Federated Learning of BGP Prefix Hijacking

Abie Safdie
University of Oregon
CS 630 - Distributed Systems

1. Introduction of Problem

The internet is divided into thousands of networks named Autonomous Systems. These Autonomous Systems (ASes) use the Border Gateway Protocol (BGP) to exchange routing information with each other. BGP provides each router the information to obtain IP prefix reachability information from neighboring ASes as well as the route to take to reach the desired destination. BGP works by establishing semi-permanent TCP connections, known as a BGP Connection, to send routing information between ASes. BGP, however, is susceptible to BGP prefix hijacking. This occurs when an AS announces it contains an IP prefix that it does not actually contain, therefore it routes internet traffic to a malicious AS. Another case of BGP hijacking is when a corrupt AS announces it can route traffic through a "better" path, forcing the traffic to go through malicious routers. These susceptibilities in BGP raise serious security concerns because internet traffic can be intercepted and manipulated by malicious actors.

Research Question: Do these malicious BGP announcements follow a similar signature making them reliability detected via machine learning algorithms? Additionally, will conducting machine learning in a federated manner, where each AS learns of its BGP prefix hijacking announcements independently and then shares their discoveries to a central model improve efficiency and reliability?

2. Current State-of-the-Art solutions

There are currently a few methods employed to prevent BGP prefix hijacking. First, the deployment of BGPsec. In standard BGP, the attribute AS-PATH contains the list of ASes that the advertisement has passed through. In BGPsec, however, the AS-PATH attribute is replaced by the SEC-PATH attribute, which cryptographically verifies the path that the ASes advertise. This attempts to prevent false announcements. However, the overhead associated with BGPsec is severe. Each path update requires cryptographic key verification, which hinders the performance of the system. Furthermore, for BGPsec to work more efficiently it requires wide-spread adoption. Another method to prevent BGP hijacking is BGP monitoring. This is the monitoring of path updates to look for suspicious behavior in real-time. Lastly, similar to BGPsec, is Route Origin Validation (ROV). This method cryptographically validates the origin of the BGP

announcement. Again, this creates a similar overhead. Ultimately, these solutions attempt to mitigate the effects of BGP prefix hijacking.

3. My Solution

My solution aims to apply federated learning to recognize signatures in malicious BGP announcements. By restricting the machine learning to each individual AS, each network can share information to a central model. Additionally, by distributing the learning, each AS receives data that is most relevant to their own system. I will attempt to accomplish this by using machine learning libraries in Python, such as TensorFlow, PyTorch, and/or SciKit-Learn. For the BGP data, I intend to use the RIPE and/or Routeview databases. I believe there will be a few significant milestones to achieve when developing this project. First, creating a sophisticated model that interprets and understands the BGP data it receives. Second, applying that understanding of BGP to train a wide variety of models, each one specific to an AS. And third, gathering data on how efficient the models are at recognizing prefix hijacking.

Due to the complexities of BGP, there is not a single solution that has solved BGP hijacking at its source. However, by applying machine learning, there could be a statistically significant improvement in preventing malicious announcements before they officially get advertised. Additionally, machine learning algorithms could provide unique insights into what constitutes a suspicious or malicious announcement.

4. List of Deliverables

- Complete Project Report
- Final version of Project Presentation Slides
- Zip file of all source code/manuals/databases
- Other deliverables will be added if needed

5. Timeline of Tasks and Deliverables

Task	Date
Project Proposal	**Week2 - Week 3**
Research/Review BGP	**Week 2 - As necessary**
Learn and Apply Python Machine Learning Libraries on BGP	**Week 3 - Week 7**
Gather BGP Hijacking Data	**Week 3 - Week 7**

Develop Models and Distribute (Federate) the learning	**Week 4 - Week 7**
Project Report	**Week 6 - Week 10**
Project Slides	**Week 8 - Week 10**
Zip File of All Source Code/Manuals/etc…	**Week 9 - Week 10**