BGP Exploration and Attack Lab

1 Overview

Border Gateway Protocol (BGP) is the standard exterior gateway protocol designed to exchange routing and reachability information among autonomous systems (AS) on the Internet. It is the "glue" of the Internet, and is an essential piece of the Internet infrastructure. It is also a primary attack target, because if attackers can compromise BGP, they can disconnect the Internet and redirect traffics.

The goal of this lab is to help students understand how BGP "glues" the Internet together, and how the Internet is actually connected. We have built an Internet emulator, and will use this emulator as the basis for the lab activities. Due to the complexity of BGP, the explanation of how BGP works is provided in a separate document. It is essential for the lab activities, so students should read the document before working on the lab. This lab covers the following topics:

- How the BGP protocol works
- Autonomous systems
- BGP configuration, BGP Large Communities
- Routing
- Internet Exchange (IX)
- BGP attack, network prefix hijacking

Supporting materials. BGP is quite complicated, especially its practice side. To help students work on this lab, I have written a chapter on BGP, which will be included in the 3rd edition of my book. I will make this chapter a sample chapter, so it is free for everybody to download. The link to this chapter can be found from the web page of this lab. Without reading this chapter or covering it in the lecture, it will be quite hard for students to work on this lab.

Note for instructors. Tasks 1 to 4 are designed to help students understand the technical details of BGP, it is for the instructors who do cover BGP in-depth in their classes. Only Task 5, BGP attacks, is related to security, so if instructors only want students to focus on the security aspect of BGP, they can skip Tasks 1 - 4, and assign Task 5 directly to students, as this task does not depend on the earlier tasks. High-level knowledge on BGP should be sufficient for Task 5.

Lab environment. This lab has been tested on our pre-built Ubuntu 20.04 VM, which can be downloaded from the SEED website. Since we use containers to set up the lab environment, this lab does not depend much on the SEED VM. You can do this lab using other VMs, physical machines, or VMs on the cloud.

2 The Lab Setup and the SEED Internet Emulator

This lab will be performed inside the SEED Internet Emulator (simply called the emulator in this document). We provide a pre-built emulator in two different forms: Python code and container files. The container files are generated from the Python code, but students need to install the SEED Emulator source code from the GitHub to run the Python code. The container files can be directly used without the emulator source code. Instructors who would like to customize the emulator can modify the Python code, generate their own container files, and then provide the files to students, replacing the ones included in the lab setup file.

Download the emulator files. Please download the `Labsetup.zip` file from the web page, and unzip it. The files inside the `output` sub-folder are the actual emulation files (container files) that are generated from the Python code `mini-internet.py`.

Start the emulation. We will directly use the container files in the `output` folder. Go to this folder, and run the following docker commands to build and start the containers. We recommend that you run the emulator inside the provided SEED Ubuntu 20.04 VM, but doing it in a generic Ubuntu 20.04 operating system should not have any problem, as long as the docker software is installed. Readers can find the docker manual from this link.

```
$ docker-compose build
$ docker-compose up

// Aliases for the Compose commands above (only available in the SEED VM)
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

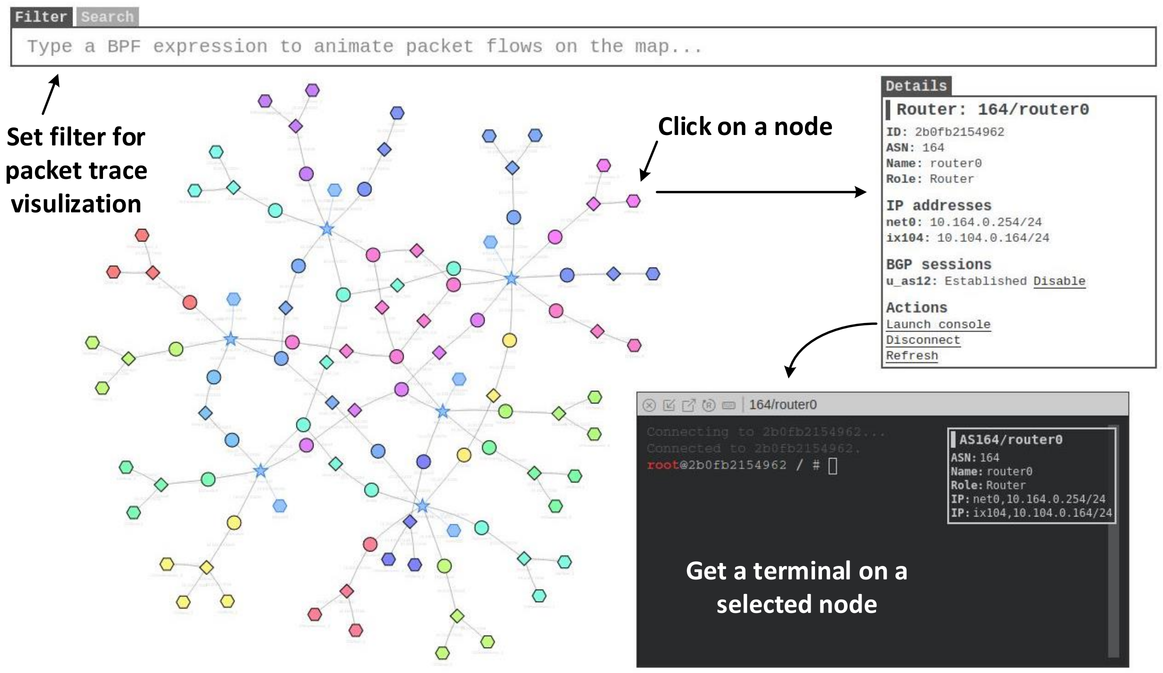
2.1 The Network Map

Each computer (hosts or routers) running inside the emulator is a docker container. Users can access these computers using docker commands, such as getting a shell inside a container. The emulator also comes with a web application, which visualizes all the hosts, routers, and networks. After the emulator starts, the map can be accessed from this URL: `http://localhost:8080/map.html`. See Figure 1. Users can interact with this map, such as getting a terminal from any of the container, disabling BGP sessions (see Figure 2). Users can also set filters to visualize network traffic. The syntax of the filter is the same as that in `tcpdump`; actually, the filter is directly fed into the `tcpdump` program running on all nodes.

2.2 Modifying the BGP Configuration File

We need to modify the BGP configuration file in several tasks. We can do that by directly modifying the configuration file inside a container. Anther way is to copy the file into the host VM, do the editing from the host VM, and then copy it back. Let us see an example (assuming that we want to modify the BGP configuration file of AS-180):

```
// Find out the IP of the AS-180's BGP router container
$ dockps | grep 180
6bf0bcda8d06  as180h-host_1-10.180.0.72
437874056b15  as180h-webservice_0-10.180.0.71
29676d5034ce  as180r-router0-10.180.0.254     ← This is AS-180's BGP router

// Copy the configuration file from the container to the host machine
$ docker cp 2967:/etc/bird/bird.conf ./as180_bird.conf
```


Filter
Search
Type a BPF expression to animate packet flows on the map...
Set filter for packet trace visulization
Click on a node
Details
Get a terminal on a selected node
164/router0

```
Router: 164/router0
ID: 2b0fb2154962
ASN: 164
Name: router0
Role: Router

IP addresses
net0: 10.164.0.254/24
ix104: 10.104.0.164/24

BGP sessions
u_as12: Established Disable

Actions
Launch console
Disconnect
Refresh
```

```
Connecting to 2b0fb2154962...
Connected to 2b0fb2154962.
root@2b0fb2154962 / #
```

```
AS164/router0
ASN: 164
Name: router0
Role: Router
IP: net0,10.164.0.254/24
IP: ix104,10.104.0.164/24
```

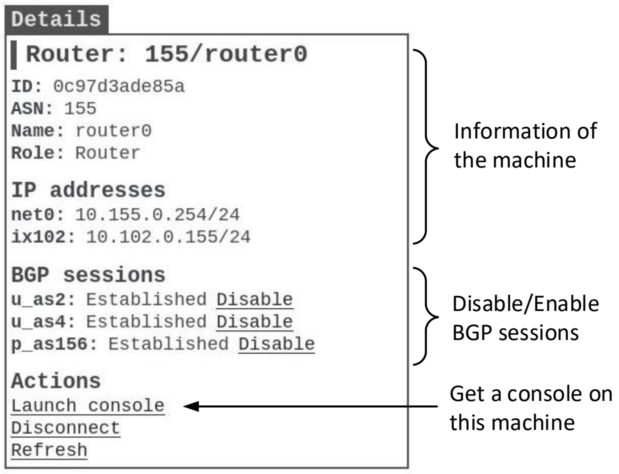
Figure 1: The SEED Internet Emulator

```
... edit the file ...

// Copy the file back to the container
$ docker cp ./as180_bird.conf 2967:/etc/bird/bird.conf

// Reload the configuration on the container
$ docker exec 2967 birdc configure      ← Run "birdc configure"
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured
```

2.3 Convention used in the Emulator

To make it easy to identify the roles of each node in the emulator, we have created a set of conventions when assigning various numbers to nodes. These conventions are only for the emulator, and they do not hold in the real world.

- Autonomous System Number (ASN) assignment:
  - ASN `2 - 9`: for large transit ASes (e.g., national backbone).
  - ASN `10 - 19`: for smaller transit ASes.
  - ASN `100 - 149`: for Internet Exchanges (IX).
  - ASN `150 - 199`: for stub ASes.
- Network prefixes and IP addresses:


Details

Router: 155/router0

ID: 0c97d3ade85a
ASN: 155
Name: router0
Role: Router

IP addresses
net0: 10.155.0.254/24
ix102: 10.102.0.155/24

BGP sessions
u_as2: Established Disable
u_as4: Established Disable
p_as156: Established Disable

Actions
Launch console
Disconnect
Refresh

Information of the machine
Disable/Enable BGP sessions
Get a console on this machine

Figure 2: Interact with a node

- For an autonomous system N, its first internal network's prefix is `10.N.0.0/24`, the second internal network is `10.N.1.0/24`, and so on.
- In each network, the address `200` to `255` are for routers. For hosts (non-router), their IP address start from `71`. For example, in AS-155, `10.155.0.255` is a BGP router, while `10.155.0.71` is a host.

3 Task 1: Stub Autonomous System

An autonomous system (AS) is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators on behalf of a single administrative entity or domain. It is a basic unit in BGP. A stub AS the type of AS that does not provide transit service to others. Most of end users are stub ASes, including universities, organization, and most companies. Another type of AS is called transit AS. They provide transit services for other ASes, and they are Internet service providers.

In this task, we focus on stub ASes, see how it peers with others. For this type of ASes, we will gain a glimpse of how BGP works. Students should read Sections 1- 7 before working on this task.

3.1 Task 1.a: Understanding AS-155's BGP Configuration

AS-155 is a stub AS, which has one network (`10.155.0.0/24`) and one BGP router (`10.155.0.254`). Please get a terminal on the container of this router, study its BGP configuration in `/etc/bird/bird.conf`, and then do the following tasks.

- **Task 1.a.1:** From the BGP configuration file, identify who AS-155 peers with. You can ignore the filtering part of the configuration for now. Here is one of the BGP entries in the configuration file. See Section 6 of the provided BGP tutorial for the explanation of each entry.

```
protocol bgp u_as2 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(PROVIDER_COMM);
            bgp_local_pref = 10;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
        next hop self;
    };
    local 10.102.0.155 as 155;
    neighbor 10.102.0.2 as 2;
}
```

- **Task 1.a.2:** AS-155 peers with several ASes, so if AS-155 loses the connection with one of them, it can still connect to the Internet. Please design an experiment to demonstrate this. You can enable/disable BGP sessions either from the graph (see Figure 2) or using the `birdc` command (see the following examples). In your experiment, please show how the routing table changes when a particular BGP session is disabled/enabled (you can use the "`ip route`" to see the content of a routing table).

```
root@0c97d3ade85a / # birdc show protocols
BIRD 2.0.7 ready.
Name       Proto      Table      State  Since         Info
u_as2      BGP        ---        up     14:51:40.447  Established
u_as4      BGP        ---        up     14:51:39.500  Established

root@0c97d3ade85a / # birdc disable u_as2   ← Disable peering with AS-2
BIRD 2.0.7 ready.
u_as2: disabled

root@0c97d3ade85a / # birdc show protocols
BIRD 2.0.7 ready.
Name       Proto      Table      State  Since         Info
u_as2      BGP        ---        down   16:32:14.883
u_as4      BGP        ---        up     14:51:39.500  Established
```

3.2 Task 1.b: Observing BGP UPDATE Messages

The objective of this task is to understand the BGP UPDATE messages. Run `tcpdump` on AS-150's BGP router, use it to monitor the BGP traffic. This command will save the captured BGP packets into `/tmp/bgp.pcap`.

```
# tcpdump -i any -w /tmp/bgp.pcap "tcp port 179"
```

Your job is to do something on AS-155's BGP router to trigger at least one BGP route withdrawal and one BGP route advertisement messages. These UPDATE messages should be captured by the `tcpdump` command on AS-150 and stored inside `bgp.pcap`. Copy this file to the host computer using the "`docker cp`" command (from the host), and then load it into Wireshark. Pick a route advertisement message and a route withdraw message, provide explanation on these two messages. Screenshots should be provided in the lab report.

3.3 Task 1.c: Experimenting with Large Communities

When a BGP router sends routes to its peers, they do not send all the routes they know. What routes are sent depends on many factors, such as the region of the peers, the business relationship between the peers, and policies. To help BGP routers make such decisions, additional information needs to be attached to each route, as the predefined set of route attributes cannot capture such information. The BGP Large Communities are created to serve this goal. The objective of this task is to learn how it is used in the emulator to reflect the business relationship among peers.

Let us assume that due to some technical issue, the connection between AS-4 and AS-156 is broken. We can emulate this by disabling the peering between AS-4 and AS-156. Since AS-4 is the only service provider for AS-156, this essentially disconnects AS-156 from the Internet. If we ping another host from one of the hosts in AS-156, we can see the following results (please do not run ping from the BGP router; only run it from a host):

```
// On 10.156.0.72
# ping 10.155.0.71
PING 10.155.0.71 (10.155.0.71) 56(84) bytes of data.
64 bytes from 10.155.0.71: icmp_seq=1 ttl=62 time=14.6 ms
64 bytes from 10.155.0.71: icmp_seq=2 ttl=62 time=0.363 ms

# ping 10.161.0.71
PING 10.161.0.71 (10.161.0.71) 56(84) bytes of data.
From 10.156.0.254 icmp_seq=1 Destination Net Unreachable
From 10.156.0.254 icmp_seq=2 Destination Net Unreachable
From 10.156.0.254 icmp_seq=3 Destination Net Unreachable
```

We can see that `10.155.0.71` is still reachable, because it belongs to AS-155, which is still peered with AS-156. However, `10.161.0.71` (belonging to AS-161) cannot be reached, because nobody will route the packet for AS-156. The question is, AS-156 still peers with AS-155, which is connected to the Internet, so why is AS-156 not able to connect to the Internet? This is because whether an AS forwards traffic for another AS depends on their business relationship.

While AS-156 and AS-4 are trying to solve the problem, AS-156 holds an emergency meeting with AS-155, agreeing to pay AS-155, so its traffic can temporarily go through AS-155 to reach the Internet. This requires some changes on AS-155's BGP router. Please make such changes, so AS-155 can temporarily provide a transit service to AS-156. Please read Section 9 before working on this task. After making the changes, please make sure run the following command to reload the BIRD configuration.

```
# birdc configure
BIRD 2.0.7 ready.
Reading configuration from /etc/bird/bird.conf
Reconfigured
```

3.4 Task 1.d: Configuring AS-180

AS-180 is already included in the emulator. It connects to the IX-105 Internet exchange (Houston), but it does not peer with anybody, so it is not connected to the Internet. In this task, students need to complete the configuration of AS-180's BGP router and all the related BGP routers, so the following goals are achieved:

- Peer AS-180 with AS-171, so they can directly reach each other.
- Peer AS-180 with the AS-2 and AS-3 transit autonomous systems, so they can reach other destination via these transits.

Shell script. In this task, we need to modify several BIRD configuration files. Instead of going to each container to make changes, we can copy all the BIRD configuration files from the containers to the host VM, make changes, and then copy them back to the containers. We have included two shell scripts in the `task1` folder to facilitate the process:

- `import_bird_conf.sh`: get all the needed BIRD configuration files from the containers. If a configuration file already exists in the current folder, the file will not be overwritten.
- `export_bird_conf.sh`: copy the BIRD configuration files to the containers and run `"birdc configure"` to reload the configuration.

Debugging. If the result is not what you have expected, you may need to debug to find out what has gone wrong. In particular, you want to know where your packets go. For example, if you run `ping`, but you do not get a reply, you want to know where the problem is. You can use the filter option in the map client, and visualize the traffic flow. The syntax of the filter is the same as that in `tcpdump`. We give a few examples in the following.

```
"icmp"                         ← show all icmp traffic
"icmp and src 10.180.0.71"  ← show icmp traffic from 10.180.0.71
"icmp and dst 10.180.0.71"  ← show icmp traffic to 10.180.0.71
```

Lab report. In your lab report, please include the content that you add to the BIRD configuration files, and provide proper explanation. Please also include screenshots (such as traceroute) to demonstrate that your task is successful.

4 Task 2: Transit Autonomous System

If two ASes want to connect, they can peer with each other at an Internet exchange point. The question is how two ASes in two different locations can reach each other. It is hard for them to find a common location to peer. To solve this problem, a special type of AS is needed.

This type of AS have BGP routers in many Internet Exchange Points, where they peer with many other ASes. Once packets get into its networks, they will be pulled from one IX to another IX (typically via some internal routers), and eventually be handed over to another AS. This type of AS provides the transit service for other ASes. That is how the hosts in one AS can reach the hosts in another AS, even though they are not peers with each other. This special of AS is called *Transit AS*.

In this task, we will first understand how a transit AS works and then we will configure a transit AS in our Internet emulator. Students should read Section 10 of the tutorial before working on this task. We pick AS-3 transit autonomous system in this task. This AS has four BGP routers, each at a different Internet exchange (IX). We pick the one connected to the Miami Internet exchange (IX-103).

4.1 Task 2.a: Experimenting with IBGP

For the task, we first need to find some traffic that goes through AS-3. We will ping `10.164.0.71` from a host in AS-162. Using the map client program, we can see that the packets go through the AS-3 transit AS. If this is not consistent with your observation, do find some other traffics that go through AS-3.

We will now disable the IBGP sessions on AS-3's BGP router at IX-103 either using the map client or from the command line (see the following example).

```
# birdc
bird> show protocols
Name        Proto       Table       State  Since          Info
...
ibgp1       BGP         ---         up     20:19:03.800   Established
ibgp2       BGP         ---         up     20:19:11.921   Established
ibgp3       BGP         ---         up     20:20:50.238   Established

bird> disable ibgp3
bird> show protocols ibgp3
Name        Proto       Table       State  Since          Info
ibgp3       BGP         ---         down   20:26:44.526
```

Before disabling IBGP, show the routing table on the BGP router (using "`ip route`"). Compare the results before and after disabling IBGP, and explain your observations.

4.2 Task 2.b: Experimenting with IGP

In this task, we will use the same BGP router. We will disable the OSPF routing protocol, and see how it affects the routing. There are several ways to disable OSPF. One way is to do it inside `birdc`:

```
# bridc
birdc> show protocols
...
ospf1       OSPF        t_ospf      up     19:49:43.343   Running
...

birdc> disable ospf1
birdc> show protocols ospf1
ospf1       OSPF        t_ospf      down   19:57:37.187
```

Before and after disabling OSPF, show the routing table on the BGP router (using "`ip route`"). Compare the results. Based on the observation, explain why the IGP is essential for the transit autonomous systems.

4.3 Task 2.c: Configuring AS-5

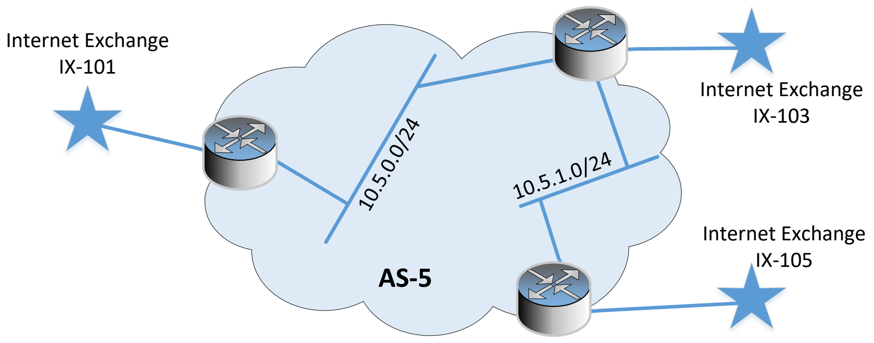


Figure 3: AS-5's network diagram

The transit autonomous system AS-5 is already included in the emulator. It connects to three Internet exchanges: IX-101, IX-103, and IX-105, but it does not peer with any AS. Its topology can be found in Figure 3. In this task, students need to conduct the following tasks:

- The IBGP peering for AS-5 is already established in the emulator. Please use AS-5's BGP router at IX-101 as an example, explain the meaning of its IBGP configuration.
- AS-5 will provide the transit service for AS-153 (at IX-101), AS-160 (at IX-103), and AS-171 (at IX-105). Please configure their EBGP peering accordingly. Students can learn from the other transit ASes, and see how their peering is configured.
- AS-5 and AS-3 are both transit ASes, and they decide to peer at IX-103 (Miami). Since they are roughly the same size, they both benefit from the peering equally, so they decide that their relationship should be peer-to-peer, not provider-to-customer. They do not pay each other.

In this task, we need to modify several BIRD configuration files. Just like in Task 1, we have created two shell scripts in the `task2` folder. They can be used to automate the downloading/uploading of the BIRD configuration from/to the containers.

In your lab report, please include the content you add to the BIRD configuration file, and provide proper explanation. Please also include screenshots (such as traceroute) to demonstrate that your task is successful.

5 Task 3: Path Selection

BGP routers typically receive multiple routes to the same network. All these routes will be kept, but BGP will run a best route selection algorithm to choose one route as the current best route. This route will be the one announced to the peers; it is also the one given to the kernel routing table, so routing is based on this selected route. When the best route is retracted, BGP router re-runs the best route selection algorithm to find the current best route.

The top-2 criteria used by BIRD is the following: (1) prefer route with the highest Local Preference attribute, (2) prefer route with the shortest AS path. In this task, we will experiment with these two criteria, and see how they affect the path selection. Students should read Section 8 of the tutorials before working on this task.

Task 3.a. The "`birdc show route all > all-routes`" command can list all the BGP routes (the output is quite long, so it is better to save the output to a file). The "`ip route`" command can list all the entries in the kernel routing table.

Go to AS-150's BGP router, and show all the BGP routes. Find a network prefix that has more than one routes. Explain their differences. Compare these routes to the entries in the kernel routing table, and identify which route is selected as the best route, and explain why this route is selected?

Task 3.b. AS-150 peers with both AS-2 and AS-3, which provide the Internet services to AS-150. However, because the link to AS-2 is slower than the link to AS-3, AS-150 wants to use AS-2 only as the backup upstream link, i.e., AS-150's inbound and outbound traffic should always use AS-3, unless the link is broken. Please modify AS-150's BGP configuration to achieve this goal.

6 Task 4: IP Anycast

IP Anycast is a network addressing and routing methodology in which a single IP address is shared by multiple machines (usually in different locations). When we send a packet to this IP address, one of the computers will get the packet. Exactly which one will get it is hard to tell, because it depends on the routing. IP anycast is naturally supported by BGP. One of the well-known applications of IP anycast is DNS, as all 13 root servers A–M exist in multiple locations.

On the emulator map, type `190` in the search box, you will find out that AS-190 has two networks, but they are disconnected. One network is connected to IX-100, and the other is connected to IX-105. A closer look at these two networks will reveal that these two networks have the identical network prefix `10.190.0.0/24`. The only hosts on these two networks have the identical IP address `10.190.0.100`.

Please find two different hosts in the emulator, so when you ping `10.190.0.100` from these two hosts, the destinations are different (even though the destination IP address is the same). Please set the filter on the map to `icmp` to visualize the packet trace, then look at the BGP routers on the path, and explain why the packets reach two different destinations. Students should read Section 11 of the tutorial before working on this task.

7 Task 5: BGP Prefix Attack

BGP Route Hijacking, also called prefix hijacking, is a typical attack on BGP. In this attack, the attackers' BGP routers announce the IP prefix that are not assigned to them, so the traffic to this IP prefix can get rerouted to the attackers, who can intercept or modify the traffic. Many of the incidents on the Internet are caused by such an "attack", although they are mostly caused by the mis-configured BGP routers. Students should read Section 12 of the tutorial before working on this task.

7.1 Task 5.a. Launching the Prefix Hijacking Attack from AS-161

In this task, we will launch the prefix hijacking attack using a BGP router in AS-161. Our goal is to hijack the IP prefix owned by AS-154. If the attack is successful, all the packets going to AS-154 will be rerouted to AS-161, where they will be dropped. Essentially, we are blackholing AS-154.

In BIRD, the prefixes announced by a BGP router can come from different sources: they can come from the `direct` protocol, i.e., they are obtained from the actual network interface attached to the BGP router. They can also come from the `static` protocol, which contains pre-defined routes. The easiest way for a BGP router to announce a prefix that it does not own is to use the `static` protocol. The following example creates a pre-defined route to `10.130.0.0/16`. See Section 3 of the tutorial for detailed explanation of the `static` protocol. It should be noted that in the filter part of the route, we need to add the route to the `LOCAL_COMM` community; otherwise, the BGP router will not export it to the outside.

```
protocol static {
    ipv4 {
        table t_bgp;
    };
    route 10.130.0.0/16 blackhole { bgp_large_community.add(LOCAL_COMM); };
}
```

7.2 Task 5.b. Fighting Back from AS-154

AS-154 has a detection system. After the attack was launched, it immediately detected the attack. It tried to contact the operators of AS-3, which is the upstream service provider for AS-161, asking them to block the attack. Unfortunately, it was midnight for AS-3's operators, so nobody could be reached. The loss of the service is very significant, so the operators of AS-154 decide to fight back: they want to "steal" back their own network prefixes, without the help of AS-3. Please reconfigure the BGP routers of AS-154, so you can get the traffic back.

7.3 Task 5.c. Fixing the Problem at AS-3

Eventually, the operators of AS-3 are reached. Without knowing whether this is a misconfiguration on the AS-161 side or an intentional attack, AS-3 decides not to cut the peering with AS-161, so the users of AS-161 can still reach the Internet (AS-3 is the only service provider to AS-161). However, AS-3 must stop the propagation of the fake announcement. This can be done by removing the fake routes from its own announcement to its own peers. More specifically, AS-3 can add some code to its filters, so the fake routes can be discarded. See Section 12 of the BGP tutorial to learn how to write filters.

8 Submission

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.

Acknowledgment

This lab was developed with the help of Honghao Zeng, a graduate student in the Department of Electrical Engineering and Computer Science at Syracuse University. The SEED project was funded in part by the grants from the US National Science Foundation and Syracuse University.