

Hochschule für Technik und Wirtschaft Berlin

**Projektarbeit**

**Projektdokumentation Informationssicherheit**

**Studiengang Wirtschaftsinformatik**

Verfasser(in):

Moritz Jürgens, Placeholder

Matrikelnummer:

0581194, Placeholder

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>ii</b>
<b>Abkürzungsverzeichnis</b>	<b>iii</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Projektphase 1 – Firewall</b>	<b>3</b>
2.1 Offene Konfiguration . . . . .	3
2.1.1 Script . . . . .	3
2.1.2 Penetrationstest – Inward . . . . .	5
2.1.3 Penetrationstest - Outward . . . . .	9
2.2 Geschlossene Konfiguration . . . . .	9
2.2.1 Script . . . . .	9
2.2.2 Penetrationstest – Inward . . . . .	12
2.2.3 Penetrationstest – Outward . . . . .	15
2.3 Standard Konfiguration . . . . .	16
2.3.1 Script . . . . .	17
2.3.2 Penetrationstest – Inward . . . . .	17
2.3.3 Penetrationstest – Outward . . . . .	18
<b>3 Projektphase 2 – Webserver</b>	<b>20</b>
3.1 Setup . . . . .	20
3.1.1 Datenbank . . . . .	20
3.1.2 API . . . . .	21
3.1.3 Frontend . . . . .	21
3.2 Cross-Site-Scripting . . . . .	21
3.3 SQL Injection . . . . .	23
<b>Literaturverzeichnis</b>	<b>24</b>

# Abbildungsverzeichnis

Abbildung 1.1 Aufbau des Netzwerks . . . . .	2
Abbildung 2.1 Datenverkehr bei offener Firewall . . . . .	7
Abbildung 2.2 Datenverkehr bei offener Firewall – UDP . . . . .	8
Abbildung 2.3 Datenverkehr bei geschlossener Firewall . . . . .	13
Abbildung 2.4 Datenverkehr bei geschlossener Firewall – UDP . . . . .	14
Abbildung 2.5 Datenverkehr der Opfermaschine . . . . .	18
Abbildung 2.6 Fort.: Datenverkehr der Opfermaschine . . . . .	19
Abbildung 2.7 Datenverkehr der Opfermaschine – UDP . . . . .	19
Abbildung 3.1 Database Setup . . . . .	20
Abbildung 3.2 Blog-Frontend . . . . .	22

# Abkürzungsverzeichnis

**XSS**      Cross-Site-Scripting

# **1 Einleitung**

**Maybe Project preview or smth...**

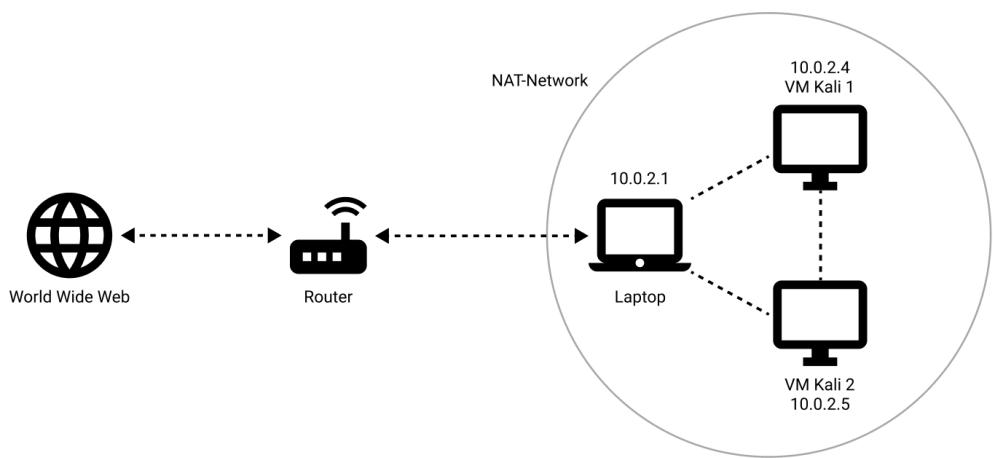


Abbildung 1.1: Aufbau des Netzwerks

# 2 Projektphase 1 – Firewall

Die Firewall-Konfigurationen wurden mittels IP-Tables auf der Opfermaschine erstellt. Dabei wurden Bash-Scripts implementiert, um die jeweils notwendigen Befehle nachvollziehen zu können. Zudem wird sämtlicher Datenverkehr der Penetrationstests aufgezeichnet und interpretiert.

## 2.1 Offene Konfiguration

Zunächst wird auf der Opfermaschine eine offene Konfiguration der Firewall erstellt. Das bedeutet in diesem Fall, dass jeglicher eingehender sowie jeglicher ausgehender Datenverkehr ungehindert zugelassen wird. In dieser Konfiguration werden weder Kommunikationsquellen noch Inhalt gefiltert oder geblockt.

### 2.1.1 Script

Der erste Schritt setzt die IP-Tables auf ihre Ursprungswerte zurück. Das heißt, dass alle einkommenden, ausgehenden und weitergeleiteten Pakete ungefiltert zugelassen werden. Zusätzlich zu der normalen IP-Tables Konfiguration wird hier auch noch die NAT-Konfiguration gesetzt. Das Script besteht aus drei wesentlichen Prozessen:

- Einstellung der Regeln (inklusive NAT)
- Flushing der Regeln
- Logging

Als nächstes werden alle noch überbleibenden Ausnahmeregelungen ‘geflushed’. Das bedeutet, dass diese gelöscht und auf den Standardwert zurückgesetzt werden. Hierdurch wird eine saubere und reproduzierbare Basiskonfiguration sichergestellt.

Abschließend wird das Logging behandelt. Damit am Ende des Vorgangs nachvollzogen werden kann, welche Pakete eingetroffen bzw. ausgetreten sind, wird der Datenverkehr vom Script aus protokolliert. Zusätzlich zu den drei Funktionen des Scripts wird nach Abschluss die Konfiguration in der Konsole ausgegeben.

```
1 #!/bin/sh
2
3 printf '=%.0s' {1..50}
4 printf "\n\t\tFirewall-Open\n"
5 printf '=%.0s' {1..50}
6
7 IPTABLES="/sbin/iptables"
8 printf "\n\nResetting default policies...\n"
9 $IPTABLES -P INPUT ACCEPT
10 $IPTABLES -P FORWARD ACCEPT
11 $IPTABLES -P OUTPUT ACCEPT
12
13 printf "Resetting NAT policies...\n"
14 $IPTABLES -t nat -P PREROUTING ACCEPT
15 $IPTABLES -t nat -P POSTROUTING ACCEPT
16 $IPTABLES -t nat -P OUTPUT ACCEPT
17
18 printf "Flushing rules...\n\n"
19 $IPTABLES -F
20 $IPTABLES -t nat -F
21 $IPTABLES -X
22 $IPTABLES -t nat -X
23 $IPTABLES -L -v
24 printf '=%.0s' {1..50}
25 printf "\n\t\tProcess finished"
26
27 $IPTABLES -A INPUT -p tcp -j LOG -m limit --limit 5/m \
28 --log-prefix 'ACCEPT IN TCP: '
29 $IPTABLES -A INPUT -p udp -j LOG -m limit --limit 5/m \
30 --log-prefix 'ACCEPT IN UDP: '
31 $IPTABLES -A OUTPUT -p tcp -j LOG -m limit --limit 5/m \
32 --log-prefix 'ACCEPT OUT TCP: '
33 $IPTABLES -A OUTPUT -p udp -j LOG -m limit --limit 5/m \
34 --log-prefix 'ACCEPT OUT UDP: '
```

Output:

```

1 =====
2          Firewall - Open
3 =====
4
5 Resetting default policies...
6 Resetting NAT policies...
7 Flushing rules...
8
9 Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
10   pkts bytes target     prot opt in     out    source      destination
11
12 Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
13   pkts bytes target     prot opt in     out    source      destination
14
15 Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
16   pkts bytes target     prot opt in     out    source      destination
17 =====
18
18          Process finished

```

## 2.1.2 Penetrationstest – Inward

Da nun die Firewall der Opfermaschine konfiguriert wurde, können Penetrationstests durchgeführt werden. Zunächst wird geprüft, welche Informationen über einen Portscan mit NMAP herausgefunden werden können. Während der laufenden Scans wird zusätzlich der Datenverkehr der Opfermaschine mit Hilfe von Wireshark aufgezeichnet. Dies hilft dabei die Reaktion – bzw. das Fehlen einer Reaktion – der Opfermaschine nachzuvollziehen.

### **nmap 10.0.2.4 -p- -A -T4**

Der erste NMAP-Scan wird mit vier Parametern versehen. Zunächst muss die Zieladresse des Opfers angegeben werden. In diesem Fall befindet sich das Opfer im NAT-Netzwerk an der Adresse 10.0.2.4. Als nächstes wird die gewünschte Menge der Ports angegeben. Standardmäßig werden die 1000 wichtigsten Ports geprüft. Auf Grund der Gründlichkeit dieses Tests werden jedoch alle 65.535 Ports der Opfermaschine gescannt – zu erkennen am Parameter -p-. Darauf folgend wird erneut ein Parameter zu Gunsten der Gründlichkeit gesetzt. Der Parameter -A setzt die Aggressivität des Scans. So werden durch diesen Parameter Funktionen wie OS-Detection, Version-Scanning, Script-Scanning und Traceroute

verwendet. Zuletzt wird ein Geschwindigkeitsparameter gesetzt.

Output:

```
1 Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-14 04:42 EST
2 Nmap scan report for 10.0.2.4
3 Host is up (0.00049s latency).
4 All 65535 scanned ports on 10.0.2.4 are closed
5 MAC Address: 08:00:27:BE:A1:CA (Oracle VirtualBox virtual NIC)
6 Too many fingerprints match this host to give specific OS details
7 Network Distance: 1 hop
8
9 TRACEROUTE
10 HOP RTT      ADDRESS
11 1    0.49 ms 10.0.2.4
```

Im Output ist zu erkennen, dass alle gescannten Ports geschlossen sind. Dies war zu erwarten, da auf der Opfermaschine keine Services laufen und somit keine Ports genutzt werden. Dennoch lässt der Fakt, dass die Ports als geschlossen gezeigt werden, auf weitere Informationen schließen. Zum einen wird dadurch bekannt, dass die Anfragen ungefiltert an die Opfermaschine durchgekommen sind. Im gleichen Zuge wird erkannt, dass die Angreifermaschiene Antworten auf ihre Anfragen erhalten hat. Demnach kann der Angreifer zu dem Entschluss kommen, dass die Firewall offen ist. Zusätzlich zu dem Portstatus wird die MAC-Adresse und der Hersteller des Geräts identifiziert. Zudem versucht das Tool Angaben über das Betriebssystem zu machen. In diesem Fall konnte kein Betriebssystem identifiziert werden, da die Antworten zu ungenau bzw. zu generisch waren. Als letzte Information gibt NMAP die Traceroute an. Das heißt, es wird angegeben welchem Pfad die Anfragen gefolgt sind und wie lange sie für den Rundentrip gebraucht haben.

Im Wireshark-Mitschnitt (Abbildung 2.1) wird der Datenverkehr der Opfermaschine gezeigt. Zu erkennen in grau, sind die eingehenden Anfragen der Angreifermaschiene. Hier werden TCP SYN Anfragen an das Opfer geschickt und eine ACK bzw. RST Antwort erwartet. Dies ist die Einleitung einer TCP Verbindung, welche bei diesem Scan nie vollständig durchgeführt wird. Da die Ports der Opfermaschine nicht gefiltert werden, werden die Anfragen verarbeitet und dem Angreifer wird geantwortet. Bei Eingang einer ACK oder RST Nachricht auf der Angreifermaschiene wird der Kontakt zum Port abgebrochen und somit keine Verbindung erstellt. Alleine durch das Antworten verrät das Opfer bereits den Status der Ports. Sollte der Angreifer eine ACK Nachricht erhalten, so ist der gefragte Port offen und "hört auf einkommende Anfragen. Im Falle einer RST Nachricht ist der Port geschlossen. Sollte keine Antwort verschickt werden interpretiert NMAP den Port als gefiltert. Ein SYN Scan eignet sich besonders aufgrund der Geschwindigkeit des Scans und der Verlässlichkeit der Ergebnisse.

No.	Time	Source	Destination	Protocol	Length	Info
1309...	2.389326613	10.0.2.4	10.0.2.5	TCP	56	9518 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389330345	10.0.2.4	10.0.2.5	TCP	56	24125 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389334849	10.0.2.4	10.0.2.5	TCP	56	9890 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389338603	10.0.2.4	10.0.2.5	TCP	56	50466 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389342290	10.0.2.4	10.0.2.5	TCP	56	23869 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389345911	10.0.2.4	10.0.2.5	TCP	56	45862 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389350431	10.0.2.5	10.0.2.4	TCP	62	45488 → 3786 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389350461	10.0.2.5	10.0.2.4	TCP	62	45488 → 9367 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389350487	10.0.2.5	10.0.2.4	TCP	62	45488 → 64108 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389350516	10.0.2.5	10.0.2.4	TCP	62	45488 → 33194 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389350542	10.0.2.5	10.0.2.4	TCP	62	45488 → 16112 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389350569	10.0.2.5	10.0.2.4	TCP	62	45488 → 57786 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389350596	10.0.2.5	10.0.2.4	TCP	62	45488 → 34349 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389350623	10.0.2.5	10.0.2.4	TCP	62	45488 → 21900 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389354145	10.0.2.4	10.0.2.5	TCP	56	3786 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389358154	10.0.2.4	10.0.2.5	TCP	56	9367 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389362116	10.0.2.4	10.0.2.5	TCP	56	64108 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389365713	10.0.2.4	10.0.2.5	TCP	56	33194 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389369550	10.0.2.4	10.0.2.5	TCP	56	16112 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389373174	10.0.2.4	10.0.2.5	TCP	56	57786 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389376960	10.0.2.4	10.0.2.5	TCP	56	34349 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389380690	10.0.2.4	10.0.2.5	TCP	56	21900 → 45488 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1310...	2.389387418	10.0.2.5	10.0.2.4	TCP	62	45488 → 65028 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389387444	10.0.2.5	10.0.2.4	TCP	62	45488 → 38789 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389387471	10.0.2.5	10.0.2.4	TCP	62	45488 → 8009 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389387498	10.0.2.5	10.0.2.4	TCP	62	45488 → 34444 [SYN] Seq=0 Win=1024 Len=0 MSS=1
1310...	2.389387526	10.0.2.5	10.0.2.4	TCP	62	45488 → 43681 [SYN] Seq=0 Win=1024 Len=0 MSS=1

Abbildung 2.1: Datenverkehr bei offener Firewall

**nmap 10.0.2.4 -sU -T4**

Zu prüfen, ob die Verbindung via UDP ebenfalls offen ist, wird als nächster Schritt ein UDP Scan mit NMAP durchgeführt. Wie bei dem vorherigen Scan sind hier ebenfalls die Parameter für die Geschwindigkeit gesetzt. Der -A Parameter wurde gegen -sU ausgetauscht. Der -sU Parameter gibt den Scantyp an. So wird anstatt eines TCP SYN Scans ein UDP Scan durchgeführt. Obwohl TCP den Großteil des Internetverkehrs prägt, ist UDP kein zu vernachlässigendes Protokoll. Bei diesem Scan werden UDP Pakete an die Ports geschickt. NMAP erwartet im Gegenzug ICMP Antworten. Entweder die Opfermaschine schickt eine Nachricht, dass der gewünschte Port nicht erreichbar ist – Port ist geschlossen – oder es wird eine andere Errornachricht verschickt, in welchem Falle der Port als gefiltert gesehen wird. Sollte das Opfer eine UDP Antwort verschicken, so ist der Port offen. Sei es der Fall, dass der Angreifer keine Antwort erhält wird der Port als entweder offen oder gefiltert interpretiert. Ein großes Hindernis an UDP Scanning ist der Zeitaufwand, da NMAP bei offenen Ports keine Antwort erhält und daraufhin auf den Timeout warten muss. Auf Grund dessen werden bei diesem Scan nur die wichtigsten 1000 Ports gescannt. Zudem ist die Verbindung via UDP wesentlich unverlässlicher als über TCP, was zu verfälschten Ergebnissen führen kann.

Auf der Abbildung 2.2 ist zu erkennen, wie der UDP Scan verläuft. Wie bereits beschrieben werden der Opfermaschine UDP Pakete geschickt (zu sehen in Blau). Darauf folgt die Antwort des Opfers – in diesem Fall als ICMP Error – dass der gewünschte Port nicht zu erreichen ist. Aus diesem Grund interpretiert NMAP die Ports der Opfermaschine als geschlossen.

No.	Time	Source	Destination	Protocol	Length	Info
3384...	8716.8904913...	10.0.2.5	10.0.2.4	UDP	84	53558 → 49161 Len=40
3384...	8717.6904645...	10.0.2.5	10.0.2.4	UDP	84	53559 → 49161 Len=40
3384...	8717.6904910...	10.0.2.4	10.0.2.5	ICMP	112	Destination unreachable (Port unreachable)
3384...	8718.5001970...	10.0.2.5	10.0.2.4	UDP	62	53558 → 30718 Len=0
3384...	8718.5002380...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8719.3123998...	10.0.2.5	10.0.2.4	UDP	62	53558 → 9 Len=0
3384...	8719.3124315...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8720.1258298...	10.0.2.5	10.0.2.4	UDP	62	53558 → 445 Len=0
3384...	8720.1258636...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8720.9562444...	10.0.2.5	10.0.2.4	UDP	84	53558 → 34555 Len=40
3384...	8721.7595114...	10.0.2.5	10.0.2.4	UDP	62	53559 → 34555 Len=10
3384...	8721.7595353...	10.0.2.4	10.0.2.5	ICMP	82	Destination unreachable (Port unreachable)
3384...	8722.5746090...	10.0.2.5	10.0.2.4	UDP	62	53558 → 26407 Len=0
3384...	8722.5746432...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8723.3912965...	10.0.2.5	10.0.2.4	UDP	62	53558 → 4000 Len=0
3384...	8723.3913316...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8724.2025668...	10.0.2.5	10.0.2.4	UDP	62	53558 → 21366 Len=0
3384...	8724.2025968...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8725.0081718...	10.0.2.5	10.0.2.4	UDP	62	53558 → 789 Len=0
3384...	8725.57461957...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8725.8223053...	10.0.2.5	10.0.2.4	UDP	62	53558 → 19165 Len=0
3384...	8726.6355016...	10.0.2.5	10.0.2.4	UDP	62	53586 → 19683 Len=0
3384...	8726.6355388...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8727.4552902...	10.0.2.5	10.0.2.4	UDP	62	53558 → 19165 Len=0
3384...	8727.4553166...	10.0.2.4	10.0.2.5	ICMP	72	Destination unreachable (Port unreachable)
3384...	8728.2867524...	10.0.2.5	10.0.2.4	UDP	116	53558 → 1056 Len=72
3384...	8728.2867850...	10.0.2.4	10.0.2.5	ICMP	144	Destination unreachable (Port unreachable)

Abbildung 2.2: Datenverkehr bei offener Firewall – UDP

```

1 Nmap scan report for 10.0.2.4
2 Host is up (0.0011s latency).
3 Not shown: 992 closed ports
4 PORT      STATE            SERVICE
5 49/udp    open|filtered  tacacs
6 800/udp   open|filtered  mdb�_daemon
7 17505/udp open|filtered  unknown
8 19075/udp open|filtered  unknown
9 19332/udp open|filtered  unknown
10 26872/udp open|filtered  unknown
11 27892/udp open|filtered  unknown
12 33281/udp open|filtered  unknown
13 MAC Address: 08:00:27:BE:A1:CA (Oracle VirtualBox virtual NIC)
14
15 Nmap done: 1 IP address (1 host up) scanned in 1088.71 seconds

```

Am Output des Scans lässt sich erkennen, dass dieser wesentlich länger gebraucht hat als der vorherige TCP Scan. So hat der UDP Scan 18 Minuten gebraucht um die wichtigesten 1000 Ports des Opfers zu scannen. Des Weiteren wird berichtet, dass acht Ports offen bzw. gefiltert sind. Da keine Services auf der Opfermaschine laufen oder Firewall-Filter gesetzt sind lässt sich vermuten, dass diese Ports fehlerhaft als offen | gefiltert angezeigt werden. So kann es sein, dass die Opfermaschine nicht schnell genug auf die Anfragen des Angreifers geantwortet haben und somit der Timeout von NMAP erreicht wurde.

### 2.1.3 Penetrationstest - Outward

Als nächstes wird getestet, welchen Einfluss die Firewall auf die Funktionalität der Maschine hat. So wird zunächst getestet

## 2.2 Geschlossene Konfiguration

Als nächstes wird eine Konfiguration getestet, in der alle Verbindungen geblockt werden. Zu dieser Konfiguration gehören die Ausnahmen: SSH und localhost. Das Ziel dieses Tests ist es zu prüfen, wie die Opfermaschine von außen angesprochen werden kann – bzw. welche Informationen ein potentieller Angreifer erhalten könnte.

### 2.2.1 Script

Dieses Script besteht aus sechs Teilen:

- Flushing der Regeln
- Ausnahmeregelungen
- Standardregelungen
- Localhost
- Bestehende Verbindungen
- Logging

Zunächst werden alle bestehenden Regeln gelöscht. Dies geschieht wie in dem vorherigen Script mit Hilfe des Flushingbefehls. Darauf folgen die Ausnahmeregelungen. Diese werden hier als nächstes gesetzt, da die Reihenfolge der Regeln bei den IP-Tables durchaus einen Einfluss hat. In diesem Fall wird der Port 22 freigegeben. Sowohl eintreffende wie auch ausgehende Nachrichten dürfen über den Port 22 gehandhabt werden. Dies erlaubt es mit der Virtual Machine eine SSH Verbindung aufzubauen.

Um dann die Firewall so einzustellen, dass keine weiteren Verbindungen zugelassen werden, werden nun die Standardregelungen gesetzt. Im Gegenzug zum vorherigen Script werden nun die eingehenden, ausgehenden und weitergeleiteten Verbindungen "gedropppt". So werden alle Pakete die über die betroffenen Ports laufen fallen gelassen und nicht bearbeitet. Damit trotzdem der localhost weiterhin funktioniert, werden noch einmal Ausnahmen beschrieben. Zusätzlich werden bestehende Verbindungen weiterhin zugelassen, um keine laufenden Prozesse zu stören. Somit kann gesichert sein, dass plötzliche Änderungen in der

Firewall keine Probleme in der Anwendung mitsichziehen. Abschließend werden erneut die Loggingbefehle gesetzt.

```
1 #!/bin/bash
2
3 printf '=%.0s' {1..50}
4 printf "\n\t\tFirewall-Closed\n"
5 printf '=%.0s' {1..50}
6
7 IPTABLES="/sbin/iptables"
8 printf "\nFlushing rules..."
9 $IPTABLES -F
10 printf "\nDenying incoming connections..."
11 $IPTABLES -I INPUT -p tcp --dport 22 -j ACCEPT
12 $IPTABLES -I OUTPUT -p tcp --dport 22 -j ACCEPT
13 # Denying incoming and forwarding
14 $IPTABLES -P INPUT DROP
15 $IPTABLES -P FORWARD DROP
16 $IPTABLES -P OUTPUT DROP
17
18 # Allowing localhost
19 $IPTABLES -A INPUT -i lo -j ACCEPT
20 $IPTABLES -A OUTPUT -o lo -j ACCEPT
21
22 # Allow established sessions to receive traffic
23 $IPTABLES -A INPUT -m conntrack --ctstate \
24 ESTABLISHED,RELATED -j ACCEPT
25
26 $IPTABLES -L -v
27
28 printf '=%.0s' {1..50}
29 printf "\n\t\tProcess finished"
30
31 $IPTABLES -A INPUT -p tcp -j LOG -m limit --limit 5/m \
32 --log-prefix 'DROP IN TCP: '
33 $IPTABLES -A INPUT -p udp -j LOG -m limit --limit 5/m \
34 --log-prefix 'DROP IN UDP: '
35 $IPTABLES -A OUTPUT -p tcp -j LOG -m limit --limit 5/m \
36 --log-prefix 'DROP OUT TCP: '
37 $IPTABLES -A OUTPUT -p udp -j LOG -m limit --limit 5/m \
38 --log-prefix 'DROP OUT UDP: '
```

```

1 =====
2          Firewall-Closed
3 =====
4 Flushing rules...
5 Denying incoming connections...Chain INPUT (policy DROP 0 packets, 0
6 pkts bytes target      prot opt in      out      source      destination
7      0      0 ACCEPT      tcp   --  any      any      anywhere  anywhere
8      tcp dpt:ssh
9      0      0 ACCEPT      all   --  lo       any      anywhere  anywhere
10     0      0 ACCEPT      all   --  any      any      anywhere  anywhere
11      ctstate RELATED,ESTABLISHED
12
13 Chain FORWARD (policy DROP 0 packets, 0 bytes)
14 pkts bytes target      prot opt in      out      source      destination
15
16 Chain OUTPUT (policy DROP 0 packets, 0 bytes)
17 pkts bytes target      prot opt in      out      source      destination
18      0      0 ACCEPT      tcp   --  any      any      anywhere  anywhere
19      tcp dpt:ssh
20      0      0 ACCEPT      all   --  any      lo       anywhere  anywhere
21 =====
22          Process finished

```

## 2.2.2 Penetrationtest – Inward

Nun sind alle Ports der Opfermaschine geblockt bzw. gefiltert. Um zu überprüfen, ob das Script funktioniert hat werden nun eine Reihe an Tests abgeschlossen. Zunächst werden wieder NMAP Scans die Grundlage bilden, von der wichtige Informationen über das Ziel abgeleitet werden können.

### nmap 10.0.2.4 -p- -A -T4

Es wird nun erneut mittels eines TCP SYN Scans geprüft, welche Ports des Opfers offen sind. Zu Gunsten der Genauigkeit und der Vergleichbarkeit des Scans wurde die Konfiguration der Parameter aus dem Kapitel 2.1.2 übernommen. Ebenfalls wird erneut der Datenverkehr der Opfermaschine aufgezeichnet.

No.	Time	Source	Destination	Protocol	Length	Info
3574...	32939.282592...	10.0.2.5	10.0.2.4	TCP	62	47499 → 7787 [SYN] Seq=0 Win=1024 Len=0 MSS=1
3574...	32939.282592...	10.0.2.5	10.0.2.4	TCP	62	47498 → 62850 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.282593...	10.0.2.5	10.0.2.4	TCP	62	47498 → 65200 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.282593...	10.0.2.5	10.0.2.4	TCP	62	47498 → 15980 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.282655...	10.0.2.5	10.0.2.4	TCP	62	47498 → 37137 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.282655...	10.0.2.5	10.0.2.4	TCP	62	47498 → 24510 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425289...	10.0.2.5	10.0.2.4	TCP	62	47499 → 24510 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425289...	10.0.2.5	10.0.2.4	TCP	62	47499 → 37137 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425289...	10.0.2.5	10.0.2.4	TCP	62	47499 → 15980 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425289...	10.0.2.5	10.0.2.4	TCP	62	47499 → 65200 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425289...	10.0.2.5	10.0.2.4	TCP	62	47499 → 62850 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425289...	10.0.2.5	10.0.2.4	TCP	62	47498 → 21771 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425289...	10.0.2.5	10.0.2.4	TCP	62	47498 → 50740 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425289...	10.0.2.5	10.0.2.4	TCP	62	47498 → 48381 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425324...	10.0.2.5	10.0.2.4	TCP	62	47498 → 29526 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.425324...	10.0.2.5	10.0.2.4	TCP	62	47498 → 62859 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526541...	10.0.2.5	10.0.2.4	TCP	62	47499 → 62859 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526541...	10.0.2.5	10.0.2.4	TCP	62	47499 → 29526 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526541...	10.0.2.5	10.0.2.4	TCP	62	47499 → 48381 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526541...	10.0.2.5	10.0.2.4	TCP	62	47499 → 50740 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526541...	10.0.2.5	10.0.2.4	TCP	62	47499 → 21771 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526541...	10.0.2.5	10.0.2.4	TCP	62	47498 → 17498 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526541...	10.0.2.5	10.0.2.4	TCP	62	47498 → 28395 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526541...	10.0.2.5	10.0.2.4	TCP	62	47498 → 20250 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526711...	10.0.2.5	10.0.2.4	TCP	62	47498 → 53183 [SYN] Seq=0 Win=1024 Len=0 MSS=
3574...	32939.526711...	10.0.2.5	10.0.2.4	TCP	62	47498 → 3070 [SYN] Seq=0 Win=1024 Len=0 MSS=1
3574...	32939.627004...	10.0.2.5	10.0.2.4	TCP	62	47499 → 3070 [SYN] Seq=0 Win=1024 Len=0 MSS=1

Abbildung 2.3: Datenverkehr bei geschlossener Firewall

```

1 Nmap scan report for 10.0.2.4
2 Host is up (0.00083s latency).
3 All 65535 scanned ports on 10.0.2.4 are filtered
4 MAC Address: 08:00:27:BE:A1:CA (Oracle VirtualBox virtual NIC)
5 Too many fingerprints match this host to give specific OS details
6 Network Distance: 1 hop
7
8 TRACEROUTE
9 HOP RTT      ADDRESS
10 1    0.83 ms 10.0.2.4

```

An der Ausgabe des Scans lässt sich erkennen, dass trotz der verschärften Regelungen die MAC Adresse sowie der Hersteller des Geräts erkannt wurde. Der wichtigste Punkt ist jedoch, dass alle Ports des Opfers nun als gefiltert anstatt als geschlossen erkannt werden. Dies liegt daran, dass keine Antwort auf die SYN Pakete des Angreifers verschickt werden. Wie bereits beschrieben, interpretiert NMAP bei einem TCP SYN Scan das Fehlen einer Antwort als Filterung des angesprochenen Ports. Dies wird anhand der Abbildung 2.3 deutlich. Dort ist zu sehen, wie die Pakete der Angreifermaschiene von der Adresse 10.0.2.5 auf der Opfermaschine eintreffen. Jedoch antwortet die Opfermaschine nicht, anders als in Abbildung 2.1.

Source	Destination	Protocol	Length	Info
9773... 10.0.2.5	10.0.2.4	UDP	62	46330 → 18818 Len=0
9773... 10.0.2.5	10.0.2.4	UDP	62	46330 → 6347 Len=0
9773... 10.0.2.5	10.0.2.4	UDP	62	46330 → 8181 Len=0
9773... 10.0.2.5	10.0.2.4	UDP	62	46330 → 25462 Len=0
9773... 10.0.2.5	10.0.2.4	UDP	62	46330 → 27482 Len=0
9773... 10.0.2.5	10.0.2.4	TFTP	84	Unknown (0x30bb)
9773... 10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xb06a)
2427... 10.0.2.5	10.0.2.4	TFTP	116	Unknown (0x705a)
2427... 10.0.2.5	10.0.2.4	UDP	62	46329 → 31731 Len=0
2427... 10.0.2.5	10.0.2.4	UDP	62	46329 → 1993 Len=0
2427... 10.0.2.5	10.0.2.4	UDP	62	46329 → 21344 Len=0
2427... 10.0.2.5	10.0.2.4	UDP	62	46329 → 16430 Len=0
2427... 10.0.2.5	10.0.2.4	TFTP	84	Unknown (0x0000)
2427... 10.0.2.5	10.0.2.4	TFTP	84	Unknown (0x58bf)
2427... 10.0.2.5	10.0.2.4	UDP	62	46329 → 5010 Len=0
2475... 10.0.2.5	10.0.2.4	UDP	62	46329 → 21609 Len=0
2475... 10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xf04e)
3600... 10.0.2.5	10.0.2.4	TFTP	84	Unknown (0x20cc)
3600... 10.0.2.5	10.0.2.4	UDP	62	46330 → 21609 Len=0
3600... 10.0.2.5	10.0.2.4	UDP	62	46330 → 5010 Len=0
3600... 10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xa0be)
3600... 10.0.2.5	10.0.2.4	TFTP	84	Unknown (0x70ad)
3794... 10.0.2.5	10.0.2.4	UDP	62	46330 → 16430 Len=0
3794... 10.0.2.5	10.0.2.4	UDP	62	46330 → 21344 Len=0
3794... 10.0.2.5	10.0.2.4	UDP	62	46330 → 1993 Len=0
3794... 10.0.2.5	10.0.2.4	UDP	62	46330 → 31731 Len=0
3959... 10.0.2.5	10.0.2.4	TFTP	116	Unknown (0x705a)

Abbildung 2.4: Datenverkehr bei geschlossener Firewall – UDP

**nmap 10.0.2.4 -sU -T4**

Um zu Prüfen, welchen Einfluss das Script auf UDP Pakete nimmt, wird als nächstes ein UDP Scan durchgeführt. Wie auch beim vorherigen Scan, bleiben die Parameter gleich. Die Erwartung an dieses Scanergebnis ist, dass alle Ports der Opfermaschine als offen | gefiltert angezeigt werden. Zudem wird erwartet, dass dieser Scan wesentlich mehr Zeit in Anspruch nimmt, als der Scan aus dem Kapitel 2.1.2. Diese Erwartungshaltung bildet sich aus der Art und Weise, wie der UDP Scan funktioniert. Da die Ports des Opfers durch das Bash-Script geschlossen sein sollten, sollten keine Antwortpakete an den Angreifer geschickt werden. Somit müsste der Angreifer auf den Timeout von NMAP warten, welcher zur gleichen Zeit die Ports als offen bzw. gefiltert interpretiert.

Output:

```

1 Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-14 14:16 EST
2 Nmap scan report for 10.0.2.4
3 Host is up (0.0026s latency).
4 All 1000 scanned ports on 10.0.2.4 are open|filtered
5 MAC Address: 08:00:27:BE:A1:CA (Oracle VirtualBox virtual NIC)

```

Der Output bestätigt die Hypothese. Da der Angreifer keine Antworten erhalten hat, kann keine genaue Aussage über den Status der Ports getroffen werden.

### **2.2.3 Penetrationtest – Outward**

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum fermentum semper mollis. Ut porta odio in ultrices fermentum. Aenean id pulvinar nisi. Phasellus porttitor condimentum suscipit. Nulla facilisi. Mauris in malesuada arcu. Morbi tempus bibendum tempor.*

*Vivamus sit amet tellus pellentesque, commodo est vitae, interdum ante. In eu vestibulum dolor, in venenatis felis. Ut risus odio, convallis in aliquet quis, luctus quis sem. Vivamus malesuada ex eget vehicula finibus. Mauris iaculis finibus diam vel ultrices. Morbi luctus nulla euismod sodales posuere. Integer dignissim gravida orci, at porttitor nibh varius ac. Donec vel efficitur nulla. In fermentum mauris elit, a efficitur elit accumsan id. Interdum et malesuada fames ac ante ipsum primis in faucibus.*

*Proin lacinia malesuada leo sit amet bibendum. Integer dictum ultrices enim id condimentum. Ut egestas vulputate quam. Pellentesque bibendum nec turpis ac ultricies. Sed at ligula nec urna bibendum convallis consequat at turpis. Sed id dolor eros. Nullam maximus turpis in massa ornare accumsan. Sed id dapibus eros, tincidunt ullamcorper risus. In tincidunt, nunc at suscipit placerat, nulla justo imperdiet est, id convallis libero elit a nulla. Vivamus velit odio, convallis sed dignissim vitae, pharetra eu tellus. Sed eu dignissim felis, ac rhoncus est. Vivamus ac aliquam urna. Pellentesque lectus risus, porta eu diam in, volutpat pharetra justo. Donec elit diam, maximus vitae congue et, condimentum a magna.*

## 2.3 Standard Konfiguration

```
1 #!/bin/bash
2
3 printf '=%.0s' {1..50}
4 printf "\n\t\tFirewall-Standard\n"
5 printf '=%.0s' {1..50}
6
7 IPTABLES="/sbin/iptables"
8
9 printf "\nFlushing rules..."
10 $IPTABLES -F
11
12 # Denying incoming and forwarding
13 $IPTABLES -P INPUT DROP
14 $IPTABLES -P FORWARD DROP
15 $IPTABLES -P OUTPUT DROP
16
17 printf "\nDenying incoming connections..."
18 $IPTABLES -I INPUT -p tcp --dport 22 -j ACCEPT
19 $IPTABLES -A OUTPUT -p tcp -m tcp -m multiport \
20 --dports 20,22,53,80,443 -j ACCEPT
21 $IPTABLES -A OUTPUT -p udp --dport 53 -j ACCEPT
22
23 # Allowing localhost
24 $IPTABLES -A INPUT -i lo -j ACCEPT
25 $IPTABLES -A OUTPUT -o lo -j ACCEPT
26
27 # Allow established sessions to receive traffic
28 $IPTABLES -A INPUT -m conntrack --ctstate \
29 ESTABLISHED,RELATED -j ACCEPT
30 $IPTABLES -A OUTPUT -m conntrack --ctstate \
31 ESTABLISHED,RELATED -j ACCEPT
32
33 $IPTABLES -L -v
34
35 printf '=%.0s' {1..50}
36 printf "\n\t\tProcess finished"
```

### 2.3.1 Script

Das Script zur Erstellung einer Firewall, wie sie möglicherweise in Unternehmen vorkommen könnte, ähnelt dem aus Kapitel 2.2.1 sehr. Es besteht aus den gleichen sechs Teilen, obwohl das Logging hier auf Grund der Länge des Scripts vernachlässigt wird. Zunächst werden alle bestehenden Regeln gelöscht – gefluscht – um die Firewall auf ihre Basiskonfiguration zurückzusetzen. Dies verhindert mögliche Konflikte mit bestehenden Regeln im weiteren Verlauf.

Darauf folgend werden alle eingehenden, weiterleitenden und ausgehenden Verbindungen blockiert, bestehende Verbindungen sowie Antworten auf Etablierte werden jedoch weiterhin zugelassen. Abschließend werden die Ausnahmeregelungen behandelt, welche den Hauptunterschied zu der vorherigen Konfiguration darstellen.

Hier werden die Ports für SSH sowohl eingehend als auch ausgehend zugelassen. Zusätzlich werden die Ports 22, 53, 80 und 443 ausgehend zugelassen, hauptsächlich um Verbindung zum Internet herstellen zu können. Zudem sind dem Port 53 UDP Verbindungen gestattet, da das DNS Protokoll hauptsächlich über UDP funktioniert.

### 2.3.2 Penetrationstest – Inward

**nmap 10.0.2.4 -p- -A -T4**

Wie auch bei den vorherigen Konfigurationen, wird hier ebenfalls zunächst ein TCP SYN Scan durchgeführt. Da in dieser Firewall-Konfiguration keine fundamentalen Veränderungen durchgeführt wurden, werden den Vorgängern ähnliche Ergebnisse erwartet. Es sollte der Angreifermaschine nur ein Port ersichtlich sein, da alle Anderen nur von innen nach außen funktionieren.

Output:

```
1 Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-15 15:04 EST
2 Nmap scan report for 10.0.2.4
3 Host is up (0.00066s latency).
4 Not shown: 65534 filtered ports
5 PORT      STATE      SERVICE      VERSION
6 22/tcp     closed     ssh
7 MAC Address: 08:00:27:BE:A1:CA (Oracle VirtualBox virtual NIC)
8 Too many fingerprints match this host to give specific OS details
9 Network Distance: 1 hop
10
11 TRACEROUTE
12 HOP      RTT      ADDRESS
13 1        0.66 ms   10.0.2.4
```

No.	Time	Source	Destination	Protocol	Length	Info
1235...	37200.585877...	10.0.2.5	10.0.2.4	TCP	62	39285 → 64628 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.585878...	10.0.2.5	10.0.2.4	TCP	62	39285 → 15177 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586070...	10.0.2.5	10.0.2.4	TCP	62	39284 → 58340 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586076...	10.0.2.5	10.0.2.4	TCP	62	39284 → 19772 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586070...	10.0.2.5	10.0.2.4	TCP	62	39284 → 1810 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586070...	10.0.2.5	10.0.2.4	TCP	62	39284 → 8464 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586070...	10.0.2.5	10.0.2.4	TCP	62	39284 → 58474 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586078...	10.0.2.5	10.0.2.4	TCP	62	39284 → 6629 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586070...	10.0.2.5	10.0.2.4	TCP	62	39284 → 42128 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586152...	10.0.2.5	10.0.2.4	TCP	62	39284 → 3401 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586152...	10.0.2.5	10.0.2.4	TCP	62	39284 → 41853 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586152...	10.0.2.5	10.0.2.4	TCP	62	39284 → 57846 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586193...	10.0.2.5	10.0.2.4	TCP	62	39284 → 44662 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586193...	10.0.2.5	10.0.2.4	TCP	62	39284 → 15875 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586193...	10.0.2.5	10.0.2.4	TCP	62	39284 → 11581 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586193...	10.0.2.5	10.0.2.4	TCP	62	39284 → 49421 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586193...	10.0.2.5	10.0.2.4	TCP	62	39284 → 54481 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586322...	10.0.2.5	10.0.2.4	TCP	62	39284 → 48620 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586322...	10.0.2.5	10.0.2.4	TCP	62	39284 → 25642 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586322...	10.0.2.5	10.0.2.4	TCP	62	39284 → 26912 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586322...	10.0.2.5	10.0.2.4	TCP	62	39284 → 36998 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586322...	10.0.2.5	10.0.2.4	TCP	62	39284 → 50393 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.586322...	10.0.2.5	10.0.2.4	TCP	62	39284 → 49317 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.588928...	10.0.2.5	10.0.2.4	TCP	62	39284 → 27612 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.588928...	10.0.2.5	10.0.2.4	TCP	62	39284 → 30914 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.588928...	10.0.2.5	10.0.2.4	TCP	62	39284 → 34941 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.588928...	10.0.2.5	10.0.2.4	TCP	62	39284 → 43943 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.588928...	10.0.2.5	10.0.2.4	TCP	62	39284 → 2771 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.588928...	10.0.2.5	10.0.2.4	TCP	62	39284 → 51248 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.588928...	10.0.2.5	10.0.2.4	TCP	62	39284 → 5913 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1235...	37200.588929...	10.0.2.5	10.0.2.4	TCP	62	39284 → 63618 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Abbildung 2.5: Datenverkehr der Opfermaschine

Der Output dieses Scans ist identisch mit dem aus Kapitel 2.2.2, mit der Ausnahme des Ports 22. Ein maßgeblicher Unterschied wird sich erst bei einem ausgehenden Test erkennen lassen.

### nmap 10.0.2.4 -sU -T4

Im Script wurde der Port 53, welcher für das DNS Protokoll verwendet wird, für ausgehende UDP Pakete geöffnet. Diese Ausnahmeregelung ist die Einzige, die UDP betrifft. Da die Ausnahme nur für ausgehende Pakete gilt, ist es zu erwarten, dass ähnlich wie in Kapitel 2.2.2 alle Ports als “open | filtered” gezeigt werden.

Output:

```

1 Starting Nmap 7.91 ( https://nmap.org ) at 2022-01-07 14:13 EST
2 Nmap scan report for 10.0.2.4
3 Host is up (0.0017s latency).
4 All 1000 scanned ports on 10.0.2.4 are open|filtered
5 MAC Address: 08:00:27:BE:A1:CA (Oracle VirtualBox virtual NIC)

```

### 2.3.3 Penetrationstest – Outward

Als nächstes wird getestet, welchen Einfluss

No.	Time	Source	Destination	Protocol	Length	Info
1176...	84.759761928	10.0.2.5	10.0.2.4	ICMP	164	Echo (ping) request id=0x13f8, seq=29
1177...	84.798223894	10.0.2.5	10.0.2.4	ICMP	194	Echo (ping) request id=0x13f9, seq=29
1177...	84.834298326	10.0.2.5	10.0.2.4	UDP	344	40675 → 31183 Len=300
1177...	84.871566274	10.0.2.5	10.0.2.4	TCP	76	[TCP Port numbers reused] 40721 → 22 [
1177...	84.871629878	10.0.2.4	10.0.2.5	TCP	56	22 → 40721 [RST, ACK] Seq=1 Ack=1 Win=
1177...	84.9088718536	10.0.2.5	10.0.2.4	TCP	76	[TCP ACKed unseen segment] [TCP Previous
1177...	84.908870991	10.0.2.4	10.0.2.5	TCP	56	22 → 40722 [RST] Seq=822747082 Win=0 L
1177...	84.934396625	10.0.2.5	10.0.2.4	TCP	76	[TCP Previous segment not captured] 40
1177...	84.961358541	10.0.2.5	10.0.2.4	ICMP	164	Echo (ping) request id=0x13f8, seq=29
1177...	85.000536399	10.0.2.5	10.0.2.4	ICMP	194	Echo (ping) request id=0x13f9, seq=29
1177...	85.027513876	10.0.2.5	10.0.2.4	UDP	344	40675 → 31183 Len=300
1177...	85.063105803	10.0.2.5	10.0.2.4	TCP	76	[TCP Retransmission] 40723 → 22 [FIN,
1177...	85.088309258	10.0.2.5	10.0.2.4	ICMP	164	Echo (ping) request id=0x13f8, seq=29
1177...	85.118465523	10.0.2.5	10.0.2.4	ICMP	194	Echo (ping) request id=0x13f9, seq=29
1177...	85.162241532	10.0.2.5	10.0.2.4	UDP	344	40675 → 31183 Len=300
1177...	85.188201825	10.0.2.5	10.0.2.4	TCP	76	[TCP Retransmission] 40723 → 22 [FIN,
1177...	85.213551715	10.0.2.5	10.0.2.4	ICMP	164	Echo (ping) request id=0x13f8, seq=29
1177...	85.254763867	10.0.2.5	10.0.2.4	ICMP	194	Echo (ping) request id=0x13f9, seq=29
1177...	85.2806568201	10.0.2.5	10.0.2.4	UDP	344	40675 → 31183 Len=300
1177...	85.311731752	10.0.2.5	10.0.2.4	TCP	76	[TCP Retransmission] 40723 → 22 [FIN,
1177...	120.326115378	0.0.0.0	255.255.255.255	DHCP	326	DHCP Request - Transaction ID 0x2844a
1177...	120.337303879	10.0.2.3	10.0.2.4	DHCP	592	DHCP ACK - Transaction ID 0x2844a

Abbildung 2.6: Fort.: Datenverkehr der Opfermaschine

No.	Time	Source	Destination	Protocol	Length	Info
1191...	2907.6654273...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xb0c4)
1191...	2907.6654273...	10.0.2.5	10.0.2.4	UDP	62	43854 → 19374 Len=0
1191...	2907.7524344...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0x90bd)
1191...	2907.7524345...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0x20c1)
1191...	2907.7524346...	10.0.2.5	10.0.2.4	UDP	62	43855 → 19482 Len=0
1191...	2907.7524346...	10.0.2.5	10.0.2.4	UDP	62	43855 → 6004 Len=0
1191...	2907.7524346...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xb0c4)
1191...	2907.7524347...	10.0.2.5	10.0.2.4	TFTP	116	Unknown (0x706a)
1191...	2907.7657275...	10.0.2.5	10.0.2.4	UDP	62	43855 → 19374 Len=0
1191...	2907.7657276...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xfc00)
1191...	2907.7657276...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xe0ba)
1191...	2907.7657277...	10.0.2.5	10.0.2.4	UDP	62	43855 → 19503 Len=0
1191...	2907.8628476...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xf0be)
1191...	2907.8628477...	10.0.2.5	10.0.2.4	TFTP	116	Unknown (0x706a)
1191...	2907.8628477...	10.0.2.5	10.0.2.4	UDP	62	43854 → 16938 Len=0
1192...	2907.8628477...	10.0.2.5	10.0.2.4	UDP	62	43854 → 68 Len=0
1192...	2907.8628478...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xe0ba)
1192...	2907.8628478...	10.0.2.5	10.0.2.4	UDP	62	43854 → 19075 Len=0
1192...	2907.8826537...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0x30c0)
1192...	2907.8826538...	10.0.2.5	10.0.2.4	TFTP	84	Unknown (0xc0d2)
1192...	2907.8826539...	10.0.2.5	10.0.2.4	UDP	62	43854 → 22695 Len=0
1192...	2907.8826539...	10.0.2.5	10.0.2.4	UDP	62	43854 → 17018 Len=0
1192...	2907.9706285...	10.0.2.5	10.0.2.4	UDP	62	43855 → 19075 Len=0

Abbildung 2.7: Datenverkehr der Opfermaschine – UDP

# 3 Projektphase 2 – Webserver

## 3.1 Setup

### 3.1.1 Datenbank

Für das Projekt wurde PostgreSQL (auch einfach als Postgres bezeichnet) als Datenbank gewählt. So wurde über Docker eine Container-Instanz einer Postgres Datenbank gestartet, die auf einem Port auf der Hostmaschine auf Nachrichten wartet und über die Adresse localhost:5432 erreichbar ist.

Die Datenbank enthält ein Schema "blog", welches die beiden Tabellen – User und Post – beinhaltet. Diese beiden Tabellen besitzen jeweils eine inkrementierende ID als Primärschlüssel.

Die Tabelle User enthält dabei zusätzlich den Benutzernamen und das Passwort des Nutzers – wobei beide bewusst im Klartext gespeichert werden. Die Tabelle Post enthält neben der ID noch die Attribute Title und Content, welche bei der Ausgabe der Posts angezeigt werden. Als Vorbereitung für die Demonstration beider Attacken wurden bereits Beispiel-Posts und User angelegt, um ein anschaulicheres Ergebnis zu erhalten.

user		post	
<b>id</b>	int	<b>id</b>	int
username	varchar	title	varchar
password	varchar	content	text

Abbildung 3.1: Database Setup

### 3.1.2 API

Zur Demonstration der beiden Attacken werden zwei APIs genutzt. Beide werden mit Hilfe des Express.js Frameworks implementiert.

Die erste – und umfangreichere – API stellt die Schnittstelle zur Datenbank der Blog-Applikation dar. Hier werden die typischen CRUD Operationen durchgeführt und ans Frontend angebunden. Zudem wird die Session-Erstellung hier geregelt.

Zu Demonstrationszwecken wurden bewusst Sicherheitsfunktionen in dieser API umgangen. Es wurde zum Beispiel darauf verzichtet, das HTTPOnly Attribut des Cookies zu setzen, was den Zugriff auf den Cookie aus einem Skript heraus verhindern würde. Zudem wird der Cookie im Klartext verschickt, anstatt verschlüsselt zu werden, was ebenfalls die potentielle Sicherheit der Demonstrationsapplikation untergräbt.

Die andere API – im Weiteren als bösartige API bezeichnet – stellt den Angreifer dar. Sie verfügt nur über eine einzige Methode, über welche der Session-Cookie des Opfers abgefangen wird.

### 3.1.3 Frontend

Das Frontend bildet einen Weblog ab, welcher das Posten von Textbeiträgen ermöglicht. Im Eingabeformular werden vom Besucher Titel und Inhalt eingetragen, bestätigt ("SSubmit") und über die API an das Backend übermittelt, wo sie in der Datenbank gespeichert werden.

Das Frontend wurde mit Vue.js entwickelt und bietet aus diesem Grund einige vordefinierte und standardmäßig aktivierte Sicherheitsfunktionen, die eine Cross-Site-Scripting (XSS) Attacke verhindern sollen. Um diese Sicherheitsmechanismen zu umgehen, werden die Inhalte aus der Datenbank mit Hilfe des v-HTML Attributs in das Frontend eingebunden, anstatt mit der üblichen "Mustache" Notation oder dem v-text Attribut. Diese füllen das innerText Attribut der DOM-Node mit dem gewünschten Inhalt, werden jedoch automatisch "escaped", was im Sinne der Sicherheit ist, jedoch gerade die hier beabsichtigte Simulation eines XSS-Angriffes verhindert. .

## 3.2 Cross-Site-Scripting

Unter dem Begriff XSS versteht man einen Typ der Injektionsattacken, bei dem schadhafter Code – oft in Form eines browserseitigen Scripts – bei einem Endnutzer ausgeführt wird. Diese werden unter anderem durch mangelhafte bzw. fehlende Überprüfung von Nutzereingaben ermöglicht.

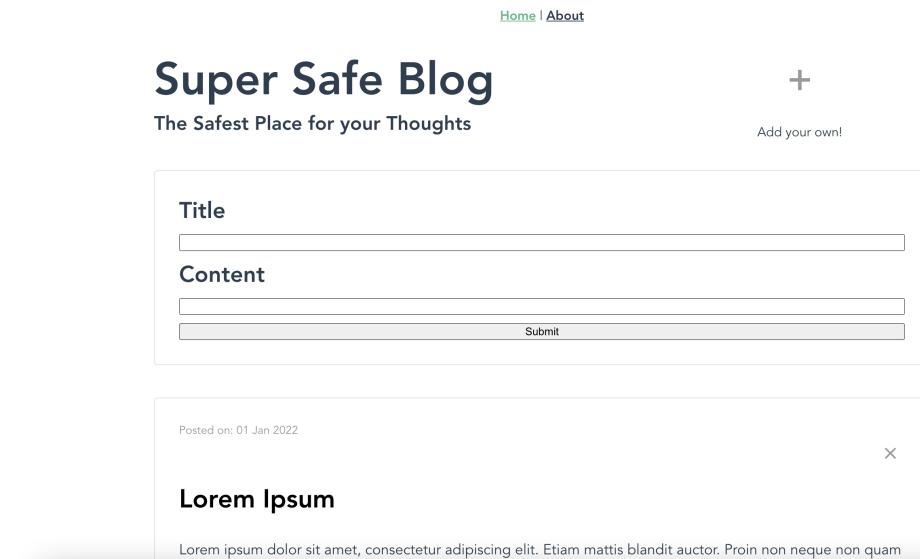


Abbildung 3.2: Blog-Frontend

Um dieses zu demonstrieren, wurde bei diesem Projekt bewusst darauf verzichtet, Nutzereingaben zu überprüfen. Das Ziel der Demonstration ist es den Session-Cookie zu stehlen. Um dies zu erreichen wird ein neuer Blog-Post erstellt, bei dem an Stelle eines "einfachen" Textes als Inhalt ein schadhaftes HTML <img> Element eingegeben wird. Da es sich bei diesem Element letztlich nur um Text handelt und die Eingabe nicht überprüft wird, wird der Wert in der Datenbank gespeichert. Beim erneuten Aufrufen der Seite versucht der Browser, das <img> Element wie jedes andere auch als Bild zu rendern, erhält jedoch nur einen Fehler, da die im src-Attribut als Bildquelle angegebene URL bewusst fehlerhaft ist.

Nun ist es möglich mittels des HTML Attributs "onerror" der Applikation vorzuschreiben, wie mit dem Fehler umzugehen ist. Dies wird mittels validem JavaScript Code geschafft.

```

1  window.alert('XSS'))"
5 >
```

Im onerror Attribut wird nun zunächst der Session-Cookie in der Variable cookies gespeichert und dann an die Adresse des Angreifers über die bösartige API übermittelt. Sobald der Vorgang abgeschlossen ist wird im Browser-Fenster des Opfers über ein Alert angezeigt, dass das Script funktioniert hat. Abschließend ist der Session-Cookie auf der Console des bösartigen Servers zu finden.

Der Angreifer wäre nun in der Lage diesen Session-Cookie zu nutzen, um sich auf der

betroffenen Webseite als “Opfer” auszugeben. Demnach könnte der Angreifer die digitale Identität des Opfers stehlen. Somit hätte er Zugriff auf die Kontoeinstellungen des Opfers oder er könnte Bestellungen im Namen und mit den Zahlungsmitteln des Opfers tätigen. Große Online-Shops, wie zum Beispiel Amazon, versuchen solche (und ähnliche) Probleme zu verhindern, indem sie vor der Transaktion eine erneute Authentifizierung fordern. So braucht der Angreifer zusätzlich noch Passwort und Benutzername des Opfers, wenn nicht sogar MFA eingerichtet ist.

### 3.3 SQL Injection

Als zweite Demonstration wird eine SQL Injection durchgeführt. SQL Injections sind eine Art der Injektionsattacken, bei der durch Nutzereingaben valide SQL Queries in die betroffene Applikation injiziert werden. Die OWASP Foundation schreibt dazu:

A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.  
[1]

Das Ziel dieser Demonstration ist es, durch die Eingabe einer bestimmten Zeichenkette einen fremden Post gezielt zu löschen. Dafür wird ein neuer Post eröffnet, bei welchem der Titel frei gewählt, im Content jedoch der folgende String eingegeben wird:

```
1 ') ; DELETE FROM blog.post WHERE title = 'test' ;
```

Der String funktioniert, indem zunächst die ursprüngliche Query escaped wird und darauf folgend eine weitere bösartige Query. Standardmäßig nutzt die API eine Query, welche mit der Schreibweise “VALUES ('\${Title}', '\${Content}')” endet, um neue Posts in der Datenbank anzulegen. Wenn nun in der Variable Content die Zeichen '); in Reihenfolge auftreten, werden diese von der Datenbank als Abschluss der Query interpretiert. Daraufhin wird dann eine weitere bösartige Query angehängt, welche von der Datenbank aufgegriffen und verarbeitet wird. Diese Query gibt der Datenbank die Anweisung alle Posts zu löschen, bei denen der Titel “test” lautet. In einer Produktionsumgebung, wie zum Beispiel die eines Online-Shops, könnte ein derartiger unauthorisierter Eingriff erhebliche Folgen mitschließen. So könnte ein bösartiger Akteur sämtliche Inhalte der Datenbank löschen oder publizieren. Ähnlich wie die XSS Attacken können SQL Injections verhindert werden, indem Nutzereingaben escaped werden.

# Literaturverzeichnis

- [1] *SQL Injection*. URL: [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection).