

---

# **Praktikum Rechnernetze**

Protokoll zu Versuch 5 (Paketfilter-Firewall unter Linux)  
von Gruppe 1

Jakob Waibel, Daniel Hiller, Elia Wüstner, Felix Pojtinger

2021-11-16

## Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>3</b>
1.1	Mitwirken . . . . .	3
1.2	Lizenz . . . . .	3
<b>2</b>	<b>Wordpress Konfigurieren</b>	<b>4</b>
<b>3</b>	<b>Portscan durchführen</b>	<b>7</b>
<b>4</b>	<b>Blockieren von Services</b>	<b>8</b>
<b>5</b>	<b>Whitelist-Ansatz per Shell-Skript</b>	<b>10</b>
<b>6</b>	<b>ICMP und Prometheus Node-Exporter</b>	<b>11</b>
<b>7</b>	<b>Besprechung, Musterlösung und Einbindung als System-Service</b>	<b>13</b>

# 1 Einführung

## 1.1 Mitwirken

Diese Materialien basieren auf [Professor Kiefers “Praktikum Rechnernetze”-Vorlesung der HdM Stuttgart](#).

**Sie haben einen Fehler gefunden oder haben einen Verbesserungsvorschlag?** Bitte eröffnen Sie ein Issue auf GitHub ([github.com/pojntfx/uni-netpractice-notes](https://github.com/pojntfx/uni-netpractice-notes)):



**Abbildung 1:** QR-Code zum Quelltext auf GitHub

Wenn Ihnen die Materialien gefallen, würden wir uns über einen GitHub-Stern sehr freuen.

## 1.2 Lizenz

Dieses Dokument und der enthaltene Quelltext ist freie Kultur bzw. freie Software.



**Abbildung 2:** Badge der AGPL-3.0-Lizenz

Uni Network Practice Notes (c) 2021 Jakob Waibel, Daniel Hiller, Elia Wüstner, Felix Pojtinger

SPDX-License-Identifier: AGPL-3.0

## 2 Wordpress Konfigurieren

**Auf ihrem Server ist Wordpress vorinstalliert / vorkonfiguriert. Lediglich die abschließende Einrichtung ist noch nicht erfolgt... Führen Sie die Einrichtung durch und stellen Sie die Funktion sicher. Rufen Sie dazu die IP der Servers in einem Web-Browser auf.**

Zur Fertigstellung der Konfiguration muss zuerst folgender Dialog ausgefüllt werden:

The image shows the WordPress installation configuration screen. At the top center is the WordPress logo. Below it, the heading "Welcome" is followed by a brief introduction. The main section is titled "Information needed" and contains several input fields: "Site Title" with the value "Ubumut", "Username" with the value "Ubongo", "Password" with a strong password "6w^7apD^4!SaXntNploM" and a "Hide" button, "Your Email" with the value "jakwai01@gmail.com", and a checkbox for "Search engine visibility" which is currently unchecked. A note at the bottom says "Double-check your email address before continuing." At the very bottom is a button labeled "Install WordPress".

**Abbildung 3:** Wordpress-Einrichtungsbildschirm

Im darauffolgenden Dashboard kann im [Pages](#) Reiter eine neue page erstellt werden.



Abbildung 4: Wordpress-Dashboard

Diese Page kann nun mit arbiträrem Inhalt gefüllt werden.



Abbildung 5: Wordpress-Post

Der Resultierende Post kann nun im Web gefunden werden. Eine Eingabe der IP führt auf eine kleine Übersicht mit diesem Post und einem weiteren “Hello-World” Post.



**Abbildung 6:** Wordpress-Home

### 3 Portscan durchführen

**Überprüfen Sie mit einem Portscanner welche Ports an Ihrem Server öffentlich erreichbar sind. Welche Ports/Services sind das? Müssen diese Services öffentlich erreichbar sein?**

Zur Sicherheit starten wir bevor wir mit dem Portscanning beginnen den VPN unseres Vertrauens.



**Abbildung 7:** VPN-Einrichtung

`nmap` zeigt die offenen Ports:

```
1 $ sudo nmap 65.21.244.249
2 Starting Nmap 7.91 ( <https://nmap.org> ) at 2021-11-16 14:59 CET
3 sendto in send*ip*packet_sd: sendto(4, packet, 44, 0, 65.21.244.249,
  16) => Operation not permitted
4 Offending packet: TCP 10.108.48.108:61668 > 65.21.244.249:53 S ttl=55
  id=22537 iplen=44 seq=1930045695 win=1024 <mss 1460>
5 sendto in send*ip*packet_sd: sendto(4, packet, 44, 0, 65.21.244.249,
  16) => Operation not permitted
```

```

6 Offending packet: TCP 10.108.48.108:61669 > 65.21.244.249:53 S ttl=50
  id=21891 iplen=44 seq=1930111230 win=1024 <mss 1460>
7 Nmap scan report for static.249.244.21.65.clients.your-server.de
  (65.21.244.249)
8 Host is up (0.035s latency).
9 Not shown: 990 closed ports
10 PORT      STATE      SERVICE
11 22/tcp    open      ssh
12 25/tcp    filtered  smtp
13 53/tcp    filtered  domain
14 80/tcp    open      http
15 139/tcp   filtered  netbios-ssn
16 445/tcp   filtered  microsoft-ds
17 1900/tcp  filtered  upnp
18 2869/tcp  filtered  icslap
19 3306/tcp  open      mysql
20 9100/tcp  open      jetdirect

```

Ein Auszug des Wireshark-Captures zeigt hier zum Beispiel die Ergebnisse des Port-Scans.

No.	Time	Source	Destination	Protocol	Length	Info
...	192.9...	65.21.244...	31.59.230...	TCP	54	443 → 22073 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	192.9...	5.53.40.10	65.21.244...	TCP	74	55090 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1300 SACK_PERM=1 TSval=105319215 TSecr=0 WS=256
...	192.9...	65.21.244...	5.53.40.10	TCP	54	443 → 55090 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	192.9...	5.53.40.10	65.21.244...	TCP	74	55088 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1300 SACK_PERM=1 TSval=105319215 TSecr=0 WS=256
...	192.9...	65.21.244...	5.53.40.10	TCP	54	443 → 55088 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	192.9...	5.53.40.10	65.21.244...	TCP	74	55092 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1300 SACK_PERM=1 TSval=105319215 TSecr=0 WS=256
...	192.9...	65.21.244...	5.53.40.10	TCP	54	443 → 55092 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	193.0...	2.185.91...	65.21.244...	TCP	74	43226 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1360 SACK_PERM=1 TSval=235063706 TSecr=0 WS=512
...	193.0...	65.21.244...	2.185.91...	TCP	54	443 → 43226 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	193.0...	31.59.230...	65.21.244...	TCP	74	22074 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1400 SACK_PERM=1 TSval=2774291790 TSecr=0 WS=256
...	193.0...	65.21.244...	31.59.230...	TCP	54	443 → 22074 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	193.0...	31.59.230...	65.21.244...	TCP	74	22075 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1400 SACK_PERM=1 TSval=2774291890 TSecr=0 WS=256
...	193.0...	65.21.244...	31.59.230...	TCP	54	443 → 22075 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	193.1...	31.59.230...	65.21.244...	TCP	74	22076 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1400 SACK_PERM=1 TSval=2774291961 TSecr=0 WS=256
...	193.1...	65.21.244...	31.59.230...	TCP	54	443 → 22076 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	193.1...	141.62.31...	65.21.244...	TCP	66	[TCP Keep-Alive] 30606 → 80 [ACK] Seq=2067 Ack=903340 Win=424192 Len=0 TSval=2353742750 TSecr=3...
...	193.1...	65.21.244...	141.62.31...	TCP	66	[TCP Keep-Alive ACK] 80 → 30606 [ACK] Seq=903340 Ack=2068 Win=35456 Len=0 TSval=308221 TSecr=23...
...	193.1...	31.59.230...	65.21.244...	TCP	74	22077 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1400 SACK_PERM=1 TSval=2774292015 TSecr=0 WS=256
...	193.1...	65.21.244...	31.59.230...	TCP	54	443 → 22077 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	193.1...	31.59.230...	65.21.244...	TCP	74	22078 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1400 SACK_PERM=1 TSval=2774292025 TSecr=0 WS=256
...	193.1...	65.21.244...	31.59.230...	TCP	54	443 → 22078 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
...	193.1...	2.185.91...	65.21.244...	TCP	74	43228 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1360 SACK_PERM=1 TSval=235063722 TSecr=0 WS=512

Abbildung 8: Wireshark-Capture von nmap

Da der Aufgabenstellung zu entnehmen ist, dass wir im Moment nur einen Webserver betreiben, können wir getrost allen externen Zugriff auf alle Ports bis auf Port 80 und Port 22 für den SSH-Zugriff blockieren. Für spätere Aufgabenstellungen können die Regeln dann angepasst werden.

## 4 Blockieren von Services

Sie haben in Aufgabe 2 mindestens einen Service identifiziert, der nicht öffentlich verfügbar sein muss. Blockieren Sie den externen Zugriff auf diesen Service in Ihrer Firewall (Blacklist-Ansatz).



Der “Blacklist-Ansatz” bedeutet, dass mit einer ACCEPT policy und negativen Regeln gearbeitet wird, sodass alles erlaubt ist, sofern es nicht durch eine Regel explizit verboten wird.

Blocken aller Ports neben 22 und 80:

```

1 $ sudo iptables -F INPUT
2 $ sudo iptables -A INPUT -p tcp --dport 25 -j DROP
3 $ sudo iptables -A INPUT -p tcp --dport 53 -j DROP
4 $ sudo iptables -A INPUT -p tcp --dport 139 -j DROP
5 $ sudo iptables -A INPUT -p tcp --dport 445 -j DROP
6 $ sudo iptables -A INPUT -p tcp --dport 1900 -j DROP
7 $ sudo iptables -A INPUT -p tcp --dport 2869 -j DROP
8 $ sudo iptables -A INPUT -p tcp --dport 3306 -j DROP
9 $ sudo iptables -A INPUT -p tcp --dport 9100 -j DROP
10 $ sudo iptables -L INPUT
11 Chain INPUT (policy ACCEPT)
12 target      prot opt source                destination          tcp dpt
13 DROP        tcp  --  anywhere              anywhere             tcp dpt
14             :9100
15 DROP        tcp  --  anywhere              anywhere             tcp dpt:
16             smtp
17 DROP        tcp  --  anywhere              anywhere             tcp dpt:
18             domain
19 DROP        tcp  --  anywhere              anywhere             tcp dpt:
20             netbios-ssn
21 DROP        tcp  --  anywhere              anywhere             tcp dpt:
22             microsoft-ds
23 DROP        tcp  --  anywhere              anywhere             tcp dpt
24             :1900
25 DROP        tcp  --  anywhere              anywhere             tcp dpt
26             :2869
27 DROP        tcp  --  anywhere              anywhere             tcp dpt:
28             mysql

```

Check auf dem lokalen System:

Es kann erkannt werden, dass nun alle blockierten Ports als “filtered” angezeigt werden.

```

1 $ nmap 65.21.244.249
2 Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-16 15:23 CET
3 Nmap scan report for static.249.244.21.65.clients.your-server.de
4 (65.21.244.249)
5 Host is up (0.079s latency).
6 Not shown: 991 closed ports
7 PORT      STATE      SERVICE
8 22/tcp    open      ssh
9 25/tcp    filtered  smtp
10 80/tcp    open      http
11 139/tcp   filtered  netbios-ssn
12 445/tcp   filtered  microsoft-ds
13 1900/tcp  filtered  upnp

```

```

13 2869/tcp filtered icslap
14 3306/tcp filtered mysql
15 9100/tcp filtered jetdirect
16
17 Nmap done: 1 IP address (1 host up) scanned in 3.39 seconds

```

## 5 Whitelist-Ansatz per Shell-Skript

**Stellen Sie den gleichen Zustand der Firewall (Damit meine ich, dass die gleichen Services erreichbar sind) her wie in Aufgabe 3, allerdings verfolgen Sie jetzt den Whitelist-Ansatz.**

Der “Whitelist-Ansatz” bedeutet, dass die default policy DROP verwendet wird und dass alles verboten ist, was nicht explizit durch eine Regel erlaubt wurde.

Inhalt von `iptables-rules.sh`:

```

1 # $HOME/iptables-rules.sh
2
3 #!/usr/bin/env bash
4
5 sudo iptables -F INPUT
6
7 sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
8 sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

```

Zur Sicherheit wurde hier noch gecheckt, ob auch wirklich die richtigen Regeln angewandt wurden:

```

1 $ sudo iptables -L
2 Chain INPUT (policy ACCEPT)
3 target    prot opt source                destination            tcp dpt:
4 ACCEPT    tcp  --  anywhere              anywhere               tcp dpt:
    ssh
5 ACCEPT    tcp  --  anywhere              anywhere               tcp dpt:
    http

```

Nun wurde noch die Default Deny Rule aktiviert:

```

1 $ sudo iptables -P INPUT DROP

```

Und die korrekte Anwendung sichergestellt:

```

1 $ sudo iptables -L INPUT
2 Chain INPUT (policy DROP)
3 target    prot opt source                destination            tcp dpt:
4 ACCEPT    tcp  --  anywhere              anywhere               tcp dpt:
    ssh
5 ACCEPT    tcp  --  anywhere              anywhere               tcp dpt:
    http

```

nmap zeigt nun auch die Ports nicht mehr als `filtered` an:

```
1 $ nmap 65.21.244.249
2 Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-16 15:48 CET
3 Nmap scan report for static.249.244.21.65.clients.your-server.de
  (65.21.244.249)
4 Host is up (0.053s latency).
5 Not shown: 997 filtered ports
6 PORT      STATE SERVICE
7 22/tcp    open  ssh
8 53/tcp    closed domain
9 80/tcp    open  http
10
11 Nmap done: 1 IP address (1 host up) scanned in 11.56 seconds
```

## 6 ICMP und Prometheus Node-Exporter

**Der Prometheus Node-Exporter liefert Metriken für Prometheus (<https://prometheus.io/>). Konfigurieren Sie ihre Firewall so, dass diese Metriken nur von Ihren IP-Adressen aus erreichbar sind (Nutzen Sie <https://ifconfig.co/> um Ihre öffentliche IP-Adresse in Erfahrung zu bringen). Das selbe gilt für ICMP. Die Angriffsvektoren für ICMP sind zwar ziemlich eingeschränkt, trotzdem reicht es, wenn Sie in der Lage sind Probes an den Server zu senden.**

Unsere IP-Adresse:

```
1 $ curl https://ifconfig.io
2 193.27.14.134
```

Nun müssen zwei `ACCEPT`-Rules erstellt werden; zuerst für den Prometheus Node-Exporter:

```
1 $ sudo iptables -A INPUT -p tcp --dport 9100 -j ACCEPT -s 193.27.14.134
2 $ curl http://65.21.244.249:9100
3 <html>
4     <head><title>Node Exporter</title></head>
5     <body>
6     <h1>Node Exporter</h1>
7     <p><a href="/metrics">Metrics</a></p>
8     </body>
9     </html>
10 $ nmap 65.21.244.249
11 Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-16 15:54 CET
12 Nmap scan report for static.249.244.21.65.clients.your-server.de
  (65.21.244.249)
13 Host is up (0.039s latency).
14 Not shown: 996 filtered ports
```

```
15  PORT      STATE  SERVICE
16  22/tcp    open   ssh
17  53/tcp    closed domain
18  80/tcp    open   http
19  9100/tcp  open   jetdirect
20
21  Nmap done: 1 IP address (1 host up) scanned in 5.74 seconds
```

Wie zu sehen ist, ist der Port von dieser öffentlichen IP zu erreichen. Von einem anderen Host ist dies nicht der Fall:

```
1  $ curl https://ifconfig.io
2  141.62.31.100
3  $ nmap 65.21.244.249
4  Starting Nmap 7.80 ( https://nmap.org ) at 2021-11-16 15:55 CET
5  Nmap scan report for static.249.244.21.65.clients.your-server.de
   (65.21.244.249)
6  Host is up (0.024s latency).
7  Not shown: 996 filtered ports
8  PORT      STATE  SERVICE
9  22/tcp    open   ssh
10 25/tcp    closed smtp
11 80/tcp    open   http
12 445/tcp   closed microsoft-ds
13
14  Nmap done: 1 IP address (1 host up) scanned in 5.18 seconds
```

Nun die Rule für ICMP:

```
1  $ sudo iptables -A INPUT -p icmp -j ACCEPT -s 193.27.14.134
```

Nun lässt sich von unserem System aus der Host anpingen:

```
1  $ ping 65.21.244.249
2  PING 65.21.244.249 (65.21.244.249) 56(84) bytes of data.
3  64 bytes from 65.21.244.249: icmp_seq=1 ttl=52 time=51.3 ms
4  64 bytes from 65.21.244.249: icmp_seq=2 ttl=52 time=51.5 ms
5  64 bytes from 65.21.244.249: icmp_seq=3 ttl=52 time=52.5 ms
6  64 bytes from 65.21.244.249: icmp_seq=4 ttl=52 time=29.2 ms
7  64 bytes from 65.21.244.249: icmp_seq=5 ttl=52 time=32.4 ms
8  ^C
9  --- 65.21.244.249 ping statistics ---
10 5 packets transmitted, 5 received, 0% packet loss, time 4007ms
11 rtt min/avg/max/mdev = 29.212/43.383/52.477/10.323 ms
```

Von einer anderen Workstation (193.27.14.134) aus ist dies nicht der Fall:

```
1  $ ping 65.21.244.249
2  PING 65.21.244.249 (65.21.244.249) 56(84) bytes of data.
3  ^C
4  --- 65.21.244.249 ping statistics ---
```

```
5 9 packets transmitted, 0 received, 100% packet loss, time 8226ms
```

Diese beiden Regeln wurden noch zu `$HOME/iptables-rules.sh` hinzugefügt, damit sie im folgenden auch nach dem Reboot persistiert werden. Ebenso wurde die Default-Deny-Rule in das Skript mit aufgenommen.

## 7 Besprechung, Musterlösung und Einbindung als System-Service

**Diese Aufgabe führen wir zusammen durch, dokumentieren Sie trotzdem die Schritte und Ergebnisse!**

```
1 $ systemctl cat iptables
2 # /etc/systemd/system/iptables.service
3 [Unit]
4 Description=firewall service
5 Before=network.target
6 AssertPathExists=/root/iptables-rules.sh
7
8 [Service]
9 Type=oneshot
10 RemainAfterExit=yes
11 ExecStart=/root/iptables-rules.sh
12 StandardOutput=syslog
13 StandardError=syslog
14
15 [Install]
16 WantedBy=multi-user.target
17 $ sudo systemctl enable --now iptables.service
18 $ sudo systemctl reboot
```

Wie zu sehen ist, wurden die Regeln auch nach dem Reboot angewandt:

```
1 $ sudo iptables -L INPUT
2 Chain INPUT (policy DROP)
3 target      prot opt source                destination
4 ACCEPT      tcp  --  anywhere               anywhere      tcp dpt:
5             ssh
5 ACCEPT      tcp  --  anywhere               anywhere      tcp dpt:
6             http
6 ACCEPT      icmp --  193.27.14.134          anywhere
7 ACCEPT      tcp  --  193.27.14.134          anywhere      tcp dpt
              :9100
```

Auf einer anderen Workstation:

```
1 $ nmap 65.21.244.249
2 Starting Nmap 7.80 ( https://nmap.org ) at 2021-11-16 16:14 CET
```

```
3 Nmap scan report for static.249.244.21.65.clients.your-server.de
  (65.21.244.249)
4 Host is up (0.049s latency).
5 Not shown: 996 filtered ports
6 PORT STATE SERVICE
7 22/tcp open  ssh
8 25/tcp closed smtp
9 80/tcp open  http
10 445/tcp closed microsoft-ds
11
12 Nmap done: 1 IP address (1 host up) scanned in 10.74 seconds
```

Auf der allowlisteten Workstation:

```
1 $ nmap 65.21.244.249
2 Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-16 16:14 CET
3 Nmap scan report for static.249.244.21.65.clients.your-server.de
  (65.21.244.249)
4 Host is up (0.036s latency).
5 Not shown: 996 filtered ports
6 PORT STATE SERVICE
7 22/tcp open  ssh
8 53/tcp closed domain
9 80/tcp open  http
10 9100/tcp open jetdirect
11
12 Nmap done: 1 IP address (1 host up) scanned in 5.43 seconds
```

Letztendlich sieht unser überarbeitetes Skript wie folgt aus:

```
1 #!/usr/bin/env bash
2
3 # exit on error
4 set -e
5
6 IPT="/sbin/iptables"
7
8 $IPT -F INPUT
9
10 $IPT -A INPUT -p tcp --dport 22 -j ACCEPT
11 $IPT -A INPUT -p tcp --dport 80 -j ACCEPT
12 $IPT -A INPUT -p icmp -j ACCEPT -s 193.27.14.134
13 $IPT -A INPUT -p tcp --dport 9100 -j ACCEPT -s 193.27.14.134
14
15 # default targets
16 $IPT -P INPUT DROP # we want to block all incoming traffic (whitelist-
  approach)
17 $IPT -P OUTPUT ACCEPT # we trust the installed software
18 $IPT -P FORWARD DROP # we don't want to forward traffic at all
```