Ain’t How You Deploy: An Analysis of BGP Security Policies Performance Against Various Attack Scenarios with Differing Deployment Strategies[1]

Seth Barrett
School of Computer and Cyber Sciences
Augusta University
Augusta, USA
sebarrett@augusta.edu

Calvin Idom[2]
College of Engineering and Science
Louisiana Tech University
Ruston, USA

German Zavala Villafuerte
School of Computer Science
Texas A&M University - San Antonio
San Antonio, USA
gzava010@jaguar.tamu.edu

Andrew Byers
School of Computing and Augmented Intelligence
Arizona State University
Tempe, USA
adbyers1@asu.edu

Berk Gulmezoglu
Electrical and Computer Engineering
Iowa State University
Ames, USA
bgulmez@iastate.edu

***Abstract*—This paper investigates the performance of various Border Gateway Protocol (BGP) security policies against multiple attack scenarios using different deployment strategies. Through extensive simulations, we evaluate the effectiveness of defensive mechanisms such as Root Origin Validation (ROV), Autonomous System Provider Authorization (ASPA), and PeerROV across distinct AS deployment types. Our findings reveal critical insights into the strengths and limitations of current BGP security measures, providing guidance for future policy development and implementation.**

***Index Terms*—BGP, Security, Simulation, Defensive Strategies**

I. Introduction

The Border Gateway Protocol (BGP) forms the backbone of the internet, routing information between autonomous systems (AS) and facilitating data exchange across vast networks. However, BGP’s inherent lack of security measures makes it a frequent target for various cyber-attacks, including prefix hijacking and route leaks. These vulnerabilities not only compromise the integrity of data transmission but can also disrupt global communications. Existing defensive strategies, while beneficial, are often limited by partial deployment and lack comprehensive effectiveness across diverse network scenarios. This paper explores the performance of various BGP security policies through extensive simulations, providing insights into their effectiveness against different attack vectors under varied deployment strategies. We aim to highlight the strengths and weaknesses of current measures and propose robust solutions to enhance the security and resilience of BGP.

II. Background

The evolution of the BGP, the de facto system for routing information between autonomous systems on the internet, has been marked by persistent security challenges. Despite being critical for global internet infrastructure, BGP’s original design lacks sufficient safeguards against malicious activities, leading to significant security vulnerabilities. Comprehensive surveys by Mitseva *et al.* [1] and Huston *et al.*. [2] outline the extent of these vulnerabilities, ranging from prefix hijacking to route leaks. Moreover, efforts to secure BGP have been chronicled by Siddiqui *et al.* [3], highlighting recent standardization attempts to mitigate these risks. These surveys collectively underscore the urgency of adopting robust security mechanisms, such as Resource Public Key Infrastructure (RPKI) [4] and the Autonomous System Provider Authorization (ASPA) [5], to strengthen BGP’s resilience against enduring threats. The ongoing research and development in this domain are crucial for the implementation of effective countermeasures, as documented in the literature.

III. Related Work

Previous studies regarding BGP security have mostly focused on detailed explanations of singular security aspects, or only a few. For example, the most comparable work in scope to our proposed idea is a survey on recent efforts

[1] The work is supported by Grant H98230-21-1-0317.
[2] Calvin Idom is partially supported by NSF Grant 1754048.

979-8-3503-6491-0/24/$31.00 ©2024 IEEE

in BGP security by Mitseva *et al.* [1], which outlines and gives a comprehensive review of all recent known attacks and relevant security measures, seeking to deal with these attacks. While it does encompass a large range of different kinds of attacks, it does not test them within any sort of simulation or environment in order to provide data to back up these claims. It contains real-world examples but does not take advantage of the versatility that a simulation environment can provide. Even more prevalent is the fact that this survey was completed in 2018. It has no mention of ASPA [5], which is the newest and arguably most relevant innovation within BGP security, and it only has brief mentions of ROV and its variations [6], which have become the new cutting edge for the future of BGP security and are generally considered to have high potential.

In regards to ASPA works, the paper on Analysis of ASPA by Umeda *et al.* [7] is a more recent study, but still, it is only an analysis of ASPA's performance and does not compare it at a grander scale to other forms of security. Even though this work gives some insights into ASPA and utilizes LOTUS as a simulation tool in order to create quantitative data, its lack of comparison makes it challenging to relate the findings with BGP security. It doesn't provide a direction for security to move whilst referencing our results. Our study seeks to take this route to provide a comprehensive plan of action based on quantitative data that is recent and relevant. When these methods are evaluated in a simulation tool, it will recommend more realistic directions for the BGP community to develop new policies and implementations with a quantitative approach, as opposed to a comprehensive qualitative approach.

Aforementioned reasons lead us into BGPy [8] that provides a simulation framework for our experiments and the quantitative data for a better security analysis. BGPy is a simulation framework that allows users to run different kinds of attacks leveraging the latest CAIDA datasets [9], and evaluate how effective a chosen defense mechanism is toward the chosen attack. This laid the foundation of our empirical study, as many studies on BGP security has no empirical studies over the efficiency of available BGP defense mechanisms. For this reason, this study fills this gap in the research literature, and based on the results, recommend potential changes to further enhance the sustainability of BGP.

IV. Methodology

Our purpose is to perform an empirical research, which evaluates the efficiency of defense mechanisms against a various of prominent attacks targeting BGP. The empirical results allow us to improve the advancement of BGP security and lead to an impact on the selection of implemented security protocols in future designs.

A. Simulation Setup

To evaluate the efficacy of different defensive strategies against BGP attacks, we ran a series of simulations using BGPy [8], a simulator created for BGP security evaluation. Two separate sessions were used to run the simulations on Apple M1 and M2 MacBook architectures.

1) First Simulation Session: The first session was run using BGPy at commit `de44b54` in the `master` branch. The execution of simulations was made easier by a custom simulation runner script, which is accessible on our GitHub repository [10]. We experimented with several deployment types of ASes, attack scenarios, and defensive policy combinations. `ROV`, `ASPA`, `PeerROV`, and `AS-Cones` were the defensive policies, whereas `Accidental Route Leak`, `Prefix Hijacking`, `Subprefix Hijacking`, and `Forged-Origin Prefix Hijacking` were the attack scenarios. There were 64 possible permutations due to the AS deployment types, which were `Input Clique`, `Stubs`, `Multihomed`, and `No Deployment Type`. There were 1000 trials in each combination, split up across six policy adoption percentages. The Simulation Details subsection (IV-B) dives into greater detail about the simulations, including the scenarios, policies, and AS deployment types.

2) Second Simulation Session: The second simulation session was run off of a M2 MacBook. We had originally intended to incorporate the ROV++ [11] defensive policies in our first simulation session. This posed major problems in when attempting to implement code into BGPy [8]. After contacting the creators of the BGPy library, they gave us a test version on a different branch named `real_v8_no_bgpisec` which since been merged into `master`. This version included the `ROVPPv1Lite`, `ROVPPv2Lite`, and `ROVPPV2ImprovedLite` policy files. As in the last session, we executed ROV++ simulations across all attacks and deployment types, excluding 'Accidental Route Leak'; this attack-policy combination gave errors from the simulation software that we were unable to completely fix.

B. Simulation Details

1) Policies: We incorporated in our simulations a range of protective measures that were intended to solve various BGP security issues:

- **ROV:** One of the key components of the BGPy simulator for assessing the potency of BGP security measures is ROV. In order to stop misrouting brought on by prefix hijacking, it verifies a route's origin AS against a trusted registry [12], [13]. The FCC recently issued a directive mandating that the top 10 most connected American ASes adopt ROV, highlighting the technology's vital role in improving network security and impacting our attention to the Input Clique deployment strategy of ROV in our simulations [14].
- **ASPA:** As previously discussed, ASPA is a protocol that aims to avoid route leaks and forged-origin prefix hijacks by enabling ASes to publish cryptographic objects that define approved routing connections [5]. This improves the security of BGP. The authors of BGPy built-in an implementation of ASPA, and we have incorporated it in our simulations to assess its efficacy because of its importance in tackling these particular BGP vulnerabilities, especially in light of recent guidelines that support its use in highly linked networks [15].

- **PeerROV:** PeerROV is an implementation variation of ROV that filters BGP announcements only on the basis of peer connections. Within the adaptable simulation environment of BGPy, it provides a specific, context-dependent validation method by discarding notifications that ROA deems invalid if they come from peers.
- **AS-Cones:** Compared to ASPA, AS-Cones is a less developed path plausibility method that uses RPKI to protect AS relationship data in an attempt to reduce route leaks. Customer cones, or groups of ASes under the hierarchical supervision of a senior AS, are the emphasis of AS-Cones as opposed to ASPA. Since fewer ASes are needed to participate in AS-Cones than in ASPA, each AS publishes information detailing its customer cone, making it easier to discover and mitigate route leaks. Although it is not fully supported in BGPy yet, our implementation, which was motivated by Rodday et al. [15], employs `OnlyToCustomers` as its basic policy, corresponding with the intended usage of AS-Cones in a real-world situation. This implementation is not perfect, and we are looking forward to its inclusion in BGPy. We believe that including it in the future will improve simulation power and accuracy while illustrating the dynamics of AS relationships.
- **ROV++:** ROV++ Versions 1 and 2 [11] are security measures proposed by Morillo *et. al.*, which is meant to extend the base ROV protection over a BGP fabric that only has partial implementation of ROV. The different versions of ROV++ are included in BGPy as `ROVPPv1Lite`, `ROVPPv2Lite` and `ROVPPV2ImprovedLite`. Version 1 attempts to drop announcements sent to hijack sub-prefixes. Version 2 builds off of version one by using the blackhole announcement to compete for the sub-prefix in question, hopefully protecting other ASes from falling for the same false announcement. There is also a Version 3 being proposed, but was not covered in this analysis. The three policies mentioned are `LITE` versions of the original, which essentially means they were designed to be strictly software that is almost as effective as their full counterparts and are much easier to implement.

2) Scenarios: A range of attack scenarios were simulated in order to evaluate the resilience of each defensive strategy:

- **Accidental Route Leak** involves the unintentional spread of routing announcements beyond their intended scope, frequently as a result of setup errors, which has an impact on traffic flow and network efficiency.
- **Prefix Hijacking** occurs when a malevolent entity wrongfully asserts ownership of IP address blocks, diverting traffic meant for the rightful owner to themselves instead.
- **Subprefix Hijacking** focuses form of prefix hijacking in which smaller, more focused IP blocks are declared fraudulently; because of their specificity, these blocks are frequently harder to find.
- **Forged-Origin Prefix Hijacking** involves deceiving routers about the real source of the route by the malicious broadcast of prefixes with a fabricated origin AS.

3) Deployment Types: Various AS deployment options were considered to assess how the general efficacy of policies is impacted by their acceptance with different types of ASes:

- **No Deployment Type:** Since no particular AS type is the focus of this deployment strategy, the policies are implemented consistently to all AS types in the network. This kind of deployment is the most extensive and wide-ranging deployment option in our simulations, affecting all 77,029 ASes in the CAIDA Serial-2 dataset [9]. As seen by the adoption data in Table I, this technique naturally targets a greater number of ASes than any specific deployment type.
- **Input Clique:** Usually Internet Service Providers (ISPs) or backbone providers encompass this type of AS. The Input Clique deployment type focuses on a densely linked core set of ASes. Because of their important locations within network architecture, these ASes are distinguished by their noteworthy effect on routing decisions. Our dataset contains only 19 ASes that fall under the definition of an input clique. This deployment type is highly similar to the type of ASes that the FCC prefers to implement ROV in their new directive [14].
- **Multihomed:** ASes that link to many ISPs in order to enhance redundancy and perhaps optimize routing decisions are known as multihomed ASes. Although these ASes don't have clients, they stay in touch with several peers or providers, which improves their resilience and connectivity. With 37,614 multihomed ASes in our dataset, their importance in preserving stable and adaptable network infrastructures is highlighted.
- **Stubs:** ASes that don't forward traffic for other ASes are known as stubs. These ASes, which are found near the edge of the internet, connect to a restricted set of providers and are mainly responsible for handling incoming and outgoing traffic for their own networks. 27,398 stub ASes make up a sizable fraction of the network's edge in our dataset, where edge-level security dynamics can be greatly impacted by the adoption of security rules.

C. Data Analysis

We had substantial amount of data to analyze after the completion of these two simulation sessions. Combining the data from the sessions was the first stage in our study. To simplify the data while keeping the most important fields, we created a simple Julia script that concatenates all the CSV files from the BGPy output directory into a single CSV file. Among these columns were `scenario_cls`(the attack scenario), `AdoptingPolicyCls` (the primary policy), `PolicyCls` (BGP for all), `BasePolicyCls` (base policy, varying depending on the policy; see the Policies subsubsection(IV-B1)), `percent_adopt`, `outcome` (with possible values of `ATTACKER_SUCCESS`, `VICTIM_SUCCESS`, or `DISCONNECTED`), `value` (percentage of ASes hijacked, disconnected or successfully connected), `yerr` (y-error for 90% confidence interval for the outcome), and `deployment type`.

Our primary focus was on the `VICTIM_SUCCESS` outcomes, as we considered both the `DISCONNECTED` and `ATTACKER_SUCCESS` outcomes as failures of the policy. We handled this data by utilizing many Exploratory Data Analysis (EDA) approaches in a Julia Jupyter notebook, which is also located in our repository [10]. Among our analyses were:

- **Bar Graphs:** In order to display the victim success rate of each deployment type, irrespective of adoption percentage, we created bar graphs for each policy-scenario combination. This made it easier to determine which policies worked effectively in all deployment scenarios.
- **Scenario-specific Bar Graphs:** We produced bar graphs showing the victim success percentages for every policy and deployment style for every scenario. This made it possible for us to evaluate which rules worked best against different types of scenarios by displaying them side-by-side.
- **2D Heatmaps:** We produced heatmaps for each policy-scenario permutation, where the victim success rate is represented by heat, to illustrate the link between adoption rate and deployment type. This graphic made it easier for us to identify how the cross between different deployment methods and adoption rates affected and whether the policies were effective.
- **Correlation Heat Maps:** In order to graphically represent the association between victim success rate and adoption percentage, we created correlation heat maps for every scenario-policy permutation. These maps helped identify the ways in which the percentage of ASes that adopted the policy in each scenario affected the success of the policy.
- **Cross Bar Graphs:** These plots revealed the distribution of y-error for every deployment type across all possible scenario-policy combinations, offering valuable information on the accuracy of our data.
- **Multi-line Graphs:** We displayed the victim success rate for each deployment type as a function of adoption percentage in each scenario-policy combination, enabling us to see trends over a range of policy adoption levels.

Furthermore, for every combination of policy, scenario, and deployment type, we produced summary statistics that included mean, median, standard deviation, maximum, and lowest values for y-error, victim success rate, and adoption percentage. Our visual evaluations were quanitively supported by these statistics.

Our comprehension of how effectively each policy worked against various situations across deployment types and adoption percentages was improved by this statistical analysis and visualization. We developed preliminary theories concerning the performance of policies based on the knowledge gained from our literature review. The following section on Findings (Section V) discusses how we tested these hypotheses against our empirical data and how our findings supported or refuted them, along with which particular visualizations and EDA techniques contributed to these insights.

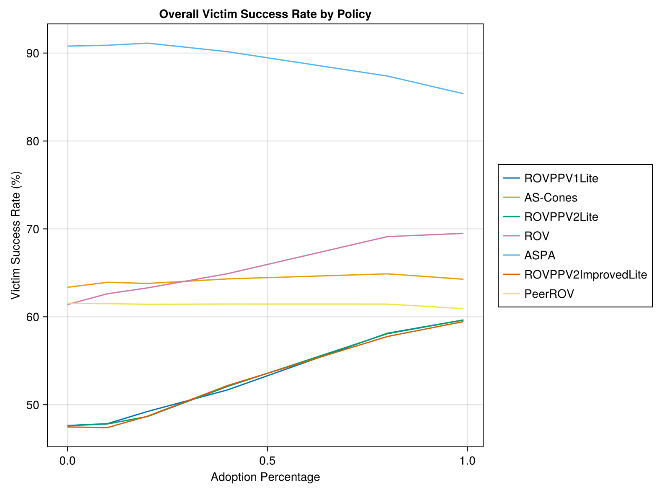

Fig. 1. Multi-line graph representing the relationship between adoption percentage and victim success rate for each defensive policy

V. Findings

A. Findings Overview:

There are many key issues and details that we have discovered over the course of analyzing the various graphs based on our experiments. These key patterns we noticed was input-clique only being 0.025% of total AS type still performed well, at some points better than other deployment types. The policy that consistently performed well is ASPA with ROV as a base. We also noted no deployment type tends to outperform the other deployment types, this could be due to encompassing nearly all the ASes in the network. In the following section we will go more into details over our hypotheses and what conclusion we came to after our experiments.

B. Hypothesis and Results:

- **Hypothesis 1:** Higher adoption percentage by any policy or deployment combo should lead to a greater success rate.
  This hypothesis was shown to be proven wrong as the higher the percentage of adoption increased the victim success rate stayed the same, with the exceptions being ASPA and ROV. ASPA can actually be seen decreasing in victim success rate as the adoption percentage goes up, while ROV and its variations increase. This can be seen in Figures 1 and 8.
- **Hypothesis 2:** ASPA should work best when deployed on stubs against any strategy at any adoption percent [15].
  This hypothesis was proven to be mostly correct as ASPA was the most effective during a stub deployment type, although no deployment type did have a slight led at a lower adoption percentage. These results can be seen in Figures 2 and 3. Therefore, if one prefers to implement ASPA in a real world scenario it will be more

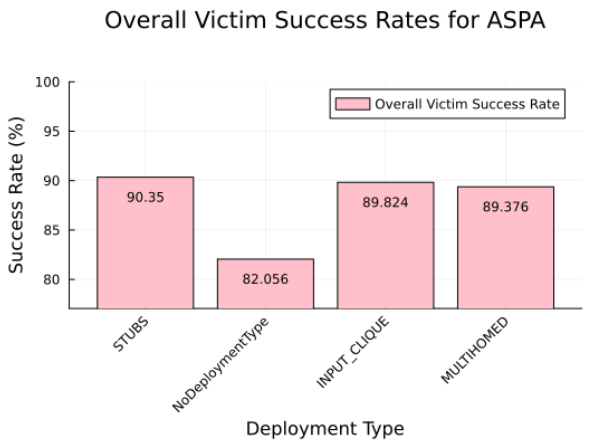

Fig. 2. Bar graph representing the relationship between adoption percentage and victim success rate for each ASPA deployment strategy

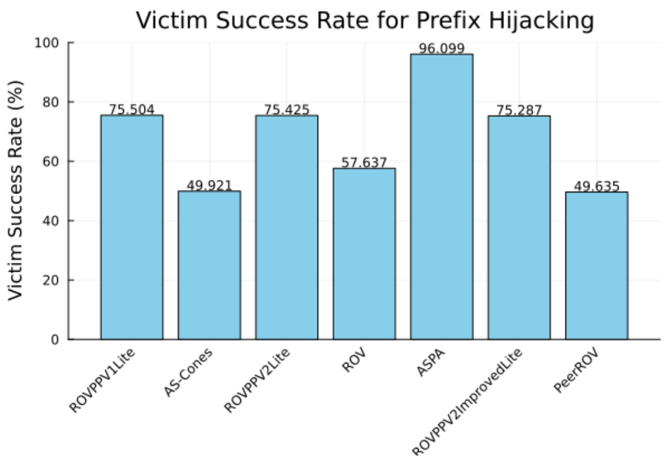

Fig. 4. Bar graph representing victim success rate of defensive policies against Prefix Hijacking Attacks

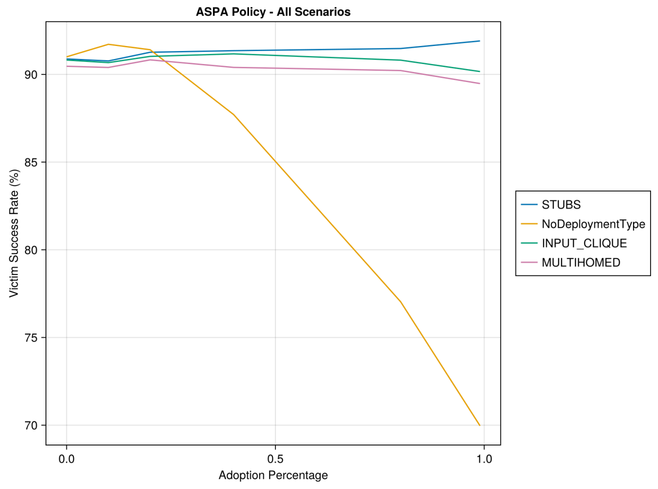

Fig. 3. Multi-line graph representing the relationship between adoption percentage and victim success rate for each ASPA deployment strategy

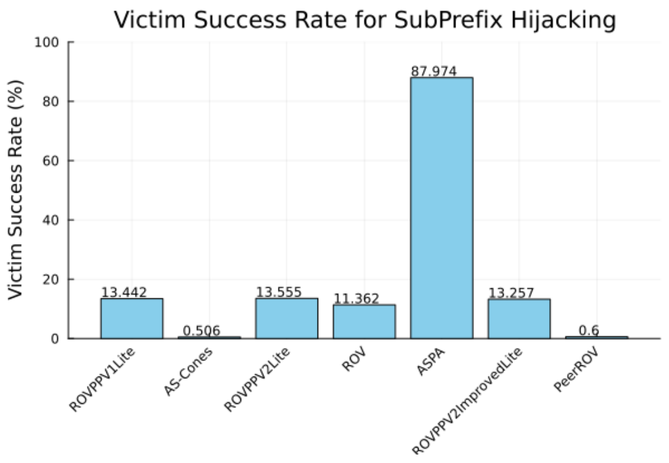

Fig. 5. Bar graph representing victim success rate of defensive policies against Sub-Prefix Hijacking Attacks

efficient to deploy it at the stub, although the benefits are marginal. The cost of deployment would be a far more important factor to deployment type rather than the marginal differences in efficiency.

- **Hypothesis 3:** ROV will have the highest victim success rate compared to any policy against prefix and sub-prefix hijacking [5], [7].
  Based on Figures 4 and 5, it can be seen that ROV performs well in both prefix and sub-prefix but it is outshadowed by ASPA. The reason is that since ASPA has ROV implementation as part of it with some additional policies and protections, which could cause the victim success rate to be higher. Additionally, enhanced ROV defence polices such as ROV++ performs better than ROV.
- **Hypothesis 4:** Any policy, especially ROV, will have a higher victim success rate as input-clique as the deployment compared to any other deployment type [2], [3], [8], [14].
  By thoroughly examining the graphs in Figures 6 and 7, it can be observed that input-clique outperforms the other deployment types in certain combinations of attacks and defenses, but most of the time no deployment type is better. Although input-clique did not perform better than no deployment type, it would be far more simpler and cost effective deploying defenses to 19 ASes than nearly all of them, thus verifying the FCC's deployment strategy.

C. Issues

We ran into several issues when conducting our study, including:

1) **ROV++ Simulation:** One of the policies we originally set out to compare to was ROV++ and its different iterations (v1, v2, v3). Within the BGPy library, there were several useful pre-implemented policies that we were able to take advantage of in order to simulate security methods without having to implement it ourselves. This

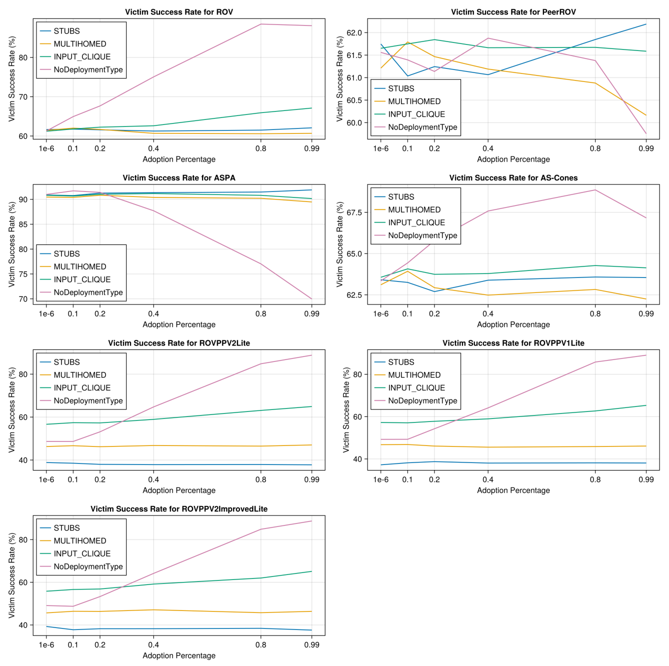

Fig. 6. Multi-line Graphs representing victim success rates compared to adoption percentage for each deployment type and defensive policy

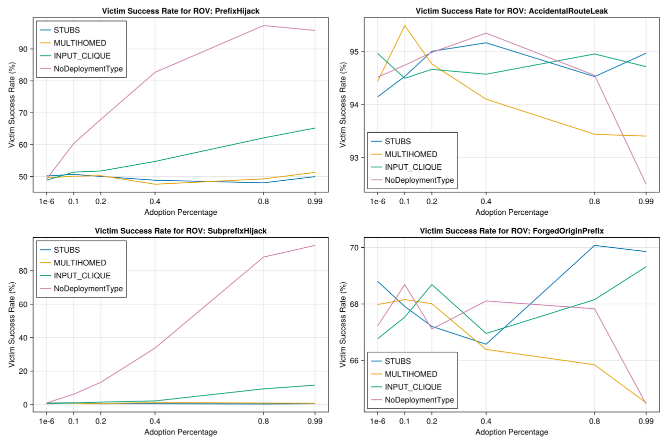

Fig. 7. Multi-line graphs highlighting the comparison rate between adoption rate and victim success rate for each deployment type of ROV for each attack scenario

was effective for many methods, but not for ROV++. In its iteration on the website, it was unable to run, and its base heavily implemented very old code and imports that were no longer relevant in the current version, but were still dependencies. However, we reached out to the creators of BGPy [8] and they responded very quickly saying they would be able to make an ROV++ policy simulation for us on short notice, but could not include v3 as it has potential security threats. We were only able to run it towards the end of our session time, and some of the data looked very similar across different versions of ROV++, but we were still able to get this data in for our final conclusion.

2) **BGPy Multi-Platform Setup:** Since BGPy is a relatively new and small simulation tool, it was not as easy of a tool to get set up on our systems. We ran into several issues while trying to install it on all of our different systems due to differences in architecture and operating system from what BGPy was designed for. Even though our environments were set up correctly, attempting to run the program on our systems would be extremely challenging through several errors regarding each different dependency requiring a different version of Python, which had to be solved by manually opening each module and changing the required version in order to run.
3) **Simulation Runtime:** We set up our simulations to run 5 different levels of adoption percentages 200 times for a total of 1000 simulations for each possible combination of policy, scenario, and deployment type. In order to cover all possible variations of these 3 parameters, we had to run 64 simulations. Each of these simulations took approximately 15 minutes to run, for a total of 960 minutes (or 16 hours) for the simulations alone, even on our best system. Since we were able to leave it running on it's own, it wasn't terribly taxing, but running the simulation required all of the computer's power, rendering it useless to being able to perform any other tasks. Thankfully, we were able to get our simulations up and running rather early in the research cycle, but due to our brief time constraints, it could have potentially been a large problem if we ran into any other errors.
4) **No Native Support for AS-Cones:** We stated in our original proposal that we wanted to include ASPA as a policy to analyze and consider in our grand conclusion, but one other approach that is essentially the same idea as ASPA is AS-Cones. AS-Cones are worth considering because we knew that ASPA would hold strong results from other papers, and AS-Cones are built on the same fundamentals as ASPA, which is creating AS objects that contain data about other authorized ASes. BGPy unfortunately did not have native support for this policy, but we believed this policy to be worth our time, so we instead implemented our own AS-Cones policy by referencing the structure defined by Rodday *et al.* [15]
5) **BGPy Output Could Use More Values:** Although BGPy has been an incredibly useful tool and has been the cornerstone for our empirical evaluations, we found that the data it produced in its CSV files could have provided more values, as the only values it gave us were the y-error value and the percentage of cases that received the type of propagation. Also of note, the creators of BGPy notified us that their `all but one` value does not work very well, so we had to use 99% to get close enough.
6) **Real-World Architecture Simulation:** During our study, we opted to simulate real-world scenarios rather than replicating the exact architecture used by ISPs.

This decision was driven by the significant computing, time, and cost constraints associated with mimicking such complex infrastructures. While BGPy performed exceptionally well in its intended role as a simulation framework, enabling us to conduct our research without the need for high-performance computing (HPC), we recognize the potential value of conducting similar experiments on real-world architectures. We would be particularly interested in seeing an entity with the necessary resources undertake such a study, as it could provide deeper insights into BGP security mechanisms in practice. Additionally, the CAIDA dataset [9] we used was from 2020, and while it included real-world information from ASes, utilizing a more up to date dataset might produce differing results.

VI. Future Work

We plan to improve the implementation of ROV++ and AS-Cones, ensuring they work properly and are up-to-date. Additional simulations will focus on deploying these defenses to the top 10 ASes, evaluating the effectiveness of FCC's BGP regulation proposal. We also aim to deploy defenses to all ASes except one, rather than using 99%. These efforts will provide valuable insights to enhance BGP security.

There are several sections we would like to improve on, such as the implementation of ROV++ and AS-Cones. Due to us implementing ROV++ based on technical explanation and a deadline, we would like to revisit it and perform some additional simulations to make sure it works properly. Similarly, for AS-Cones due to it still being in a developmental phase; there may be changes in the future we would need to into account for. Once we make sure these defense mechanisms are running properly and are up to date, we want to run a simulation where we only deploy the top 10 US-based ASes. The reason for this is to better determine the effectiveness of the proposal for BGP regulation created by FCC [14]. This would allow us to see if the regulations would be enough to provide significant protection for BGP, or if a different direction would need to take place. Finally, we would want to deploy defenses to all ASes except one, rather than using 99% in place of it. By running these simulations and obtaining more results, we hope to provide insight and knowledge to policy makers, researchers, and companies to better improve the security of BGP which it lacks.

VII. Conclusion

Our study identifies ASPA as the most effective BGP security mechanism, though broader deployment and further research are needed. PeerROV, despite its ease of implementation, offers limited protection and may not be the best priority for defense. Implementing defenses at the input clique level shows promise due to its effectiveness with fewer ASes involved, making it a practical and scalable solution. Future work would concentrate on cost-effective and practical strategies for deploying robust BGP security measures.

Acknowledgments

We would like to acknowledge Neil Ziring from NSA for his proposal of the idea for this research. We also thank Iowa State University for hosting us onsite to work on this project. Additionally, Dr. Furuness, the creator of BGPy, was exceptionally helpful with his responses to our questions about BGPy.

References

[1] A. Mitseva, A. Mitseva, A. Panchenko, A. Panchenko, T. Engel, and T. Engel, "The state of affairs in bgp security: A survey of attacks and defenses," *Computer Communications*, 2018.

[2] G. Huston, G. Huston, G. Huston, M. Rossi, M. Rossi, G. Armitage, and G. Armitage, "Securing bgp — a literature survey," *IEEE Communications Surveys and Tutorials*, 2011.

[3] M. S. Siddiqui, M. S. Siddiqui, M. S. Siddiqui, D. Montero, D. Montero, R. Serral-Gracià, R. Serral-Gracià, X. Masip-Bruin, X. Masip-Bruin, X. Masip-Bruin, M. Yannuzzi, and M. Yannuzzi, "A survey on the recent efforts of the internet standardization body for securing inter-domain routing," *Computer Networks*, 2015.

[4] NIST, "NIST RPKI Monitor — rpki-monitor.antd.nist.gov," https://rpki-monitor.antd.nist.gov/ROV/20240613.00/All/All/6, [Accessed 13-06-2024].

[5] A. Azimov, E. Bogomazov, R. Bush, K. Patel, J. Snijders, and K. Sriram, "BGP AS_PATH Verification Based on Autonomous System Provider Authorization (ASPA) Objects," Internet Engineering Task Force, Internet-Draft draft-ietf-sidrops-aspa-verification-17, Feb. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification/17/

[6] N. Rodday, Í. Cunha, R. Bush, E. Katz-Bassett, G. D. Rodosek, T. C. Schmidt, and M. Wählisch, "Revisiting rpki route origin validation on the data plane," in *Proc. of Network Traffic Measurement and Analysis Conference (TMA), IFIP*, 2021.

[7] N. Umeda, T. Kimura, and N. Yanai, "The juice is worth the squeeze: Analysis of autonomous system provider authorization in partial deployment," *IEEE Open Journal of the Communications Society*, 2023.

[8] J. Furuness, C. Morris, R. Morillo, A. Herzberg, and B. Wang, "Bgpy: The bgp python security simulator," *CSET @ USENIX Security Symposium*, 2023.

[9] CAIDA, "Caida as relationships dataset," 2020. [Online]. Available: https://www.caida.org/catalog/datasets/as-relationships/

[10] S. Barrett, C. Idom, G. Z. Villafuerte, and A. Byers, "GitHub - sethbarrett50/BGPy_Analysis — github.com," https://github.com/sethbarrett50/BGPy_Analysis, [Accessed 25-06-2024].

[11] R. Morillo, R. Morillo, J. Furuness, J. Furuness, C. Morris, C. Morris, J. T. Breslin, J. Breslin, A. Herzberg, A. Herzberg, B. Wang, and B. Wang, "Rov++: Improved deployable defense against bgp hijacking." *Network and Distributed System Security Symposium*, 2021.

[12] T. Hlaváček, H. Shulman, N. Vogel, and M. Waidner, "Keep your friends close, but your routeservers closer: Insights into rpki validation in the internet," *USENIX Security Symposium*, 2023.

[13] Y. Gilad, T. Halvacek, A. Herzberg, M. Schapira, and H. Shulman, "Perfect is the enemy of good: Setting realistic goals for interdomain routing security."

[14] F. C. Commission, "Reporting on border gateway protocol risk mitigation progress; secure internet routing," https://www.fcc.gov, 2024, accessed: 2024-06-25.

[15] N. M. Rodday, G. D. Rodosek, A. Pras, and R. M. van Rijswijk-Deij, "Exploring the benefit of path plausibility algorithms in bgp," in *IEEE/IFIP Network Operations and Management Symposium, NOMS 2024*. IFIP, 2024.

Appendix

Deployment Type	**Only One**	**10%**	**20%**	**40%**	**80%**	**99%**
Input Clique	1	2	4	8	16	19
Stubs	1	2740	5480	10960	21919	27125
Multihomed	1	3762	7523	15046	30092	37238
No Deployment Type	1	7703	15406	30812	61624	76259

TABLE I
Number of ASes Adopting Defensive Policy Based on Deployment Type and Adoption Percentages

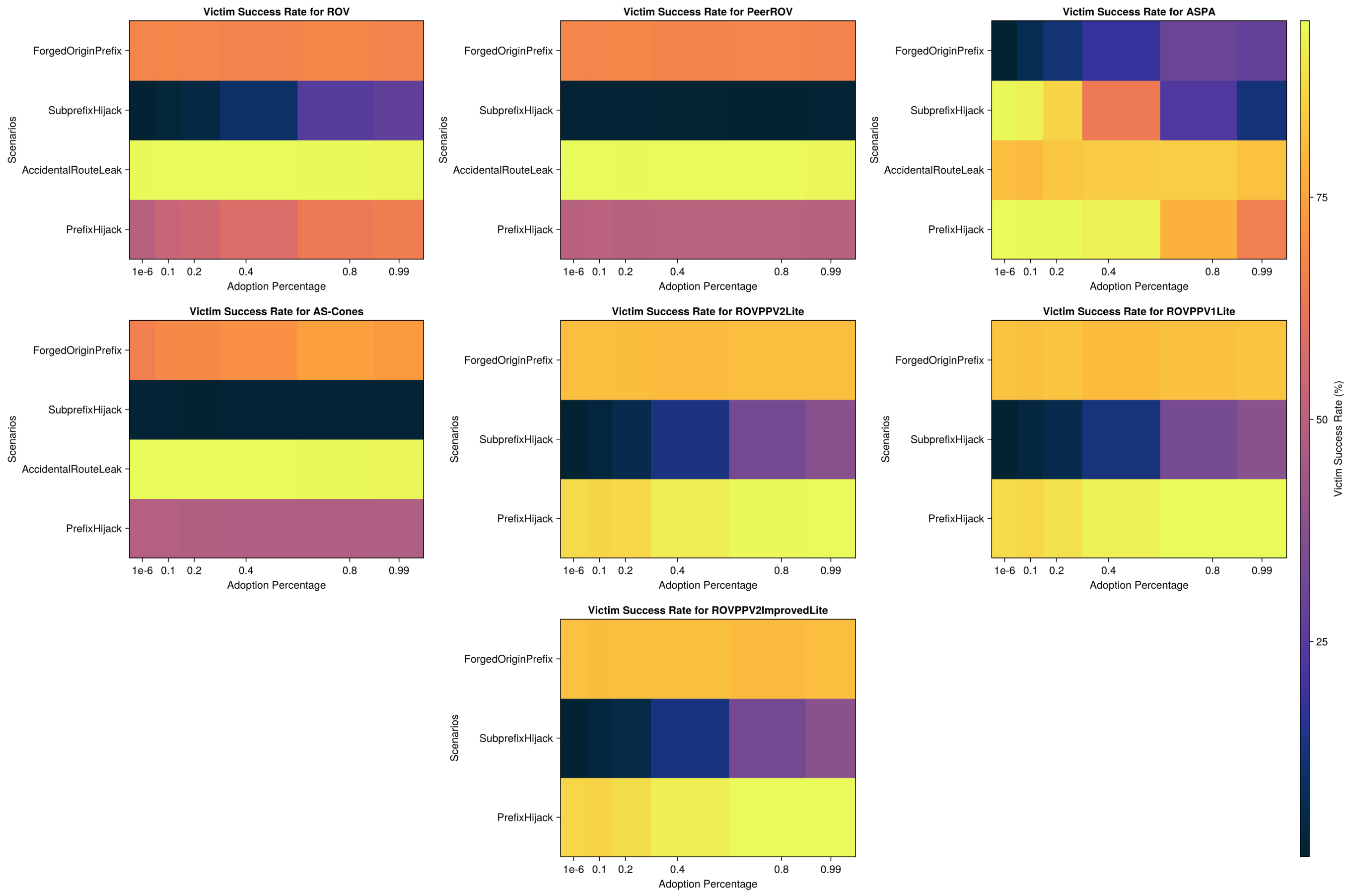

Fig. 8. Heat-map illustrating the relationship between policy adoption rates and victim success rates across various attack scenarios.