vier Tabellen und ihre Eigenschaften beschrieben[ipt].

---

Tabelle	Beschreibung
<b>filter</b>	Standardtabelle, hier werden nur die grundlegenden Regeln hinterlegt.
<b>nat</b>	(Network Address Translation) Tabelle für die Adressumsetzung und für das Port-Forwarding.
<b>mangle</b>	Tabelle bei gewünschter Manipulation von Paketen.
<b>raw</b>	Tabelle für Ausnahmen beim Connection tracking.

Tabelle 2: IP-Tables: Tabellen

### 3.4.4 Chains

In jeder Tabelle sind sog. "Chains" enthalten, welche festlegen wann ein Paket geprüft wird, bspw. bevor es verschickt werden soll. In der folgenden Tabelle werden alle fünf verschiedenen möglichen "Chains" beschrieben[ipt].

Chain	Tabelle	Beschreibung
<b>INPUT</b>	filter, mangle	Relevant für alle Pakete, welche an einen lokalen Prozess gerichtet sind.
<b>OUTPUT</b>	filter, nat, mangle, raw	Relevant für alle Pakete, welche von einem lokalen Prozess abstammen.
<b>FORWARD</b>	filter, mangle	Relevant für alle Pakete, welche geroutet werden sollen.
<b>PREROUTING</b>	nat, mangle, raw	Relevant für alle Pakete, bevor sie geroutet werden sollen.
<b>POSTROUTING</b>	nat, mangle	Relevant für alle Pakete, nachdem sie geroutet wurden.

Tabelle 3: IP-Tables: Chains

### 3.4.5 Filterregeln

Sobald ein Paket auf eine Filterregel trifft muss definiert werden, wie mit dem Paket umgegangen werden soll. Dafür gibt es vier Optionen, welche am häufigsten Anwendung finden [ipt]. Diese sind

- **ACCEPT**: Das Paket wird akzeptiert und angenommen,
- **DROP**: Das Paket wird, ohne Nachricht an den Absender, abgelehnt,
- **REJECT**: Das Paket wird, mit Nachricht an den Absender, abgelehnt,
- **LOG**: Die Daten des Pakets werden in den System Log geschrieben und es wird mit der nächsten Regel der Chain fortgefahren.

Für dieses Laborpraktikum sind die beschriebenen vier Filterregeln ausreichend, im Kontext des IP-Tables Pakets bietet Debian/Linux noch weitere Optionen an.



---

### 3.4.6 Policies

Da sich aus den beschriebenen Möglichkeiten eine große Anzahl an Möglichkeiten für Filterregeln für die einzelnen Pakete ergibt, gibt es sog. "Policies". Diese "Policies" setzen sich aus der Filterregel, der Chain und Aktion zusammen [ipt]. Exemplarisch könnte ein Befehl für Input, Drop oder Forward wie folgt aussehen:

Listing 1: Policies

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

Alle beschriebenen Möglichkeiten von IP-Tables wurden im Rahmen des Praktikums für die Lösung der Problemstellung eingesetzt. Dies ist jedoch nur ein Teil der Möglichkeiten von IP-Tables und spiegelt nicht den kompletten Funktionsumfang wieder. Im folgenden Kapitel wird eine Einführung in die Grundlagen einer Public Key Infrastructure gegeben, wie sie auch für das Praktikum benötigt wird.

## 3.5 Public Key Infrastructure

Eine Public Key Infrastructure ist ein System anhand dessen digitale Zertifikate ausgestellt, geprüft und verteilt werden können. Diese ausgestellten Zertifikate werden dann für die sichere Kommunikation zwischen Netzwerkteilnehmern genutzt.

### 3.5.1 Aufbau einer PKI

Eine PKI besteht grundsätzlich aus folgenden Bestandteilen

**Digitale Zertifikate:** Mit Hilfe der digitalen Zertifikate wird die Echtheit eines Objekts sichergestellt, zudem kann die Authentizität und Integrität mit Hilfe von kryptographischen Verfahren nachgeprüft werden. Ein Zertifikat enthält die Schlüssel, welche zur Authentifizierung und Ver- und Entschlüsselung der zu übertragenden Daten, sorgen. Die Ausstellung eines digitalen Zertifikats erfolgt durch die Zertifizierungsstelle (CA)

---

**Zertifizierungsstelle (CA):** Eine Zertifizierungsstelle (CA), bildet den Kern einer PKI. Sie ist für das Herausgeben der Zertifikate verantwortlich und trägt die Verantwortung für die Integrität dieser.

**Registrierungsstelle (RA):** Eine Registrierungsstelle (RA), ist eine Organisation bei der Zertifikate beantragt werden können. Dies können Personen, Maschinen oder auch untergeordnete Zertifizierungsstellen sein. Die RA prüft den Zertifizierungsantrag und nach erfolgreicher Prüfung, kann dieses dann durch die Zertifizierungsstelle signiert werden.

**Zertifikatssperrliste (CRL):** Eine Zertifikatssperrliste(CRL), enthält alle Zertifikate, welche vor ihrem eigentlich Ablaufdatum ungültig geworden sind. Dies kann bspw. durch einen geknackten Algorithmus bei einem der Schlüssel passieren.

**Verzeichnisdienst (Directory Service):** Ein Verzeichnisdienst, enthält alle ausgestellten Zertifikate. In Unternehmen ist dieser Dienst meistens durch einen LDAP-Server realisiert.

**Validierungsdienst (VA):** Ein Validierungsdienst(VA), stellt die Möglichkeit bereit ein Zertifikat in Echtzeit zu prüfen. Im Normalfall ist dies durch OCSP oder SCVP realisiert.

Im Rahmen des Praktikums wurde eine solche PKI bereits zur Verfügung gestellt. Im nächsten Abschnitt wird die Erstellung eines Server-Zertifikats gezeigt, wie es auch für die Lösung der Aufgabenstellung erforderlich ist.

### 3.5.2 Erstellung eines Server-Zertifikats

Der Ablauf für die Erstellung eines Server-Zertifikats besteht aus insgesamt sechs Schritten, diese werden nun ihrer Reihenfolge nach beschrieben.

1. Der Benutzer muss sich zuerst ein Schlüsselpaar erzeugen (privater/öffentlicher)[ND18].
2. Auf der CA-Seite, kann sich der Benutzer nun die req.cnf runterladen. In dieser Datei sind alle Grundeinstellungen der CA enthalten[ND18].
3. Aus der req-Datei und dem Schlüssel kann der Benutzer nun einen Zertifikats-Request erstellen (Request, Fileextension csr). Dieser Request enthält den öffentlichen Schlüssel, welcher dann von der CA beglaubigt wird[ND18].
4. Jetzt kann die csr-Datei auf die CA-Seite hochgeladen werden[ND18].

- 
5. Der Request wird nun auf dem CA-Server verarbeitet und das signierte Zertifikat für den Benutzer auf der Seite der CA zum Download bereitgestellt[ND18].
  6. Diese crt-Datei, welche nun heruntergeladen werden kann, kann nun einfach für die Sicherung der Netzwerkkomponenten genutzt werden, bspw. für den VPN-Client oder VPN-Server[ND18].

Für die Lösung der Aufgabenstellung muss dieser Vorgang insgesamt zwei Mal durchgeführt werden, da für Client und Server ein solches Server-Zertifikat benötigt wird. Zu beachten ist auch, dass die Software OpenSSL zwingend, für die Erstellung des privaten Schlüssels, installiert sein muss.

Der Gesamtprozess zur Erstellung eines Server-Zertifikats wird in Abbildung 3 nochmals verdeutlicht.

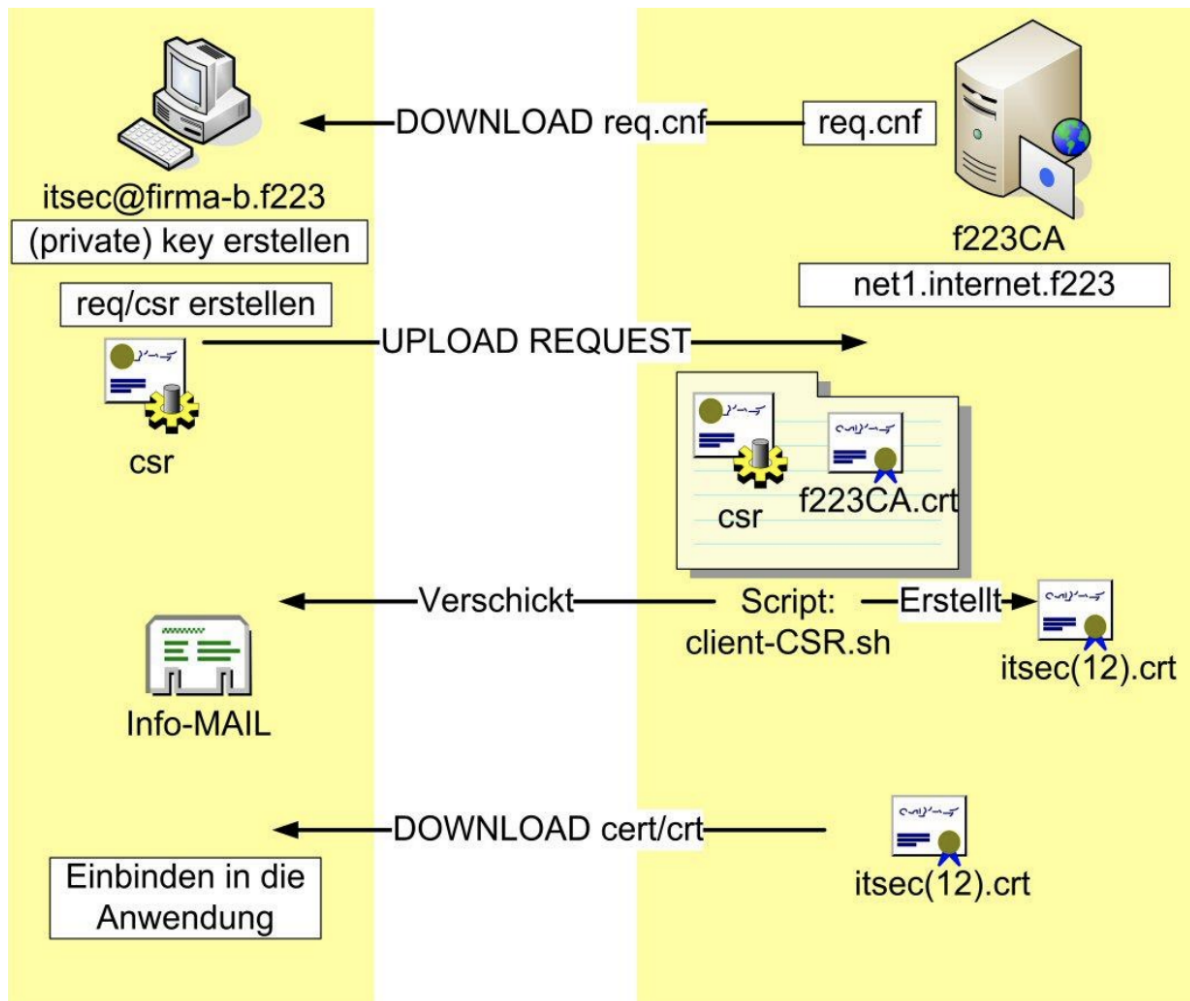


Abbildung 3: Zertifikatserstellung im Labor[ND18]

Im folgenden Abschnitt wird, die für die Umsetzung der VPN-Verbindung benötigte Software OpenVPN eingeführt.

---

## 3.6 OpenVPN

Für die Lösung der Problemstellung wurde die Software OpenVPN genutzt, da sie den Anforderungen an die Aufgabenstellung entspricht. Da OpenVPN zwingend die Software OpenSSL voraussetzt und diese ohnehin für die Erstellung der privaten Schlüssel genutzt wird, war die Wahl schnell getroffen[oep].

### 3.6.1 Grundlagen

OpenVPN ist eine frei verfügbare Software für die Einrichtung eines virtuellen Netzwerks über eine verschlüsselte TLS/SSL-Verbindungen. Wie bereits angesprochen, erfolgt die Verschlüsselung mit Hilfe von OpenSSL. Es kann wahlweise UDP oder TCP als Transportprotokoll eingesetzt werden. Zur Authentifizierung bietet OpenVPN das sog. Pre-shared Key Verfahren oder eine Authentifizierung über Zertifikate, welches im Rahmen des Praktikums auch eingesetzt wurde[oep].

OpenVPN bietet grundlegend folgende Eigenschaften:

- Verfügbarkeit verschiedenen Betriebsmodi (Bridging oder Routing),
- Tunnelung in beliebige Subnetze
- Verschiedene Verschlüsselungsarten (Pre-shared Key / Zertifikate)
- Skalierbarkeit bei VPN-Serverfarmen

### 3.6.2 Authentifizierung

Wie schon angesprochen verfügt OpenVPN über zwei verschiedene Methoden zur Authentifizierung, folgend werden beide gegenübergestellt.

**Pre-shared Key:** ist ein symmetrisches Verfahren, der Schlüssel wird im VPN-Server erstellt und an die Clients verteilt. Es handelt sich um ein sehr einfaches Verfahren. Jedoch ist es nicht sehr sicher, da es nur einen Schlüssel gibt und falls dieser korrumpiert werden sollte, müssten sämtliche Kommunikationsteilnehmer neue Schlüssel erhalten und wären potentiell in Gefahr[oep].

---

**Zertifikatsbasiert:** ist ein auf dem TLS-Protokoll basierendes Verfahren, welches aktuell als sehr sicher klassifiziert wird. Dieses Verfahren beruht auf privaten und öffentlichen Schlüsselpaaren. Der Server und der Client erhalten jeweils ein öffentliches und privates Zertifikat. Mit Hilfe von easy-rsa könnten auch sehr einfach die benötigten Zertifikate erstellt werden. Diese Aufgabe wird jedoch im Praktikum durch die PKI übernommen. Allgemein ist dieses Verfahren sicherer als das Pre-shared Key, es bringt aber auch eine höhere Komplexität bei der Umsetzung mit[ope].

### 3.6.3 Kommunikation

Mittels OpenVPN kann entweder eine Client-Server Verbindung aufgebaut werden, auch Roadwarrior genannt. Oder, wie in der Aufgabenstellung gefordert, eine Site-to-Site Verbindung. Im Standard wird die Kommunikation über das zustandslose UDP-Protokoll abgewickelt. Diese kann jedoch auch auf das TCP-Protokoll umgestellt werden[ope].

Zudem bietet OpenVPN auch verschiedene Netzwerkmodi an Bridging-Mode (Tap-Device) und Routing-Mode (Tun-Device). Diese zwei Modi werden folgend gegenübergestellt.

**Bridging-Mode(Tap-Device):** in diesem Modus wird ein vollständiges Tunneln des Ethernet-Frames (Layer 2) vorgenommen. Der Client erhält in diesem Fall die IP-Adresse des aktuellen Subnetzes. Die Broadcast müssen in dem Fall auch weitergeleitet werden, um eine potentielle Namensauflösung via SMB-Protokoll zu gewährleisten[ope].

**Routing-Mode (Tun-Device):** in diesem Modus werden ausschließlich IP-Pakete (Layer 3) über einen verschlüsselten Tunnel verschickt. Jeder Kommunikationspartner bekommt eine virtuelle IP-Adresse eines fiktiven Subnetzes. Somit ist der Zugriff auf das dahinter liegenden Netz standardmäßig nicht möglich. Dies muss dann über die Routing-Tabellen der Firewalls oder IP-Forwarding realisiert werden[ope].

In der folgenden Tabelle 4 werden abschließend für dieses Kapitel noch die Vor- und Nachteile der beiden beschriebenen Modi gegenüber gestellt.

---

	<b>Bridging-Modus (Tap-Device)</b>	<b>Routing-Modus(Tun-Device)</b>
<b>Vorteile</b>	Agiert wie ein eigenständiger Netzwerkadapter	Geringer Traffic-Overhead
	Verschiedene Netzwerkprotokolle verfügbar	Da kein Ethernet-Layer, geringe Bandbreite
	Client ist transparent im Zielnetz	Gute Skalierbarkeit
<b>Nachteile</b>	Schlechte Skalierbarkeit	Nur IP-Pakete möglich
	Hoher Broadcast-Overhead am VPN-Tunnel	Keine Broadcasts möglich

Tabelle 4: Bridging-Mode vs. Routing-Mode

Für die Lösung der Aufgabe wurde der Routing-Mode (Tun-Device) eingesetzt.

Mit Abschluss dieses Kapitels wurden alle theoretischen Grundlagen, welche für den Rahmen des Laborpraktikums relevant sind, eingeführt. Im anschließenden Kapitel wird die Lösungsskizze für die Problemstellung beschrieben.

---

## 4 Lösungsskizze

In diesem Kapitel wird sich mit dem Thema befasst, wie sich eine Side-to-Side VPN Verbindung, die die Firmen A und B wie in Kapitel 2.1 beschrieben verbindet. Dazu waren zu Beginn mehrere Vorüberlegungen zu tätigen. Dies sind zum einen, wie wird das innere Netz der Firmen A und B geschützt, dann welcher VPN Dienst kann für diesen Zweck verwendet werden und welche weiteren Strategien braucht es bei der Umsetzung der Security Policies der Firmen.

### 4.1 Firewall

Wenn eine Firewall aufgebaut werden soll, muss sich zu Beginn überlegt werden, welche allgemeine Strategie mit der Firewall gefahren werden soll. Das heißt sollen alle Verbindungen Standardmäßig freigegeben werden und nur explizit nicht erlaubt Verbindungen geblockt werden (Default-Allow-Strategy), oder sollen alle Verbindungen blockiert werden und nur die die explizit erlaubt wurden, geöffnet werden (Default-Deny-Strategy). Im Standardfall wird in Unternehmen, die Strategie Default-Deny verwendet, weshalb auch im Versuchsaufbau diese Strategie gewählt wurde. Eine Firewall wird mit Hilfe des Userspace-Programmes IPTABLES (siehe Kapitel 3.4) unter Linux realisiert. Dafür wird eine Rules Datei (nachfolgend Skript genannt) angelegt, in dieser werden die Einstellungen und Regeln für die Firewall definiert. Zu Beginn des Skriptes wird der Interpreter für den Code angegeben. Dann wird begonnen die durch die Default Strategie vorgegebenen Default Policies umzusetzen (siehe Auflistung 2). Dabei sagt das -P, dass die Policy angesprochen werden soll, das INPUT, OUTPUT und FORWARD bezieht sich auf die im Kapitel 3.4 vorgestellten Chains. DROP sagt dabei aus, dass die eintreffenden Pakete, wenn keine weiteren Regeln definiert oder zutreffen nicht weitergeleitet und “fallengelassen” werden sollen.

Listing 2: Default Policy IPTABLE

---

```
#Verwenden der Tabelle FILTER
*filter
-P INPUT DROP
-P OUTPUT DROP
-P FORWARD DROP
```

---



---

Nach dem die Default Policy erfolgreich eingeführt wurde, werden die feingranulareren Regeln definiert. Dies setzt die Verwendung des Befehles -F voraus. Dieser löscht alle bisherigen Filterregeln, um die nach folgenden neu definierten Regeln einzuführen. Bei Linux ist dabei zu beachten das ein Teil der Interprozesskommunikation über das interne Netzwerk läuft. Dafür ist es nötig diese Kommunikation zuzulassen, dies geschieht wie in der nachfolgenden Auflistung 3 zusehen ist.

Listing 3: Interprozesskommunikation zulassen

```
# Interprozesskommunikation Verbindungen erlauben
-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
```

---

Dabei sagt das -A an welche Chain diese Regel angehängt werden soll. Das -i ist dabei die Option über welches Netzwerkinterface das Paket eingegangen ist, beziehungsweise -o für das Paket versenden. In diesem Fall das "lo" Netzwerkinterface. Wenn alle Prüfregeln auf das Paket zutreffen wird mit -j entschieden wie mit dem Paket verfahren werden soll, in diesem Beispiel soll es mit ACCEPT akzeptiert werden.

Listing 4: Weitere allgemeine Firewallregeln

```
# Erlaube ICMP Befehle
-A INPUT -p icmp -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT

# Erlaube SSH Verbindung
-A INPUT -p tcp --dport 22 -j ACCEPT

# Alle Verbindungen von innen nach aussen zulassen
-A FORWARD -i eth0 -o eth1 -m state --state NEW -j ACCEPT

# Erlaube nur bereits aufgebaute
# Verbindungen von aussen nach innen
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

---

Die in Auflistung 4 dargestellten Regeln sind allgemeine Regeln für die Firewall, diese erlauben das empfangen von Pingenfragen, den Fernzugriff mittels SSH, alle neuen Verbindungen von innen nach außen und alle bereits aufgebauten Verbindungen beziehungsweise alle Verbindungen die einen Bezug auf eine andere Verbindung besitzen. In

---

der nachfolgenden Tabelle 5 werden die hier verwendeten Optionen nochmals näher beschrieben. Nach den allgemeine Regeln müssen nun die VPN spezifischen Regeln definiert

Optionen	Beschreibung
-p	Gibt das Protokoll an welches verwendet werden soll hier icmp(Ping), tcp
-dport	Gibt den Destinationport an, auf den zugegriffen werden soll hier 22
eth0,eth1	bezieht sich auf das verwendete Netzwerkinterface
-m state	die enttreffenden Pakete sollen auf den Status überprüft werden
-state	Status der eintreffenden Pakete, hier NEW, RELATED, ESTABLISHED

Tabelle 5: IPTABLES Optionen und Beschreibung

werden. Dazu muss man die auf den VPN Ports eintreffenden und ausgehenden Pakete betrachten. Diese Ports sind entweder 1194 oder 1195. In der Auflistung 5 sind VPN Regeln zusehen. Eine Besonderheit bei VPN ist das zum einen alle verbindungen die über das Netzwerkinterface tun0 eintreffenden Pakete ohne Überprüfung an das Inteface eth0 übergeben werden. Des weiteren wird noch die Tabelle NAT benötigt, was das ”\*nat” angibt. In dieser Tabelle gibt es die Chain POSTROUTING, über diese kann nachträglich der Verkehrsheader eines Paketes verändert werden. MASQUERADE hat dabei die Funktion das wenn ein Paket versendet wird die Source-IP-Adresse so verändert wird das nur noch die IP-Adresse des Firewallserver ersichtlich ist. Dies hat den Grund das von außen nicht ersichtlich wird, was sich für IP-Adressen hinter der Firewall befinden und es so Angreifern erschwert wird diese anzugreifen.

Listing 5: Weitere VPN Firewallregeln

---

```
# Erlaube alle Verbindungen auf den VPN Ports
-A INPUT -i eth1 -p udp --dport 1194 -m state --state NEW -j ACCEPT
-A INPUT -i eth1 -p udp --dport 1195 -m state --state NEW -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i tun0 -j ACCEPT

#Forwarding fuer die VPN Verbindungen
-A FORWARD -i tun0 -o eth0 -m state --state NEW -j ACCEPT
-A FORWARD -i eth0 -o tun0 -m state --state NEW -j ACCEPT
-A FORWARD -i tun0 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth0 -o tun0 -m state --state RELATED,ESTABLISHED -j ACCEPT
COMMIT
```

---

---

```
#Verwenden der Tabelle NAT
*nat

#Loeschen der vorhandenen Regeln
-F

-A POSTROUTING -o eth1 -j MASQUERADE
COMMIT
```

---

Nach dem erstellen des Skriptes, muss dieses nun eingespielt werden. Dazu wird der Befehl "iptables-restore" verwendet. Dieser wird exemplarisch in der Auflistung 6 veranschaulicht, dabei liegt das Skript im Ordner "etc".

Listing 6: Einspielen des Firewallskriptes der Firma a  
iptables-restore < etc/iptables.firewall-a.rules

Nach dem das Skript eingespielt wurde, muss dieses noch so konfiguriert werden, dass es nach einem Serverneustart automatisch neu eingespielt wird. Dazu könne zwei Arten verwendet werden. Zum einen mit Hilfe des "if-pre-up.d" Directories. Dabei wird eine neue Datei angelegt mit dem Namen "iptables" angelegt. In der Auflistung 7 wird der Inhalt, der neu angelegten Datei gezeigt. Nun muss die Datei noch Ausführbar gemacht werden, dies geschieht mit Hilfe des Befehls "chmod +x"

Listing 7: Automatisches Laden des Iptablesskriptes bei Serverneustart

```
#!/bin/bash
/sbin/iptables-restore < /etc/iptables.fwirewall-a.rules
```

Die zweite Möglichkeit des automatischen Neuladens der Firewall ist mit Hilfe des Tools "iptables-persistent". Dieses Tool muss mit "aptitude" installiert werden. Nach dem das Tool installiert wurde, können mit dem Befehl "iptables-save > /etc/iptables/rules.v4" die Regeln für jeden Neustart automatisch geladen werden.

Eine weitere Konfiguration die vorgenommen werden muss, ist das Akzeptieren des Forwardings. Dazu muss im Verzeichnis "/proc" die Datei "/proc/sys/net/ipv4/ip\_forward" die Zeile in der Auflistung 8 eingefügt werden. Zu Beginn steht in dieser Datei lediglich

---

eine "0" diese muss ersetzt werden durch eine "1" dies aktiviert das Forwarding.

Listing 8: Forwarding aktivieren

```
/bin/echo "1" > /proc/sys/net/ipv4/ip_forward
```

Wurden all diese Einstellungen vorgenommen, kann begonnen werden mit der Konfiguration der Site-toSite VPN Verbindung, was im nachfolgenden Kapitel beschrieben wird.

## 4.2 VPN

Nach dem die Firewall auf den Servern der Firma A und B eingerichtet wurde, muss nun die VPN Verbindung erstellt werden. Dazu dient der Firewallserver der Firma A als Server und der Firewallserver der Firma B als Client. Diese Einrichtung wird in diesem Kapitel näher beschrieben.

Dafür muss zu Beginn auf beiden Servern das in Kapitel 3.6 vorgestellte Tool OpenVPN installiert werden. Dies geschieht wie schon im Kapitel 4.1 erwähnt mit dem Tool "aptitude". Nach der Installation müssen wie nachfolgend erklärt für den Server der Firma A und B ein Serverzertifikat erstellt werden.

### 4.2.1 Serverzertifikate

Wie schon in Kapitel 3.5 erklärt wird für die Einrichtung eines VPN's Zertifikate benötigt. Diese werden verwendet um eine Verbindungsverschlüsselung zu realisieren. In diesem Kapitelteil wird anhand der Firma A erklärt, wie ein solches Zertifikat in der Laborumgebung erstellt wird, um ein Zertifikat für Firma B zu erstellen, ist der gleiche Prozess notwendig.

Für das Erstellen eines Zertifikates ist das Tool OpenSSL notwendig. Über OpenSSL wird ein öffentlicher und privater Schlüssel erzeugt. Nach dem diese erstellt wurden, muss von der Zertifizierungsstelle die Konfigurationsdatei geladen werden. Mit Hilfe dieser Konfigurationsdatei des privaten Schlüssels und des Tools OpenSSL wird nun eine CSR (Certificate Signing Request) Datei erzeugt. Nach dem diese Datei erzeugt wurde, muss diese zur Zertifizierungsstelle hochgeladen werden. Nach dem Upload der Datei (Zertifikat), muss diese nun Signiert werden. Dies wird im Labor mit einem Shellskript

---

realisiert. Nach dem Signieren kann nun das fertige Zertifikat geladen und auf dem Server abgelegt werden. In der Auflistung 9 werden die einzelnen Befehle zur Erstellung der Schlüssel und der CSR Datei gezeigt und die einzelnen Befehle in der Tabelle 6 näher beschrieben.

#### Listing 9: Erzeugen eines privaten Schlüssels mit OpenSSL

```
#Privaten Schluessel erzeugen
openssl genrsa -out firma-A.key 2048

#Request erstellen - CSR-Datei
openssl req -new -key firma-A.key -config req.cnf -reqexts v3_req_srv -o
```

Befehle	Beschreibung
openssl	Tool zur Erstellung von privaten Schlüsseln
genrsa	Befehl zum erstellen des Schlüssels mittels RSA
-out	Ausgabe des Schlüssels in angegebene Datei
firma-A.key	Name der Schlüsseldatei
2048	Schlüssellänge in Bits
req -new	Erzeugen eines neuen Requests
-config	laden der Konfigurationsdatei req.cnf
-reqexts	Erweiterung des X.509 Formates

Tabelle 6: Befehle zur Erstellung eines Privaten Schlüssels mittels OpenSSL

### 4.2.2 Server Firma A

Nach dem die Zertifikate wie im vorherigen Kapitel erstellt wurden, kann nun mit dem Konfigurieren des VPN-Servers begonnen werden. Dazu wird zuerst mittels des Root-Users das Verzeichnis `"/etc/opnevpn"` erstellt. Nach dem dieses erstellt wurde werden zwei Unterverzeichnisse `"/etc/openvpn/certs"` und `"/etc/openvpn/keys"` angelegt. In das Unterverzeichnis `"certs"` werden das Serverzertifikat `"fw2-firma-A.crt"` und die Zertifikatschain (von der Zertifizierungsstelle) kopiert. In das Unterverzeichnis `"keys"` wird der private Schlüssel verschoben. Da der private Schlüssel geheimgehalten werden soll, muss die Berechtigung für das Unterverzeichnis noch geändert werden. Dies wird mit dem Befehl `"chmod -R 600 /etc/openvpn/keys/"` bewerkstelligt. Der nächste Schritt ist das

---

erstellen des Diffie-Hellman Schlüssels. Dieser Schlüssel wird später benötigt um später die Verbindung symmetrisch zu verschlüsseln. Der dafür notwendige Befehl wird in der Auflistung 10 gezeigt.

Listing 10: Erzeugen des Diffie-Hellman Schlüssels

openssl dhparam -out dh2048.pem 2048

Eine besondere Eigenschaft des Site-to-Site VPN's ist das man für jeden einzelnen Clienten eine besondere Konfiguration "pushen" kann. Dies bringt den Vorteil, dass wie hier verwendet, ein Client die Infrastruktur hinter der Firewall so verwenden kann, als wäre sie ein Teil seiner eigenen Infrastruktur. Dazu muss ein weiteres Unterverzeichnis mit dem Namen "client\_configs" erstellt werden. Darin muss eine Datei angelegt werden, dass den gleichen Namen trägt wie das Clientzertifikat. In dieser Datei werden die Konfigurationsdaten für den Clienten geschrieben, wie in Auflistung 13 zusehen ist.

Listing 11: Clientkonfigurationsdatei fw-firma-b

iroute 192.168.70.0 255.255.255.0  
iroute 192.168.80.0 255.255.255.0

Das "iroute" Statement gibt dem Client an welche IP-Adressen von ihm angesprochen werden können.

Nun muss als letztes die OpenVPN Konfigurationsdatei erstellt und angepasst werden. Dazu können die bei der Installation von OpenVPN mitgelieferten Beispieldateien verwendet werden und nach und nach an den Verwendungszweck angepasst werden. Für die Konfiguration für den Server wird die "server.conf" benötigt.

Listing 12: server.conf Datei der Firma A

script-security 3

port 1194

proto udp

dev tun

---

```
tls-server
auth SHA1

ca /etc/openvpn/cert/f223CA.chain.crt
cert /etc/openvpn/cert/fw-firma-a.crt
# This file should be kept secret
key /etc/openvpn/keys/firma-A.key

dh /etc/openvpn/dh2048.pem

server 192.168.100.0 255.255.255.0

ifconfig-pool-persist ipp.txt

route 192.168.70.0 255.255.255.0
route 192.168.80.0 255.255.255.0
push "route 192.168.30.0 255.255.255.0"
push "route 192.168.40.0 255.255.255.0"

client-config-dir /etc/openvpn/client-configs

keepalive 10 120

cipher aes-256-cbc

comp-lzo

user nobody
group nogroup

persist-key
persist-tun

verb 3
```

---

Die Auflistung 12 zeigt die "server.conf" der Firma A. Dabei wird von oben nach unten die folgenden Konfigurationsbefehle abgearbeitet.

**script-security 3** Beschreibt die Skript Sicherheit, das heißt hier wird definiert wie OpenVPN externe Programme und Skripte verwenden darf. Die drei erlaubt die Übergabe von Passwörtern an Skripte über Umgebungsvariablen.

**port 1194** Definiert den Port über den der VPN Verkehr abläuft. Alternativport 1195

---

**proto udp** Definiert über welches Protokoll kommuniziert wird.

**dev tun** Erzeugt einen gerouteten IP-Tunnel.

**tls-server auth SHA1** Authentifikationskonfiguration

**ca** Pfadangabe zur Zertifikatschain

**cert** Pfadangabe zum Serverzertifikat

**key** Pfadangabe zum Privaten Schlüssel

**dh** Pfadangabe zum Diffi-Hellman Key

**server** Spezifiziert den Server als Server und gibt den IP-Adress Raum des VPN Netzes an.

**ifconfig-pool-persist** Datei in dem IP-Adressen der Clients gespeichert werden, um beim nächsten Verbinden von dem selben Client diesem die gleiche IP-Adresse zu geben.

**route** Definiert die hinter dem Client liegenden IP-Adressen auf die von Serverseite aus zugegriffen werden kann.

**push** IP-Adressraum der an den Client "gepusht" werden, die der Client annehmen darf.

**client-config-dir** Pfadangabe zur Clientkonfiguration

**keepalive** Gibt an das nach 10 Sekunden Inaktivität ein Ping an den Client gesendet wird, nach 120 Sekunden Inaktivität wird ein weiterer Versuch unternommen sich mit dem Client zu verbinden.

**cipher** Kryptographische Chiffre, über die die Verbindung zwischen Client und Server verschlüsselt wird. Dies muss beim Client der gleiche Chiffre sein.

**comp-lzo** Aktiviert die Kompression der Verbindung, dies muss auch beim Client aktiviert sein.

**user** Nach der Initialisierung werden die Rechte des OpenVPN Nutzers auf nobody gesetzt, da zum starten eines VPN's Rootrechte benötigt und dies ist aber während der Sitzung nicht gewollt ist.

**group** Nach der Initialisierung werden die Rechte des OpenVPN Nutzers auf nogroup gesetzt, da zum starten eines VPN's Rootrechte benötigt und dies ist aber während der Sitzung nicht gewollt ist.



---

**persist-key** Sichert vor einem Keepalive Neustart die Schlüssel, um diese nicht nochmals erzeugen zu müssen.

**persist-tun** Ähnlich wie persist-key nur für das tun-interface.

**verb 3** Logging Optionen, über diese Einstellung kann die Genauigkeit des Loggens eingestellt werden.

### 4.2.3 Client Firma B

Ähnlich wie beim Server der Firma A, wie im Kapitel-teil zuvor, wird der Client der Firma B eingerichtet. Allerdings wird hier die "client.conf" benötigt und es wird kein "client.conf" Verzeichnis benötigt. Nachfolgend wird in der Auflistung die Konfigurationsdatei gezeigt und die einzelnen Begriffe nochmals erklärt.

---

Listing 13: client.conf Datei der Firma B

---

```
port 1194

client

dev tun

proto udp

remote 10.1.0.131 1194

user nobody
group nogroup

persist-key
persist-tun

ca /etc/openvpn/f223CA.chain.crt
cert /etc/openvpn/fw2.firma-b.f223.crt
key /etc/openvpn/vpnb.key

pull
auth SHA1

cipher aes-256-cbc

comp-lzo
```

---

verb 3

---

**client** Spezifiziert den Client als Client

**remote** Gibt die IP-Adresse und den Port des VPN-Servers an, mit dem sich der Client verbinden soll.

**tls-client** Spezifiziert den Client als tls-Client

**pull** holt sich über diesen Befehl die von Server "gepushten" Konfigurationen

## 5 Auswertung

In diesem Kapitel werden die im Kapitel 4 vorgestellten Konfigurationen verwendet und eine Auswertung der VPN Verbindung von Firma B(Client) zur Firma A(Server) gemacht.

Um die VPN Verbindung zu starten wird zuerst der Server mit Hilfe des Befehls "openvpn /etc/openvpn/server.conf" gestartet. Nach dem der Server gestartet wurde, wird bei Firma B der Client mit "openvpn /etc/openvpn/client.conf" gestartet. Dabei initiiert der Client den Verbindungsaufbau. Die Abbildung 4 stellt den drei Wege SSL Handshake

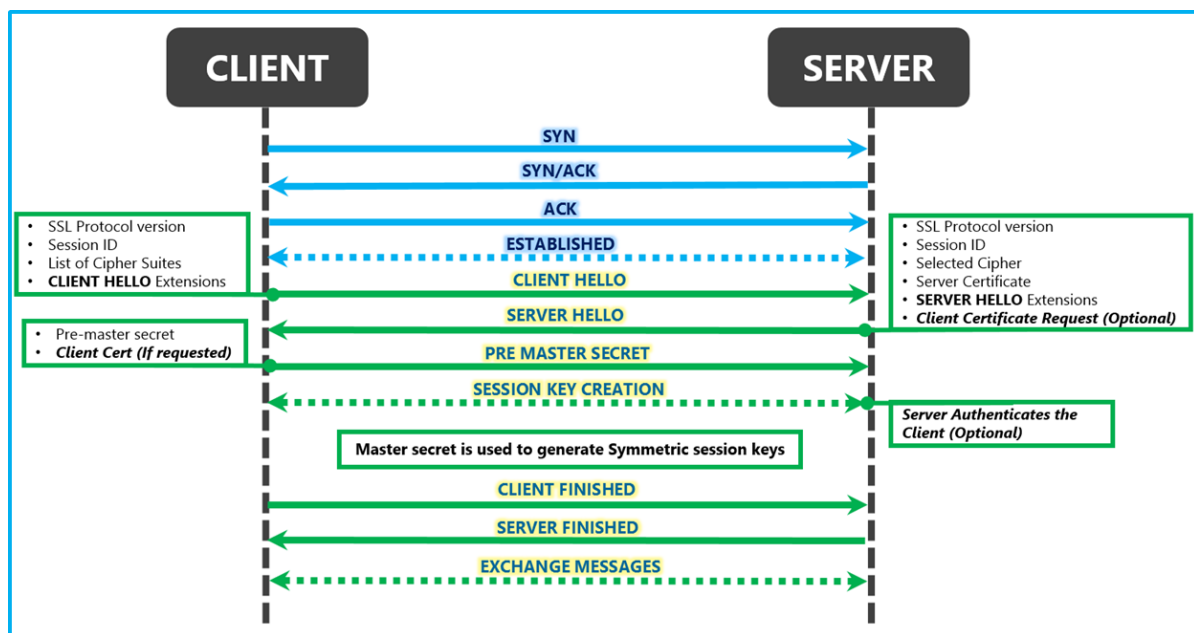
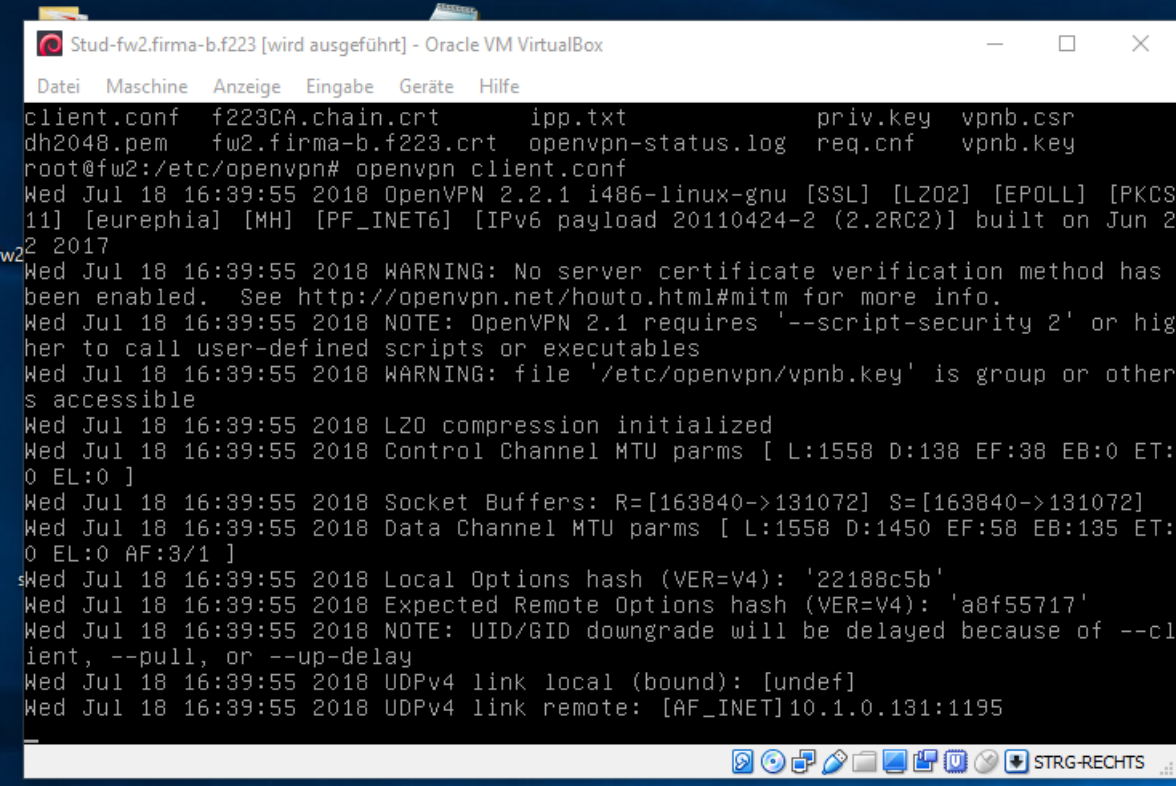


Abbildung 4: Verbindungsaufbau eines VPN's[Rij18]

zwischen Client und Server dar. Dabei wird zu Beginn eine unverschlüsselte Verbindung aufgebaut, indem dem Client die Server-IP und der Port mitgeteilt wird. Nach dem diese Verbindung "ESTABLISHED" ist sendet der Client einen "Client Hallo" auf die nun bekannte IP-Adresse und den Port, dabei liefert er zum einen das von ihm verwendete SSL Protokoll, eine Session ID, eine Liste von Chiffren die er verwenden kann, an den Server aus. Auf dieses "Hallo" sendet der Server selbst ein "Server Hallo" und liefert dabei dem Client ebenfalls seine SSL Protokoll Version, die Session ID, den von ihm gewählten Chiffre, sein Serverzertifikat und Gegeben falls seine CCR (Client Certificate Request)

aus. Hat der Client den "Server Hallo" erhalten, überprüft dieser die Gültigkeit, das Vertrauen in die CA und den öffentlichen Schlüssel mit der Digitalen Signatur. Wenn er dies Überprüft hat erzeugt der Client mit Hilfe des öffentlichen Schlüssels des Servers ein Pre-master-secret. Dieses Secret sendet er nun an den Server. Der Server entschlüsselt das Pre-master-secret. Nun erstellen Server und Client aus dem pre-master-secret einen symmetrischen Schlüssel über den die Kommunikation verschlüsselt und entschlüsselt wird.

Im Labor stießen wir auf ein nicht zu überbrückendes Hindernis. Dies entstand nach dem TCP-Handshake, im SSL-Handshake. Dabei konnte der Server sein "Server Hallo" nicht an den Client zurück schicken konnte, was darin mündete das der Client kein pre-master-secret erstellen konnte und den Vorgang abbrach und nach 120 Sekunden erneut versuchte eine Verbindung aufzubauen, mit dem gleichen Ergebnis. Dies ist zum einen in der Abbildung 5 und Abbildung 6 zuerkennen



```
client.conf f223CA.chain.crt ipp.txt priv.key vpnb.csr
dh2048.pem fw2.firma-b.f223.crt openvpn-status.log req.cnf vpnb.key
root@fw2:/etc/openvpn# openvpn client.conf
Wed Jul 18 16:39:55 2018 OpenVPN 2.2.1 i486-linux-gnu [SSL] [LZO2] [EPOLL] [PKCS
11] [eurephia] [MH] [PF_INET6] [IPv6 payload 20110424-2 (2.2RC2)] built on Jun 2
2 2017
Wed Jul 18 16:39:55 2018 WARNING: No server certificate verification method has
been enabled. See http://openvpn.net/howto.html#mitm for more info.
Wed Jul 18 16:39:55 2018 NOTE: OpenVPN 2.1 requires '--script-security 2' or hig
her to call user-defined scripts or executables
Wed Jul 18 16:39:55 2018 WARNING: file '/etc/openvpn/vpnb.key' is group or other
s accessible
Wed Jul 18 16:39:55 2018 LZO compression initialized
Wed Jul 18 16:39:55 2018 Control Channel MTU parms [ L:1558 D:138 EF:38 EB:0 ET:
0 EL:0 ]
Wed Jul 18 16:39:55 2018 Socket Buffers: R=[163840->131072] S=[163840->131072]
Wed Jul 18 16:39:55 2018 Data Channel MTU parms [ L:1558 D:1450 EF:58 EB:135 ET:
0 EL:0 AF:3/1 ]
Wed Jul 18 16:39:55 2018 Local Options hash (VER=V4): '22188c5b'
Wed Jul 18 16:39:55 2018 Expected Remote Options hash (VER=V4): 'a8f55717'
Wed Jul 18 16:39:55 2018 NOTE: UID/GID downgrade will be delayed because of --cl
ient, --pull, or --up-delay
Wed Jul 18 16:39:55 2018 UDPv4 link local (bound): [undef]
Wed Jul 18 16:39:55 2018 UDPv4 link remote: [AF_INET]10.1.0.131:1195
```

Abbildung 5: Log Ausgabe des gestarteten VPN-Client

```
Stud-fw2.firma-a.f223 [wird ausgeführt] - Oracle VM VirtualBox
Datei  Maschine  Anzeige  Eingabe  Geräte  Hilfe
Wed Jul 18 16:40:12 2018 UDPv4 link local (bound): [undef]
Wed Jul 18 16:40:12 2018 UDPv4 link remote: [undef]
Wed Jul 18 16:40:12 2018 MULTI: multi_init called, r=256 v=256
Wed Jul 18 16:40:12 2018 IFCONFIG POOL: base=192.168.100.4 size=62, ipv6=0
Wed Jul 18 16:40:12 2018 IFCONFIG POOL LIST
Wed Jul 18 16:40:12 2018 Initialization Sequence Completed
Wed Jul 18 16:40:26 2018 MULTI: multi_create_instance called
Wed Jul 18 16:40:26 2018 10.1.0.132:1195 Re-using SSL/TLS context
Wed Jul 18 16:40:26 2018 10.1.0.132:1195 LZ0 compression initialized
Wed Jul 18 16:40:26 2018 10.1.0.132:1195 Control Channel MTU parms [ L:1558 D:13
8 EF:38 EB:0 ET:0 EL:0 ]
Wed Jul 18 16:40:26 2018 10.1.0.132:1195 Data Channel MTU parms [ L:1558 D:1450
EF:58 EB:135 ET:0 EL:0 AF:3/1 ]
Wed Jul 18 16:40:26 2018 10.1.0.132:1195 Local Options hash (VER=V4): 'a8f55717'
Wed Jul 18 16:40:26 2018 10.1.0.132:1195 Expected Remote Options hash (VER=V4):
'22188c5b'
Wed Jul 18 16:40:26 2018 10.1.0.132:1195 TLS: Initial packet from [AF_INET]10.1.
0.132:1195, sid=698e33a1 d0ad4769
Wed Jul 18 16:40:26 2018 10.1.0.132:1195 write UDPv4 []: Network is unreachable
(code=101)
Wed Jul 18 16:40:28 2018 10.1.0.132:1195 write UDPv4 []: Network is unreachable
(code=101)
Wed Jul 18 16:40:32 2018 10.1.0.132:1195 write UDPv4 []: Network is unreachable
(code=101)
```

Abbildung 6: Log Ausgabe des gestarteten VPN-Servers

---

## 6 Fazit

Trotz der nicht Aufklärung des Fehlers in der konfigurierten Lösung konnten einige sehr wichtige Erkenntnisse aus diesem Praktikum gewonnen werden. Der Bereich der Netzwerkarchitektur in Verbindung mit den vielen Sicherheitsaspekten wird im heutigen Zeitalter immer entscheidender. Fast jeder Mitarbeiter in einem globalen Unternehmen nutzt fast täglich die Möglichkeiten einer sicheren VPN-Verbindung, ohne sich darüber im klaren zu sein, was für eine Komplexität bei der Bereitstellung einer solchen Verbindung überwunden werden muss.

Nach der Absolvierung des Laborpraktikums wurden viele Aspekte, welche vorher einfach hingenommen wurden, kritisch und fachlich beleuchtet. Das Einrichten einer VPN-Verbindung mit den dazugehörigen Zertifikate, sowie die Konfiguration der internen Firewalls, hat mal wieder die Annahme bestätigt, dass es bei der Konfiguration von Netzwerken auf der Basis von Linux auf jede Kleinigkeit ankommt. So führt eine Leerzeile in einem Skript dazu, dass ein komplettes System nicht mehr so reagiert wie es sollte. Die Schwierigkeit solch einen Fehler zu finden ist, wie das sprichwörtliche Suchen der Nadeln in einem Heuhaufen. Zudem gestaltet sich die Fehlersuche in einem Netzwerk deutlich komplizierter, als bspw. in der Anwendungsentwicklung. Bei der Anwendungsentwicklung kann durch Einsatz einer IDE und dem darin enthaltenen Debugger relativ schnell auf einen Fehler geschlossen werden. Da bei der Fehlersuche eher die Gewohnheiten aus der Anwendungsentwicklung zu Grunde lagen, brauchte es etwas Zeit bis es Möglich war, in dieser Domäne eine effektive und erfolgreiche Fehlersuche durchzuführen.

Der Punkt der Fehlersuche war auch der, mit dem der größte Zeitaufwand verbunden war. Die rein logische Umsetzung der Problemstellung war relativ schnell erledigt. Auch das parallele Aufsetzen mehrerer Systeme mit dem Lösungsansatz führte immer zum gleichen Fehlerfall, wie in der Auswertung bereits beschrieben. Selbst die intensive Hilfe der Laborassistenten konnte den Fehler nicht aufklären. Da es bei der Laborumgebung um eine virtualisierte Umgebung handelte, liegt die Vermutung nahe, dass der Fehler in der Konfiguration der virtuellen Umgebung zu finden sein könnte. Dies bleibt jedoch nur eine reine Vermutung und konnte nicht mit Fakten belegt werden.

Es bleibt jedoch festzuhalten, dass es die Lernkurve im Rahmen des Praktikums sehr steil war. Die Bereiche der Netzwerkarchitekturen bis hin zur Erstellung eines Server-Zertifikats sind aktuell eine sehr wichtige Domäne in der Informatik.

---

## Literatur

- [bas] <https://wiki.ubuntuusers.de/Bash/>
- [edi] <https://wiki.ubuntuusers.de/Editoren/>
- [Gab] <https://wirtschaftslexikon.gabler.de/definition/internet-37192>
- [Hea] <https://blog.milsystems.de/2011/11/netzwerktechnologie/>
- [int] <https://wirtschaftslexikon.gabler.de/definition/intranet-38840>
- [int16] *DMZ, demilitarisierte Zone.* <https://www.iternas.com/dmz>. Version: 2016, Abruf: 20.07.2018
- [ipt] <https://wiki.ubuntuusers.de/iptables2/>
- [KR08] KUROSE, James F. ; ROSS, Keith W.: *Computernetzwerke: der Top-Down-Ansatz*. 4., aktual. Aufl. München [u.a.] : Pearson Studium, 2008 (IT - Informatik). – 896 Seiten S. <http://www.gbv.de/dms/ilmenau/toc/558366325.PDF>. – ISBN 978-3-8273-7330-4. – HTWG Konstanz
- [ND18] NEUSCHWANDER, Jürgen ; DÜSTERHÖFT, Sabine: *IT-Sicherheitsarchitekturen Aufgabenstellung zum Praktikum*. March 2018
- [ope] [https://www.thomas-krenn.com/de/wiki/OpenVPN\\_Grundlagen](https://www.thomas-krenn.com/de/wiki/OpenVPN_Grundlagen)
- [Rij18] RIJN, Robert van: An overview of the SSL Handshake. In: *Medium* (2018), May. <https://medium.com/@robertvanrijn/an-overview-of-the-ssl-handshake-3885c37c3e0f>
- [she] [https://wiki.ubuntuusers.de/Shebang\\_f%C3%BCr\\_Shellskripte/](https://wiki.ubuntuusers.de/Shebang_f%C3%BCr_Shellskripte/)