# Labreport #5

Patrick Eickhoff, Alexander Timmermann

## Aufgabe 1: Netzwerkeinstellungen

**2.**
- Client-VM

  **IP-Adresse:** `192.168.254.44`

  **Gateway:** `192.168.254.2`

  **Nameserver:** `10.1.1.1`
- Router-VM

  **IP-Adresse eth0:** `172.16.137.222`

  **IP-Adresse eth1:** `192.168.254.2`
- Server-VM

  **IP-Adresse:** `172.16.137.144`

## Aufgabe 2: Absichern eines Einzelplatzrechners mit iptables

**1.** Auf der Client-VM sind keine iptables-Regeln vorhanden, die man löschen könnte. Zum Löschen könnte man sonst folgende Befehle benutzen:

```
1 sudo iptables -F              # flush chains in 'filter' table
2 sudo iptables -t nat -F       # flush chains in 'nat' table
3 sudo iptables -t mangle -F    # flush chains in 'mangle' table
4 sudo ipables -X               # delete custom chains
```

**2.** Um das Surfen auf Webseiten zu erlauben, müssen wir den Datenverkehr über die Ports 80 (HTTP), 443 (HTTPS) und 53 (DNS) der RouterVM erlauben:

```
1 iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
2 iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
3 iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT
4
5 iptables -A INPUT -p udp --dport 53 -j ACCEPT
6 iptables -A INPUT -p tcp --dport 80 -j ACCEPT
7 iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Desweiteren wollen wir sowohl als ICMP-Nachrichten senden und empfangen, als auch SSH-Verbindungen (Port 22) aufbauen können:

```
iptables -A INPUT -p icmp-j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
```

Letzendlich wollen wir jeglichen anderen Traffic unterbinden:

```
iptables -A INPUT -j REJECT
iptables -A OUTPUT -j REJECT
```

3. Die SSH-Verbidung von der CLientVM auf die RouterVM (`ssh user@192.168.254.2`) wird verweigert ("refused"), während die Verbindung von RouterVM auf ClientVM (`ssh user@192.168.254.1`) problemlos funktioniert.
   Per `nc -l 5555` setzen wir einen Server auf der CLientVM auf. Wenn wir diesen jedoch von der RouterVm mit `nc 192.168.254.2 5555` ansprechen wollen, wird die Verbindung verweigert ("refused").
   Wenn wir statt REJECT DROP für unsere Firewall verwenden, bekommen wir bei einem Verbindungsversuch keine Refused-Nachricht mehr zurück. Da die Firewall das Packet einfach ignoriert.

4. Mithilfe dynamischer Regeln können wir einfach definieren, dass ein- und ausgehende Packete, die zu bereits etablierten Verbindungen gehören (ESTABLISHED,RELATED), automatisch akzeptiert werden:

   ```
   iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
   iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
   ```

   Die restlichen Regeln definieren sich dann wie folgt:

   ```
   iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
   iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
   iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT

   iptables -A INPUT -p tcp --dport 22 -j ACCEPT
   iptables -A INPUT -p icmp-j ACCEPT
   iptables -A INPUT -j REJECT
   ```

   Dynamische Regeln sind sehr angenehm, da sie erlauben Packete abhängig von ihrem Zustand zu behandeln. So werden deutlich weniger Regeln benötigt, um die Kommunikation bereits aufgebauter Verbindungen zu erlauben.

## Aufgabe 3: Absichern eines Netzwerks

1. Folgenden Befehl haben wir in der *rc.local* auf der RouterVM gefunden:
   `iptables -t nat -A POSTROUTING -o eth0 -s 192.168.254.0/24 -j MASQUERADE`

Die *NAT-Tabelle* gibt die Firewall-Regeln an, für Packete, die neue Verbindungen aufbauen wollen. Die Postrouting-Chain behandelt Packets nachdem sie geroutet wurden. Der Befehl maskiert, bildet die Source-IP, der Packete, die aus dem internen Netzwerk von Ip-Adressen 192.168.254.0/24 gesendet und über das Interface eth0 geroutet wurden, auf eine interne IP-Adresse ab.

2. Die ClietVM kann die ServerVM anpingen, die ServerVM jedoch nicht die ClientVM. Da die ClientVM nur eine IP-Adresse für ein lokales Netzwerk hat, kann die ServerVM höchstens die RouterVM erreichen. Die Antwort auf den Ping der ClientVM erreicht diese, da die RouterVM per Masquerading die IP-Adresse der Client intern mapped.

3. Die minimale Firewallkonfiguration sollte wie folgt aussehen:

```
iptables -A FORWARD -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -p tcp -d 172.16.134.144 --dport 80 -j DROP
iptables -A FORWARD -p tcp -d 10.1.1.2 -j DROP
iptables -A FORWARD -p tcp -d 10.0.0.0/8 -j DROP
iptables -A FORWARD -i eth1 -o eth0 -p udp -m udp --dport 53 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp -m tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp -m tcp --dport 443 -j ACCEPT
iptables -A FORWARD -j DROP
```

4. Damit eine SSH-Verbindung aufgebaut werden kann, müssen wir Forwarding zu Port 22 erlauben: `iptables -I FORWARD 2 -i eth1 -o eth0 -dport 22 -j ACCEPT`

5. Zum forwarden der SSH-Verbindung an die ClientVM brauchen wir nur eine Regel:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 5022 -j DNAT
         --to-destination 192.168.254.44:22
```

   Die Packete die an Port 5022 des äußeren Interfaces eth0 eingehen, werden vorm Routing zur ClientVM (192.168.254.44) auf Port 22 weitergeleitet.

6. Nachdem wir eth0 eine weitere IP-Adresse 172.16.137.223 eingerichtet haben, geben wir an, dass alle Packete, die an diese IP-Adresse geschickt werden, zur ClientVM weitergeleitet werden:

```
iptables -t nat -A PREROUTING -i eth0 -d 172.16.137.223 -j DNAT
         --to-destination 192.168.254.44
```

## Aufgabe 4: SSH-Tunnel

1. Als erstes konfigurieren wir die Firewall:

```
iptables -A FORWARD -m contrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -j DROP
```

2. Ein SSH-Tunnel lässt sich einfach über den ssh-Befehl aufbauen:
`ssh -L 9000:172.16.137.144:80 user@172.16.137.144`
Mit obigem Befehl binden wir Port 80 der ServerVM an Port 9000 der ClientVM per SSH-Verbindung. Dann lässt sich über *localhost:9000* auf den Webserver der ServerVM zugreifen. Über Wireshark sehen wir nur die SSH-Packete.
(Quelle http://blog.trackets.com/2014/05/17/ssh-tunnel-local-and-remote-port-forwarding-explained-with-examples.html)

3. Local-Forwarding zum Browsen wäre sehr aufwendig, da wir jeden Port, den wir zum Surfen brauchen, einzeln geforwarded werden muss. Statdessen benutzen wir Dynamic-Forwarding, um unsere SSH-Verbindung als SOCKS-Proxy zu verwenden:
`ssh -C -D 1080 user@172.16.137.223`
Im Browser müssen wir nun den Socks Proxy konfigurieren: SOCKS-Host=127.0.0.1 und SOCKS-Port 1080.
(Quelle https://help.ubuntu.com/community/SSH/OpenSSH/PortForwarding)

4. Zuerst starten wir auf der ClientVM einen Server mittels Netcat `nc -L 5555` und bauen dann unseren SHH-Tunnel zum Remote-Forwarding auf `ssh -R 9000:localhost:5555 user@172.16.137.144`. Nun können wir uns auf der ServerVM mittels `nc localhost:5555` verbinden.

## Aufgabe 5: OpenVPN

1. Filterregeln:

```
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j DROP
```

2. Der Server und Client sind wie in der *client.conf* (App. A) und *server.conf* (App. B) angegeben, konfiguriert.

3. Die Firewall erlaubt Packete die Port 1194 für OpenVPN ansteuern und Packete die vom Interface tun0, welches von der VPN genutzt wird, eingehen. Ausgehen dürfen nur Packete die zu einer bereits etablierten VPN-Verbindung gehören:

```
iptables -A INPUT -i tun0 -j ACCEPT
iptables -A INPUT -p tcp --dport 1194 -j ACCEPT
iptables -A INPUT -j DROP
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -j DROP
```

4. Wir starten erst VPN-Client und -Server: `openvpn client.conf` und `openvpn server.conf`. Auf der ClientVM sehen wir dann bei erfolgreicher Verbindung:
`Peer Connection Initiated with [AF_INET]172.16.137.222:54286`
`Initialization Sequence Completed`.

Auf der ServerVM respektive:
`Peer Connection Initiated with [AF_INET]172.16.137:1194`
`Initialization Sequence Complete`

5. Für den Tunnel der VPN wird das Interface tun0 verwendet.
Der Webserver der ServerVM lässt sich einfach im Browser über *10.8.0.1:80* aufrufen. Der Aufbau einer SSH-Verbindung ist nicht erlaubt, da dies kein Antwortpacket der VPN ist und somit im Output geblockt wird.

## Aufgabe 6: HTTP-Tunnel

1. Firewall:

```
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p udp --dport 53 -j ACCEPT
```

2. Wir überlisten die Firewall einfach, indem wir Packete die bei der ServerVM auf Port 80 eingehen an Port 22 Weiterleiten:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 22
```

3. Firewall:

```
iptables -A INPUT -i eth1 -p tcp --dport 3128 -j ACCEPT
iptables -A INPUT -i eth1 -j DROP
iptables -A FORWARD -j DROP
iptables -A OUTPUT -o eth0 -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -o eth0 -j DROP
```

4. In Firefox müssen wir als HTTP-Proxy nur *192.168.254.2 3128* einstellen.

5. Als erstes verbinden wir uns über Netcat mit dem SSH-Server der ServerVM:
`nc -x192.168.254.2:3128 -Xconnect 172.16.134.144 80`
Danach konfigurieren wir unseren ssh-Befehl mittels Corkscrew in der */.ssh/config*:

```
Host *
 ProxyCommand corkscrew 192.168.254.2 3128 %h %p
```

Jetzt können wir einfach einen HTTP-Tunnel via SSH aufbauen:
`ssh -p 80 user@172.16.134.144`

6. Nachdem wir httptunnel installiert haben, müssen wir nur mit `hts -F localhost 80` den Server auf der ServerVM öffnen und den Client mittels `htc -P 192.168.254.2:3128 -F 8888 172.16.134.144 80` auf der ClientVM starten. Danach können wir einen HTTP-Tunnel von der ClientVM zur ServerVM aufbauen: `ssh -p 8888 user@localhost`

# Anhang A Server-Config

Listing 1: server.conf

```
port 1194
dev tun
ifconfig 10.8.0.1 10.8.0.2
secret static.key
keepalive 10 120
;status openvpn-status.log
verb 3
```

# Anhang B Client-Config

Listing 2: client.conf

```
dev tun
remote 172.16.137.144 1194
;resolve-retry infinite
;nobind
ifconfig 10.8.0.2 10.8.0.1
secret static.key
verb 3
```