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:

```
sudo iptables -F &\Comment{\# flush chains in 'filter' table}&
sudo iptables -t nat -F &\Comment{\# flush chains in 'nat' table}&
sudo iptables -t mangle -F &\Comment{\# flush chains in 'mangle' table}&
sudo ipables -X &\Comment{\# delete custom chains}&
```

Mit

```
sudo apt-get update
sudo apt-get install openssh-server
```

installieren wir den OpenSSH Server.

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:

```
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 udp --dport 53 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
 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.

1 Absichern eines Netzwerks

1.1

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.

1.2

- Ein Ping von der ClientVM an die ServerVM ist problemlos möglich.
- Das Pingen der ClientVM von der ServerVM funktioniert nicht.
- Die ServerVm von der ClientVM anzupingen funktioniert problemlos, da die Packete über die RouterVM per Masquerading weitergeleitet werden. Wenn dann die ICMP-Antwort an der RouterVM ankommt, wird die Destination-IP auf die zugehörige IP im internen Netzwerk abgebildet. Die ClientVM können wir jedoch nicht von der ServerVM direkt ansprechen, da die das Netzwerkinterface der ClientVM nur mit dem Netz unter der RouterVM verbunden ist. Da wir das interne Mapping der RouterVM nicht kennen, können wir auch nicht die gemappte IP-Adresse der ClientVM ansprechen.

1.3

Die minimale Firewallkonfiguration sollte wie folgt aussehen:

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

1.4

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`

1.5

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

```
1 iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 5022 -j DNAT
2           --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.

1.6

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:

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

2 SSH-Tunnel

2.1

Als erstes konfigurieren wir die Firewall:

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

2.2

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)

2.3

- 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.