SVS Bachelor-Projekt Network Security

Blatt 5: Beschreibung der Experimentierumgebung

Louis Kobras
6658699

Utz Pöhlmann
6663579

1 Netzwerkeinstellungen

1.2

ClientVM:

IP-Adresse (`ifconfig -a`): 192.168.254.44	**Standard-Gateway** (`route -n`): 192.168.254.2	**DNS-Nameserver** (`nslookup ubuntu.com`): 10.1.1.1

RouterVM:

eth0	eth1
172.16.137.222	192.168.254.2

ServerVM:
IP-Adresse der Server-VM: 172.16.137.144

2 Absichern eines Einzelplatzrechners mit iptables (ClientVM)

2.1

Anzeigen der Firewall-Regeln mit `sudo iptables -L`; alle Regeln löschen mit `sudo iptables -F`[1]; OpenSSH-Server nach Paketquellen-Update via apt-get installiert (automatisch gestartet).

2.2

Regelwerk siehe [2.2: ClientVM-Filterregeln (S. 5)].
`iptables` säubern mit `sudo iptables -F`, eingeben der Regeln aus dem Anhang als Root.

2.3

- SSH-Verbindungsversuch von RouterVM mit `sudo ssh user@192.168.254.44` erfolgreich
- SSH-Verbindungsversuch in die andere Richtung nicht erfolgreich (Connection refused)
- hosten eines Servers mit `netcat -l 5555` erfolgreich, Verbindung (`sudo netcat 192.168.254.44 5555`) erwartungsgemäß fehlgeschlagen
- Firefox ist bei DROP schneller als bei REJECT. Bei REJECT erhält der Remote Host die Ablehnung als Antwort, während bei DROP das Paket nur verworfen wird, ohne dem Remote mitzuteilen, dass die Verbindung verweigert wird.

2.4

Dynamische Regeln vgl. [2.4: ClientVM Stateful Filtering (S. 5)].
Man muss nicht jeden Port und jedes Protokoll einzelnd abdecken. Stateful Filter sind effizienter, da sie sich nur die Paket-Header ansehen.

[1] löscht alle Regeln nacheinander

3 Absichern eines Netzwerks (RouterVM)

3.1

Der Aufruf bedeutet (nach [1]): “Maskiere alles, was an eth0 ausgeht”.
Es wird die Adressumsetzung (NAT) aktiviert und die Schnittstelle markiert ([2]).
Source: 192.168.254.0; Maske: 24

3.2

Die Client-VM kann die Server-VM anpingen; umgekehrt geht dies nicht.
Vermutung: Die Client-VM ist von außen nicht direkt ansprechbar, da sie hinter der RouterVM versteckt ist.

3.3

Regelsatz im Anhang unter [3.3: Filterregeln (S. 5)]

3.4

Folgender Eintrag in der iptable *filter an Stelle [0] öffnet den SSH-Tunnel einseitig: `FORWARD -p tcp -syn -dport 22 -destination 172.16.137.144 -m conntrack -cstate NEW -j ACCEPT`

3.5

Folgende Regeln sollte die Aufgabe erfüllen:

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

Die erste Zeile erledigt die eigentliche Weiterleitung, während die zweite Zeile die Firewall für die umgeleiteten Pakete öffnet. Zusätzlich muss der öffentliche Port mithilfe von netcat geöffnet werden: `nc -l 5022`.

3.6

- Zuweisen der IP 172.16.137.42 mit `ifconfig eth0:1 172.16.137.42 netmask 255.255.255.0`
- PREROUTING-Regel: `-A PREROUTING -i eth0 -j DNAT --to 192.168.254.44`
- FORWARD-Regel: `-A FORWARD -d 192.168.254.44 -j ACCEPT`
- Login von der Server-VM mit `ssh user@172.16.137.42 -L 2000:172.16.137.42:22`
- Testweise Client-VM von der Server-VM aus neugestartet

4 SSH-Tunnel

4.1

iptables-Regeln vgl. [4.1: SSH-Ausgang (S. 5)].

4.2

Tunnelerzeugung mit `ssh user@172.16.137.42 -L 2000:172.16.137.42:80`
Beobachtung mit Wireshark ergibt TCP-Pakete, die an SSH weitergeleitet werden. Auslesen ist nicht möglich, Gesprächspartner stimmen überein.

4.3

Es ist erforderlich, den Zielserver zu kennen, sowie lokal einen Port zu öffnen und einen freien Port auf dem Server zu wissen.
Als Alternative bietet sich Dynamic Forwarding an (ssh-Aufruf um `-D` erweitern) [3].
Dem Browser muss mitgeteilt werden, einen Proxy zu verwenden (HOWTO: [4]).

4.4

Aufbauen einer Reverse-Verbindung von der Client-VM zur Server-VM: `ssh -R 5555:localhost:22 user@172.16.137.144`
Rücktunneln von der Server-VM: `ssh localhost -p 5555` [5]

5 OpenVPN

5.1

Konfiguration vgl. [5.1 (S. 6)]

5.2

Key wird erzeugt durch `openvpn -genkey -secret static.key`. Key liegt dann in $pwd/static.key in ASCII.
server.conf:

```
dev tun
ifconfig 192.168.1.2 192.168.1.1
secret static.key
```

5.3

Konfiguration vgl. [5.3 (S. 6)]. **Verbindung schlägt fehl.**

5.4

client.conf:

```
dev tun
remote 172.16.137.144
ifconfig 192.168.1.2 192.168.1.1
secret static.key
```

5.5

OpenVPN auf der ServerVM mit Link auf die Server-Config starten (tunneling interface):
`openvpn --config servervm.conf --dev tun`
OpenVPN auf der ClientVM mit Link auf die Client-Config starten (tunneling interface):
`openvpn --config clientvm.conf --dev tun`
Output im syslog in folgender Form:

```
<<Timestamp>> <<Version>> <<Meta-Infos>> <<Build>>
<<Timestamp>> <<Response>>
```

5.6

Server und Client benutzen das Interface `tun0`. Eine SSH-Verbindung konnte aufgebaut werden.

6 HTTP-Tunnel

6.1

Konfiguration vgl. [6.1 (S. 6)].

6.2

Umgehen der Einschränkung durch Umleiten von SSH auf HTTP:
HTTP-Port freimachen (Daemon ausschalten): `daemon-start-stop -K httpd /etc/ssh/ssh_config`: Port auf 80 setzen
SSH-Service neu starten: `service ssh restart`
Verbindung testen: `ssh user@172.16.137.144 -p 80`

6.3

Konfiguration vgl. [6.3 (S. 6)].

6.4

Datei `/etc/squid/squid.conf` wurde angepasst, Proxy-Einstellungen im Browser wurden entsprechend vorgenommen (Preferences > Advanced > Network > Connection > Settings > Manual proxy configuration).

6.5

Es sind wiederholt Probleme aufgetreten, weswegen die Zeit an dieser Stelle nicht mehr gereicht hat.

6.6

s.o.

Literatur

[1] `www.netfilter.org/documentation/HOWTO/de/NAT-HOWTO-6.html`

[2] `https://wiki.ubuntuusers.de/Router/`

[3] `https://help.ubuntu.com/community/SSH/OpenSSH/PortForwarding`

[4] `https://help.ubuntu.com/community/SSH/OpenSSH/PortForwarding#Dynamic_Port_Forwarding`

[5] `https://howtoforge.com/reverse-ssh-tunneling`

ANHANG

2.2: ClientVM-Filterregeln

```
# DNS
iptables -i eth0 -I INPUT -p udp --sport 53 --source 10.1.1.1 -j ACCEPT
iptables -o eth0 -I OUTPUT -p udp --dport 53 --destination 10.1.1.1 -j
    ACCEPT
# HTTP, HTTPS, SSH
iptables -i eth0 -I INPUT -p tcp -m multiport --sports 22,80,443 -j ACCEPT
iptables -o eth0 -I OUTPUT -p tcp -m multiport --dports 22,80,443 -j ACCEPT
# PING
iptables -i eth0 -I INPUT -p icmp -j ACCEPT
iptables -o eth0 -I OUTPUT -p icmp -j ACCEPT
```

2.4: ClientVM Stateful Filtering

```
# Antworten in beide Richtungen zulassen
iptables -i eth0 -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -o eth0 -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# HTTP und HTTPS
iptables -o eth0 -I OUTPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
# DNS
iptables -o eth0 -I OUTPUT -p udp --dport 53 --destination 10.1.1.1 -j
    ACCEPT
# SSH
iptables -i eth0 -I INPUT -p tcp --dport 22 --soure 192.168.254.1/24 -m
    conntrack --cstate NEW -j ACCEPT
```

3.3: Filterregeln

```
# DNS auf 10.1.1.1 zulassen
iptables -A FORWARD -p udp --dport 53 --destination 10.1.1.1 -m conntrack
    --cstate NEW -j ACCEPT
# Labornetz sperren
iptables -A FORWARD --destination 10.0.0.0/8 -j REJECT
# HTTP und HTTPS zum Rest der Welt außer zur ServerVM zulassen
iptables -A FORWARD -p tcp --syn -m multiport --dports 80,443 ! --
    destination 172.16.137.144 -m --conntrack --cstate NEW -j ACCEPT
```

4.1: SSH-Ausgang

```
# Generated by iptables-save v1.4.4 on Thu Jun 16 12:48:17 2016
*filter
:INPUT ACCEPT [15:1139]
:FORWARD ACCEPT [115:11372]
:OUTPUT ACCEPT [15:1073]
-A FORWARD -i eth1 -p tcp --sport 22 -d 172.16.137.144 -j ACCEPT
-A FORWARD -i eth1 -p udp --sport 53 -j ACCEPT
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
COMMIT
# Completed on Thu Jun 16 12:48:17 2016
# Generated by iptables-save v1.4.4 on Thu Jun 16 12:48:17 2016
*nat
```

```
:PREROUTING ACCEPT [15:952]
:POSTROUTING ACCEPT [4:301]
:OUTPUT ACCEPT [3:241]
-A POSTROUTING -s 192.168.254.0/24 -o eth0 -j MASQUERADE
COMMIT
# Completed on Thu Jun 16 12:48:17 2016
```

5.1

```
# Generated by iptables-save v1.4.4 on Thu Jun 16 14:00:49 2016
*filter
:INPUT ACCEPT [3737:5146437]
:FORWARD ACCEPT [347:23103]
:OUTPUT ACCEPT [730:74822]
-A FORWARD -s 192.168.2.0/24 -d 172.0.0.0/8 -j ACCEPT
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD -j REJECT
COMMIT
# Completed on Thu Jun 16 14:00:49 2016
# Generated by iptables-save v1.4.4 on Thu Jun 16 14:00:49 2016
*nat
:PREROUTING ACCEPT [327:20291]
:POSTROUTING ACCEPT [35:2494]
:OUTPUT ACCEPT [35:2494]
-A POSTROUTING -s 192.168.254.0/24 -o eth0 -j MASQUERADE
COMMIT
# Completed on Thu Jun 16 14:00:49 2016
```

5.3

```
iptables -i eth0 -I INPUT -p udp --dport 1194 -m state --state NEW,
    ESTABLISHED -j ACCEPT
iptables -o eth0 -I OUTPUT -p udp --sport 1194 -m state --state ESTABLISHED
     -j ACCEPT
```

6.1

```
# HTTP in beide Richtungen als Antwort öffnen
-A FORWARD -p tcp --port 80 -m state --state ESTABLISHED,RELATED -j ACCEPT
# ausgehende HTTP-Requests öffnen
-A FORWARD -p tcp --dport 80 -m state --state NEW -j ACCEPT
# DNS in beide Richtungen öffnen
-A FORWARD -p udp --port 53 -m state --state NEW,ESTABLISHED -j ACCEPT
```

6.3

```
## INPUT
# Verbindungen auf Squid-Port öffnen
-i eth1 -A INPUT -p tcp --dport 3128 -m state --state NEW,ESTABLISHED -j
    ACCEPT
# HTTP und HTTPS Responses erlauben
-i eth1 -A INPUT -p tcp -m multiport --sports 80,443 -m state --state
    ESTABLISHED -j ACCEPT
```

```
6  # DNS
7  -i eth1 -A OUTPUT -p udp --sport 53 -m state --state NEW,ESTABLISHED -j
       ACCEPT
8  ## OUTPUT
9  # Verbindungen auf Squid-Port öffnen
10 -o eth1 -A OUTPUT -p tcp --sport 3128 -m state --state ESTABLISHED -j
       ACCEPT
11 # HTTP und HTTPS Responses erlauben
12 -o eth1 -A OUTPUT -p tcp -m multiport --dports 80,443 -m state --state
       ESTABLISHED -j ACCEPT
13 # DNS
14 -o eth1 -A OUTPUT -p udp --dport 53 -m state --state ESTABLISHED -j ACCEPT
```