BGP Security in the Age of Machine Learning: Why Current Defenses Fall Short

Abstract—**Border Gateway Protocol (BGP) remains a foundational yet vulnerable component of internet routing, frequently targeted by hijacks and other malicious activities. While the Resource Public Key Infrastructure (RPKI) mitigates some threats, it is incompletely deployed and vulnerable to attacks like forged-origin hijacks. Consequently, ML-based monitoring systems analyzing public BGP data have become crucial defenses. However, these systems implicitly trust the integrity of the monitored data. This paper demonstrates that this trust is misplaced. We show that state-of-the-art ML-based BGP monitoring systems, specifically DFOH and BEAM, are susceptible to adversarial manipulation. Attackers can poison the historical data used by these systems or pollute the statistical metrics they rely on. Through analysis and simulation, we illustrate how carefully crafted BGP announcements can allow malicious hijacks, including RPKI-compliant forged-origin attacks, to evade detection. Our findings expose the fundamental limitations of relying solely on passively collected public BGP data for security and highlight the urgent need for more robust, multi-faceted strategies to protect the internet's core routing infrastructure.**

The Border Gateway Protocol (BGP) serves as the backbone of the internet, directing traffic between the vast network of Autonomous Systems (ASes) that comprise the global infrastructure. However, BGP was designed in an era prioritizing connectivity over security, leaving it without inherent mechanisms to verify the authenticity of routing information [1]. This fundamental vulnerability makes it susceptible to malicious manipulation, most notably BGP hijacking, where an attacker illegitimately claims ownership of IP address blocks (prefixes), and route leaks, where routing announcements are propagated beyond their intended scope. These incidents can redirect vast amounts of internet traffic, enabling espionage, causing widespread service outages, and undermining the stability of the internet [2]–[5]

To combat these threats, the Resource Public Key Infrastructure (RPKI) was developed [6]. RPKI provides a way to cryptographically verify that an AS is authorized to originate routes for specific IP prefixes, significantly mitigating traditional prefix hijacks. However, RPKI deployment is far from universal [7], [8], leaving large parts of the internet unprotected. Furthermore, even where RPKI is deployed, it remains vulnerable to specific manipulations, such as forged-origin attacks [9]. In these attacks, an adversary announces a prefix belonging to a victim but prepends the victim's legitimate AS to the path. This makes the announcement appear valid to RPKI checks while still allowing the attacker to attract traffic, highlighting the need for security measures beyond RPKI alone. Another cryptographic solution, BGPSec [10], aims to secure the entire path of BGP announcements, offering stronger protection against path manipulation attacks. However, similar to RPKI, its deployment has been extremely slow, limiting its practical effectiveness in the current internet landscape.

Given the limitations of cryptographic solutions like RPKI and the slow adoption of BGPSec, the BGP security community has increasingly turned towards BGP monitoring. Systems leverage data from public collectors like RouteViews [11] and RIPE RIS [12], which aggregate BGP updates from hundreds of vantage points worldwide. In recent years, Machine Learning (ML) and data-driven techniques have been integrated into these monitoring systems. By analyzing historical and real-time BGP data, these systems extract complex features (e.g., AS-path characteristics, topological changes) and train models to distinguish legitimate routing behavior from anomalies indicative of hijacks or leaks. Prominent examples of such data-driven systems include academic proposals like ARTEMIS [9], DFOH [13], and BEAM [14], as well as commercial offerings like Cisco's ThousandEyes [15].

However, like many ML applications, these systems face challenges regarding the integrity of their input data. Their effectiveness can be compromised if the observed BGP data does not accurately reflect ground truth, particularly if it is subject to deliberate manipulation by sophisticated adversaries. Attackers, aware of these monitoring systems, can craft malicious BGP announcements specifically designed to deceive the ML models. This constitutes an adversarial attack, a well-known vulnerability in the ML domain, where inputs are subtly altered to cause misclassification [16], [17].

This paper demonstrates that current state-of-the-art ML-based BGP monitoring systems are susceptible to such adversarial manipulation. We show how attackers can poison the knowledge bases used by these systems or pollute the metrics they rely on, allowing malicious hijacks (including RPKI-compliant forged-origin attacks) to evade detection. Through analysis and simulation targeting systems like DFOH [13] and BEAM [14], we expose the fundamental limitations of relying solely on publicly visible BGP data for security. Our findings underscore the urgent need for more robust, multi-faceted security strategies to protect the internet's core routing infrastructure from sophisticated adversaries.

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.[23|24]xxxx
www.ndss-symposium.org

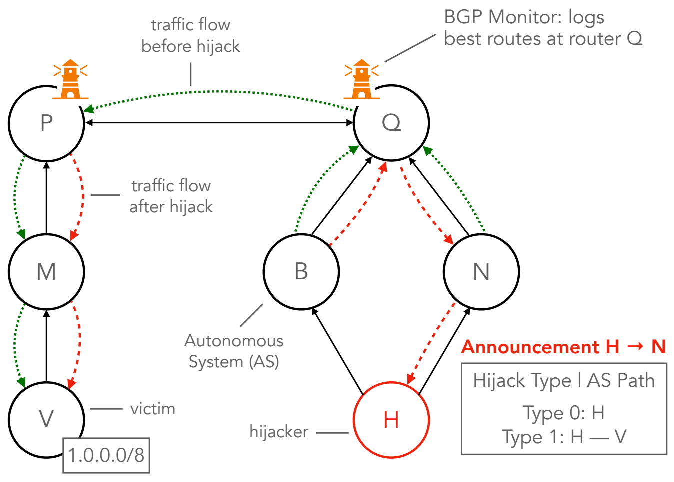


Fig. 1. Example of hijack for prefix 1.0.0.0/8 owned by AS *V*. Single-headed solid arrows indicate the direction of the money in customer-provider links; double-headed arrows represent charge-free links.

I. Background

A. Prefix Hijacking

Prefix hijacking is a malicious activity where an attacker illegitimately claims ownership of IP address blocks (prefixes) they do not control. This is possible because the BGP protocol, which manages routing between ASes, was designed with an inherent trust model, lacking built-in mechanisms to verify the authenticity of route announcements. When an attacker successfully hijacks a prefix, they can redirect internet traffic intended for the legitimate owner, leading to severe consequences such as service outages, espionage, and financial losses, as seen in attacks targeting cryptocurrency platforms like KlaySwap [4] and MyEtherWallet [5].

Hijacks can be categorized based on their complexity. Following the taxonomy proposed in [9], a *Type-0 hijack* involves an attacker announcing a prefix belonging to another AS without altering the origin AS in the announcement. This is the simplest form of hijack. A *Type-1 hijack*, or forged-origin hijack, occurs when the attacker announces the victim's prefix but lists the victim's AS as the origin. While the Resource Public Key Infrastructure (RPKI) [6], which cryptographically links prefixes to their legitimate origin ASes, can effectively prevent Type-0 hijacks when deployed, it is less effective against Type-1 hijacks if the attacker manipulates the AS path in specific ways (e.g., prepending the legitimate origin). Figure 1 illustrates both Type-0 and Type-1 hijack scenarios.

B. BGP monitors

BGP monitors are crucial infrastructure components for observing the global state of internet routing. Prominent examples include the RouteViews project [11] and the RIPE Routing Information Service (RIS) [12]. These projects operate globally distributed collectors that establish BGP peering sessions with numerous ASes worldwide, acting as vantage points. By passively listening to the BGP updates (including announcements and withdrawals) exchanged in these sessions, monitors collect vast amounts of routing data. This data typically includes periodic snapshots of the Routing Information Base (RIB) from their peers and continuous streams of UPDATE messages reflecting real-time changes. The data is typically stored and distributed in the Multi-threaded Routing Toolkit (MRT) format [18] and access is provided through near real-time streams, often using the BGP Monitoring Protocol (BMP) [19], and periodic archival dumps (e.g., RIB snapshots every few hours and update logs every few minutes). While invaluable, this data provides an incomplete view of the internet's routing dynamics, as monitors only capture routes propagated towards their specific vantage points. Despite this limitation, in the current data-driven era, BGP monitors are indispensable for security analysis, performance monitoring, and research, offering the most comprehensive publicly available source of information for understanding BGP behavior and detecting anomalies like hijacks.

C. Data-driven Hijack Detection

Given the limitations of inherent BGP security and the incomplete deployment of RPKI, data-driven hijack detection systems have emerged as a critical layer of defense. These systems analyze BGP data, primarily sourced from public monitors, to identify suspicious routing events.

More sophisticated systems employ machine learning (ML) and anomaly detection techniques to uncover complex attacks, including forged-origin hijacks (Type-1) and route leaks, which might evade simpler checks. These systems analyze features extracted from BGP updates, such as AS-path characteristics, prefix propagation patterns, and consistency with historical data, to build models of normal routing behavior and flag significant deviations [9], [13], [14], [20], [21]. A general overview of such a system is depicted in Figure 2. Typically, these systems operate in three phases: first, they are *triggered* by new BGP updates observed by monitors; second, they *check* the legitimacy of the update against a knowledge base of historical data or learned models; finally, if the calculated metrics exceed certain thresholds, they raise an *alert*. For instance, DFOH is triggered by new AS links, checks them by computing features (e.g., topological, PeeringDB-based) against its historical graph, and alerts based on a Random Forest classifier's output. While their specific architectures and algorithms vary, a common characteristic is their reliance on historical BGP announcements collected by monitors to establish a baseline or knowledge base against which new routes are evaluated. This reliance on potentially manipulable public data is a key focus of this paper.

II. Overview

A. Threat Model

We consider an adversary who operates their own AS and is capable of injecting malicious BGP announcements into the global routing system with the intent of disrupting internet traffic, specifically through prefix hijacking. The attacker's primary objective is to execute a Type-1 hijack. This type of

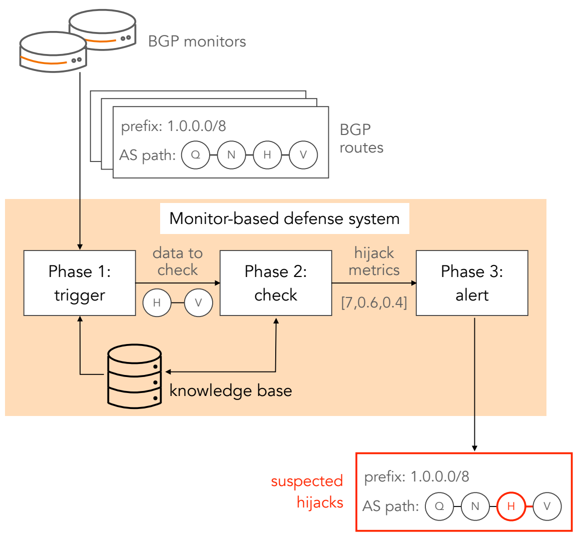


Fig. 2. General architecture of a monitor-based hijack detection system.

attack is particularly relevant in the context of our study, as it targets scenarios where the RPKI is widely deployed. While RPKI provides cryptographic validation of route origins, it remains vulnerable to forged-origin attacks, which are the focus of the defenses analyzed in this paper.

We assume that the attacker operates in an environment where their upstream providers do not enforce strict IP prefix filtering or AS path filtering. This assumption is grounded in real-world observations of past hijacks, where such filtering mechanisms were absent, allowing malicious announcements to propagate [2]–[5]. The lack of filtering by upstream providers is a critical enabler for the adversary, as it permits the injection of illegitimate routes into the global BGP routing system.

Furthermore, the adversary may leverage readily available commercial transit services, often offered by hosting providers, which allow users to acquire BGP sessions relatively easily and cheaply, frequently through Virtual Machines (VMs) [22], [23]. We assume the adversary can establish additional BGP sessions with such transit providers. These additional sessions might be used for legitimate purposes or to aid in the propagation of poisoning announcements (as described in Section II-C), but the core hijack announcements rely on the lack of filtering by the main upstream provider mentioned previously. We assume these additional transit services are used in a way that does not directly implicate them in the hijack itself, thus avoiding their specific abuse detection mechanisms.

B. Fundamental Limitations of Hijack Monitors

Monitor-based defense systems suffer from inherent limitations that stem from the nature of BGP data collection and analysis, creating vulnerabilities that sophisticated attackers can exploit.

It is impossible to identify invalid or expired BGP data. These systems face the challenge of determining data validity over time. When a new route is observed and classified as legitimate, there's no fundamental way to determine how long it should remain in the knowledge base as valid. BGP paths vary greatly in their stability—some persist for months while others are ephemeral. Since the BGP protocol itself provides no explicit mechanism to signal when paths or links expire, defense systems must make arbitrary decisions about data retention. At any point, previously announced paths may still be valid despite not being currently used, or they might have legitimately expired.

BGP routes implementing a hijack cannot be distinguished from policy changes. Monitor-based systems cannot reliably distinguish between actual hijacks and legitimate policy changes. For any newly observed BGP path, two plausible scenarios always exist: either the path results from a malicious hijack attempt or it reflects a legitimate topology change resulting from new business agreements or updated BGP policies. These scenarios are fundamentally indistinguishable from the perspective of external observers. As a result, these defense systems must rely on inherently imprecise heuristics based on factors like AS geography or inferred business relationships to evaluate route legitimacy.

Hijackers can manipulate BGP data. Lastly, and perhaps the most critical, attackers can deliberately manipulate the BGP data that these detection systems rely upon. Since monitors collect BGP routes without discrimination as to their origin or legitimacy, adversaries can inject arbitrary routes in addition to their actual hijack attempts. This capability allows attackers to strategically craft BGP announcements that condition the knowledge base over time, gradually poisoning it with seemingly benign routes that later enable undetected hijacks. Such manipulation requires minimal additional infrastructure beyond what's already needed to execute hijacks, and can remain stealthy if performed gradually with carefully crafted announcements.

Collectively, these limitations create a fundamental vulnerability: monitor-based defenses are forced to operate on data that may be partially crafted by potential attackers, with no reliable way to detect when this manipulation is occurring. This vulnerability undermines the core assumptions of data-driven BGP security approaches and creates opportunities for evasion that sophisticated adversaries can exploit.

C. Poisoning Monitors

A fundamental issue with relying on public BGP monitors is that they primarily act as passive data collectors. They record the BGP announcements they observe from their peers but generally do not perform deep validation of the legitimacy or accuracy of the routing information contained within those announcements. Their goal is data aggregation, not verification.

This lack of inherent validation poses a significant problem for ML-based hijack detection systems. These systems often build their models or knowledge bases by learning from the historical data collected by monitors, implicitly trusting this data as a representation of legitimate routing behavior. If the

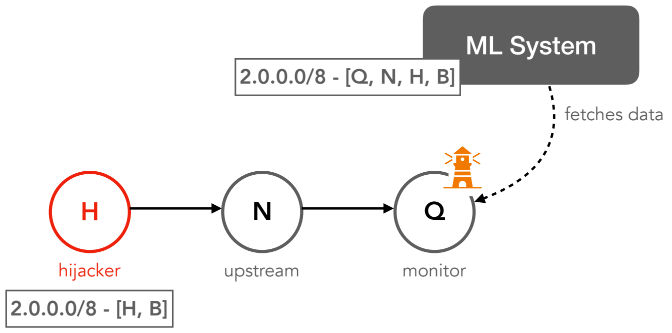


Fig. 3. Monitor Poisoning Attack: Attacker H announces a prefix with forged origin B. The announcement propagates via H's upstream to a monitor. The ML system ingests this polluted data, corrupting its knowledge base.

underlying data is tainted, the resulting model or knowledge base will be flawed, leading to inaccurate classifications.

An attacker can exploit this by deliberately "poisoning" the data collected by monitors. This involves injecting carefully crafted BGP announcements designed to mislead the ML systems that consume this data. The goal is to manipulate the system's understanding of normal network topology or routing policies.

Specifically, an attacker can poison the knowledge base by announcing a prefix they legitimately control (or a sub-prefix), but forging the origin AS in the announcement to be a different, strategically chosen AS. The attacker ensures this prefix does not have a Route Origin Authorization (ROA) in the RPKI system, or creates a ROA that includes the forged origin. Since RPKI deployment is still far from universal and many prefixes lack ROAs [8], announcements for such prefixes often result in an RPKI validation state of 'NotFound'. Many networks do not drop routes with 'NotFound' status [24]. Consequently, the poisoned announcement can propagate through the network, be collected by public monitors, and subsequently ingested by ML-based detection systems, corrupting their knowledge base with inaccurate information about AS relationships or path validity. This poisoned data can then be used by the attacker to make subsequent hijack attempts appear legitimate or to cause false alarms. The carefully selected forged-origin AS is typically one not directly connected to the attacker's actual infrastructure and is chosen specifically to appear plausible enough to evade initial detection by the monitoring system.

Figure 3 illustrates this process. An attacker (AS H) announces a prefix it controls, but falsely claims AS B as the origin, an AS to which H is not actually connected. H's upstream provider propagates this announcement. A public monitor receives the announcement from its peer and records the path containing the forged origin. Subsequently, an ML-based detection system fetches data from this monitor, ingesting the polluted information and potentially adding an incorrect link or relationship involving AS B to its knowledge base.

III. Exploiting DFOH

A. DFOH Architecture

DFOH (Detecting Forged-Origin Hijacks) is designed to identify forged-origin BGP hijacks by analyzing AS paths observed in public BGP data [13]. Its architecture centers around a three-stage pipeline:

1) **New Link Detection:** DFOH monitors BGP updates from public collectors (e.g., RouteViews, RIPE RIS) and compares observed AS links against a historical topology graph (built from $\approx$300 days of data). Links not present in the historical graph are flagged as "new," triggering further analysis, as these often correlate with hijack events. The core idea is that forged-origin hijacks often introduce a new, previously unobserved link in the AS path, typically between the attacker and the (forged) origin AS.
2) **Feature Computation:** For each new link, DFOH calculates a feature vector encompassing four categories to assess legitimacy:
   - *Topological:* Measures the impact of the new link on AS graph structure (e.g., centrality, neighborhood changes).
   - *Peering:* Infers peering likelihood based on shared infrastructure (IXPs, facilities) or geography, using data from sources like PeeringDB [25] and focusing on neighbors' data to resist manipulation.
   - *AS-Path-Pattern:* Evaluates path validity against routing policy expectations (e.g., Gao-Rexford valley-free model [26]) using learned models based on AS degree and customer cone sequences.
   - *Bidirectionality:* Checks if the link is observed in both directions (using BGP and IRR data), a strong indicator of legitimacy.
3) **Inference:** The computed features are fed into a Random Forest classifier trained to distinguish between legitimate links and forged-origin hijacks. This classifier is trained daily using a balanced sampling strategy that clusters ASes and ensures representative sampling across different AS types and potential attack scenarios, mitigating biases inherent in the AS topology. If multiple paths contain the new link, results are aggregated.

The system aims to provide timely detection by focusing analysis on these newly observed links and leveraging a combination of topological, policy-based, and metadata features. However, as discussed in Section II-B, its reliance on publicly observable data forms the basis for potential adversarial attacks.

B. Poisoning DFOH's Knowledge Base

While DFOH aims to detect forged-origin hijacks by identifying new AS links and analyzing associated features, its reliance on historical data gathered from public monitors creates a significant vulnerability - as attackers can manipulate the data ingested by these monitors. This allows for a knowledge base poisoning attack specifically tailored to circumvent DFOH's defenses.

The core principle of the attack is to strategically introduce carefully crafted, non-existent AS links into DFOH's historical topology graph *before* launching an actual hijack. The goal is to manipulate the data against which future, malicious announcements are compared. The methodology involves the following steps:

1) **Identify Poisonous Links:** The attacker identifies potential "poisonous" ASes. These are typically ASes not directly connected to the attacker but chosen strategically to manipulate DFOH's feature calculations favorably for a future hijack attempt against a target victim AS. The attacker leverages the small inherent error rate (false negatives) of the DFOH classifier, aiming for these crafted links to be misclassified as legitimate and added to the knowledge base.
2) **Craft Poisoning Announcements:** The attacker announces a prefix they legitimately control (or a subprefix) but forges the origin AS in the BGP announcement to be one of the chosen poisonous ASes (see II-C).
3) **Manipulate Feature Scores:** The poisonous links are chosen specifically to degrade DFOH's ability to detect a subsequent hijack involving the attacker (H) and a victim (V). For instance, by poisoning the knowledge base with links to ASes geographically close to the victim, the attacker can artificially lower the suspicion score derived from the PeeringDB features when the actual hijack link (H, V) appears later. As shown in Table I, features like PeeringDB carry significant weight in the classification decision, making them effective targets for manipulation.
4) **Execute Hijack:** After a sufficient period, allowing the poisoned data to be incorporated into DFOH's knowledge base (typically minutes), the attacker launches the actual forged-origin hijack. The presence of the previously injected poisonous links makes the new, malicious link appear less anomalous to DFOH's classifier, increasing the chance of evasion.

By manipulating DFOH's input data, attackers can undermine its detection capabilities with minimal extra resources, posing a practical threat.

TABLE I
FEATURE CATEGORY IMPORTANCE SCORES FOR THE DFOH RANDOM FOREST CLASSIFIER, DERIVED FROM TRAINING ON DATA FROM 2024-03-01.

Feature Category	Importance Score
ASPath Patterns	0.59
PeeringDB	0.23
Topological	0.16
Bidirectionality	0.02

C. Experimental Evaluation

To evaluate the effectiveness of the knowledge base poisoning attack against DFOH, we conducted a large-scale simulation-based experiment. The simulations were performed on a cluster consisting of 2 nodes, each equipped with a double-socket Intel Xeon Gold 5118 CPU. While attacking a single target AS requires only a few seconds, the scale of the experiment - simulating 1,000 attacker ASes targeting each of the over 80,000 ASes in the internet topology - necessitated significant computational resources.

We simulated 1,000 attacker ASes, chosen randomly to represent a diverse set of network types and locations. For each simulated attacker, we attempted to poison DFOH's knowledge base to enable subsequent forged-origin hijacks against all other ASes. The simulation for each attacker utilized real-world AS paths observed from public collectors that originated from the attacker's AS. This represents a conservative approach, as many observed paths might be transient and not reappear, potentially underestimating the attack's effectiveness in a real-world scenario where attackers could sustain announcements.

The poisoning process involved identifying potential false negatives in DFOH's classification – links that DFOH would incorrectly classify as legitimate. From this set, we selected poisonous links using a heuristic designed to maximize the impact on DFOH's feature calculations for future hijack attempts. Specifically, we targeted features with high importance scores, such as PeeringDB (see Table I), by selecting poisonous ASes geographically close to potential victim ASes. These crafted links were then assumed to be injected into the BGP data stream (as described in Section II-C) and subsequently incorporated into DFOH's historical topology graph after being misclassified.

During initial simulations, we observed that some stub ASes, particularly those in remote or poorly connected regions, exhibited very low false negative rates according to DFOH's model. While seemingly positive for detection, this paradoxically implies that DFOH would likely generate frequent false positives for legitimate announcements originating from these networks. From the attacker's perspective, this finding initially limited the pool of viable poisonous AS candidates when targeting certain victims. To address this and model a more realistic attacker capability, we simulated the scenario where an attacker acquires additional connectivity through readily available commercial BGP transit services [22], [23]. By announcing legitimate prefixes through a new provider connection, the attacker could expand their observed connectivity. Although DFOH's mechanism would initially flag and quarantine this new (legitimate) link for a period (30 days), an attacker willing to wait could subsequently leverage this expanded connectivity, significantly increasing their pool of potential poisonous ASes for manipulating the knowledge base. Our final evaluation incorporates this possibility.

Figure 4 presents the results of our simulation incorporating this enhanced attacker model. The left plot shows the distribution of hijack success rates achieved by each attacker after the poisoning phase. While the first 400 attackers achieved success rates below 40%, a significant portion demonstrated high effectiveness. Notably, over 100 distinct attacker ASes

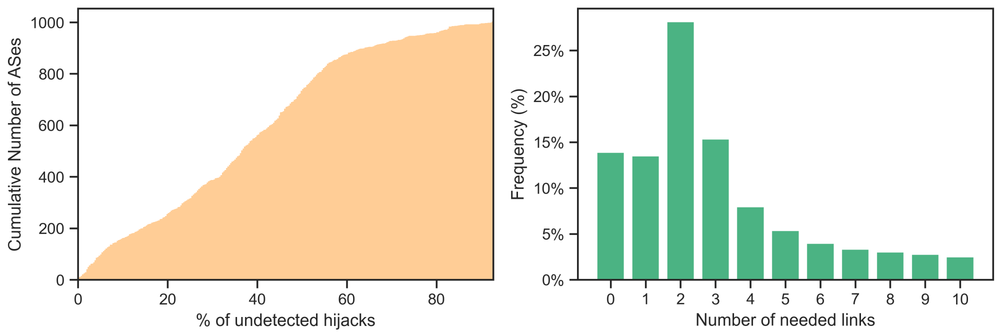


Fig. 4. Evaluation of the DFOH poisoning attack. Left: Distribution of hijack success rates across 1,000 simulated attackers targeting all other ASes. Each bin represents the percentage of successful hijacks achieved by one attacker after poisoning. Right: Distribution of the number of poisonous links required per attacker to achieve their respective success rates.

were able to successfully hijack more than 80% of all other ASes on the internet after poisoning DFOH's knowledge base.

The right plot in Figure 4 illustrates the number of poisonous links required to achieve these success rates. Crucially, the attack often requires only a small number of carefully chosen links. Approximately 15% of the attackers needed zero additional links (which represents the false negatives), while around 14% required only one link and 28% needed two links. This demonstrates that attackers do not need to inject a large volume of malicious announcements to significantly compromise DFOH's detection capabilities.

These findings highlight the attack's practicality: by exploiting public data reliance, classification errors, and the potential to augment connectivity, a few crafted links can severely degrade DFOH's hijack detection.

D. Naive Defense

A seemingly straightforward approach to counter the knowledge base poisoning attack might be to remove features from the DFOH model that are considered easily manipulable. For instance, given that PeeringDB information can be influenced by crafted announcements targeting specific geographic or infrastructural overlaps (as discussed in Section II-C), one might consider removing the PeeringDB feature category entirely.

However, this naive defense strategy has significant drawbacks. Firstly, DFOH's strength lies in its combination of diverse feature categories, each contributing uniquely to detection accuracy across different scenarios, as indicated by their importance scores (Table I). Removing a feature category, even a potentially vulnerable one, can significantly degrade the system's overall performance, leading to lower detection rates (True Positive Rate) and potentially higher false alarms (False Positive Rate) for legitimate events.

Secondly, attackers are not necessarily limited to manipulating only the most obvious features; they can craft announcements to influence topological features or AS-path patterns as well. A defense strategy based solely on removing features ignores the potential for broader manipulation and sacrifices detection capability. Therefore, simply removing potentially manipulable features is not a robust defense against strategic poisoning attacks.

IV. Exploiting BEAM

A. BEAM's Architecture

BEAM (BGP sEmAntics aware network eMbedding) is a system designed to detect BGP anomalies by understanding the typical behavior of each AS within the Internet's routing landscape. Instead of just looking at path changes, BEAM learns these roles by analyzing the underlying structure of AS business relationships (like provider-customer or peer-to-peer links), which fundamentally dictate how routes are propagated [14].

The core process involves two main steps:

1) **AS Graph Construction:** BEAM utilizes datasets describing AS relationships (e.g., from CAIDA [27]) to build a graph representing the inter-domain topology. In this graph, ASes are nodes, and directed edges represent relationships like provider-to-customer.
2) **AS Embedding:** BEAM then employs a network representation learning model to generate a low-dimensional vector (an embedding) for each AS. This process is specifically designed to capture two key semantic properties derived from the AS graph:
   - *Proximity:* This measures how similar ASes are, considering both direct connections and shared neighbors (i.e., having similar connections to other ASes).
   - *Hierarchy:* This captures an AS's position within the Internet's structure, mainly based on provider-customer links. Tier-1 providers sit high in the hierarchy, while customer ASes are lower.

   The model optimizes these embedding vectors so that the distance between vectors reflects the difference in the ASes' routing roles based on both proximity and hierarchy. The difference between any two AS roles can be measured using a function derived from these embeddings.

BEAM uses these learned AS embeddings to evaluate BGP route changes. When a new route announcement for a prefix is observed, replacing a previous path, the system calculates a **path difference score**. This score quantifies the overall change in routing roles between the sequence of ASes in the old path and the new path. It is computed by comparing the embedding vectors of the ASes in both paths, often using an algorithm like Dynamic Time Warping (DTW) which measures the similarity between two temporal sequences [28].

To determine if a calculated path difference score indicates an anomaly, BEAM uses a **dynamic threshold**. This threshold is not fixed; it is periodically recalculated based on the distribution of path difference scores observed for route changes in a recent historical window (e.g., the previous hour). A path difference score exceeding this dynamic threshold is flagged as suspicious.

B. Polluting the Threshold

Beyond manipulating the path characteristics themselves, BEAM's architecture presents a distinct vulnerability related to its dynamic threshold mechanism. Unlike the knowledge base poisoning attack effective against DFOH (Section III), this attack targets the statistical basis used by BEAM to distinguish between normal and anomalous route changes.

The core idea of the threshold pollution attack is for an adversary to inject a controlled volume of crafted BGP announcements. These announcements are designed to generate path difference scores that fall just below the *current* threshold, thereby initially evading detection. However, these injected ”slightly abnormal but not anomalous” scores are then included in the pool of data used to calculate the threshold for the *next* time window.

By persistently injecting such announcements, the attacker artificially inflates the statistics (e.g., mean, standard deviation) of the path difference scores considered ”normal”. This manipulation forces the system to calculate a higher threshold for the subsequent period. An elevated threshold makes it easier for the attacker's actual malicious announcements, such as those implementing a hijack (which inherently have significantly different routing roles and thus higher path difference scores), to fall below the inflated threshold and avoid being flagged as anomalous.

An attacker can anticipate or calculate the dynamic threshold because the inputs to its calculation are derived from publicly observable BGP data. By accessing public BGP monitoring feeds and potentially having access to a trained BEAM model or a functional equivalent, an attacker can observe the same route changes as the detection system and estimate the resulting threshold. This knowledge allows them to calibrate their injected announcements effectively for the pollution attack.

A potential challenge for this attack is injecting a sufficient volume of announcements within the observation window to significantly skew the statistics. However, the natural behavior of BGP provides an amplification factor. Due to routing oscillations and convergence processes, a single logical route announcement for a prefix is often repeated multiple times by routers across the network. Based on measurements from March 2024 [29], a single prefix announcement was observed, on average, 6.43 times per hour (with a standard deviation of 17.79). This inherent repetition means an attacker only needs to initiate a relatively small number of distinct crafted announcements, as BGP dynamics will amplify their presence in the data streams monitored by systems like BEAM, making threshold manipulation feasible with less effort than might be initially assumed.

C. Evaluation

We simulated the threshold pollution attack against BEAM to evaluate its practical impact. The simulation was performed on a machine equipped with an Apple M3 Pro CPU. We selected 5 random ASes to act as the attackers. For each attacker, we simulated the injection of a series of crafted BGP announcements. These announcements involved modifying legitimate paths originating from the attacker's AS by inserting a forged origin, carefully chosen such that the resulting path difference score would fall just below BEAM's current detection threshold. This strategy aims to have the announcements classified as legitimate while contributing to the statistics used for the next threshold calculation.

To model the amplification effect inherent in BGP, we analyzed real BGP update data from March 2024 [29] specifically for prefixes announced by the chosen attacker ASes. We measured the typical oscillation for these prefixes and used this observed rate to simulate how many times each distinct crafted announcement would likely appear in the monitoring data within the relevant time window. We varied the number of distinct polluting announcements initiated by the attacker from 1 to 50. After injecting the amplified set of announcements based on the observed oscillations, we recalculated BEAM's detection threshold based on the polluted data distribution. Finally, we determined the percentage of prefixes that could subsequently be hijacked using a forged-origin attack without triggering detection under the newly inflated threshold.

The results, illustrated in Figure 5, demonstrate the effectiveness of this approach. We found that for most of the simulated attacker ASes, initiating just 10 distinct polluting announcements (amplified according to their observed prefix oscillation rates) was sufficient to raise the detection threshold by approximately 5%. This seemingly modest increase in the threshold translated into a significant practical advantage for the attacker, allowing them to successfully hijack an additional 10% of prefixes without being detected by BEAM compared to the baseline scenario without pollution. This highlights the practical risk posed by threshold pollution attacks, exploiting the statistical nature of the detection mechanism and the inherent amplification within BGP.

D. Naive Defense

A seemingly straightforward defense against the threshold pollution attack described in Section IV-B would be to abandon the global dynamic threshold and instead maintain a separate, dynamically calculated threshold for each individual IP prefix. The intuition is that polluting the threshold for

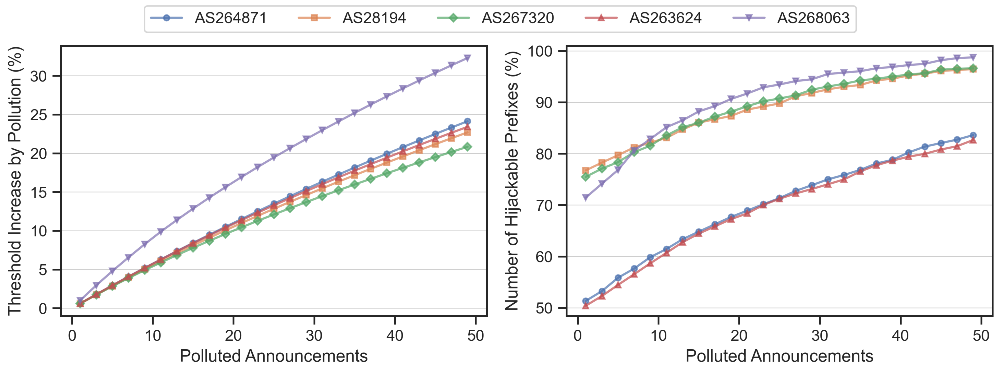


Fig. 5. Effectiveness of the BEAM threshold pollution attack. The plot shows the percentage of prefixes an attacker can successfully hijack (evading detection) as a function of the number of distinct polluting announcements initiated.

one prefix would not affect the thresholds for others, thus containing the attack's impact.

However, this per-prefix threshold approach, while conceptually simple, faces significant practical challenges that render it infeasible for a global monitoring system like BEAM:

- **Scalability Issues:** The global routing table contains over one million IPv4 prefixes and over 220,000 IPv6 prefixes (as of mid 2025) [30]. Maintaining, calculating, and storing a dynamic threshold for each of these prefixes would introduce immense computational and storage overhead, making the system difficult to scale and manage effectively.
- **Data Sparsity and Threshold Instability:** Calculating a reliable dynamic threshold requires a sufficient volume of historical data (legitimate route changes) for statistical significance. Many prefixes, especially more specific ones or those belonging to smaller organizations, exhibit very infrequent route changes. For such prefixes, there would be insufficient data within typical recalculation windows (like an hour) to establish a stable and accurate threshold. Attempting to calculate thresholds with sparse data would lead to high volatility, causing the threshold to fluctuate wildly and trigger constant false positives for minor, legitimate changes or, conversely, become too permissive and miss actual anomalies.

Therefore, while per-prefix thresholds might theoretically isolate pollution effects, the practical hurdles related to scalability and the need for robust statistical baselines based on sufficient data make this approach unworkable for a comprehensive BGP anomaly detection system.

V. Countermeasures

The vulnerabilities exposed in monitor-based defenses necessitate exploring more robust security strategies. Long-term solutions aim to fundamentally enhance BGP's security architecture. Protocols like BGPSec [10] offer cryptographic validation of the entire AS path, while alternative architectures like SCION [31] propose a complete overhaul of inter-domain routing with security built-in. However, despite significant research and efforts to incentivize adoption [32], [33], widespread deployment of such fundamental changes faces immense practical hurdles due to the scale and inertia of the existing internet infrastructure. Modifying a protocol as pervasive as BGP remains a formidable challenge.

Given the slow progress of long-term solutions, short-term countermeasures focus on mitigating the risks within the current BGP ecosystem. One direction involves reducing the transparency of the detection system to potential attackers, essentially making it a ”blackbox”. By keeping the specific algorithms, features, thresholds, and potentially the training data confidential (e.g., in commercial offerings or non-open-source systems), defenders aim to make it significantly harder for attackers to probe the system for weaknesses, calculate false negative rates, or precisely tailor poisoning attacks like those demonstrated against DFOH and BEAM. However, this ”security through obscurity” approach has inherent limitations. Determined attackers might still infer system behavior through careful observation or probing, and it hinders collaborative security research and independent verification.

Another, potentially complementary, short-term strategy involves augmenting public data with private BGP data feeds or out-of-band validation mechanisms. Systems like ARTEMIS [9] exemplify this by combining publicly available monitor data with routing information directly obtained from the Routing Information Bases (RIBs) of participating routers. This allows for cross-validation; if a route change observed publicly is inconsistent with the private RIB data, it raises suspicion. However, even incorporating private data is not a panacea. While ARTEMIS and similar approaches can effectively detect anomalies for prefixes announced by ASes contributing private data, their visibility remains limited. Attackers can still exploit the vastness of the internet topology by using random or unused ASes, which are unlikely to be covered by private monitoring arrangements, to inject polluted routes or launch hijacks. This manipulation can still poison the public data components relied upon by hybrid systems or evade detection entirely if the attack path does not traverse the privately monitored infrastructure.

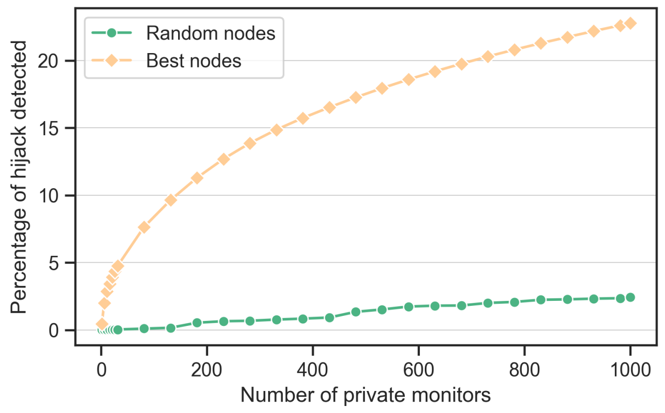


Fig. 6. Hijack detection rate improvement when using an increasing number of private monitors, comparing random selection versus best-case selection.

To quantify the potential benefits and limitations of private monitoring, we simulated its impact on detecting the knowledge base poisoning attacks described earlier. The simulation was conducted on a machine equipped with an Apple M3 Pro CPU. We modeled scenarios where a varying number of ASes (from 1 to 1,000) act as private monitors, providing ground truth for routes they observe. The core logic involved checking if any of the poisoned links injected during the simulated DFOH attack (Section III-C) would be directly observed by one of the designated private monitors. If a poisoned link involved a direct connection to a private monitor AS, it was considered detected. We compared two selection strategies: randomly choosing monitor ASes versus an optimal (best-case) selection that maximizes visibility into potential attack paths based on the simulated attack data. As shown in Figure 6, increasing the number of monitors improves detection. However, even with 1,000 randomly selected private monitors — a significant deployment — the detection rate for poisoned links remained below 3%. This poor performance stems from the vast scale of the internet topology; with tens of thousands of ASes, the probability that an attacker's chosen AS for poisoning happens to have a direct peering relationship with one of the randomly placed private monitors is inherently low. The best-case selection strategy yielded better results, detecting over 20% of attacks with 1,000 monitors, but still leaving a large fraction undetected. This highlights that even substantial private monitoring infrastructure struggles against attackers who can carefully choose where to inject malicious announcements, underscoring the challenge of achieving comprehensive BGP security through monitoring alone.

VI. Related Work

Securing the Border Gateway Protocol (BGP) has been a long-standing challenge. The Resource Public Key Infrastructure (RPKI) represents a significant effort to cryptographically validate the origin of route announcements [7], [34]. However, its incomplete deployment and inherent vulnerability to specific attacks like forged-origin hijacks necessitate complementary defense mechanisms. Consequently, BGP monitoring systems, often leveraging public data collectors like RouteViews [11] and RIPE RIS [12], have become crucial. Early approaches focused on heuristic-based anomaly detection; simpler tools like BGPmon [35] and BGPalerter [36] alert operators to basic events like origin AS changes or RPKI-invalid announcements. Some systems, such as Cloudflare Radar [37], incorporate additional data sources like Internet Routing Registries (IRR), though IRR data limitations regarding consistency and completeness restrict its reliability [38], [39]. Other systems, like ARTEMIS [9], aim to improve detection speed and mitigation by combining public monitor data with private RIB feeds, addressing some limitations of purely public monitor-based approaches. More recent approaches increasingly incorporate machine learning (ML) to identify complex suspicious routing events. Systems like DFOH [13] utilize ML to detect forged-origin hijacks by analyzing path features derived from public data.

The application of ML in security domains, however, is fraught with challenges. Researchers have identified common pitfalls in the design, implementation, and evaluation of learning-based security systems, which can lead to unrealistic performance claims and hinder practical deployment [40]. A major concern is the vulnerability of ML models to adversarial attacks, where malicious inputs are crafted to cause misclassification [16], [17]. Evaluating the robustness of ML models against such attacks is critical, especially in security contexts [41]. The threat of adversarial attacks against ML specifically in network security has been surveyed, highlighting the adversarial nature inherent in tasks like intrusion and malware detection [42].

Within BGP security, research has explored how attackers can evade monitoring systems. Studies have demonstrated techniques to launch hijacks, such as sub-prefix hijacks using communities, that remain hidden from public monitors [43]. Further investigations have analyzed the effectiveness of attackers deliberately crafting BGP paths to avoid propagation to route collectors, showing that even with expanded monitoring infrastructure, visibility gaps can persist [44]. While these works address the fundamental limitations of monitor visibility, they do not specifically target the vulnerabilities within the ML models used by modern defense systems. Our work differs by focusing explicitly on the susceptibility of ML-based BGP hijack detection systems like DFOH and BEAM to adversarial manipulation, demonstrating how techniques like knowledge base poisoning and threshold pollution can undermine their effectiveness, an area that remains relatively underexplored.

VII. Conclusion

This paper investigated the security vulnerabilities inherent in modern, data-driven BGP hijack detection systems that rely heavily on public monitoring infrastructure. While these systems, often employing sophisticated machine learning techniques like those in DFOH and BEAM, represent advancements, we demonstrated their susceptibility to adversarial manipulation. Our analysis and simulations revealed that attackers can effectively undermine these defenses by strategically injecting crafted BGP announcements to poison knowledge bases or pollute statistical thresholds, allowing malicious hijacks to evade detection. This vulnerability stems from the fundamental limitations of relying on passively collected, publicly available BGP data, which lacks inherent verification mechanisms.

We examined potential countermeasures, including black-boxing systems and augmenting public data with private feeds. While these approaches offer partial mitigation, they are not foolproof. Security through obscurity has inherent weaknesses, and even significant private monitoring infrastructure struggles to provide comprehensive coverage. Our findings underscore a critical need: securing inter-domain routing requires moving beyond defenses solely reliant on observing public BGP data streams. The ease with which this data can be manipulated necessitates a shift towards more robust, multi-faceted security strategies that incorporate stronger validation mechanisms and consider the adversarial nature of the environment in their core design, ultimately demanding a more holistic approach than passive monitoring alone can provide.

References

[1] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of bgp security issues and solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2010.

[2] A. Siddiqui, "Not just another BGP Hijack," https://manrs.org/2020/04/not-just-another-bgp-hijack/, 2020.

[3] R. NCC, "YouTube Hijacking: A RIPE NCC RIS case study," https://www.ripe.net/about-us/news/youtube-hijacking-a-ripe-ncc-ris-case-study/, 2008.

[4] A. Siddiqui, "KlaySwap – Another BGP Hijack Targeting Crypto Wallets," https://www.manrs.org/2022/02/klayswap-another-bgp-hijack-targeting-crypto-wallets, 2022.

[5] R. B. bm, "Hackers emptied Ethereum wallets byreaking the basic infrastructure of the internet," 2018.

[6] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1," RFC 8210, Sep. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8210

[7] N. Rodday, Í. Cunha, R. Bush, E. Katz-Bassett, G. D. Rodosek, T. C. Schmidt, and M. Wählisch, "The resource public key infrastructure (rpki): A survey on measurements and future prospects," *IEEE TNSM*, 2023.

[8] D. Madory, "RPKI ROV Deployment Reaches Major Milestone," https://manrs.org/2024/05/rpki-rov-deployment-reaches-major-milestone/, 2024.

[9] P. Sermpezis, V. Kotronis, P. Gigis, X. Dimitropoulos, D. Cicalese, A. King, and A. Dainotti, "ARTEMIS: Neutralizing BGP Hijacking Within a Minute," *IEEE/ACM ToN*, 2018.

[10] M. Lepinski and K. Sriram, "BGPsec Protocol Specification," RFC 8205, Sep. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8205

[11] U. of Oregon, "RouteViews Project," www.routeviews.org/routeviews/, 2024.

[12] R. NCC, "Routing Information Service (RIS)," www.ripe.net/data-tools/stats/ris/, 2024.

[13] T. Holterbach, T. Alfroy, A. D. Phokeer, A. Dainotti, and C. Pelsser, "A System to Detect Forged-Origin Hijacks," in *Proc. USENIX NSDI*, 2024.

[14] Y. Chen, Q. Yin, Q. Li, Z. Liu, K. Xu, Y. Xu, M. Xu, Z. Liu, and J. Wu, "Learning with Semantics: Towards a Semantics-Aware Routing Anomaly Detection System," in *Proc. USENIX Security*, 2024.

[15] Cisco Systems, "Thousandeyes," n.d. [Online]. Available: https://www.thousandeyes.com/

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015. [Online]. Available: https://arxiv.org/abs/1412.6572

[17] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, p. 317–331, Dec. 2018. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2018.07.023

[18] L. Blunk, C. Labovitz, and M. Karir, "Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format," RFC 6396, Oct. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6396

[19] J. Scudder, R. Fernando, and S. Stuart, "BGP Monitoring Protocol (BMP)," RFC 7854, Jun. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7854

[20] T. Shapira and Y. Shavitt, "A deep learning approach for ip hijack detection based on asn embedding," in *Proceedings of the Workshop on Network Meets AI & ML*, ser. NetAI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 35–41. [Online]. Available: https://doi.org/10.1145/3405671.3405814

[21] Y. Dong, Q. Li, R. O. Sinnott, Y. Jiang, and S. Xia, "Isp self-operated bgp anomaly detection based on weakly supervised learning," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, 2021, pp. 1–11.

[22] "Servperso systems – bgp and lir services for small network operators," https://www.servperso.net/, 2025.

[23] "ifog gmbh – web hosting, vps, ip transit, and lir services," https://ifog.ch/en, 2025.

[24] R. Bush, "Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI)," RFC 7115, Jan. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7115

[25] "The Interconnection Database," https://www.peeringdb.com/.

[26] L. Gao and J. Rexford, “Stable internet routing without global coordination,” *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 681–692, 2001.

[27] “Caida as relationships dataset,” https://www.caida.org/catalog/datasets/as-relationships/, 2025.

[28] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series,” in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, ser. AAAIWS’94. AAAI Press, 1994, p. 359–370.

[29] U. of Oregon Route Views Project, “Route views bgp update archive - march 2024,” https://archive.routeviews.org/route-views.wide/bgpdata/2024.03/UPDATES/, 2024.

[30] T. Bates, P. Smith, and G. Huston, “Cidr report,” https://www.cidr-report.org/, 2025, accessed: 2025-04-29.

[31] C. de Kater, N. Rustignoli, and A. Perrig, “SCION Overview,” Internet Engineering Task Force, Tech. Rep., 2024.

[32] P. Gill, M. Schapira, and S. Goldberg, “Let the market drive deployment: a strategy for transitioning to BGP security,” in *Proc. ACM SIGCOMM*, 2011.

[33] T. Hlavacek, I. Cunha, Y. Gilad, A. Herzberg, E. Katz-Bassett, M. Schapira, and H. Shulman, “Disco: Sidestepping rpki’s deployment barriers,” in *Proc. NDSS*, 2020.

[34] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman, “Are we there yet? on rpki’s deployment and security,” in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 02 2017.

[35] H. Yan, R. Oliveira, K. Burnett, D. Matthews, L. Zhang, and D. Massey, “Bgpmon: A real-time, scalable, extensible monitoring system,” in *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*, ser. CATCH ’09. USA: IEEE Computer Society, 2009, p. 212–223. [Online]. Available: https://doi.org/10.1109/CATCH.2009.28

[36] N. G. I. Network, “Bgpalerter,” https://github.com/nttgin/BGPalerter, 2019.

[37] M. Zhang and C. Martinho, “Cloudflare Radar’s new BGP origin hijack detection system,” https://blog.cloudflare.com/bgp-hijack-detection/, 2023.

[38] D. R. McPherson, S. Amante, E. Osterweil, L. Blunk, and D. Mitchell, “Considerations for Internet Routing Registries (IRRs) and Routing Policy Configuration,” RFC 7682, Dec. 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7682

[39] B. Du, G. Akiwate, T. Krenc, C. Testart, A. Marder, B. Huffaker, A. C. Snoeren, and K. Claffy, “Irr hygiene in the rpki era,” in *Passive and Active Measurement*, O. Hohlfeld, G. Moura, and C. Pelsser, Eds. Cham: Springer International Publishing, 2022, pp. 321–337.

[40] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, “Pitfalls in machine learning for computer security,” *Commun. ACM*, vol. 67, no. 11, p. 104–112, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3643456

[41] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, “On evaluating adversarial robustness,” 2019. [Online]. Available: https://arxiv.org/abs/1902.06705

[42] O. Ibitoye, R. Abou-Khamis, M. el Shehaby, A. Matrawy, and M. O. Shafiq, “The threat of adversarial attacks on machine learning in network security – a survey,” 2023. [Online]. Available: https://arxiv.org/abs/1911.02621

[43] H. Birge-Lee, M. Apostolaki, and J. Rexford, “Global bgp attacks that evade route monitoring,” 2024. [Online]. Available: https://arxiv.org/abs/2408.09622

[44] A. Milolidakis, T. Bühler, K. Wang, M. Chiesa, L. Vanbever, and S. Vissicchio, “On the effectiveness of bgp hijackers that evade public route collectors,” *IEEE Access*, vol. 11, pp. 31 092–31 124, 2023.

APPENDIX

This appendix offers a few other insights into the work I did throughout this thesis, covering simulation details, implementation challenges, and ideas for future exploration.

I. GETTING THE PATHS RIGHT: SIMULATING HIJACK PROPAGATION

One of the tricky parts of simulating these hijack attacks was figuring out which paths would actually make it to the public BGP monitors that systems like DFOH and BEAM rely on.

At first, I tried building a simulator based on the classic Gao-Rexford model. The result of that was a Python framework, which could be connected to a React interface to visualize the propagation of BGP hijacks (as shown in Figure 7), but it was too theoretical. Real-world BGP is messy – ASes have their own preferences, use BGP communities, have hidden policies, and the network is always changing. The simple model just couldn't capture all that.

To get a more grounded evaluation, I switched to a more conservative strategy. Instead of trying to predict propagation, I looked backward. I gathered all the AS paths observed by public monitors (RouteViews and RIPE RIS) that originated from the simulated attacker's AS over a long period (around 300 days, using data collected by DFOH). By feeding the defense systems this superset of potentially propagatable paths, we get a conservative estimate – essentially assuming the attacker could potentially reach any monitor they've historically reached. This avoids the pitfalls of inaccurate propagation modeling.

II. MAKING DFOH SIMULATIONS BEARABLE: PERFORMANCE TUNING

Evaluating the DFOH poisoning attack meant running simulations on a massive scale: testing 1,000 randomly chosen attacker ASes against each of the roughly 80,000 other ASes in the March 2024 topology. With each simulation taking several seconds, the total compute time was looking prohibitive, even on a 96-core machine.

Digging into the performance bottlenecks revealed the culprits weren't the ML parts, but rather specific feature calculations:

- *PeeringDB features:* Calculating cosine distances between large vectors representing AS presence in IXPs and facilities was slow.
- *Topological features:* Running numerous graph algorithms on the AS topology also took significant time.

These were originally implemented in Python. To lower the compute time, I decided to rewrite these parts in C, which is much faster for heavy computations. The process involved three main steps:

1) **Reverse Engineering:** Carefully analyzed the original code and reviewed the bottlenecks.
2) **Rewriting in C:** The computationally heavy parts (PeeringDB and topological feature calculations) were re-implemented in C, using more efficient algorithms and data structures.
3) **Python Bindings:** Created a Python library to easily call the optimized C code from the existing simulation framework.

This optimization yielded substantial performance gains. As shown in Figure 8, the PeeringDB feature calculation sped up by about 13.6x, and topological features improved by a factor of 21.1x (based on the mean of 100 runs). Crucially, the overall time for a single attack simulation dropped from around 6 seconds to just about 1 second. This brought the total simulation time down from an infeasible duration to a manageable number of days on the 96-core machine.

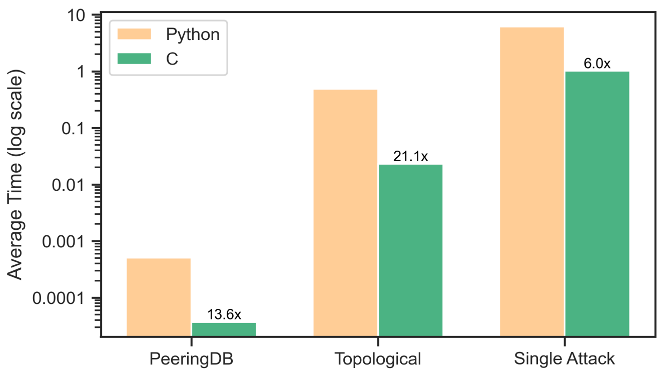


Fig. 8. Performance boost from rewriting DFOH's feature calculation bottlenecks in C. The optimized C versions significantly outperformed the original Python implementations, making large-scale simulations feasible.

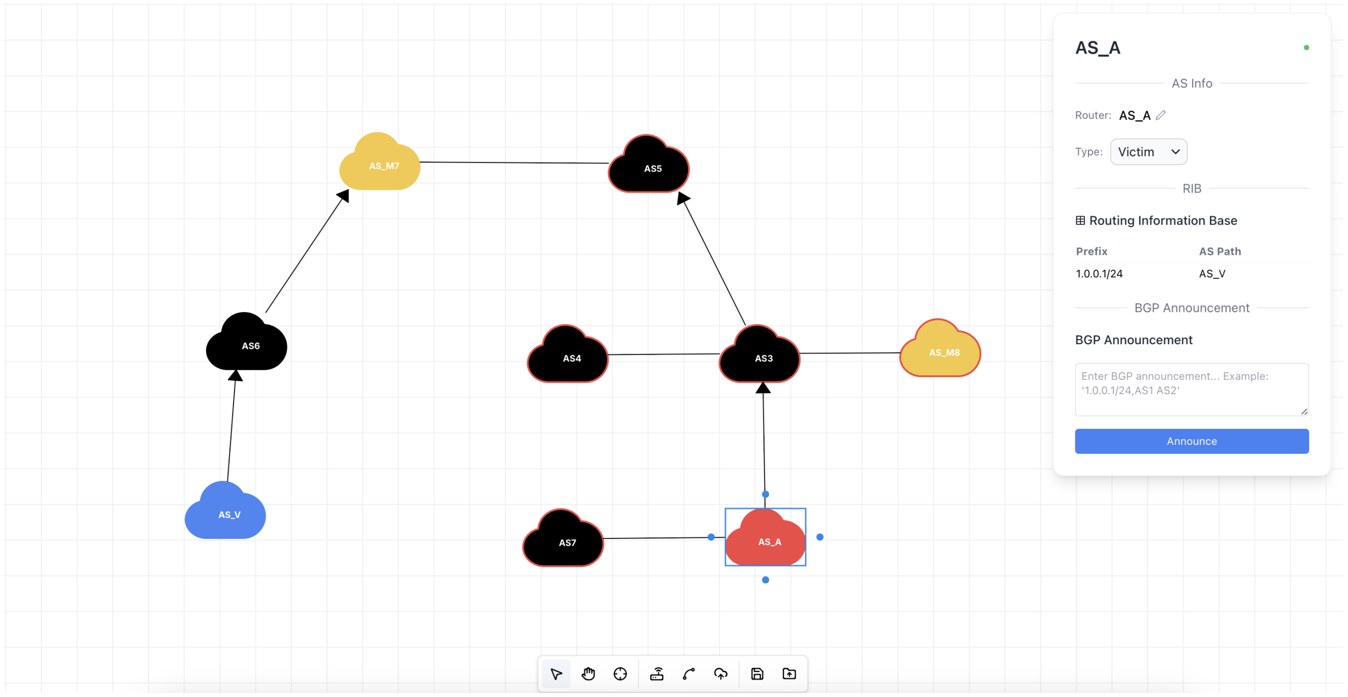
III. BEAM: CHALLENGES AND POTENTIAL ATTACK VECTORS

Working with BEAM presented its own set of challenges. Training the AS embedding model was time-consuming (several hours), and reverse-engineering the system revealed potential issues in its design.

One notable observation concerns the generation of training samples. BEAM uses the CAIDA AS relationships dataset, treating observed links as positive samples. Negative samples are generated, in part, by inverting positive samples. For example, if 'AS A - AS B' (customer-provider) is a positive sample, 'AS B - AS A' might be included as a negative sample. The model is trained to decrease the distance between embeddings of positive pairs and increase the distance for negative pairs. This creates a conflict: the model simultaneously tries to pull A and B closer (due to the positive sample) and push them further apart (due to the inverted negative sample). This design flaw could lead to suboptimal embeddings, potentially affecting the model's performance.

Given these complexities, exploring attacks beyond threshold pollution proved difficult. However, two potential attack vectors are worth noting:

1) **AS Relationship Poisoning:** Could an attacker manipulate the input AS relationship dataset to make their AS appear "closer" to a target victim AS in the embedding space? This is non-trivial because BEAM's path differ-



AS_A

AS Info

Router: AS_A

Type: Victim

RIB

Routing Information Base

Prefix	AS Path
1.0.0.1/24	AS_V

BGP Announcement

BGP Announcement

Enter BGP announcement... Example: '1.0.0.1/24,AS1 AS2'

Announce

Fig. 7. Screenshot of the initial BGP propagation simulator interface, based on the Gao-Rexford model. While useful for visualization, it lacked the fidelity needed for accurate attack simulation.

ence score considers the entire sequence of ASes, not just the attacker and victim.

2) **Path Padding/Pollution:** An attacker might try to manipulate the path difference score directly by padding the AS path of a hijack announcement. For an attacker H hijacking victim V's prefix, they could announce a path like 'H - X - V', where X is an intermediary AS chosen specifically to minimize the overall path difference score calculated by BEAM. The main drawback is that longer paths are generally less preferred in BGP routing, potentially limiting the reach and effectiveness of such a hijack.

IV. Future Work

Based on the findings and challenges encountered in this work, several areas for future research emerge:

- **Making BEAM Attacks More Feasible:** The threshold pollution attack demonstrated against BEAM worked in simulation, but it assumes the attacker can easily amplify their announcements. Future research could explore more sophisticated ways to inject malicious data or investigate entirely different attack strategies that might be stealthier or more practical in the real world.
- **Testing Monitor Poisoning in the Wild:** The idea that attackers can poison public monitor data by carefully crafting BGP announcements needs real-world validation. Experiments using platforms like PEERING to inject controlled announcements and see if they actually get picked up by monitors would provide strong evidence for (or against) this attack vector's feasibility.
- **Building Stronger Defenses:** The vulnerabilities shown here highlight that defenses relying solely on passively observed public data might be too fragile. The next step should be designing and evaluating more robust countermeasures. This could involve smarter ways to combine public and private BGP data, exploring different machine learning models less prone to poisoning, or developing techniques to cross-validate BGP data using diverse, independent sources.