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, 29. Mai 2024

Inhaltsverzeichnis

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 der Informationsverbund sowie die Konfiguration der einzelnen Komponenten beschrieben.

1.1 Skripts und Anwendungen

Alle genutzten Skripts und 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` 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:

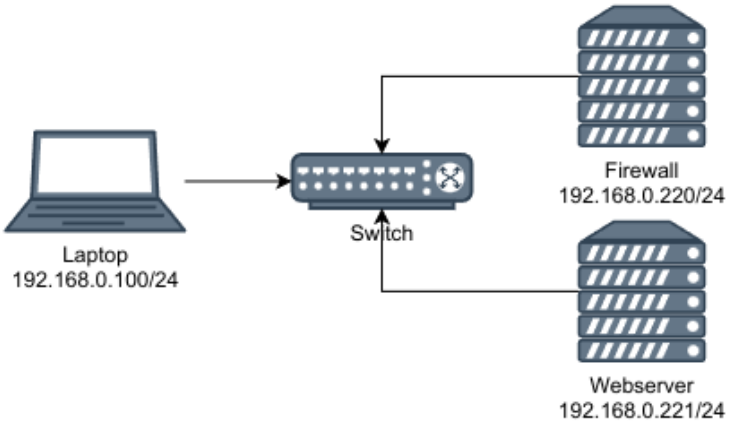

Abbildung 1: Aufbau des Informationsverbunds

Der Laptop dient als Client, um auf den Webserver zuzugreifen und als Client für Pentests.
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, ein NETGEAR ProSAFE GS105GE mit 5 RJ45 Ports, verbindet alle Geräte im Informationsverbund. Zu den gezeigten Komponenten können noch zusätzlich 2 weitere Clients angeschlossen werden.

2.2 Informationsfluss

Folgendes Diagramm stellt den Informationsfluss im Verbund dar:

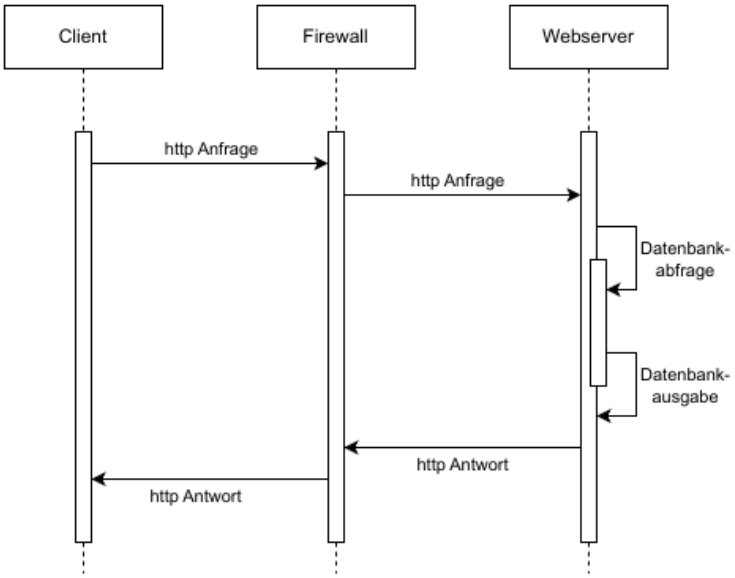

Abbildung 2: Aufbau des Informationsverbunds

Beginnend beim Client wird eine Anfrage an die Firewall gesendet, diese leitete die Anfrage, mittels einer Forwading-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.

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 Konfiguration der Server

Folgende Abschnitte beschreiben die Konfiguration des Firewall-Servers und des Webservers.

3.1 Firewall-Server

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
nvim	Texteditor zum bearbeiten von Konfigurationsdateien
zsh	Shell mit besserer Auto-Complete-Funktion
tmux	Terminal Multiplexer zum einfachen Verwalten über SSH
suricata	Intrusion Detection System

Tabelle 1: Zusätzlich installiere Software auf dem Firewall-Server

Der standardmäßig aktivierte nftables-Dienst wurde deaktiviert um Konflikte mit iptables zu verhindern, siehe Firewall-Server 4.1 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 Konfiguration des Intrusion Detection Systems 5 für die Konfiguration des IDS.

Dem Server wurde die statische IP-Adresse `192.168.0.220` zugewiesen, siehe Setzen einer statischen IP-Adresse 7.3.

3.2 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 2: Zusätzlich installiere Software auf dem Webserver

Der standardmäßig aktivierte nftables-Dienst wurde deaktiviert um Konflikte mit iptables zu verhinden, siehe Webserver 4.2 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 Hinzufügen eines Cron-Jobs 7.2.

Dem Server wurde die statische IP-Adresse `192.168.0.221` zugewiesen, siehe Setzen einer statischen IP-Adresse 7.3.

3.2.1 Python-Anwendung

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

Diese bietet ein einfaches HTML Login Form, 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.

3.2.2 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.

3.2.3 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 Konfiguration der Firewall

Folgende Abschnitte zeigen und erläutern die Konfiguration der einzelnen Firewalls.

4.1 Firewall-Server

Im folgenden werden die Anforderungen und die Konfiguration von iptables auf dem Firewall-Server erläutert.

4.1.1 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

Tabelle 3: Benötigte Ports auf dem Firewall-Server

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

Port 22 wird für SSH genutzt um den Server remote 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, um vom Server aus Internetseiten aufzurufen und weitere Pakete zu installieren.
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.

4.1.2 Skripte

Nach den Vorgaben aus Aufgabenblatt 1 und den identifizierten Anforderungen ergeben sich folgende Skripte. Alle Skripte müssen mit Root Rechten ausgeführt werden.
Iptables Regeln wurden gemäß der iptables man page [3] angelegt.

Bei allen folgenden Skripts 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
# this script blocks all incoming and outgoing traffic
# flush all rules and reset all chains
iptables -F
iptables -t nat -F
iptables -X

# block all incoming, outgoing and forwarded traffic
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

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

```
#!/usr/bin/env bash
# this script allows all incoming and outgoing traffic
# flush all rules and reset all chains
iptables -F
iptables -t nat -F
iptables -X

# allow all incoming, outgoing and forwarded traffic
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD 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 Hinzufügen eines Cron-Jobs 7.2.

```
#!/usr/bin/env bash
# script for setting up production settings
# flush all rules and reset all chains
iptables -F
iptables -t nat -F
iptables -X

# allow all local traffic, needed for local connection like databases
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

```
# allow ssh traffic on port 22
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

# allow dns traffic on port 53, both tcp and udp
iptables -A INPUT -p tcp --dport 53 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 53 -j ACCEPT
iptables -A INPUT -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -p udp --sport 53 -j ACCEPT

# allow ntp traffic on port 123
iptables -A INPUT -p udp --dport 123 -j ACCEPT
iptables -A OUTPUT -p udp --sport 123 -j ACCEPT

# allow pinging
iptables -A INPUT -p icmp -j  ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT

# allow http on port 80
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

# allow https on port 443
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT

# forward incoming traffic on port 80, http, to webserver port 80 and
   back
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-
   destination 192.168.0.221:80
iptables -A FORWARD -p tcp -d 192.168.0.221 --dport 80 -j ACCEPT
iptables -t nat -A POSTROUTING -p tcp -d 192.168.0.221 --dport 80 -j
   MASQUERADE

# allow outlook traffic to external mailserver
## imap
iptables -A INPUT -p tcp --dport 143 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 143 -j ACCEPT
iptables -A INPUT -p tcp --dport 993 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 993 -j ACCEPT
## pop3
iptables -A INPUT -p tcp --dport 110 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 110 -j ACCEPT
iptables -A INPUT -p tcp --dport 995 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 995 -j ACCEPT
## smtp
iptables -A INPUT -p tcp --dport 587 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 587 -j ACCEPT

# allow traffic for teamviewer
iptables -A INPUT -p tcp --dport 5938 -j ACCEPT
```

```
iptables -A OUTPUT -p tcp --sport 5938 -j ACCEPT

iptables -A INPUT -p udp --dport 5938 -j ACCEPT
iptables -A OUTPUT -p udp --sport 5938 -j ACCEPT

# drop  traffic that doesnt match incoming or outgoing rules, allow all
    forwarding
iptables -P INPUT DROP
iptables -P FORWARD ACCEPT # forward has to be ACCEPT, to allow
   forwarding to webserver
iptables -P OUTPUT DROP
```

4.2 Webserver

Im folgenden werden die Anforderungen der Firewall des Webservers beschrieben und der Skript der diese umsetzt gezeigt.

4.2.1 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 4: Benötigte Ports auf dem Webserver

4.2.2 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 [3] angelegt.
Der Skript wird direkt am Boot mittels eines Cron-Jobs ausgeführt, siehe Hinzufügen eines Cron-Jobs 7.2.

```
#!/usr/bin/env bash
# script for setting up production settings
# flush all rules and reset all chains
iptables -F
iptables -t nat -F
iptables -X

# allow all local traffic, needed for local connection like databases
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

```
# allow ssh traffic on port 22
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

# allow pinging
iptables -A INPUT -p icmp -j  ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT

# allow incoming http traffic from firewall and allow all outgoing http
    traffic
iptables -A INPUT -p tcp --src 192.168.0.220 --dport 80 --jump ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

# drop  traffic that doesnt match incoming or forwarding rules, allow
   all outgoing
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

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.

5 Konfiguration des Intrusion Detection Systems

Es wurde sich für das Intrusion Detection System (IDS) Suricata entschieden, da bereits Vorerfahrung bestanden.

Bei der Installation wurde der Anleitung des Herstellers [4] 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 heruntergeladen, installiert und aktiviert [5], mittels `sudo suricata-update`.
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, das jemand remote den `id` Befehl als root ausgeführt hat, was bedeuten würde das 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.

6 Pentest und Bewertung der Sicherheit

Folgende Kapitel befassen sich mit der Bewertung der Sicherheit und den Ergebnissen der Pentests.

6.1 Bewertung der Firewallregeln

Die gesetzten Regeln der Firewall werden als ”gut” bewertet. Ausschlaggebend für diese Bewertung sind folgende Punkte:

1. + Es werden nur die absolut notwendigen eingehenden Verbindungen zugelassen.
2. + Es werden nur die absolut notwendigen ausgehenden Verbindungen zugelassen.
3. - Es gibt keine Regelung welche Protokolle für die Authentifizierung mit dem Exchange- oder E-Mail-Server zulässig sind. Es können hier also mehr Ports offen sein als nötig.
4. - Pings von und zur Firewall sind zulässig, die Firewall kann dadurch schneller bei Netzwerkscans identifiziert werden.
5. + Es wird nur der HTTP-Verkehr an den Webserver weitergeleitet.
6. + Der Webserver akzeptiert HTTP-Verkehr nur von der Firewall.
7. - Port 22 kann von jeder IP-Adresse angefragt werden.
8. - Verkehr für Port 22 wird zugelassen, es besteht kein Limit für Versuche eine Verbindung aufzubauen.

6.2 SQL Injection

6.3 Cross-Site Scripting

6.4 Datenverkehr verschleiern

6.5 MITM

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," 2 2023.

[2] TeamViewer, "Ports used by TeamViewer," 4 2024.

[3] H. Eychenne, "iptables(8) - linux man page."

[4] "2. Quickstart guide — Suricata 8.0.0-dev documentation."

[5] "9.1. Rule Management with Suricata-Update — Suricata 8.0.0-dev documentation."