OMSCS GEORGIA TECH

BGP Hijacking Attacks

Project 7

CS 6250

FALL 2020

Project 7

Table of Contents

NOTE: Read through the directions completely once or twice before beginning the project!

PROJECT INTRO

In this project you will explore the vulnerability of the AS systems and the BGP protocol.

As you recall from Lesson 4, an autonomous system can be any of the tier ISP providers access (tier 3), regional (tier 2), or global (tier 1). An autonomous system can also be an IXP (where ISP's and CDN's exchange local traffic) or CDN (like Netflix and Google). An AS is a group of routers (including the links among them) that operate under the same administrative authority. The border routers of the ASes use the Border Gateway Protocol (BGP) to exchange routing information with one another.

BGP is used to implement routing policies, which makes it important for ASes to cooperate with other ASes. Even though each AS can make internal decisions, they look to each other for routing information provided by BGP. Security was not in original design of BGP, but with the internet's increasing complexity and size, so is there a need to provide security measures.

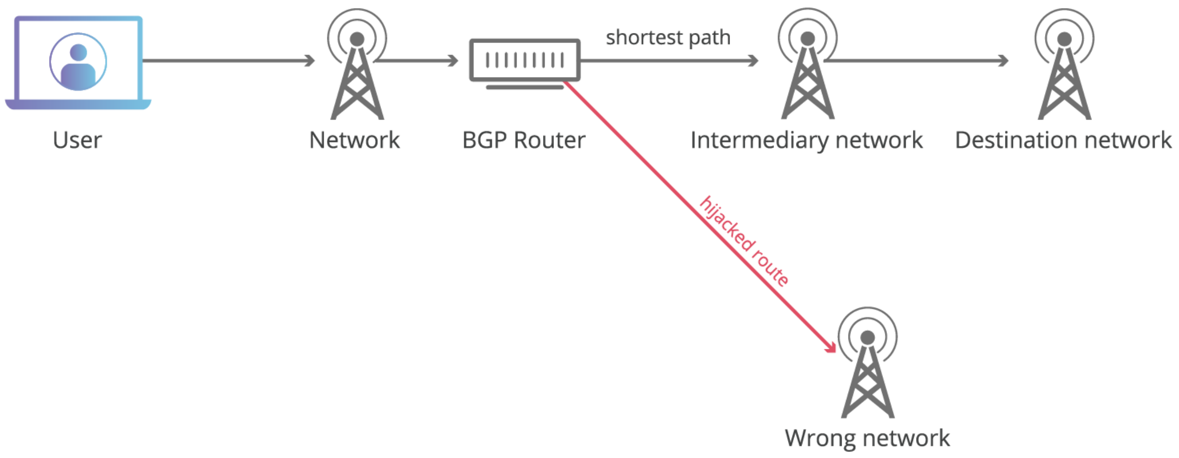
What Is BGP Hijacking?

BGP hijacking occurs when a malicious attackers or rogue AS advertises a false IP prefix that it does not own or control to reroute internet traffic. These vulnerabilities still cause routing disruptions and connectivity issues for individual hosts, networks, and sometimes even entire countries. There have been some notable recent hijacking events that we have linked in the slides for this project. BGP favors a shorter route to save money for the providers or just to decrease number of hops to an IP prefix (more specific route).

For a hijack to be successful it must:

A. Advertise a shouter route of a more specific range of IP addresses that another AS already advertised.

B. Advertise a shorter route to a block of IP addresses. This can only be made by an operator of a AS, or by a bad actor that takes control of an AS.



(Cloudflare)

What happens when BGP is hijacked?

When an attack is made, it re-routes the traffic to malicious sites to steal credentials, drops traffic to cause disruption, and/or increases latency of pages. Even though the victim AS thinks the route is shorter, the reality is it may be a lot longer than the previous advertised routes. The best-case scenario of an attack is where the route will just increase latency and steer traffic to a much longer route, at worst case the attack may re-route to a malicious site to steal confidential information from the users.

BGP hijacking in the real world

We have linked several of the notable cases of real-world hijacking in the project slides. In 2017 Russian hackers re-routed several Visa and MasterCard IP prefixes hijacking the traffic that routed to those addresses:

https://arstechnica.com/information-technology/2017/04/russian-controlled-telecom-hijacks-financial-services-internet-traffic/

Additionally, BGP hijacking has occurred when the Pakistan government re-routed YouTube traffic (link in slides) and hackers attempted to steal crypto currency. Aside from constant monitoring of how Internet traffic is routed, users and networks can do very little to prevent BGP hijacks.

A good resource that explains BGP hijacking is linked below:

https://www.cloudflare.com/learning/security/glossary/bgp-hijacking/

PROJECT GOAL

In this project, using an interactive Mininet demo [1], we will explore some of the vulnerabilities of Border Gateway Protocol (BGP). In particular, we will see how BGP is vulnerable to abuse and manipulation through a class of attacks called BGP hijacking attacks. A malicious Autonomous System (AS) can mount these attacks through false BGP announcements from a rogue AS, causing victim ASes to route their traffic bound for another AS through the malicious AS. This attack succeeds because the false advertisement exploits BGP routing behavior by advertising a shorter path to reach a particular prefix, which causes victim ASes to attempt to use the newly advertised (and seemingly better!) route.

INSTRUCTIONS

Part 1: Background reading, resources and example BGP router configurations

A. Browse this paper as a reference for subsequent tasks and for some important background on Prefix Hijack Attacks.

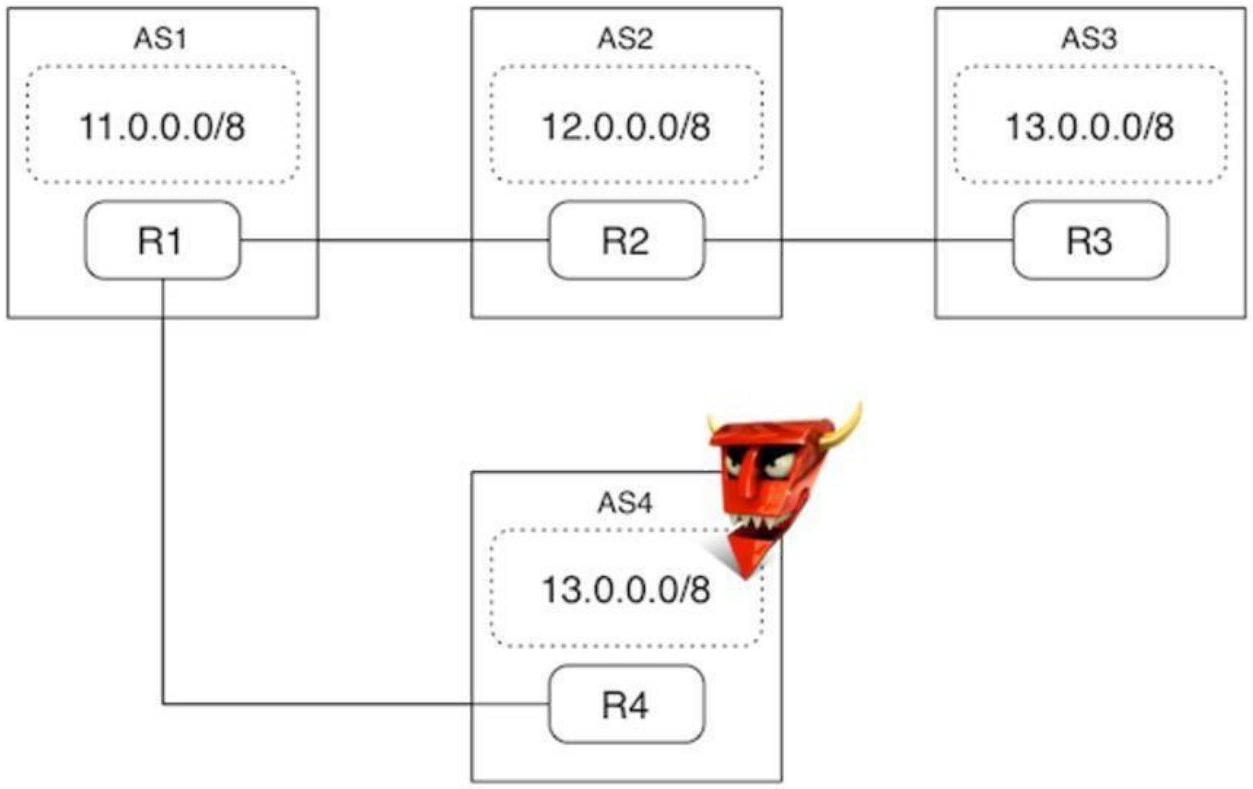
B. Refer to this resource on configuring a BGP router with Quagga.

C. Check out the following example configurations: Example 1 and Example 2

D. The "BGP Bible" is here https://www.cisco.com/c/en/us/td/docs/ios/iproute_bgp/command/reference/irg_book.html

Part 2: Interactive Demonstration using a Mininet Topology and simulated prefixes/paths

The Part 2 demo creates the network topology shown below, consisting of four ASes and their peering relationships. AS4 is the malicious AS that will mount the attack. Once again, we will be simulating this network in Mininet, however there are some important distinctions to make from our previous projects. In this setup, each object is not a single host, but an entire autonomous system. In each AS, a router runs a routing daemon (quagga), communicates with other ASes using BGP (bgpd), and configures its own isolated set of routing entries in the kernel (zebra). Each AS router has multiple IP addresses, to connect to the hosts in the AS and to other routers.



NOTE: In this topology solid lines indicate peering relationships and the dotted boxes indicate the prefix advertised by that AS.

STEPS TO START DEMO

1. Download and unzip the Project files. Modify permissions using the command:

   ```
   sudo chmod -R 777  Project-7
   ```

2. In the Project Directory open a terminal and type the following command:

   ```
   sudo python bgp.py
   ```

   Then enter the password for VM: **`mininet`**
3. After loading the topology, the Mininet CLI should be visible. Keep this terminal open throughout the experiment.
4. Open a second terminal in the Project directory. We will use this terminal to start a remote session with AS1's routing daemon. Type in the following command:

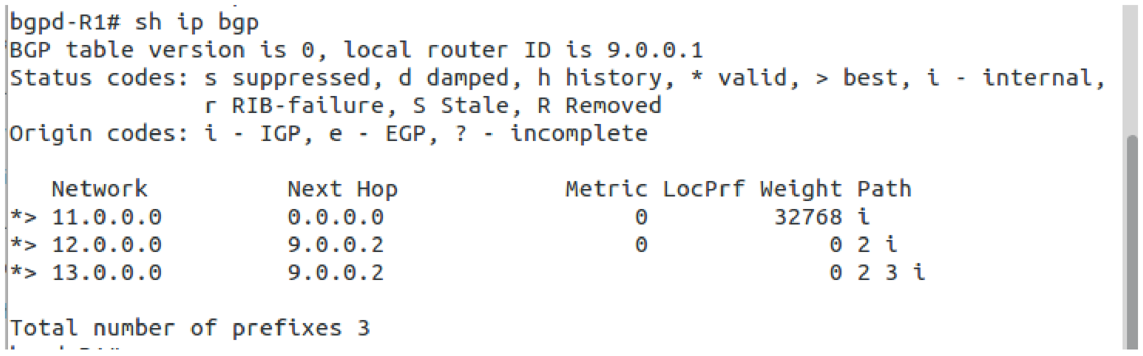
   ```
   ./connect.sh
   ```

   Then enter the password for VM. Enter the password whenever prompted.

5. This script will start quagga, which will require access verification. The password is:
   en
   You will type in "**en**" and press enter (a total of 3 times)
   This will give you access to the administration shell and R1 routing table
   When you get the bgpd-R1# prompt type the following command:

   ```
   sh ip bgp
   ```

6. You should see output very much like the screen grab below. In particular, notice that AS1 has chosen the path via AS2 and AS3 to reach the prefix 13.0.0.0/8. **NOTE: It may take a minute for the routes to settle. Try the command until you see all three routes.**



```
bgpd-R1# sh ip bgp
BGP table version is 0, local router ID is 9.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 11.0.0.0         0.0.0.0                  0         32768 i
*> 12.0.0.0         9.0.0.2                  0             0 2 i
*> 13.0.0.0         9.0.0.2                                0 2 3 i

Total number of prefixes 3
```

7. Next, let us verify that network traffic is traversing this path. While still in the project directory, open a third terminal (keeping all other terminals open). In this terminal you will start a script that continuously makes web requests from a host within AS1 to a web server in AS3. Type in the following:

   ```
   ./website.sh
   ```

8. Leaving all terminals open, open a fourth in the Project directory. Now, we will start a rogue AS (AS4) that will connect directly to AS1 and advertise the same 13.0.0.0/8 prefix. This will allow AS4 to hijack the prefix due to the shorter AS Path Length type the following:

   ```
   ./start_rogue.sh
   ```

9. Return to the third terminal window and observe the continuous web requests. After the BGP routing tables converge on this simple network, you should eventually see the attacker start responding to requests from AS1, rather than AS3.

10. Additionally, return to the second terminal and rerun the command to print the routing table. You may need to repeat the steps to establish the remote session if it closes due to inactivity. You should now see the fraudulent advertisement for the 13.0.0.0/8 prefix in the routing table, in addition to the longer unused path to the legitimate owner.

11. Finally, let's stop the attack by switching to the fourth terminal and using the following command:

    ```
    ./stop_rogue.sh
    ```

12. You should notice a fast reconvergence to the original legitimate route in the third terminal window, which should now be delivering the original traffic. Additionally, you can check the BGP routing table again to see the original path is being traversed.

Part 3: Creating a more complex topology and attack scenario

As demonstrated in Part 2, network virtualization can be very useful in demonstrating and analyzing network attacks that would otherwise require a large amount of physical hardware to accomplish. In Part 3, you are tasked with replicating a different topology and attack scenario to demonstrate the effects of a different instance of a Prefix Hijack Attack.

IMPORTANT NOTE: Build your Part 3 Attack Scenario off the demo files used in Part 2. Per step 2 below, make a backup of your demo files so you can refer to them as you modify the files for Part 3.

STEPS FOR YOUR ATTACK SCENARIO

1. First, locate Figure 2 in the referenced paper. Draw a topology map using any drawing tool of your choice. See Slide 7 of the presentation slides for an example of the level of detail desired in your topology diagram.

   You may hand-draw your topology with pencil and paper and scan or photograph your drawing. All configuration values drawn on the map must be legible! Save your topology diagram in PDF format with the name fig2_topo.pdf. You must use this filename as part of your submission to receive credit for your diagram. **We find that if you do your diagram first, it will make the following steps much easier!**

2. Next, we recommend making a copy of the code provided to you in the Project files (the full demo folder). This will make it easier to complete this project and you will likely find this project to be more approachable if you spend time exploring the demo code and fully understanding how each *part works* rather than immediately trying to edit the code.
3. Next, refer to the referenced paper in Part 1A, and locate Figures 1 and 2

4. Edit the working copy of the demo code you just made to reconstruct the topology in Figures 1 and 2

When complete, you should be able to use the commands from Part 2 to recreate the attack on the new topology you built in the Project-7 directory.

For our purposes, you can assume the following:

- **All links to be bidirectional peering links.**
- Each AS advertises a single prefix: AS1: 11.0.0.0/8, AS2: 12.0.0.0/8, AS3: 13.0.0.0/8, AS4: 14.0.0.0/8, AS5: 15.0.0.0/8, AS6: 11.0.0.0/8 (Note: We highly recommend using these prefix values in your configuration to simplify grading and for consistency in communication and discussion in Piazza. However, you may use any valid prefix values in your configuration.)
- The number of hosts in each AS is the same as in the provided code (the demo).

5. Do not change passwords in zebra and conf files. If you change the passwords, the auto-grader will fail resulting in 0 for the assignment. All passwords need to follow the demo and be ‘en’

6. Continue to adapt the code in your Project-7 directory to simulate this hijack scenario. When complete, you should be able to use the commands from Part 2 to start a Rogue AS and demonstrate a similar change in routing table information as was shown in Part 2 and see the screen printout (website.sh) as in demo. If this is not seen will result in points lost (see rubric for breakdown of points)

7. Finally, follow the directions in the **What To Turn In** section carefully. You must include all of the files necessary to run your demo in the directory - do **NOT** assume that we will provide any of the files necessary to run your demonstration for grading purposes. Include your fig2_topo.pdf file in your directory.

Part 3 Configuration Debugging Tips

- When viewing the BGP Tables note the “Status codes”. Give your topology enough time to converge before recreating the hijack simulation portion. It may take a minute or so for your topology to fully converge. You may continue to check the BGP Tables to determine whether the topology has converged
- The order that you set up your peering links using addLink() matters. In previous projects, we manually selected which port on the switch to use. There is an optional parameter to the addLink() call which allows you to specify which switch port to use. In this project, you will not use those options. Therefore, the order of the links matters.
- Some of the commands in the boilerplate code may not be necessary to complete Part 3. Some of it is there just so that you know it exists.
- Check for more descriptive errors in the /logs directory. See the zebra files for the location of additional log files.
- Run "links" on the Mininet CLI terminal to see if all links are connected and OK.
- Run "net" on the Mininet CLI terminal to see if your ethernet links are connected as you expect.
- Run "ifconfig -a" on all routers and hosts to ensure that all IP addresses are assigned correctly.
- Run "sh ip bgp" and "sh ip bgp summary" on all routers.
- The command **`pingall`** may not work and that is fine.
- The website.sh may sometimes hang intermittently. If this happens restart the simulation. We are aware of this issue, and we keep this in mind as we grade your submission. You will not lose points if website.sh hangs so long as we are eventually able to run the simulation.
- Watch the Intro presentation and read through the additional debugging tips on the intro slides.

Part 4 (Optional Extra Credit) – Design and implement a countermeasure to the attack from Part 3

This part of the project is optional, but it is worth extra credit if you complete it. Your task here is to design and implement a countermeasure to the attack demonstrated in Part 3. Start by creating a complete copy of the code you produced in Part 3 in a fresh working directory. Be sure to use the `-r` argument to `cp` to copy the subdirectories.

Next, design and implement a countermeasure to the attack from Part 3. When complete, you should be able to use the commands from Part 2 to launch the simulation and start a Rogue AS that mounts a Prefix Hijack attack as in Part 3. In this case, the attack should fail. You should be able to observe the victim AS routing table maintain (or revert back to) it's original state before the attack commences.

The paper referenced in Part 1A describes some example countermeasures, and you can implement/ modify them as required for this project. You are also free to explore other methods; this part is open ended. The first stipulation is that the solution you implement be applicable in the general case, **meaning it is not a hard-coded defense**. Your defense should work regardless of which AS is attacked, which AS mounts the attack, and what prefix is targeted. The second is that the countermeasure must be demonstrable on the course VM. It is permissible to use additional libraries in the development of your countermeasure; however, they must be documented so the grader can install them prior to grading your code.

As was done in Part 3, create a `zip` file named gtlogin-Part4.zip (substitute your gtlogin) containing your entire countermeasure demonstration. **You must include all of the files necessary to run your demo** in an empty directory - do NOT assume that we will provide any of the files necessary to run your demonstration for grading purposes. Additionally, you should make sure you provide a supplementary document (PDF format) named gtlogin-Countermeasure.pdf (see What To Turn In section). This document should provide the following and submitted separately in: Project 7: Extra Credit (.pdf submission):

1. A brief summary of how your solution counters the attack
2. A list of files you modified from Part 3 or created to implement the countermeasure
3. A brief description of what is changed in each file, (or the purpose of newly created files) including how it functions as a part of the larger system.
4. Instructions for demonstrating the countermeasure, including instructions for installing required software / libraries.
5. A brief closing containing any additional information the grader may need to reproduce your countermeasure and contact information (if different than your GT student email address) in case the grader has questions.

What to Turn In:

PART 3

For this project you need to turn in all the files that are necessary for your code to run. Please name the zip file based on your GA Tech username. **Be sure to use the -r option so your conf and logs subdirectories are part of the zip file!** Use the following command to zip for Part 3:

`zip -r gtlogin_Part3.zip Project-7` (replace gtlogin with your GT login)

You need to make sure the pdf file `fig2_topo.pdf` is present inside your `Project-7` directory along with all files and directories to run the attack scenario you created. Run the above zip command when above your project folder. **Zip the directory** `Project-7`**.** All the files and folders needed must be in the `Project-7` folder – **zip the files in the VM using the Linux command, don't zip the files in your host operating system (i.e. No MAC_OSX directory)**

<u>**Failure to have proper zip organization will result in 10 points loss!**</u>

<u>Improper zipping (not following directions) will cause problems with the auto grader so please follow the directions!</u>

gtlogin_Part3.zip should be submitted in the main assignment on canvas which is named:

Project 7 – Network Security and BGP Hijacking Attacks

As with all projects, we highly recommend after submitting that you **re-download your submission from Canvas** to check that it uploaded correctly and runs properly in the class VM.

OPTIONAL PART 4 EXTRA CREDIT

If you are going for the extra-credit part perform a similar operation for Part 4 with all files to run your code inside your Project-7 directory and run the following command:

```
zip -r gtlogin_Part4.zip <Part 4 directory name>
```

gtlogin_Part4.zip should be submitted in the extra-credit assignment on canvas which is named:

Project 7: Extra Credit (.zip submission)

The Pdf write up for part 4 should be submitted in the appropriate part-4 pdf submission in canvas **named Project 7: Extra Credit (.pdf submission**) Submit **gtlogin_Countermeasure.pdf** there. Do not zip the PDF.

Note on Creating PDFs and Turnitin Plagiarism Check

We recommend you use Microsoft Word to generate the PDF you will turn in. We use Turnitin to check for plagiarism. Turnitin needs to be able to see a PDF file with selectable text. DO NOT create a PDF with all images -- you should be able to copy the text portion from your PDF and paste it into other applications. Submitting a PDF made entirely of images may result in a 0 grade for the PDF.

Where to Submit and Double-checking Your Submission

All the submissions must be uploaded into their respective Canvas submission page. There are three submission areas:

- **Project 7: Network Security and BGP Hijacking Attacks** – This is where your part 3 zip file is submitted
- **Project 7: Extra Credit (.pdf submission)** -- this is your extra credit Part 4 PDF file
- **Project 7: Extra Credit (.zip submission)** -- this is your Part 4 zip file

Again, as with all submissions, we highly suggest you **re-download your submissions from Canvas** and **double check** that they work in the VM, that all files are present and that it is the correct version. We have seen submissions with missing files and or incorrect versions – unfortunately, we cannot accept these missing items after the due date!

What you can and cannot share

While discussion of the project in general is always permitted on Piazza, you are not permitted to share your code generated for Part 3 or Part 4. You may quote snippets of the unmodified skeleton code provided to you when discussing the project.

- You may **not** share your topology diagram you created in Part 3 Step 5. (private post to instructors is always OK)
- You may not share your IP addresses publicly (private post to instructors is always OK)
- Sharing of completed code pseudo code is ok but if in doubt please private post to instructors
- **There is no discussion allowed of Part 4 extra credit on Piazza or Slack!**

Rubric (out of 150 points)

20 pts	Submission	For turning in all the correct demo files with the correct names, and significant effort has been made towards completing the project. 10 for submission and effort 10 for following zip directions
5 pts	Fig 2 Topo Diagram	For turning in the correctly named Topology diagram file: **fig2_topo.pdf** Please use legible configuration values!
125 pts	Attack Demo	For accurately recreating the topology, links, router configuration, and attack per the instructions. **Partial credit** is available for this rubric item as follows: 40 points for accurately recreating the topology, links, router configuration 40 points for seeing default message when you run website.sh 40 points for seeing attack message after running start_rogue.sh 5 points for seeing default message after stop_rogue.sh is run
50 pts	Extra Credit	For correctly designing and implementing a **countermeasure** to the attack from Part 3. Submissions MUST include both the code and documentation - extra credit will not be considered for code without accompanying documentation. ***Some partial credit*** may be provided for thorough gtlogin_Countermeasure.pdf identifying a viable solution without accompanying code or with non-working code if the documentation acknowledges the lack of code or the failing code. Up to 5 points for a paper with no code Up to 30 points for a working hard coded example and paper Up to 50 points for working dynamic solution with documentation and paper

[1] This Project inspired by a Mininet Demo originally presented at SIGCOMM 2014.

Bibliography

Cloudflare. (n.d.). Retrieved 2020, from cloudflare.com: https://www.cloudflare.com/learning/security/glossary/bgp-hijacking/