KN-Praktikum: SDN mit Mininet und POX

Vorbereitung

Username und Passwort für die VM:

- mininet / mininet

Copy/Paste in der VM:

- STRG+SHIFT+C / STRG+SHIFT+V

Geteilte Zwischenablage anschalten:

- Im VM-Fenster unter Geräte -> Gemeinsame Zwischenablage -> bidirektional

Falls die geteilte Zwischenablage nicht funktioniert, folgendes nachinstallieren:

```
sudo apt install virtualbox-guest-x11
```

Und dann die geteilte Zwischenablage starten:

```
sudo VBoxClient --clipboard
```

Ihr könnt dieses Repo in das Home-Verzeichnis der Mininet-VM clonen, um die Dateien nicht händisch kopieren zu müssen:

```
cd ~
git clone https://github.com/Flipp-io/SDN-Praktikum.git
```

Den Befehl zum Starten der Topologie führt ihr dann aus dem geclonten Ordner heraus aus. (also ~/SDN-Praktikum)

Curl installieren:

```
sudo apt install curl
```

Wichtige Mininet-Befehle

Wenn ein Befehl mit "mininet>" beginnt, ist er in der Mininet-CLI auszuführen. Wenn er mit "h1>" beginnt, ist er im Terminal-Fenster des Hosts h1 auszuführen.

Zwischen allen Hosts pingen:

```
mininet> pingall
```

Die Flowtable des Switches ausgeben:

```
mininet> dpctl dump-flows --color=always
```

Die Topologie beenden:

```
mininet> exit
```

Aufgabe A. Aufwärmübung: Mininet mit POX verwenden

Dieses erste Szenario soll helfen, euch mit Mininet und Pox vertraut zu machen. Das Prinzip von SDN wird hier zunächst auf Layer 2 umgesetzt, indem ein Switch neue Flowtable-Einträge von einem Controller zugewiesen bekommt.

1. POX-Controller starten

Startet den POX-SDN-Controller mit dem Layer2-Lernswitch-Modul und Debug-Ausgaben.

```
~/pox/pox.py forwarding.l2_learning samples.pretty_log --DEBUG
```

2. Mininet starten

Startet in einer zweiten Bash eine Mininet-Topologie mit externem Controller:

```
sudo mn --topo=single,2 --controller=remote,port=6633 --mac -x
```

Hinweise zu den Optionen:

- '--mac' -> die Hosts erhalten einfacher zu lesende MAC-Adressen
- '-x' -> öffnet für jeden Host ein eigenes Terminal-Fenster.

3. Testen mit `ping`

```
mininet> pingall
```

Können sich die Hosts gegenseitig erreichen? Achtet auch auf die Ausgabe des Controllers.

4. Flowtable des Switches ausgeben lassen

```
mininet> dpctl dump-flows --color=always
```

Findet ihr die MAC- und IP-Adressen der Hosts wieder?
Wie soll der Switch laut Tabelle mit den Paketen der Flows umgehen?

Aufgabe B. SDN-Firewall mit statischer ACL

In diesem Versuch sollt ihr eine einfache Firewall mit statischen Regeln implementieren, die den Verkehr basierend auf IP-Adressen, Protokollen und Ports blockiert oder erlaubt. Die Filter-Regeln sollt ihr selbst festlegen und im Code umsetzen.

Vorbereitung

POX-Modul

Der Großteil des Controller-Codes ist bereits für euch vorbereitet. Speichert die Datei "pox_firewall_acl.py" im Verzeichnis "~/pox" ab. Falls ihr das Repo ins Home-Verzeichnis geclonet habt, könnt ihr die Datei mit folgendem Befehl an die richtige Stelle kopieren:

```
cp ~/SDN-Praktikum/pox_firewall_acl.py ~/pox/pox_firewall_acl.py
```

Der Controller kann mit diesem Befehl gestartet werden:

```
~/pox/pox.py samples.pretty_log --DEBUG pox_firewall_acl
```

Mininet-Topologie

Der Code für die Netzwerktopologie ist in der Datei "custom_topo.py" zu finden.
Diese Topologie enthält einen internen Client (h1), einen Server (h2) und einen externen Client (h3). Alle Hosts befinden sich im selben Subnetz (10.0.0.0/24). Auf h2 sollen ein Webserver und ein SSH-Server laufen. Durch die Firewall soll der Server vor unberechtigtem Zugriff geschützt werden.
Die Topologie kann mit diesem Befehl gestartet werden:

```
sudo mn --custom custom_topo.py --topo sdnfirewall --
controller=remote,ip=127.0.0.1,port=6633 --mac -x
```

Durchführung

- Startet den Controller mit dem Firewall-Modul:

```
~/pox/pox.py samples.pretty_log --DEBUG pox_firewall_acl
```

- Startet in einem zweiten Terminal die Mininet-Topologie:

```
sudo mn --custom custom_topo.py --topo sdnfirewall --
controller=remote,ip=127.0.0.1,port=6633 --mac -x
```

- Startet einen HTTP-Server auf h2 (den Befehl in der Mininet-CLI ausführen):

```
mininet> h2 python3 -m http.server 80 &
```

- Startet einen SSH-Server auf h2 (den Befehl in der Mininet-CLI ausführen):

```
mininet> h2 /usr/sbin/sshd
```

- Prüft die Erreichbarkeit der Hosts untereinander mit Ping:

```
mininet> pingall
```

- Prüft die Erreichbarkeit des HTTP-Servers von beiden Clients (h1 und h3):

```
h1> curl 10.0.0.2
```

- Prüft die Erreichbarkeit des SSH-Servers von beiden Clients (h1 und h3):

```
h1> ssh 10.0.0.2
```

Wenn jeder Host von allen anderen Erreicht werden konnte, ist das Netzwerk korrekt konfiguriert. Nun geht es darum, den Zugriff auf den Server und den internen Client zu steuern und ggf zu blockieren.

Aufgaben

- Schaut euch den Code des POX-Controllers an (Datei 'pox_firewall_acl.py') und versucht ihn nachzuvollziehen (bspw mit dem Editor "emacs" öffnen)
- Welche Informationen eines eintreffenden Pakets werden extrahiert?

- Überlegt euch sinnvolle Regeln, die die Sicherheit im Netzwerk erhöhen und bspw. den Zugriff auf die Server-Dienste steuern (ein bis zwei Regeln reichen zunächst aus). Die Regeln können zunächst mit Worten formuliert werden.
    - Regelsatz der Musterlösung:
        - keinen Traffic vom externen Client zulassen bis auf Zugriff auf den HTTP-Server
        - der interne Client darf alles, bis auf SSH zu h2
    - zusätzliche Mögliche Regeln:
        - ICMP (Ping) von h3 zu h2 blockieren
        - aber HTTP von h3 zu h2 erlauben
        - ICMP und HTTP von h1 zu h2 erlauben
- Implementiert die Regeln im Code (in der 'is_blocked'-Methode)
- Überprüft, ob die Regeln wirksam sind, indem ihr die Erreichbarkeit der Hosts und Server-Dienste prüft und die Flowtabelle anschaut.

Bonusaufgabe falls noch Zeit: Erweiterung auf IP-Subnetze

Erweitert die Logik-Regeln, sodass sie auf ganze Subnetze angewendet werden und nicht nur auf einzelne Hosts. Dafür müssen andere IP-Adressen an die Hosts vergeben werden (anzupassen in der Mininet-Topologie).

- Setzt die Hosts in unterschiedliche "/24"-er Subnetze (zB. 10.0.1.0/24, 10.0.2.0/24, 10.0.3.0/24, ...). Für diese Subnetze legt ihr anschließend die Firewall-Regeln fest. Damit die Hosts sich grundsätzlich ohne Routing erreichen können, setzt ihre Netzmaske in der Topologie auf "/16" (zB. 10.0.1.1/16, 10.0.2.2/16, ...). Dadurch befinden sie sich in einem größeren Subnetz, die Regeln werden jedoch auf die kleineren Subnetze angewendet.
- Passt die Filter-Regeln im Code an, sodass sie auf die neu erstellten Netze angewendet werden.
- Fügt (einen) weitere(n) Host(s) in den verschiedenen Subnetzen hinzu oder ändert die IP-Adressen der vorhandenen Hosts. Prüft, ob die Regeln weiterhin wie gewünscht funktionieren.
    - mögliche Lösung in pox_firewall_acl_subnets.py und custom_topo_subnets.py

Fragen

1. Was fällt euch auf, wenn ihr euch die Flowtable ausgeben lasst? Was passiert mit den Paketen? (Befehl in Mininet: "dpctl dump-flows --color=always")

2. Bisher werden nur für die erlaubten Pakete Flows in den Switches installiert. Was passiert mit den anderen Paketen? Was hat das für eine Auswirkung? Kann man als Angreifer dieses Verhalten ggf. ausnutzen? Wie kann man das Problem lösen?

3. Was ist der Unterschied zwischen einer SDN-basierten Firewall und einer traditionellen Firewall?

4. Welche Vorteile bietet die Regelverwaltung via SDN-Controller gegenüber einer herkömmlichen Firewall?

Deepdive: Erweiterte SDN-Implementierungen

Für fortgeschrittene Anwender und zusätzliche Experimente haben wir erweiterte SDN-Implementierungen im `deepdive/` Ordner erstellt.

Verfügbare Erweiterungen

L2 Learning Switch mit Firewall

- **Datei:** `deepdive/l2_switch_with_firewall.py`
- **Features:** MAC-Learning + IP-basierte Firewall
- **Verwendung:** `~/pox/pox.py deepdive.l2_switch_with_firewall samples.pretty_log --DEBUG`

Layer 3 Switch mit Firewall

- **Datei:** `deepdive/l3_switch_with_firewall.py`
- **Features:** IP-Routing + ARP-Handling + Subnetz-basierte Firewall
- **Verwendung:** `~/pox/pox.py deepdive.l3_switch_with_firewall samples.pretty_log --DEBUG`

Enterprise-Netzwerk Topologie

- **Datei:** `deepdive/enterprise_network_topo.py`
- **Features:** 27 Hosts in 5 Subnetzen (Büros, DMZ, Server-Farm, Management)
- **Verwendung:** `sudo mn --custom deepdive.enterprise_network_topo --topo enterprise --controller=remote,ip=127.0.0.1,port=6633 --mac -x`

Hilfedateien

- **`deepdive/firewall_help.py`:** Umfassende Firewall-Regel Beispiele
- **`deepdive/enterprise_firewall_rules.py`:** Enterprise-spezifische Sicherheitsrichtlinien

Dokumentation

Siehe `deepdive/README.md` für detaillierte Anleitungen, Demo-Szenarien und Vergleichstabellen.

Demo-Szenarien

```
# Enterprise-Topologie mit L3 Switch
~/pox/pox.py l3_switch_with_firewall samples.pretty_log --DEBUG
sudo mn --custom enterprise_network_topo --topo enterprise --
controller=remote,ip=127.0.0.1,port=6633 --mac -x

# Tests
mininet> h15 ping h8      # Externer → Webserver (erlaubt)
mininet> h15 ping h19     # Externer → MySQL (blockiert)
mininet> h1 ping h8       # Büro-Client → Webserver (erlaubt)
mininet> h6 ping h25      # IT-Admin → Monitoring (erlaubt)
```

Diese Erweiterungen demonstrieren die vollen möglichkeiten von SDN mit realistischen Enterprise-Szenarien und erweiterten Sicherheitsfunktionen.