BGPNetSim: Dockerized Border Gateway Protocol Routing Simulation with Quagga

I Wayan Budi Sentana[1],
I Made Ari Dwi Suta Atmaja[2],
I Nyoman Gede Arya Astawa[3],
and Ni Gusti Ayu Putu Harry Saptarini[4]

[1,2,3,4] Information Technology Department, Politeknik Negeri Bali, Bali, Indonesia
budisentana@pnb.ac.id

Abstract. This research introduces BGPNetSim, a Dockerized simulation framework designed to enhance the study and experimentation of Border Gateway Protocol (BGP) routing. Leveraging Docker containers and Quagga, BGPNetSim creates isolated and reproducible network environments, making it an essential tool for network researchers, educators, and practitioners. The framework integrates key components, including a Graph Generator, Network Configuration Manager, Container Actuator, Configuration Manager, and Docker Environment, to automate the creation, configuration, and management of complex BGP topologies. The effectiveness of BGPNetSim was evaluated through extensive testing in research focused on Blockjack, a blockchain-based system aimed at preventing IP prefix hijacking. The simulations encompassed various network topologies and attack scenarios, including Single Path, Multiple Path, and Random Attacks. Results revealed that Blockjack handled structured attacks with average pre-pending times of 11.445 seconds for Single Path, 24.894 seconds for Multiple Path, and 24.729 seconds for Random Attacks, while neutralization times were 0.146 seconds, 0.145 seconds, and 0.411 seconds, respectively. The findings indicate Blockjack's robust performance in neutralizing hijacking attempts, although variability in chaotic environments was observed. Overall, BGPNetSim has demonstrated its capability as a valuable tool for advancing BGP routing security. By providing a controlled yet flexible simulation environment, it enables comprehensive testing and evaluation of BGP security mechanisms. The framework's ability to accurately measure and analyze key performance metrics contributes significantly to developing more effective and secure BGP protocols.

Keywords: Network Simulator, Border Gateway Protocol, Quagga Router, Docker Container, Python

1 Introduction

In the rapidly evolving field of computer networking, the ability to simulate network protocols accurately is crucial for both educational and research purposes. Simulations provide a safe and controlled environment where network administrators, engineers,

A. A. N. G. Sapteka et al. (eds.), *Proceedings of the International Conference on Sustainable Green Tourism Applied Science - Engineering Applied Science 2024 (ICoSTAS-EAS 2024)*, Advances in Engineering Research 249,
https://doi.org/10.2991/978-94-6463-587-4_28

and researchers can test and evaluate the behavior of protocols under various conditions without the need for extensive physical infrastructure (Cho et al., 2019). The Border Gateway Protocol (BGP) is the backbone of the internet's global routing system, facilitating the exchange of routing information between autonomous systems (AS) (Kent et al., 2000). This critical role underscores the importance of understanding and experimenting with BGP to ensure the stability and performance of network operations. However, existing BGP simulation tools often fall short, being either too simplistic and lacking real-world applicability or too complex and resource-intensive to deploy effectively (Hameed, 2017). Several studies have highlighted the limitations of current simulation tools and the need for more robust solutions. For example, recent research by (Sentana et al., 2020; Sentana et al., 2021) has demonstrated the use of simulation tools to test concepts involving BGP, emphasizing the need for improved simulation environments.

This research introduces BGPNetSim, a dockerized routing simulation environment utilizing Quagga, an open-source software suite that implements various routing protocols, including BGP (Saad et al., 2019). BGPNetSim is designed as an experimental and learning tool for BGP routing configuration, leveraging the isolation capabilities of Docker containers and the flexibility of Quagga to provide a realistic and scalable simulation platform(McGlynn et al., 2019). The development of BGPNetSim incorporates the use of Python programming language and shell scripts to facilitate the system development process. Python automates several aspects of configuration and analysis in the simulation, while shell scripts assist in managing and configuring the simulator. Additionally, integration with *Telnet* allows users to interact with routers within the simulation, performing configuration, testing, and monitoring of network devices directly. The method involves the installation and configuration of Docker and its integration with Quagga to compose a simulation environment for BGP routing (Iamartino et al., 2015). This setup enables the replication of routing scenarios without impacting the production network, providing a secure and isolated environment for network practitioners to experiment and learn about BGP routing configuration and behavior. This simulator is also important for researchers to simulate IP prefix hijacking in BGP, a critical security concern that can lead to significant disruptions in network traffic (Sermpezis et al., 2018). By utilizing the Docker-Quagga approach, BGPNetSim offers high flexibility and accessibility, supporting the understanding of BGP routing concepts and their implementation. The simulator is poised to be an effective tool for learning and skill development in BGP routing configuration, ultimately contributing to enhancing the understanding and capabilities of network practitioners in managing and optimizing BGP routing.

The subsequent sections of this paper will provide a detailed overview of BGP and Quagga, the design and implementation of BGPNetSim, a series of experiments to validate the simulation environment, and a discussion of the results and potential future work.

2 System Architecture

BGPNetSim is a Dockerized simulation framework designed to facilitate the study and experimentation of Border Gateway Protocol (BGP) routing using Quagga. The framework leverages containerization to create isolated and reproducible network environments, making it an ideal tool for network researchers, educators, and practitioners. The core components of the framework include a Graph Generator, Network Configuration Manager, Container Actuator, Configuration Manager, and Docker Environment. These components work together to automate the creation, configuration, and management of complex BGP network topologies.

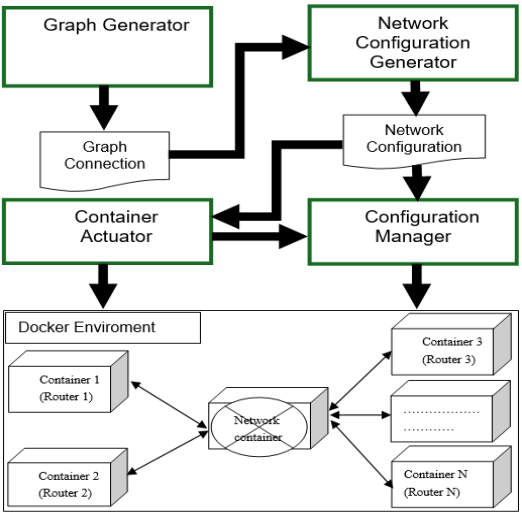

Figure 1. BGPNetSim architecture

2.1 Graph Generator

The Graph Generator is responsible for creating detailed network topologies that are used as reference for all subsequent simulations and configurations. Developed using Python, the Graph Generator leverages the NetworkX library to construct graph-based representations of network structures, defining how different nodes (routers) are interconnected. The Graph Generator can create various types of network topologies, including random, binary tree, and hierarchical structures. This versatility allows users to simulate a wide range of network scenarios and understand BGP behavior under different conditions. For instance, in a binary tree topology, the network follows a hierarchical structure where each node has two child nodes, representing a simple yet structured network design. On the other hand, a random network topology connects nodes more arbitrarily, reflecting the real-world network's unpredictability.

The connectivity level within these topologies is set to 20%, ensuring that each node connects to at least 20% of the other nodes in the network. This setting creates a robust

and resilient network structure that can better mimic real-world scenarios. The output of the Graph Generator is a text file named "Graph Connection", which details the network's topology. This file serves as a reference for the subsequent Network Configuration Manager component, providing essential data on how the nodes are interconnected.

2.2 Network Configuration Manager

The Network Configuration Manager component is responsible for translating the abstract network topology generated by the Graph Generator into concrete network configurations for each router in the topology. Developed using Python and shell scripts, the Network Configuration Manager ensures the integration with the rest of the framework. The primary function of the Network Configuration Manager is to parse the "Graph Connection" file to determine router connections and BGP peering relationships. It generates BGP configuration files for each router using Quagga's configuration syntax, ensuring that all necessary parameters such as IP addresses, Autonomous System (AS) numbers, and BGP policies are correctly specified. This detailed configuration is crucial for accurate network simulation, as it defines how routers will communicate and share routing information.

Once the configurations are generated, they are stored in a text file. This file contains all the necessary configuration parameters and will be used by the Container Actuator for container deployment. The nature of this component ensures that the resulting network is a representation of the specified topology, with all routers correctly configured to engage in BGP routing.

2.3 Container Actuator

The Container Actuator component automates the creation and management of Docker containers that emulate the routers specified in the network topology. Developed using Python and shell scripts, the Container Actuator ensures that each container is properly configured and networked to reflect the desired topology.

The Container Actuator automates the deployment process by creating Docker containers based on the configurations provided by the Network Configuration Manager. Each container is configured with details such as the router name, IP address, AS number, and memory size. These containers run the Quagga router software, which enables them to perform BGP routing as specified in the configuration files. In addition to deployment, the Container Actuator is responsible for the lifecycle management of the containers. This includes monitoring the status of the containers, starting, stopping, and restarting them as needed. By automating these tasks, the Container Actuator ensures consistency and reduces the potential for human error.

2.4 Configuration Manager

The Configuration Manager oversees the overall configuration process, ensuring that all components of the BGPNetSim framework work together seamlessly. This

component coordinates the activities of the Graph Generator, Network Configuration Manager, and Container Actuator, validating configuration files and network settings before deployment. One of the Configuration Manager's critical tasks is to set up router connectivity using the information from the network configuration files. This includes creating pairs among BGP speaker routers, which are essential for establishing BGP peering relationships. To automate this process, the Configuration Manager uses a specialized library known as Expect. Expect is an interactive shell script command line tool that automates the configuration of BGP connections, eliminating the need for manual setup and significantly reducing the potential for human error.

The Configuration Manager also provides interfaces for users to customize configuration parameters and network properties. This flexibility allows users to tailor the network simulation to their specific needs, adjusting parameters such as IP addressing schemes, AS numbers, and BGP policies. By offering this level of customization, the Configuration Manager ensures that BGPNetSim can accommodate a wide range of research and educational scenarios.

2.5 Docker Environment

The Docker Environment provides the underlying platform for containerization and network emulation. This component is crucial for maintaining isolated and reproducible network environments, which are essential for accurate and reliable network simulations. Docker ensures that each router runs in an isolated container, providing a controlled environment for simulation. This isolation prevents interference between containers, ensuring that the behavior of each router is accurately represented. The Docker Environment also manages Docker networks to accurately reflect the physical connections between routers, ensuring that the network topology is faithfully reproduced within the simulation environment.

One of the key advantages of using Docker is its scalability. The Docker Environment supports scaling, allowing multiple containers to run on a single host or across a cluster of hosts. This capability facilitates larger and more complex simulations, enabling researchers and educators to study BGP behavior in networks of varying sizes and complexities. By utilizing Docker, the BGPNetSim framework creates isolated environments for each router, ensuring reproducibility and isolation. This approach allows users to replicate their experiments and simulations with high fidelity, ensuring that their results are consistent and reliable.

3 Testing Scenario and Result

The BGPNetSim framework has undergone extensive testing to validate its effectiveness and robustness in simulating BGP routing environments. Notably, this framework has been tested in research conducted by (Sentana et al., 2020; Sentana et al., 2021). In this research, the framework was utilized by Blockjack, a blockchain-based system designed to prevent IP prefix hijacking in the Border Gateway Protocol. The primary goal of this research was to evaluate the ability of Blockjack to detect and

prevent IP prefix hijacking, a significant security threat in BGP networks. IP prefix hijacking occurs when an unauthorized entity advertises IP address spaces it does not own, causing traffic to be misrouted. Such incidents can lead to significant security breaches, including data interception and denial of service. BGPNetSim played a pivotal role in evaluating BlockJack's resiliency against prefix hijacking attacks. As a versatile framework designed for simulating BGP environments, BGPNetSim facilitated comprehensive testing under controlled and realistic scenarios. This allowed for detailed observation and analysis of BlockJack's performance, providing critical insights into its effectiveness and efficiency.

For the resiliency evaluation, BGPNetSim was deployed in a high-performance computing environment leveraging Quagga and Docker. The simulation environment was set up on a cluster server with 4-core CPU units, 128 GB of memory, and a 500 GB hard drive running Ubuntu 18.04 LTS. This robust setup ensured that the simulations could handle complex network topologies and a significant number of routers, mimicking real-world conditions. The powerful computing environment was critical for running resource-intensive simulations and maintaining the integrity of the experimental data.

3.1 Network Topologies and Attack Scenario

BGPNetSim enabled the creation of various network topologies with different numbers of routers, ranging from 20 to 60. These topologies were used to simulate three distinct attack scenarios: Single Path Attack, Multiple Path Attack, and Random Attack. Each scenario tested BlockJack's resilience under different conditions, providing a comprehensive evaluation of its capabilities.

Single Path Attack Scenario: In this scenario, a binary tree-like network topology was created, with the dispatcher router positioned at the root. Adversarial prefixes were used to hijack prefixes announced by routers at the leaf nodes, creating single paths to the BlockJack router. This scenario aimed to evaluate BlockJack's ability to neutralize attacks originating from a straightforward, single-path route. By testing with a structured topology, researchers could observe how BlockJack handles clear and predictable attack vectors. The setup involved five adversarial prefixes used by routers located in the farthest branches to hijack the original prefixes. Five sets of experiments were conducted for various router numbers (20 to 60), each set restarted to ensure the blockchain ledger only contained the genesis block at the start of each trial.

Multiple Path Attack Scenario: This scenario built upon the Single Path Attack by modifying the binary tree topology to include BGP peering at each branch level. This modification caused each announced prefix to have multiple paths when reaching the BlockJack router, simulating a more complex and dynamic routing environment. This scenario tested BlockJack's resilience in handling routing path changes that occur during BGP prefix hijacking. By introducing multiple paths, the experiment assessed BlockJack's capability to identify and neutralize hijacking attempts amid more intricate network interactions. Similar to the Single Path Attack, a total of 25 experiments were

conducted across varying router setups (20 to 60), with the blockchain network restarted for each set to ensure consistency.

Random Attack Scenario: For the Random Attack Scenario, several random network topologies were created with varying numbers of routers (20 to 60). The connectivity level was set to 25%, indicating that each node had a probability of connecting to 25% of the total nodes in the network. For experiments with 20 routers, the connectivity level was set to 50% to avoid unconnected nodes. The dispatcher and adversarial prefixes were randomly assigned, simulating a highly dynamic and unpredictable BGP environment. This scenario aimed to evaluate BlockJack's resilience in an environment where the attack vectors and network topology was less structured and more chaotic. Each experiment ran for 10 minutes, and the results were recorded over five sets of experiments for different router numbers. The randomness of the dispatcher and adversarial prefix assignments meant that some experiments had to be discarded and rerun if no hijacking effect was observed in the routing table where the dispatcher was placed.

3.2 Measurement and Evaluation

BGPNetSim facilitated the precise measurement of prefix hijacking neutralization in two stages: Prefix Prepending and Neutralization.

Prefix Prepending: This stage involved adding an ASN to the AS-path parameter in the BGP table for each AS passed by a prefix. The time taken for this process was recorded as the prepending time. Prepending time is critical as it reflects the duration taken by an adversarial prefix to disrupt the original prefix's path, thereby highlighting the immediacy of the hijacking threat.

Neutralization: This stage involved BlockJack detecting, verifying, and sending filter commands to neutralize the hijacking. The time taken for this process was recorded as the neutralization time. Neutralization time is essential as it indicates the speed at which BlockJack can respond to and mitigate the hijacking threat, showcasing its effectiveness in maintaining the integrity of the network.

By measuring the prepending and neutralization times, BGPNetSim provided critical data on the duration of BGP hijacking attacks and BlockJack's efficiency in neutralizing them. The framework enabled the collection of detailed performance metrics under each scenario, facilitating a thorough analysis of BlockJack's capabilities.

The results from BGPNetSim's simulations revealed key insights into BlockJack's performance. The average prepending time increased gradually with the addition of routers in the Single Path and Multiple Path Attack Scenarios, while it fluctuated in the Random Attack Scenario. This indicated that BlockJack effectively handled structured attacks but faced variability in more chaotic environments. The lowest prepending times for Single Path, Multiple Path, and Random Attack Scenarios were 28.068, 41.855, and 52.101 seconds, respectively, recorded during experiments with 20 routers.

The average prepending time for all experimental sets was 74.527 seconds for Single Path, 80.9088 seconds for Multiple Path, and 54.9572 seconds for Random Attack Scenarios.

Neutralization times were generally consistent across scenarios, except for the Random Attack Scenario, which recorded higher neutralization times due to the greater number of neighbors and attacks. This highlighted BlockJack's robustness but also indicated areas for potential improvement in highly dynamic networks. The average neutralization time was 0.1516 seconds for Single Path, 0.2362 seconds for Multiple Path, and 1.0484 seconds for Random Attack Scenarios.

BGPNetSim demonstrated that BlockJack could effectively scale to handle increasing numbers of routers and complex network topologies. The standard deviation of prepending and neutralization times remained relatively stable in structured scenarios, affirming BlockJack's reliability. The standard deviation for prepending and neutralization times in the Random Attack Scenario was recorded at 20.3712 seconds and 0.8998 seconds, respectively, indicating greater variability due to the unpredictable nature of the network setup.

The detailed experimental results, as presented in Table 1, underscored the efficacy of BGPNetSim in supporting BGP security research. By providing a controlled yet flexible simulation environment, BGPNetSim enabled the rigorous testing of BlockJack, contributing to the advancement of secure BGP protocol implementations. The experiment results for BlockJack scalability tests are detailed, with a specific focus on the role of BGPNetSim in the testing process. The tests evaluated the performance and resilience of BlockJack against prefix hijacking attacks within different network topo-logies generated using BGPNetSim.

The table presents several metrics: the number of routers in the network, the number of adversarial prefixes sent, the average number of prefixes impacted by hijacking, the average number of attacks received by the BlockJack router, the average prepend time for adversarial prefixes announcement, and the average neutralization time for each hijacking. These metrics were collected across different scenarios, with router counts ranging from 20 to 60.

Table 1. Experiment result for Blockjack scalability test; Router: Number of router; Sent: Number of adversarial prefixes sent to the network; Impact: Average number of prefixes impacted by the hijacking in the Blockjack router; Attack: Average number of attack received by Blockjack router; Prepend: Average Prepend time for adversarial prefixes announcement, Neutralize: Average Neutralization time for each hijacking

Router#	# of Attack	Impact (second)	Attack	Prepending	Neutralization
20	5	2.2	11	11.445	0.146
30	5	1.8	14.8	24.894	0.145
40	5	1.4	6.2	19.131	0.214
50	5	1.8	12.4	13.959	0.141
60	5	1.8	6	24.729	0.411

4 Conclusion

In this paper, we present BGPNetSim, a Docker-based Border Gateway Protocol (BGP) simulation framework using Quagga. BGPNetSim enables the creation, configuration, and management of virtual network topologies through key components: the Graph Generator, Network Configuration Manager, Container Actuator, Configuration Manager, and Docker Environment. The Graph Generator, built with Python and NetworkX, designs network topologies, which are then configured and deployed in Docker containers running Quagga.

BGPNetSim was tested to evaluate BlockJack, a blockchain system for preventing IP prefix hijacking. The tests showed BGPNetSim's effectiveness in simulating various network scenarios and its reliability in assessing BlockJack's performance under different conditions. Results demonstrated BlockJack's success in mitigating prefix attacks and highlighted BGPNetSim's scalability and efficiency.

In conclusion, BGPNetSim is a valuable tool for researchers and network engineers, offering a modular and scalable platform for BGP simulations. Future improvements could include support for additional protocols and optimization for larger networks. The successful use of BGPNetSim in testing BlockJack demonstrates its potential as a standard tool for BGP routing simulations.

Acknowledgment

We would like to express our sincere gratitude to Politeknik Negeri Bali for their generous support and funding provided through the Applied Research Schema.

References

Cho, S., Fontugne, R., Cho, K., Dainotti, A., & Gill, P. (2019). BGP hijacking classification. *TMA 2019 - Proceedings of the 3rd Network Traffic Measurement and Analysis Conference*, 25–32. https://doi.org/10.23919/TMA.2019.8784511.

Hameed, H. (2017). *Detecting IP prefix hijack events using BGP activity and AS connectivity analysis*. Plymouth University.

Iamartino, D., Pelsser, C., & Bush, R. (2015). Measuring BGP route origin registration and validation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8995*, 28–40. https://doi.org/10.1007/978-3-319-15509-8_3.

Kent, S., Lynn, C., & Seo, K. (2000). Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, *18*(4), 582–592. https://doi.org/10.1109/49.839934.

McGlynn, K., Acharya, H. B., & Kwon, M. (2019). Detecting BGP route anomalies with Deep Learning. *INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2019*, 1039–1040. https://doi.org/10.1109/INFCOMW.2019.8845138.

Saad, M., Anwar, A., Ahmad, A., Alasmary, H., Yuksel, M., & Mohaisen, A. (2019). Routechain: towards blockchain-based secure and efficient BGP routing. *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, *1*(1), 210–218. https://doi.org/10.1109/BLOC.2019.8751229.

Sentana, I. W. B., Ikram, M., & Ali Kaafar, M. A. D. (2020). BlockJack: blocking IP prefix hijacker in inter-domain routing. In *CoNEXT Student Workshop 2020 - Proceedings of the 2020 Student Workshop, Part of CoNEXT 2020*. https://doi.org/10.1145/3426746.3434052.

Sentana, I. W. B., Ikram, M., & Kaafar, D. (2021). BlockJack: Towards improved prevention of IP prefix hijacking attacks in inter-domain routing via blockchain. *Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021*. https://doi.org/10.5220/0010521106740679.

Sermpezis, P., Kotronis, V., Gigis, P., Dimitropoulos, X., Cicalese, D., King, A., & Dainotti, A. (2018). ARTEMIS: Neutralizing BGP hijacking within a minute. *IEEE/ACM Transactions on Networking*, *26*(6), 2471–2486. https://doi.org/10.1109/TNET.2018.2869798.