Redes y comunicaciones
Practica 2

Estudiante : Jorge Rebosio Cumpa

Ejercicio 1:

Para el desarrollo de este ejercicio se utilizó una máquina de real (no necesite de una virtual). Para desplegar la topología tipo árbol se utilizó el siguiente comando :

sudo mn --controller remote,ip=127.0.0.1,port=6653 --switch ovsk,protocols=OpenFlow13 –topo=tree,3,2

```
jorgerebosio@jrebosio-Aspire-A515-54:~$ sudo mn --controller remote,ip=127.0.0.1,port=665
3 --switch ovsk,protocols=OpenFlow13 --topo=tree,3,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7)
 (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
```

Asimismo, se uso Foodlight 1.2 como controlador remoto de la red.

```
mininet> sh ovs-vsctl show
a6f74bad-d5f4-4796-afe3-082ed2d9a3f2
    Bridge "s4"
        Controller "ptcp:6657"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port "s4-eth2"
            Interface "s4-eth2"
        Port "s4-eth3"
            Interface "s4-eth3"
        Port "s4-eth1"
            Interface "s4-eth1"
        Port "s4"
            Interface "s4"
                type: internal
    Bridge "s2"
        Controller "ptcp:6655"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        fail_mode: secure
        Port "s2"
```

Se desarrolló un script para aplicar las reglas ACL. Este script se desarrolló en lenguaje Python. Conociendo el Host que va ser denegado se va a calcular el Switch y el Port del Host y se deniega los paquetes que entran y salen desde el Host hacia toda la red . La dirección de la red es 10.0.0.0/24.

Es así como se aísla al Host de toda la Topología de la red . Para este ejemplo, como se ve en el código solo se van a aislar a los Host 2 y 5. Pero se pueden denegar cualquier otro Host, solo se tendría que agregar a la lista To_Deny.

```
pusher = StaticEntryPusher('127.0.0.1')
to_deny = [2, 5]

for host in to_deny:
    switch = (host - 1) // 2 + 3 + (host > 4)
    port = 2 - (host % 2)
    print(host, switch , port)
    flow = {
                "switch": "00:00:00:00:00:00:00:0" + str(switch),
                "name":"flow-mod-" + str(host),
                "priority":"32769",
                "eth_type": "0x800",
                "in_port": str(port),
                "ipv4_src":"10.0.0." + str(host)+ "/32",
                "ipv4_dst":"10.0.0.0/24",
                "active":"true",
                "actions": "drop"
            }
    pusher.set(flow)
```

Una vez ejecutado el Script y con las reglas ACL aplicadas, se puede verificar que se ha insertado el flujo en S3 y S6 , switches de Host 2 y Host 5 respectivamente.

```
mininet>  sh ovs-ofctl dump-flows s3 -O OpenFlow13
 cookie=0xa000004039d1a7, duration=3547.416s, table=0, n_packets=13, n_bytes=1274, send_flow
_rem priority=32769,ip,in_port="s3-eth2",nw_src=10.0.0.2,nw_dst=10.0.0.0/24 actions=drop
 cookie=0x0, duration=3564.092s, table=0, n_packets=597, n_bytes=49847, priority=0 actions=C
ONTROLLER:65535
mininet>  sh ovs-ofctl dump-flows s6 -O OpenFlow13
 cookie=0xa000004039d1aa, duration=3562.785s, table=0, n_packets=13, n_bytes=1274, send_flow
_rem priority=32769,ip,in_port="s6-eth1",nw_src=10.0.0.5,nw_dst=10.0.0.0/24 actions=drop
 cookie=0x0, duration=3579.508s, table=0, n_packets=597, n_bytes=50275, priority=0 actions=C
ONTROLLER:65535
```

Asimismo el Host 2 y el Host 5 no se pueden comunicar con ningún otro Host de la Topología.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 h4 X h6 h7 h8
h2 -> X X X X X X X
h3 -> h1 X h4 X h6 h7 h8
h4 -> h1 X h3 X h6 h7 h8
h5 -> X X X X X X X
h6 -> h1 X h3 h4 X h7 h8
h7 -> h1 X h3 h4 X h6 h8
h8 -> h1 X h3 h4 X h6 h7
*** Results: 46% dropped (30/56 received)
```

Ejercicio 2 :

Para este ejercicio se cargó la data input del archivo :

https://feodotracker.abuse.ch/downloads/ipblocklist_aggressive.csv

Y se desarrolló un código en Python que nos permita generar las reglas ACL que denegaran el acceso de estas IP's maliciosas. Asimismo, estas reglas se agregaran a la Red creada en el ejercicio 1.

A continuación se presenta una parte del código :

```
pusher = StaticEntryPusher('127.0.0.1')

lines = pd.read_csv(sys.argv[1])
lines = lines[['dst_ip', 'dst_port']]
lines = lines.dropna()



for line, data in lines.iterrows():
    ip, port = data['dst_ip'], data['dst_port']
    curl = {
        'switch': '00:00:00:00:00:00:00:06',
        "name":"flow-mod-" + str(line),
        'src-ip': ip + "/32",
        'nw-proto': 'TCP',
        'tp-dst': port,
        'action': 'DENY'
    }
    pusher.set(curl)
```

Con la ayuda de la librería Pandas leeremos el archivo CSV e iteraremos entre cada fila para crear una regla ACL para cada IP maliciosa.

```
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
(200, 'OK', '{"status" : "Success! New rule added."}')
```

Con este código se podría cargar cualquier Data input de IP adreess maliciosas y denegarles acceso a la red.

Se adjunta todos los códigos y archivos en el repositorio .