USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Ares: Comprehensive Path Hijacking Detection via Routing Tree

Yinxiang Tao, *Institute of Network Sciences and Cyberspace, Tsinghua University, Beijing, China;* Chengwan Zhang, *unaffiliated;* Changqing An, *Institute of Network Sciences and Cyberspace, Tsinghua University, Beijing, China;* Shuying Zhuang, *Zhongguancun Laboratory, Beijing, China;* Jilong Wang, *Quan Cheng Laboratory, 250103, Jinan, Shandong, China and Institute of Network Sciences and Cyberspace, Tsinghua University, Beijing, China;* Congcong Miao, *Tencent*

https://www.usenix.org/conference/usenixsecurity25/presentation/tao

This paper is included in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

Ares: Comprehensive Path Hijacking Detection via Routing Tree

Yinxiang Tao[1*], Chengwan Zhang[*], Changqing An[1],
Shuying Zhuang[2†], Jilong Wang[3,1], Congcong Miao[4†]

[1]Institute of Network Sciences and Cyberspace, Tsinghua University, Beijing, China
[2]Zhongguancun Laboratory, Beijing, China
[3]Quan Cheng Laboratory, 250103, Jinan, Shandong, China
[4]Tencent

tyx23@mails.tsinghua.edu.cn, *x10000zhang@gmail.com*,
{acq,wjl}@cernet.edu.cn, *zhuangsy@mail.zgclab.edu.cn*, *mccmiao@163.com*

Abstract

Since Border Gateway Protocol (BGP) lacks a strong security mechanism, prefix hijacking attacks are becoming increasingly rampant, which has drawn a lot of attention from both academia and industry. Recently, prefix hijacking has evolved from origin hijacking to more stealthy hijacking, i.e., *path hijacking*, to bypass existing hijacking detection systems. The attacker will manipulate the AS path attributes while announcing the prefix of the victim AS. However, existing systems only target origin hijacking or only address part of the path hijacking, which allows attackers to exploit vulnerabilities to hijack. In this paper, our observation shows that path hijacking triggers the creation of a new observed prefix's routing tree (OPRT) within an AS and we advocate for a radical new approach to comprehensively address all types of path hijacking. We propose a *first-of-its-kind* system, called **Ares**, to detect path hijacking in an *effective*, *accurate*, and *fast* way. At the core of Ares is weighted edit distance to quantify the differences between routing trees, combined with a clustering mechanism to accelerate anomaly detection and heuristic rules to further increase the detection accuracy. We validate Ares with historical hijacking events and large-scale simulations, For each of the 12 real-world events, Ares was able to detect the hijacking within 5 minutes of its occurrence. Additionally, simulations show that Ares detects an average of 97.2% and 99.3% of stealthy exact and sub-prefix path hijackings targeting Tier-1 and content ASes with only 1.06% of false positive rate, *outperforming* state-of-the-art methods. In addition, it generates only 2.31 suspicious alerts per hour across the entire Internet, a manageable volume for operators to investigate and respond effectively.

1 Introduction

The Internet consists of over 80,000 Autonomous Systems (ASes). To exchange information with each other, ASes use Border Gateway Protocol (BGP) [40] to establish inter-domain routes on the Internet. However, the initial design of BGP lacked information verification mechanisms, making it vulnerable to malicious users.

Attackers can manipulate BGP routes to redirect traffic and achieve their malicious goals, a technique known as prefix hijacking. Prefix hijacking can be categorized into *origin hijacking* and *path hijacking*. In origin hijacking, the attacker directly announces the victim's prefixes, which is relatively easy to detect due to the resulting Multiple Origin Autonomous Systems (MOAS) conflict. In path hijacking, the attacker manipulates the AS-PATH when announcing the victim's prefix, allowing them to hijack the path while avoiding MOAS conflicts [52]. Misconfigurations [3–5] or malicious attacks [6, 8] can lead to origin hijacking, but path hijacking is often a *deliberate malicious* action. Recently, several incidents have shown that attackers used path hijacking to steal Bitcoin [9, 10].

However, path hijacking remains an unsolved problem due to its stealthiness. Firstly, path hijackings do not raise false prefix-origin pairs, rendering RPKI-ROV [36] and origin hijacking detection systems [33, 39, 43] ineffective. Secondly, existing state-of-the-art detection methods are unable to detect path hijacking comprehensively. They fail to detect sophisticated hijacking attacks, such as the Defcon attack [7] or man-in-the-middle (MITM) attacks, and show low detection rates for specific scenarios of path hijacking. Specifically, AS link-based systems [29, 49] detect path hijacking through fake links but fail to identify the Defcon attack which does not incur any fake links. Valley-free-based systems [31, 46] would confuse path hijacking with legitimate non-valley-free routes, thus providing false alarms. Probing-based [43, 46, 51] systems cannot detect hijacking with man-in-the-middle (MITM) attacks. Artemis [45] can detect all path hijacking, but it relies on AS secret information and is limited to detecting hijacking only for the AS that deploys it. In short, no system currently exists that can comprehensively detect path hijacking, allowing latent hijacking to go undetected in the wild.

Vision. Our measurements reveal that ASes typically announce prefixes in batches, in line with their routing policies,

[*]These authors contributed equally to this work and share first authorship.
[†]These authors are the corresponding authors.

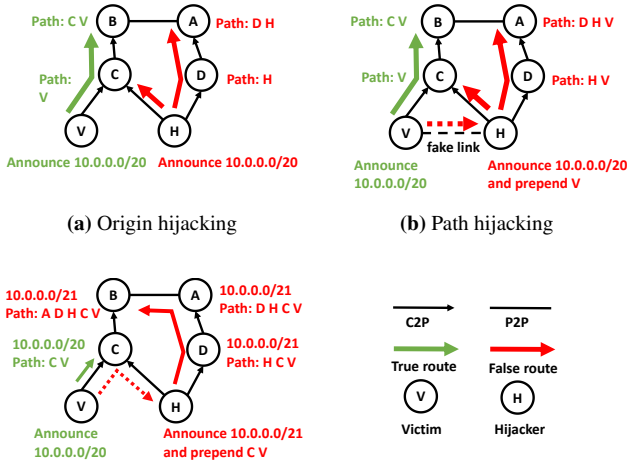
to achieve specific goals such as traffic engineering. These prefix announcements can be represented using *Observed Prefix Routing Trees (OPRTs)*, which are a rooted tree induced by all AS-PATHs from *vantage points (VP)* to target prefixes. We observe that more than 80% of ASes are associated with fewer than 10 OPRTs (see Section 3). Path hijacking inherently disrupts normal prefix announcements, causing alterations in the OPRT structure within the affected ASes and potentially creating new OPRTs. This disruption offers a promising method for detecting various forms of path hijacking.

Based on our observations, we propose *Ares*, the first routing tree-based system for efficient and accurate detection of all Internet path hijacking types. Ares detects path hijacking by monitoring changes in the number of OPRT clusters associated with an AS. It solves algorithmic challenges in detection *coverage* and *accuracy*, like distinguishing OPRTs between normal and hijacked prefixes, and system challenges like clustering *efficiency*. To achieve accurate and comprehensive path hijacking detection, we propose a weighted edit distance approach that leverages structural differences in trees and link characteristics associated with path hijacking. This minimizes normal OPRT variations while highlighting hijacked ones. To accelerate clustering, we introduce a two-phase method that uses hash values and cluster grouping. We also apply several carefully designed and effective heuristic rules to filter false alarms from normal OPRT changes.

We validate the effectiveness and efficiency of Ares with historical path hijacking events and simulation experiments. Ares can detect all 12 real path hijacking events with an average of only 1.49 alarms per hour. To more comprehensively validate Ares, we propose a novel approach that supports large-scale path hijacking simulations by combining real BGP data with crafted BGP updates. Our large-scale simulation experiments demonstrate that Ares can detect an average of 97.2% and 99.3% of malicious, stealthy exact and sub-prefix path hijackings targeting Tier-1 and content ASes with only 1.06% false positive rate (FPR). This represents a considerable average improvement over existing state-of-the-art methods ([19, 29, 49], see Section 5.2). Furthermore, we deploy Ares in the real world, and it alarmed an average of 2.31 events per hour, a number that is easy for operators to validate.

The contributions of this paper are as follows:

- We design and implement a routing tree-based *real-time* path hijacking detection system capable of detecting all types of path hijacks with *high recall rate*, *low FPR*, and *high-efficiency* performance, enabled by the design of weighted tree edit distance, two-phase clustering, and heuristic filters.
- We propose a novel approach that supports large-scale path hijacking simulations by combining real BGP data with crafted BGP updates, addressing the issue of the scarcity of real path hijacking events.
- We validate Ares' effectiveness through historical events, a large-scale simulated path hijacking dataset, and a month-long real-world deployment. Compared with existing state-of-the-art methods, Ares achieves superior accuracy, low false positive rate, and high efficiency, demonstrating its strong practicality and reliability.



(a) Origin hijacking **(b)** Path hijacking

(c) Defcon hijacking

Figure 1: Prefix hijacking

2 Background

In this Section, we introduce prefix hijacking and its classification (§ 2.1) and then discuss the existing issues of path hijacking detection (§ 2.2).

2.1 Prefix Hijacking

The Border Gateway Protocol (BGP) is introduced to establish inter-domain routes among Autonomous Systems (ASes) and exchange information with each other. However, the original design of BGP lacks a mechanism to authenticate routes, allowing malicious ASes, namely attackers to create and disseminate misleading routing information. Therefore, the traffic will be redirected by false routing information. This is known as prefix hijacking (also called BGP hijacking or route hijacking). As defined by Sermpezis et al. [45], three perspectives can be used to categorize prefix hijacking: announced AS-PATH, affected prefix, and data-plane traffic manipulation. The details are shown in Table 1. From the perspective of announced AS-PATH, prefix hijackings can be classified into two categories: origin hijacking and path hijacking.

In origin hijacking, the attacker will announce the victim AS's IP prefix to its neighbor ASes so that the traffic destined for the victim AS will be redirected to the attacker. For example, as shown in Figure 1a, attacker H announces the prefix 10.0.0.0/20 to its neighbors C and D, while the legitimate owner of this prefix is victim V. The attacker's fake route competes with the legitimate route of the victim V. Typically, two interconnected ASes maintain business relationships cat-

Table 1: The Classification of prefix hijacking

Classification Criteria	Hijacking Type	Explanation
Announced AS-PATH	Type-0 hijacking	i.e. Origin hijacking, the hijacker announces the victim's prefix directly.
	Type-N (N≥1) hijacking	i.e. Path hijacking, where N indicates the length of the path forged by the hijacker.
	Type-U hijacking	i.e. Defcon hijacking, the hijacker prepend a real path to victim.
Affected prefix	Exact prefix hijacking	The hijacker announces a path for exactly the same prefix of victim.
	Sub-prefix hijacking	The hijacker announces a sub-prefix of victim's prefix.
Data-plane traffic manipulation	BH (Blackholing)	The hijacker discards the traffic, effectively blackholing it.
	IM (Imposture)	The hijacker masquerades as the victim host to perform phishing attacks.
	MM (Man-in-the-Middle)	The hijacker redirects traffic to the victim to execute a man-in-the-middle attack.

egorized as either customer-to-provider (C2P/P2C) or peer-to-peer (P2P). We assume that an AS prioritizes routes in the order: customer > peer > provider[1], prefers shorter AS-PATHs when relationships are equal, and favors the victim's route when all else is equal. As a result, This attack results in route pollution in AS A and D, where the traffic destined for victim V will be redirected to the attacker H. As origin hijacking typically introduces false prefix-AS mappings, it can be easily identified by route origin validation (ROV) [36] or existing hijack detection systems based on multiple origin ASes (MOAS) conflicts [31, 33, 39, 45].

To avoid MOAS conflicts, attackers introduce a more stealthy hijacking, i.e., path hijacking (also called forged origin hijacking), in which the attacker manipulates the AS path attributes while announcing the prefix of the victim AS. As shown in Figure 1b, the attacker H prepends victim V when announcing the IP prefix 10.0.0.0/20 to its neighbors C and D. Similarly, this attack causes AS A and D to be polluted, while AS B and C remain unaffected. Here, path hijacking tends to introduce a fake AS link between the attacker and the victim. Recently, a notable variant of path hijacking is Defcon hijacking, or Type-U hijacking as defined by Sermpezis et al. [45]. In this attack, the attacker fabricates an AS-PATH that mimics a legitimate route to the victim AS, typically hijacking a sub-prefix. This is because ASes often prefer shorter routes, but since the forged path results in a longer overall route, Type-U attacks usually fail when attempting exact prefix hijacks. As illustrated in Figure 1c, attacker H announces the sub-prefix 10.0.0.0/21 and prepends AS C and V, causing pollution in AS A, B, and D. Here, AS C remains unaffected due to loop avoidance. Defcon hijacking resembles a route leak, the difference is the sub-prefix does not originate from the victim AS.

2.2 Existing path hijacking issues

Existing hijack detection methods can be divided into control-plane-based, data-plane-based and hybrid-plane-based methods according to the data type they used. Control-plane-based methods primarily utilize control-plane data, i.e., BGP routing data. Public repositories such as RIPE RIS [1] and RouteViews [2] deploy collectors to collect such data, with connected ASes or routers called Vantage Points (VPs). These collectors aggregate BGP data from VPs, enabling researchers and network operators to monitor global routing dynamics. Control-plane-based methods use these data to check whether forged BGP data is forwarded in the Internet. Data-plane-based methods, in contrast, detect hijacks by analyzing data traffic flows. These approaches employ tools like traceroute or ping to monitor actual propagation paths, identifying malicious traffic redirection to anomalous destinations. Although prefix hijack detection has been a long-standing problem, and many hijack detection techniques have been proposed over the past two decades, none of them can address path hijacking thoroughly because attackers have evolved to be more stealthy to hijack the path. Table 2 summarizes the limitations of existing path hijacking detection systems.

• *Capable of bypassing link-based detection systems.* Typically, path hijacking introduces new links, which may be detected by state-of-the-art link-based systems (e.g., Metis [49], DFOH [29]) with promising probability. However, attackers can employ other techniques such as Defcon attacks, which do not create any fake links, rendering these systems ineffective.

• *Detect based on valley-free violation.* Internet routing propagation typically adheres to the valley-free rule [25], which prohibits ASes from forwarding routes received from a provider or peer to another provider or peer. Hijacks often result in valley-free violations, making them detectable by systems that rely on them (e.g., Argus [46], Fingerprints [31], HEAP [43]). However, In real networks, legitimate propagation that violates valley-free rules also exists, leading to high FPR in systems that rely on detecting valley-free violations.

• *Hard to detect and verify via data-plane.* Some ping-based systems, such as iSPY [51] and Argus [46], rely on connectivity checks to hijacked ASes but cannot detect MM and IM types of path hijacking, which are typically employed by malicious attackers. Other systems, such as Fingerprints [31] and HEAP [43], may use host fingerprints to verify the authenticity of hosts, but they are ineffective against the MM type. Furthermore, the data-plane-based systems cannot provide control-plane information, resulting in great verification efforts for operators.

The various issues in existing systems motivate us to design a more comprehensive path hijacking detection system, called Ares, that can *effectively*, *accurately*, and *quickly* detect all types of path hijacking at *internet-wide*.

[1]The widely used routing model assumes that if an AS receives two routes to the same destination, it will prefer the route received from a customer over the route received from a peer or provider

Table 2: Detection of path hijacking by current BGP hijacking detection systems

Class of Path hijacking			Control-plane Based System				Data-plane Based System		Hybrid-plane Based System		
Affected prefix	AS-PATH Type	Date Plane	**Ares (Ours)**	Metis (2023) [49]	DFOH (2024) [29]	Artemis (2018) [45]	iSPY (2008) [51]	LWDS (2007) [53]	Fingerprints (2007) [31]	Argus (2012) [46]	HEAP (2016) [43]
Exact	Type-N (N>=1)	BH	✓	✓	✓	✓	✓	×	✓	✓	✓
Exact	Type-N (N>=1)	MM	✓	✓	✓	✓	×	✓	×	×	×
Exact	Type-N (N>=1)	IM	✓	✓	✓	✓	×	✓	✓	×	✓
Exact	Type-U or Defcon	BH	✓	×	×	✓	✓	×	×	✓	✓
Exact	Type-U or Defcon	MM	✓	×	×	✓	×	✓	×	×	×
Exact	Type-U or Defcon	IM	✓	×	×	✓	×	✓	×	×	✓
Sub	Type-N (N>=1)	BH	✓	✓	✓	✓	×	×	✓	✓	×
Sub	Type-N (N>=1)	MM	✓	✓	✓	✓	×	×	×	×	×
Sub	Type-N (N>=1)	IM	✓	✓	✓	✓	×	×	✓	×	×
Sub	Type-U or Defcon	BH	✓	×	×	✓	×	×	×	✓	×
Sub	Type-U or Defcon	MM	✓	×	×	✓	×	×	×	×	×
Sub	Type-U or Defcon	IM	✓	×	×	✓	×	×	×	×	×
Internet-wide Detection			✓	✓	✓	×	×	✓	✓	✓	✓

3 Observation and Motivation

In this section, we first introduce our observation from internet-wide public routing data that motivates our detecting approach (§ 3.1) and then provide the new challenges (§ 3.2).

3.1 Observation

Through observation and analysis of public BGP data from RIPE [1], we have discovered that ASes typically announce prefixes in batches according to routing policies to achieve specific goals, such as traffic engineering. Prefixes with the same policy exhibit consistent propagation paths, while those with different policies display variability. This allows us to characterize the routing policy through a set of AS-PATHs from VPs to the destination prefix, which induces a tree rooted at the origin AS.

Observed Prefix's Routing Tree (OPRT). As defined by Gill et al. [27], the set of selected routes induces a tree rooted at the destination AS, which is regarded as the routing tree of the destination AS. Let S denote the set containing all ASes in the Internet. For every AS $x \in S$, the best AS path from AS x to AS i is denoted as $P(x,i)$, which is a list of AS numbers that can be represented as the following formula. Note that the best path between a given pair of source and destination ASes is typically unique. Even when multiple best paths exist [18], the following definitions remain valid and the routing tree becomes a directed acyclic graph (DAG).

$$P(x,i) =< x, a_1, a_2 ... a_n, i >$$

A Prefix's Routing Tree (PRT) denotes the collections of best paths from all ASes to a specific prefix. Let p be a prefix and AS i be the AS it originated in. Then the best AS path from AS x to p is denoted as:

$$P(x,p) = P(x,i) =< x, a_1, a_2 ... a_n, i >$$

Therefore, the routing tree of prefix p can be denoted as:

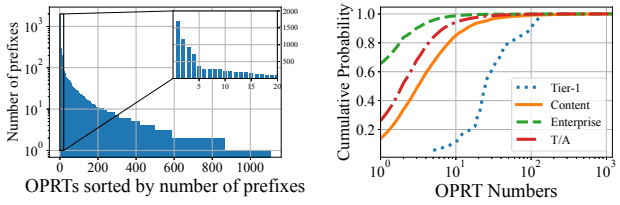
$$PRT_p = \bigcup_{x \in S} P(x,p)$$

However, it is impossible to obtain all $P(x,p)$ in practice. Thus we utilize Observed Prefix's Routing Tree (OPRT) to characterize the routing propagation of a specific prefix. Observed Prefix's Routing Tree (OPRT) denotes the collections of best paths observed by a certain set of VPs to a specific prefix. Let S_{vp} denote a set of VPs, the OPRT is denoted as:

$$OPRT_p = \bigcup_{x \in S_{VP}} P(x,p)$$

An AS is Associated with Multiple OPRTs. We observe that ASes are always associated with multiple OPRTs. Also, different prefixes may share the same OPRT topology, called OPRT equivalent prefixes. Taking a typical AS as an example, we group prefixes of AS16509 (AMAZON) by OPRT [2] and plot the distribution on the number of equivalent prefixes per OPRT in Figure 2a. We can see that most of the AS' prefixes share several common OPRTs, while over 800 OPRTs each correspond to fewer than 10 prefixes, e.g., if there are 10 prefixes have equivalent OPRT topology, we say this OPRT corresponds to those 10 prefixes. We regard OPRTs that correspond to a large number of prefixes as mainstream OPRTs and those correspond to a few number of prefixes as niche OPRTs. We also observed that some niche OPRTs have similar topology with mainstream OPRTs. This suggests that the number of OPRTs is primarily influenced by varying routing policies and incomplete BGP convergence. To illustrate this phenomenon across different types of ASes, we calculate the OPRTs of different types of ASes as defined in CAIDA AS classification [16]. Figure 2b illustrates that over 80% of ASes have fewer than 10 types of OPRT topology despite they may originating a larger number of prefixes. This indicates that

[2] We use RIB data of the RIPE RRC03 at 2024-09-01 00:00 UTC.



(a) Distribution of OPRT equivelant prefixes of AS16509

(b) Distribution of OPRT numbers owned by different types of ASes

Figure 2: Distribution of number of prefixes and OPRTs

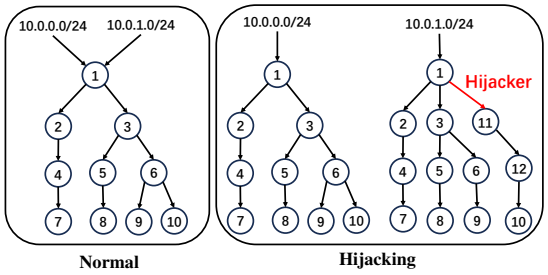


Figure 3: OPRT change while hijacking

usually there is minimal routing policy diversity among prefixes belonging to the same AS. In contrast, Tier-1 ASes, due to their larger number of prefixes and more complex policies, demonstrate a broader diversity in routing policies. Also, Content ASes often have more distinct OPRTs since they contain multiple types of services.

Hijacking Triggers the Creation of a New OPRT. BGP anomalies, such as route hijacks and route leaks, lead to changes in the OPRTs of affected prefixes, resulting in a new OPRT distinct from the normal one. For instance, in Figure 3, the prefixes 10.0.0.0/24 and 10.0.1.0/24 share the same OPRT. However, when the hijacker AS11 performs a Type-1 hijacking on the prefix 10.0.1.0/24, it causes 10.0.1.0/24 to switch to a new OPRT. Therefore, an initial intuition for detecting path hijacking is to monitor the *number change of OPRTs* of an AS. This naïve method is effective but encounters several challenges, which we will discuss later.

3.2 Challenges

Monitoring the number changes of OPRTs can effectively capture hijacks. Hijackers face great challenges in evading this method, except in the following two scenarios: 1. The hijacker hijacks the target prefix along with all its OPRT equivalent prefixes. 2. The attacker is the sole provider for the victim AS. The first scenario would cause a major traffic disruption and could easily be discovered by the victim. The second scenario is not meaningful because a sole provider of the victim does not need hijacks to attract traffic toward the victim. Despite this method's effectiveness, it faces the following challenges.

Existence of Niche and Unstable OPRTs. As Figure 2a shows, there are many niche OPRTs corresponding to no more than 10 prefixes. These OPRTs usually have similar structures to the mainstream OPRT due to similar routing policies and incomplete BGP convergence. They usually cause the frequent number change of OPRTs of an AS, resulting in many *false alarms*. To mitigate this, we apply OPRT clustering (§4.3) and monitor the number of OPRT clusters. The OPRT clustering has two-fold effects: 1. It clusters unstable niche OPRTs with stable mainstream OPRTs, thereby reducing the side effects of BGP convergence. 2. It hinders hijackers from bypassing the detection by hijacking a prefix and all its OPRT equivalent prefixes simultaneously.

Differentiation between Hijacked and Normal OPRTs. Although clustering can reduce niche OPRTs, it may group hijacked OPRTs into a normal OPRT cluster mistakenly. Therefore, measuring the differences between OPRTs is necessary to maintain accuracy. To better differentiate between normal and hijacked OPRTs, thereby ensuring the *coverage* of path hijacking detection, we propose and utilize weighted edit distance between OPRTs to amplify the differences between them (§4.2). Specifically, we assign edge weights to each unique AS link in OPRTs based on domain knowledge and subsequently calculate the edit distance of OPRTs to distinguish between OPRTs.

Monitoring All OPRTs is Costly. As Figure 2a shows, some ASes will have over 1000 distinct OPRTs, and monitoring so many OPRTs will cause high space and time resource consumption. To improve the *efficiency* of hijacking detection, we proposed a highly efficient clustering algorithm with hashing as detailed in (§4.3).

Normal Events Can Also Change OPRT Numbers. Since path hijacking always results in changes to the number of OPRTs, detection methods based on OPRT changes have strong coverage capability for path hijacking. However, there are numerous normal routing changes in the network, and these changes may also lead to changes in the number of OPRTs. Confusing these normal changes with actual path hijacking can result in network operators spending effort on meaningless investigation and handling, leading to unnecessary losses. To guarantee the *False Positive Rate* of hijacking detection, we designed four heuristic filtering methods to disregard potential normal changes as detailed in (§4.4).

4 System Design

In this section, we first present the design principles and the overall structure of Ares (§4.1), our path hijacking detection system. Then, we introduce key aspects designed in Ares to implement important design principles: weighted edit distance (§4.2), OPRT clustering (§4.3), and anomaly detector (§4.4).

4.1 Overall Design

Existing hijacking detection systems cannot detect path hijacking thoroughly and can be bypassed by smart attackers with path hijacking. This motivates us to design a novel system to effectively detect all kinds of path hijacking to solve the gap.

Design Principles. Our system design should follow these basic principles.

• *Accuracy.* Detecting path hijacking with a low False Positive Rate (FPR) and high True Positive Rate (Recall).

• *Real-time.* Capable of detecting path hijacking in real-time.

• *Comprehensiveness.* Covering all types of path hijacking.

• *Internet-wide Coverage.* Detection path hijacking for all ASes on the Internet.

• *Control-plane Information.* Output sufficient control plane information for operators to analyze and verify, such as duration time, hijacker, suspicious links, etc.

• *Lightweight.* Efficient and easy to deploy with minimal resource requirements.

System Overview. Following the above design goal, we design Ares, an effective and lightweight path hijacking detection system based on routing tree clustering. Firstly, Ares takes public BGP data as input and maintains a routing table for the entire Internet. To quantify the differences between routing trees, we propose a type of weighted edit distance based on the routing characteristics of AS links (§4.2). Moreover, to achieve efficient detection and lightweight deployment, we proposed an anomaly detection algorithm based on routing tree clustering (§4.3). Finally, to enhance detection accuracy and reduce false positives, we applied four heuristic rules to filter potential false alarms (§4.4).

Workflow. As shown in Figure 4, Firstly, Ares' route monitor takes public BGP routing table as initial input and then utilizes public BGP updates to maintain a routing table for the entire Internet. Every 5 minutes, Ares generates weighted OPRTs based on the routing information stored in the route monitor and clusters these OPRTs using a clustering algorithm. If the number of identified clusters differs between two consecutive clustering iterations, Ares outputs the newly appeared clusters as candidate alarms to the anomaly detector. Finally, the anomaly detector determines whether the newly appeared clusters should be classified as hijacks or normal routing changes.

Route Monitor. Ares's Route Monitor continuously collects BGP data from RIPE RIS via BGPStream [15]. Every 5 minutes, it aggregates updates into a snapshot and sends it to the Clustering module. However, routing paths may oscillate frequently due to traffic engineering or ongoing route convergence which can trigger false alarms. To mitigate this, Ares applies two strategies: **Route Flap Damping** and **stability confirmation**. For flap damping, we adopt the IETF approach [48] to suppress prefixes that are frequently announced and withdrawn, excluding them from clustering. For stability confirmation, Ares checks whether a prefix remains stable for over 3 minutes. If so, we consider the route converged and incorporate it into the input of OPRT clustering, following the finding in [32] that BGP typically converges within 3 minutes.

4.2 Weighted Edit Distance

We adopt the weighted edit distance definition proposed by Dinler et al. [22], which introduces a weighted edit distance adapted for labeled graphs. This definition was chosen for two primary reasons: (1) its simplicity and ease of implementation, and (2) its natural compatibility with the labeled structure of OPRTs, where the labels correspond to AS numbers. To illustrate, Figure 5 depicts an example showcasing the disparity between two labeled trees (DAGs). In this scenario, transforming the left tree into the right one necessitates deleting edge (7, 8), inserting edge (4, 8), (7, 9), and (7, 10). Assuming the weights of the rest edges remain constant and let $\omega_{i,j}$ denote the weight of edge (i, j), the weighted edit distance between these labeled trees is represented as $\omega_{7,8}+\omega_{4,8}+\omega_{7,9}+\omega_{7,10}$.

Formally, let E_A and E_B be the edge set of routing trees T_A and T_B respectively, and $\omega_{A(x)}$ and $\omega_{B(x)}$ denote the weight of edge x in routing tree T_A and T_B respectively. The weighted edit distance between T_A and T_B is defined as follows:

$$WED(T_A,T_B)=\sum_{x\in E_A-E_B}\omega_{A(x)}+\sum_{x\in E_B-E_A}\omega_{B(x)}+\sum_{x\in E_A\bigcap E_B}|\omega_{A(x)}-\omega_{B(x)}|$$

Edge Weight with Different Features. To effectively distinguish routing trees of hijacked and normal prefixes and to ensure Ares' wide coverage hijacking scenarios, we aim to maximize the edit distance between them. Path hijackings typically introduce previously unseen AS links or links connecting ASes with great geographical differences. Therefore we use link frequency and geographical location to assign edge weights to increase the weights of suspicious links.

In considering link frequency, we prioritize links that appear more frequently in the routing table, assigning them lower weights due to their higher reliability. Specifically, we denote $Cnt(A,B)$ as the count of the AS link (A, B) in all the AS-PATHs from BGP routing Table. Then the weight of link frequency is defined as follows:

$$W_{freq}(A,B)=\frac{C_1}{1+Cnt(A,B)}$$

As $Cnt(A,B)\geq 0$, the corresponding $W_{freq}(A,B)$ lies in the interval $(0,C_1]$.

Regarding the geographical feature, we expect the weight of a link to be positively correlated with the geographical distance between the ASes it connects. Specifically, for an AS link (A, B), we denote $Dis(A,B)$ as the geographical distance

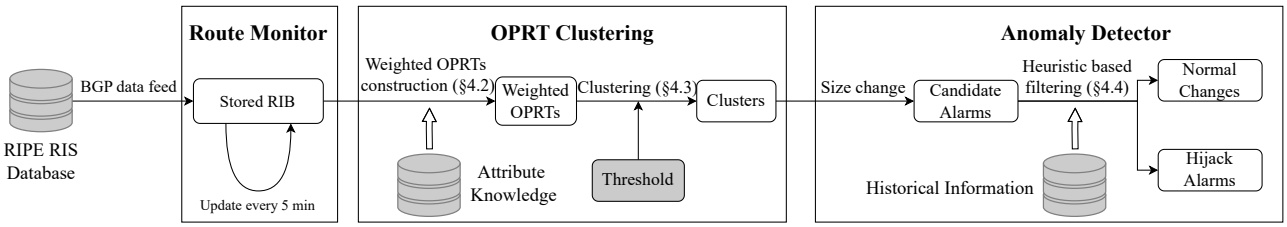


Figure 4: The architecture of Ares

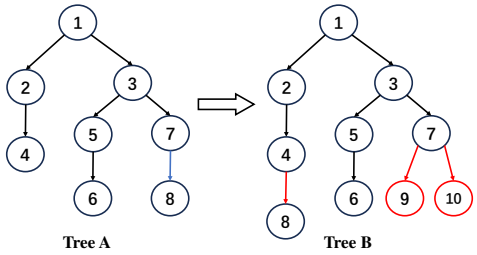


Figure 5: Edit distance with $\omega_{7,8}+\omega_{4,8}+\omega_{7,9}+\omega_{7,10}$ between two trees.

between A and B. Then the weight is defined as follows:

$$W_{geo}(A,B)=\frac{C_2\times Dis(A,B)}{C_2+Dis(A,B)}$$

The definition (based on the concept of the *harmonic mean function*) ensures that W_{geo} is positively correlated with the distance and normalizes it into the range $[0,C_2)$. The *harmonic effect* avoids extremely large weights and keeps the values stable. Since $Dis(A,B)$ ranges from 0 km to ~20,000 km in practice, we set $C_2=1000$, allowing the weight to approach C_2 when the distance is large and to grow smoothly with distance.[3] Accordingly, C_1 should be set equal to C_2 to ensure that the range of W_{freq} is the same as that of W_{geo}.

However, links involving higher-tier ASes, such as Tier-1 ASes with PoPs in multiple countries, should not receive excessively high weights, even if they span large geographical distances. To mitigate this issue, we introduce an *adjustment factor* $Adj(A,B)$ for the link (A,B). We leverage ASRank [34] to quantify the tier of an AS, where a higher ASRank (i.e., a lower numerical value) corresponds to a higher-tier AS. We define $Adj(A,B)=\frac{R_A\times R_B}{N^2}$, where R_A and R_B are the ASRanks of AS A and AS B, respectively, and N is the total number of ASes on the Internet. This definition ensures that $Adj(A,B)$ lies in the interval $(0,1)$, and that links between higher-tier ASes receive lower adjustment factor values.[4] So our adjusted W_{freq} as follows:

$$\begin{aligned}W'_{geo}(A,B)&=W_{geo}(A,B)\times Adj(A,B)\\&=\frac{C_2\times Dis(A,B)}{C_2+Dis(A,B)}\times\frac{R_A\times R_B}{N^2}\end{aligned}$$

Finally, the comprehensive weight of the two features, W_{feat} (A,B) is defined as follows:

$$W_{feat}(A,B)=w_1\times W_{freq}(A,B)+w_2\times W'_{geo}(A,B)$$

where w_1 and w_2 are weighting coefficients that control the relative importance of traffic frequency and geographical distance, respectively, with $w_1+w_2=1$. w_1 and w_2 are set by parameter tuning. Table 3 presents the detection results for six randomly sampled historical hijack events (see Section 5.1) across different weight coefficient settings. We find that only when $w_1\geq 0.6$ can Ares consistently recall all historical hijacking events, and as w_1 increases further, the number of alarms increases. To achieve complete recall of historical hijacking events while maintaining a low false alarm rate, we select w_1 to be 0.6 and w_2 to be 0.4.

Besides considering link attributes, we must also account for tree structure. Different edges within the same tree hold varying importance. For instance, in Figure 6, transitioning from Tree A to Tree B signifies a substantial change in tree structure, while transitioning from Tree A to Tree C denotes a minor change. This implies that edges (1, 10) and (10, 5) are more critical than edge (5, 9) due to the larger impact they have. Through observation, we deduce that edges closer to the origin AS and observed by more VPs carry more significance than those farther from the origin AS and observed by fewer VPs. Hence, we employ these heuristics to refine edge weights in different trees after normalizing W_{feat}. Firstly, after normalizing W_{feat}, we linearly decrease a link's weight based on its distance from the origin AS. Then we multiply each link's weight by the number of VPs observing it because a link observed by numerous VPs is deemed highly important in that routing tree. Overall, the final weight of an edge whose distance from origin AS is d and observed by v VPs is

$$W=v\times W_{feat}\times(1-min(\frac{d}{2*L_{avg}},1))$$

where L_{avg} is the average length of AS paths in the Internet.

Summary. Consequently, two types of links may be assigned

[3] C_2 controls the sensitivity of the weight to distance. If too small, the weight saturates too quickly; if too large, even the maximum distance (~20,000 km) cannot push the weight close to C_2, making the range underutilized. We set $C_2=1000$ for a good balance.

[4] For global ASes, we utilize their registered country as their location. This simplification does not considerably affect our algorithm's performance, as such ASes typically have very low ASRank values, making location exert a negligible influence on the geographical weight.

Table 3: Detection Results for Different Weight Ratios

Value of w_1/w_2	0.1/0.9	0.2/0.8	0.3/0.7	0.4/0.6	0.5/0.5	0.6/0.4	0.7/0.3	0.8/0.2	0.9/0.1
Detected history event	5	5	6	6	5	6	6	6	6
Alarm number (merged by hijacker)	53	62	60	79	78	108	117	125	128

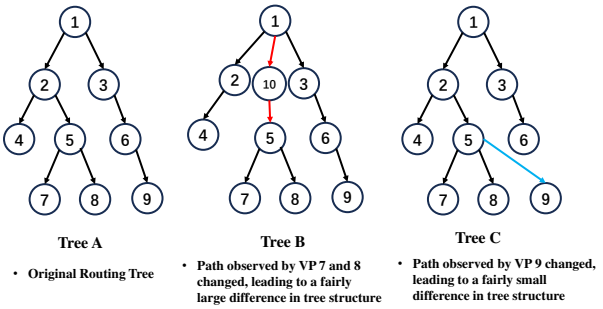


Figure 6: Different importance between edges

higher weights: (1) links deemed highly important in a routing tree and (2) links identified as suspicious based on frequency and geographical location. This approach allows link weights to capture both structural disparities among different routing trees and the suspicious nature of certain links. For instance, in Figure 6, the edit distance between Tree A and Tree B can be calculated as $|W_{A(1,2)} - W_{B(1,2)}| + W_{A(2,5)} + W_{B(1,10)} + W_{B(10,5)}$. Given the proximity of these links to the origin AS, the high edit distance suggests great structural discrepancies between the trees. Furthermore, the different weights of edge (1, 2) in Tree A and Tree B also highlight their structural distinctions. In contrast, the edit distance between Tree A and Tree C is $W_{A(6,9)} + W_{C(5,9)}$. As these edges are further from the origin AS and only observed by VP 9, the edit distance between Tree A and Tree C is relatively smaller compared to that between Tree A and Tree B, indicating minor structural differences between them. However, if edge (5, 9) is suspicious and assigned a higher weight, the edit distance between Tree A and Tree C could increase, signaling a potential anomaly.

4.3 Clustering

After defining the Weighted Edit Distance, we established a quantitative measure to evaluate dissimilarities between OPRTs. This enables effective path hijacking detection since clustering can mitigate the impact of route flapping, increase the difficulty of hijacking attacks, and improve the efficiency of detection. We use the following steps to accomplish this. The pseudocode of the clustering algorithm is shown in Algorithm 1 (see Appendix A).

Step 1. We classify OPRTs based on their tree structures (graph topology), grouping OPRTs with identical tree structures into the same category. We achieve this by hashing the set of edges with SHA-256 [28]. Note that our approach is compatible with any standard hashing method. This method ensures that OPRTs with the same set of edges will have the same hash value, enabling the classification of OPRTs.

Step 2. After completing the classification of OPRTs, we treat each category as an initial cluster and further categorize them based on the number of prefixes in each initial cluster. Based on our observations and experiments introduced in Section 3, we have found that less than 20% of OPRT topologies are shared by more than 10 OPRT equivalent prefixes. Therefore we categorize clusters with more than 10 prefixes into main clusters and the rest into subclusters to accelerate clustering.

Step 3. For each subcluster, we try to merge it into one of the main clusters. Due to the dynamic nature of routing paths and limitations of the distribution of VPs, OPRTs in subclusters may have similar tree structures with OPRTs in main clusters. Therefore we calculate the weighted edit distance between each subcluster and main cluster. If the distance is smaller than a threshold, we consider it reasonable to merge two clusters. Note that in this step each cluster contains exactly one OPRT, so the weighted edit distance between any subcluster and main cluster is equivalent to the defined weighted edit distance between their OPRTs.

Step 4. After the first attempt of merging, some subclusters may remain outside main clusters. This indicates that ASes' routing policies for these prefixes are rare. To further reduce the number of clusters to improve detection efficiency, we try to merge subclusters pair by pair. Specifically, we check weighted edit distance between subclusters and merge similar ones. In the process, a subcluster may contain multiple OPRTs. Therefore, when checking weighted edit distance between subclusters, we calculate the pairwise distances between all OPRTs across subclusters and check the maximum value.

Update Clustering. To enhance the efficiency of detection, after performing complete clustering using Algorithm 1, we will not employ it for upcoming clustering. Instead, we will utilize an incremental updating approach for clustering. Specifically, after obtaining the specific changes in the route tree structures, we will try to merge the newly emerged OPRTs with the existing clusters. For example, AS A initially has 5 clusters after performing Algorithm 1 and one of its prefix's OPRT changed. We will identify the changed OPRT as a new cluster and try to merge it with the other 5 clusters instead of re-clustering. Note that even if a certain cluster disappeared due to path changes (e.g., a cluster only contains one OPRT and that OPRT changed), we would still retain that empty cluster when performing the incremental updating. This is because sometimes an AS may change its policies for a set of prefixes simultaneously. Through employing update clustering, we can ignore a large amount of unnecessary computations, thereby processing a large number of BGP updates more quickly.

4.4 Anomaly Detector

As mentioned in Section 4.1, an increase in the number of clusters may indicate a hijack, and the newly formed cluster—referred to as a candidate alarm—is passed to the anomaly detector. While Ares flags new clusters as potential hijacks, normal route changes can also cause such clusters. To reduce false positives, Ares applies a series of heuristics to candidate alarms from the clustering algorithm. Alarms filtered by any heuristic are treated as normal changes; otherwise, they are classified as hijacks. These heuristics are carefully designed to account for BGP convergence (H1), capture typical hijack patterns (H2/H4), and leverage historical routing context (H3), effectively filtering out benign events while preserving detection sensitivity. Their effectiveness is validated in Section 5.1.

H1: Partial Edit Distance. Edit distance measures the structural similarity between two OPRTs, with smaller distances indicating greater similarity. Nevertheless, sometimes even OPRTs with fairly different tree structures should be clustered together. As shown in Figure 7, although Tree B differs from Tree A, all its VPs are also in Tree A, and the observed paths are identical. Thus, Tree B is a subset of Tree A. Such subset-shaped OPRTs may result from incomplete BGP convergence or differences in routing policies (e.g., path filtering or preference). Regardless, if no new paths appear and all observed paths already exist in a previous OPRT, the change is unlikely to be a hijack since hijacks typically introduce new, unseen paths. Therefore, instead of computing full edit distance during clustering, we calculate it only over the newly appeared links in the new OPRT. This enables us to group structurally reduced OPRTs like Tree B with Tree A, avoiding false positives.

H2: Size of new cluster. Reported historical events [21] show that a hijacking event usually impacts only a small number of prefixes. This is because hijacking numerous prefixes simultaneously can significantly disrupt Internet traffic, making it easier to detect the hijackers quickly. Moreover, compared to hijacks, routing policy changes are more likely to cause the paths of multiple prefixes to change simultaneously. Therefore, if a new cluster includes a sufficient number of prefixes, we consider it a normal change. As mentioned in Section 3, it is rare for over 10 prefixes' OPRT to share the same tree structure. Therefore newly emerged clusters with ten or more prefixes are deemed non-anomalous changes by this heuristic.

H3: Historical information. Consider the following scenario: within an AS, there exists special prefixes whose OPRTs significantly differ from those of other prefixes, making them inside a subcluster. When a prefix's OPRT in this subcluster changes, it will almost certainly generate a new OPRT that cannot be merged, potentially triggering false alarms. To mitigate this issue, we maintain a database of stable and normal tree structures observed over a certain period. When a newly emerged cluster doesn't align with existing ones, we compare

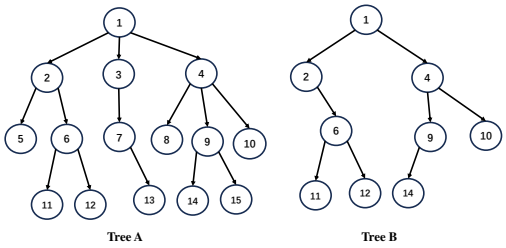


Figure 7: Example of partial edit distance

it with the recorded structures. If the weighted edit distance from the new cluster to any recorded structure falls below the set threshold, we conclude this new cluster is legitimate.

H4: Existing relationships. The complexity and dynamic nature of BGP routing imply that even legitimate routing updates may produce new OPRTs that significantly deviate from existing ones. To make more precise judgments, we design a heuristic based on the characteristics of path hijacking. Specifically, when a new cluster emerges, we extract its higher-weighted links and perform preliminary filtering. If a link either: (1) previously non-existent, which is absent from CAIDA's business relationship dataset [17], or (2) triggers a valley-free violation, we flag it as anomalous. Otherwise, the cluster is classified as normal. While normal routing changes could occasionally trigger these conditions, such criteria remain essential for effective hijacking detection.

5 Experiments

In this section, we perform our evaluation of Ares. Firstly, we validate Ares' effectiveness (§5.1) using reported historical events. Secondly, to illustrate Ares' robustness in multiple scenarios of path hijacking, we propose an evaluation method based on large-scale simulation (§5.2) and compare Ares' results with existing state-of-the-art methods. Then we validate Ares' efficiency (§5.3) by comparing its overhead with existing methods. Furthermore, we present the results of the real-world deployment of Ares to show its practicality (§5.4). Finally, we analyze several detected events (§5.5).

5.1 Historical Events Detection

Ground Truth. We collected 12 historical path hijacking events, which are combined with the dataset proposed by Cho et al. [21] and two events reported in 2022. These events are all real-world hijacks in which the attackers forged AS links to redirect Internet traffic. There is also a Defcon attack included in it. These reports provide detailed information on the hijackers involved, the victims, and the affected prefixes in hijacking events, helping us accurately pinpoint the presence of hijacked BGP updates.

Workflow. For each historical hijacking event, we build a dataset by selecting the RIB file closest in time from a specific

Table 4: Detection results of historical events and alarms filtered by each heuristics (Event Alarm means alarms can be confirmed by historical events)

Event	Date	Duration	Detected	Update	H1	H2	H3	H4	Alarm (Event Alarm)	Updates per Alarm
torg_1	2014-09-26	8 hours	Yes	743,275	0	1	7	6	4 (1)	185,818.75
torg_2	2014-09-26	8 hours	Yes	743,275	0	1	7	6	4 (1)	185,818.75
torg_3	2014-09-27	8 hours	Yes	1,015,573	1	0	4	3	9 (1)	112,841.44
backconnect_3	2016-02-20	8 hours	Yes	1,686,242	3	0	7	35	24 (1)	70,260.08
backconnect_5	2016-02-21	16 hours	Yes	2,473,606	2	3	7	11	5 (2)	494,721.20
backconnect_6	2016-04-16	8 hours	Yes	1,072,191	0	1	2	9	18 (2)	59,566.17
france_1	2012-12-28	8 hours	Yes	1,852,179	1	1	3	10	8 (6)	231,522.38
enzu_1	2015-03-26	8 hours	Yes	2,326,413	5	9	7	31	288 (257)	8,077.82
defcon_1	2008-08-10	28[6] hours	Yes	1,557,289	8	4	24	47	27 (2)	57,677.37
facebook_1	2011-03-22	8 hours	Yes	676,775	4	4	4	16	31 (1)	21,831.45
zayo_1	2022-02-03	8 hours	Yes	13,518,139	10	1	12	27	16 (1)	844,883.69
amazon_1	2022-08-17	8 hours	Yes	15,131,654	2	0	7	10	19 (1)	796,402.84

RIPE RIS [1] collector (rrc00 or rrc03 in experiments) as the initial snapshot. We then collect all BGP updates from that collector until the next RIB file following the hijack. These updates are fed into Ares, and all resulting alarms are recorded. If any alarm matches the reported prefix and hijacker of the historical event, we consider it a successful detection by Ares.

Threshold Setting. To enhance Ares' adaptability, we dynamically calculate the merging threshold for clusters during detection based on historical data. During the cold start phase, we first collect the routing tables and all BGP updates from the day before the hijacking event occurs. We feed this data into Ares to perform an initial detection run[5], and record the minimum weighted edit distance calculated each time Ares attempts a cluster merge. Finally, we determine the final threshold to be used by computing the knee point of all collected weighted edit distances via kneed [42]. The knee point is a commonly used method for optimal trade-off identification. However, in real-world data, knee points are sensitive to noise and may not serve as the best threshold choice. To mitigate the noise and fluctuations near the knee point, we adjusted the threshold to the knee point value plus four times the Median Absolute Deviation (MAD [30]) of the entire data.

Result Analysis. The detection outcomes for all historical events are depicted in Table 4. Ares effectively identifies all path hijacking incidents with an extremely low false positive rate since the number of alarms is considerably lower than the number of BGP updates during the detection period. For most events (excluding enzu_1), Ares triggers an average of only 1.49 alarms per hour, which is a minimal number for network operators to inspect. As for the enzu_1 event, we find it is primarily triggered by Enzu Inc. (AS18978) accidentally leaking prefixes to AS3257. Although the event triggers a high alarm count (288), 257 alarms are confirmed as true positives based on the event's report, demonstrating the Ares' detection effectiveness in higher-alarm scenarios.

[5] As mentioned in Section 3, monitoring all OPRTs simultaneously would significantly increase overhead. Hence, this process uses a fixed threshold for acceleration. Since this threshold is set only for efficiency purposes, it remains constant in all cases to prevent it from affecting later computed thresholds.

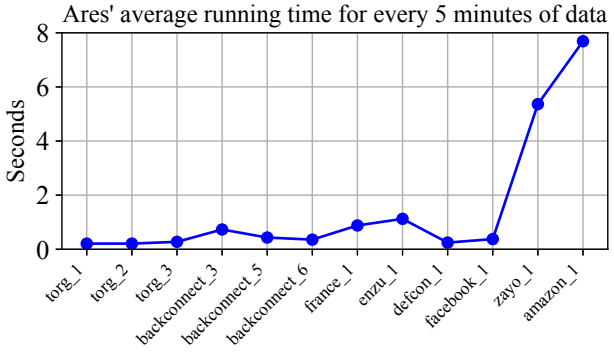


Figure 8: The runtime overhead of Ares

Heuristics Analysis. We use the detection of historical events to evaluate the effectiveness of different heuristics, i.e., how many potential alarms these heuristics filtered for Ares. The assessment results are detailed in Table 4, which suggests that H4 is the most effective heuristic, significantly reducing potential alarms compared to the other heuristics. H4 plays a crucial role in filtering out false alarms stemming from routing changes, a common occurrence in the Internet. When ASes modify their routing preferences, hidden links may surface. H4 identifies these links based on documented business relationships. However, detecting new hidden links without historical records poses challenges in distinguishing them from spoofed hijack paths. Ares may trigger an alarm in such scenarios, deeming them as potential hijacks. Increasing the number of VPs enhances Ares' ability to detect these hijacks. Furthermore, H3 also filtered a considerable number of potential alarms, proving that introducing historical OPRT information can effectively mitigate false alarms caused by the inability to cluster due to the lack of comparative data. Besides, H1 and H2 can also help filter potential false alarms in certain cases, although their targeted scenarios (e.g., incomplete BGP convergence or routing policy changes) occur less frequently in real-world networks.

[6] Due to the nature of this event being essentially a demonstration experiment, and a similar anomalous routing occurring the day before the event reporting, the detection duration of this event has been extended to 28 hours.

Summary. In conclusion, (1) Ares can effectively identify various path hijacking events that have occurred historically and successfully pinpoint suspicious links and attackers, a capability that most existing hijacking detection systems cannot achieve. (2) Ares has a low number of alarms compared to the number of BGP updates, resulting in an extremely low false positive rate. This characteristic enables users of Ares to discern the alarms triggered by Ares with minimal expenditure, allowing for effective identification of potential threats.

5.2 Large-scale Simulation

To demonstrate the effectiveness of Ares in multiple scenarios of path hijacking, we innovatively propose an approach combining simulated data and real-world routing data to evaluate Ares' detection capabilities for different scenarios of path hijacking. We also compare the effectiveness between Ares and existing state-of-the-art detection methods. Note that since Ares uses control-plane data to perform detection, it is only comparable with control-plane-based methods. Therefore, we compared Ares with Metis [49], DFOH [29] and BEAM [19].

Workflow. Initially, we categorize ASes into Content, Enterprise, Transit/Access (T/A), and Tier-1 based on the CAIDA AS classification scheme [16]. Path hijacking events are then classified into 16 types, considering the categories of hijacker and victim ASes. Subsequently, we conduct simulations for each type of hijacking event (i.e., positive event) to assess Ares' detection capabilities. Roughly 100 (hijacker, victim) AS pairs are selected per event type, and routing paths post-hijacking are obtained through simulation (1706 events are simulated in total). These simulated routing trees, representing new OPRTs of victim ASes, are compared and clustered in Ares against actual routing trees. Anomalies in these 'newly appeared' routing trees trigger Ares' detection, indicating successful identification of the hijacking event. To demonstrate the effectiveness of Ares comprehensively, we simulated exact prefix hijacking and sub-prefix hijacking, respectively. Moreover, we sampled routing changes in the real world to form non-hijack events (i.e., negative events) and applied Ares to it to test Ares' False Positive Rate (FPR).

Simulation Method. We employ the BGP route simulation technique by Sermpezis et al. [44] to model the effects of hijacking events on routing behavior. This method upholds valley-free routing principles and general routing preferences of ASes. Given Ares' dependency on routing tree clustering for detection, we compare and cluster simulated hijacking routing trees with actual victim AS routing trees for experimental analysis. Specifically, we first simulate the original route propagation of one of the victim's prefixes. Then we simulate route changes after the hijack occurs. To mirror real-world routing conditions, adjustments were made during the original route simulations: (1) allowing violations of valley-free routing principles if such violations exist in reality; (2) manually optimizing routes from specific ASes to the origin AS when discrepancies between simulated propagation and real-world observations occurred, followed by re-propagation. During the simulations, we compared the simulated data with the RIB of rrc03 [41] at 00:00 on October 1, 2024. The threshold used in the experiment was obtained based on the routing data from September 30, 2024, following the method described in Section 5.1.

Hijack Paths Construction. We focus on evaluating the recall of the path hijack detection methods by simulating Type-1, Type-2, and Type-3 hijacks, while ignoring longer forged paths due to their reduced competitiveness, lower success rates, and poor stealth. We suppose that in Type-N attacks, the hijackers select a real path of length N to the victim AS, then prepend their ASN to this path, ensuring only one fake link in the forged path for a higher hijack success probability. Furthermore, a Defcon hijack simulation (Type-U) is performed for each (hijacker, victim) AS pair, where the hijacker directly hijacks the victim AS using a path based on the existing AS topology. Hijacker ASes in simulations prioritize shorter paths but exclude length-1 paths, as directly hijacking neighbors offers minimal traffic attraction and is easily spotted.

Non-Hijack Paths Construction. To evaluate the FPR of the four detection methods, we conducted a random sampling of real-world routing changes and evaluated them using the four detection methods. Specifically, we collected routing updates from collector rrc03 [41] between 00:00 and 02:00 UTC on October 1, 2024, and performed random sampling to construct a dataset containing 1706 prefixes with observed changes. This matches the number of hijacking events (positive samples) we simulated. Since the vast majority of real-world routing changes are legitimate and no routing anomalies were publicly reported during this time window, we treated all sampled routing changes as benign.

Result Analysis and Comparison. The detection results for simulated hijacks are illustrated in Table 5, Table 6, and Tables 11-16 in Appendix B due to space limit. Furthermore, the FPRs of each compared method are presented in Table 7. The results demonstrate that Ares considerably outperforms all other methods in both detection scope and accuracy, successfully identifying all Type-U hijacks and achieving stable and superior recall rates that mostly exceed 95% while maintaining a competitive low FPR of 1.06%. We sequentially analyze the results of each method and conduct comparisons.

• *Ares.* Tier-1 and Content ASes represent high-value targets for hijackers, where successful attacks can cause significant cascading impacts. For Tier-1 and Content victims and non-Tier-1 attackers, Ares shows a high detection rate for Type-1/2/3 and Type-U hijacks (see Table 5, 6, 13 and 14), maintains an average recall rate of 97.2%/99.3% (exact/sub-prefix hijacks, respectively, same below). For hijacks targeting Enterprise and T/A ASes by non-Tier-1 AS attackers, Ares also presents outperformed detection results, ranging from 94.2% to 100%, more details see Appendix B. Although Ares has a lower detection success rate when the hijacker is a Tier-1 AS

Table 5: Recall of different methods on simulation dataset (**Type-1 Hijacks**; Victim Types: Tier-1 & Content)

Victim Attacker Affected prefix	Tier-1 Tier-1 Exact	Tier-1 Tier-1 Sub	Tier-1 Content Exact	Tier-1 Content Sub	Tier-1 Enterprise Exact	Tier-1 Enterprise Sub	Tier-1 T/A Exact	Tier-1 T/A Sub	Tier-1 All Exact	Tier-1 All Sub	Content Tier-1 Exact	Content Tier-1 Sub	Content Content Exact	Content Content Sub	Content Enterprise Exact	Content Enterprise Sub	Content T/A Exact	Content T/A Sub	Content All Exact	Content All Sub
Ares	**20.7%**	31.2%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**97.8%**	**98.0%**	**82.0%**	**81.8%**	**71.2%**	**74.4%**	**95.1%**	**96.3%**	**98.8%**	**100.0%**	**95.2%**	**98.0%**	**89.3%**	**91.4%**
Metis	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	43.7%	44.4%	58.3%	59.0%	51.8%	53.0%	36.6%	37.3%
DFOH	0.0%	0.0%	57.1%	57.1%	83.0%	84.0%	86.0%	86.9%	57.5%	55.3%	5.8%	5.8%	47.6%	46.3%	86.9%	82.0%	57.3%	58.6%	46.9%	46.0%
BEAM-auto	0.0%	0.9%	3.3%	29.8%	13.8%	59.0%	6.5%	39.4%	5.8%	31.2%	1.0%	9.9%	1.0%	2.8%	7.1%	12.0%	1.2%	11.1%	2.4%	8.9%
BEAM-tweaked	2.3%	**32.1%**	90.1%	96.7%	87.2%	99.0%	78.5%	94.9%	67.3%	80.1%	1.9%	41.3%	40.8%	52.8%	47.6%	71.0%	48.8%	66.7%	33.2%	57.0%

Table 6: Recall of different methods on simulation dataset (**Type-U Hijacks**; Victim Types: Tier-1 & Content)

Victim Attacker Affected prefix	Tier-1 Tier-1 Exact	Tier-1 Tier-1 Sub	Tier-1 Content Exact	Tier-1 Content Sub	Tier-1 Enterprise Exact	Tier-1 Enterprise Sub	Tier-1 T/A Exact	Tier-1 T/A Sub	Tier-1 All Exact	Tier-1 All Sub	Content Tier-1 Exact	Content Tier-1 Sub	Content Content Exact	Content Content Sub	Content Enterprise Exact	Content Enterprise Sub	Content T/A Exact	Content T/A Sub	Content All Exact	Content All Sub
Ares	**90.4%**	**90.2%**	**98.8%**	**100.0%**	**100.0%**	**100.0%**	**97.1%**	**100.0%**	**96.5%**	**97.0%**	**46.4%**	**62.8%**	**100.0%**	**100.0%**	**97.5%**	**100.0%**	**92.6%**	**96.4%**	**83.9%**	**87.1%**
Metis	0.0%	0.0%	0.0%	0.0%	0.0%	1.2%	0.0%	0.0%	0.0%	0.3%	0.0%	0.0%	0.0%	0.0%	2.5%	1.2%	5.6%	3.6%	1.8%	1.1%
DFOH	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
BEAM-auto	0.0%	0.9%	15.1%	55.4%	5.3%	73.5%	11.4%	68.7%	8.4%	45.9%	0.0%	14.9%	0.0%	3.5%	0.0%	24.1%	5.6%	10.8%	1.4%	13.4%
BEAM-tweaked	5.5%	75.0%	83.7%	97.8%	80.7%	**100.0%**	64.3%	96.4%	58.4%	91.1%	5.4%	46.3%	30.9%	53.5%	45.0%	75.9%	33.3%	69.9%	27.5%	59.8%

compared to other scenarios, other methods also struggle to detect such hijacks (first two columns of Tables). Moreover, this scenario is uncommon, we discuss it in Appendix D.

• *Metis.* Metis is proved ineffective for hijacks against Tier-1 ASes or launched by Tier-1 ASes because it trusts all links involving Tier-1 ASes. Moreover, it struggles to detect Type-U hijacks since it will skip the existing links. Although Metis achieves a higher recall rate for Type-1 hijacks compared to Type-2 and Type-3 hijacks, ranging from 51.5% to 81.9% (excluding hijacks involving Tier-1 ASes) for hijacks against Enterprise and T/A and even lower for Content ASes, these results are substantially worse to those achieved by Ares. [7]

• *DFOH.* Like Metis, DFOH is inherently incapable of detecting Type-U hijacks because it exclusively focuses on newly emerged links in the topology while disregarding existing ones. For Tier-1 and Content victims and non-Tier-1 attackers, DFOH achieves an average of 60.4%/65.2% recall rate for Type-1/2/3 hijacks, which is worse than Ares' results. DFOH also achieves worse results than Ares in other hijack scenarios (see Appendix B). Furthermore, results illustrate that DFOH demonstrates considerable scenario sensitivity, with recall ranging from 22.0% to 86.9% across different cases.

• *BEAM.* BEAM's auto-generated thresholds performed poorly on our dataset (BEAM-auto in Tables). To demonstrate BEAM's effectiveness, we tweaked BEAM's thresholds to boost recall while keeping its FPR below 5% (BEAM-tweaked in Tables). BEAM-tweaked achieves an average of 41.7%/77.7% recall rate for Type-1/2/3 and Type-U hijacks against Tier-1 and Content victims with non-Tier-1 attackers. BEAM-tweaked shows superior recall for sub-prefix versus exact prefix hijacks, particularly excelling at Tier-1-targeted sub-prefix hijacks with close to Ares in certain scenarios. However, its effectiveness varies by scenario, with its recall rate for exact prefix hijacks degrading as forged paths become longer (see Type-3 results in Tables 14 and 16).

False Negative Analysis. Ares achieves lower recall rates for hijacks launched by Tier-1 ASes as shown in Tables 5, 6 and 11-16. We investigated the simulated events missed by Ares and found that Tier-1 AS hijackers typically utilize real paths from the public BGP RIB for forging paths. Given their extensive transit services, Tier-1 ASes can easily access common paths for hijacking. However, such behavior is unlikely, as they neither rely on hijacking to attract traffic nor are they easily controlled by external attackers. Additionally, in those events missed by Ares and the hijackers are non-Tier-1 ASes, over 80% of these events impacted fewer than 5 VPs. This suggests that the competitiveness of forged paths in these events was limited due to the hijacker's characteristics (hierarchy or relationships) or path length. Consequently, these events had minimal impact on OPRT structures and were not detected by Ares. Overlooking these less low-impact events should not lead to significant losses, thus maintaining Ares' effectiveness. Furthermore, these events can be detected by Ares if the number of public VPs increases.

[7] Theoretically, Metis and DFOH should have an identical recall for both exact and sub-prefix hijacks. However, in simulations, some exact-prefix hijacks may fail to impact routing due to weak path competitiveness, causing them to be excluded as positive samples. This leads to the observed recall differences between exact and sub-prefix hijacks for Metis and DFOH.

Table 7: False Positive Rate Comparison on Real-world Routing Changes

Method	Ares	Metis	DFOH	BEAM-auto	BEAM-tweaked
Alarm Number	18	9	0	15	81
FPR	1.06%	0.53%	0%	0.88%	4.75%

5.3 Runtime Overhead

Ares' detection efficiency. We further evaluate the efficiency of Ares by measuring the processing time of every 5-minute update data of BGP during detecting the 12 historical path hijacking events (see Section 5.1). Our detection runs on a Linux Server with Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz. The runtime overhead of Ares is presented in Figure 8. We can see that Ares processes the first 10 events' updates in under 1 second on average, while the last two take up to 7 seconds. This is because the last two events occurred later (in 2022), and with the network's growth, the number of BGP updates within the same time frame has dramatically increased. Nevertheless, the average time Ares needs to detect BGP updates every 5 minutes remains well below 5 minutes, indicating that Ares has low runtime overhead and is capable of real-time detection.

Efficiency comparison with existing methods. We also com-

Table 8: Efficiency Comparison (Average Running Time for Detecting BGP Updates in each 5 Minutes)

Method	Ares	Metis	DFOH	BEAM
Overhead (s)	6.33	12.5	16.1	23.9

Table 9: Memory and Disk Space Comparison of Methods

Methods	Ares	Metis	DFOH	BEAM
Memory	50GB	20GB	<4.5GB	<5GB
Disk Space	2GB	21GB (2GB if lively)	>166GB	4GB BGP data 155MB model

pare the detection efficiency between Ares and existing state-of-the-art detection methods. We collected all BGP updates of rrc03 on October 1, 2024, and applied the methods to process the data on the same device. Table 8 records the average running time overhead of handling updates in every 5 minutes (RIPE RIS [1] provides an update file every 5 minutes). The results demonstrate that Ares considerably outperforms existing detection methods, requiring only 6.33 seconds on average to process all updates within a five-minute window. Notably, even BEAM, which exhibits the highest computational overhead among the four methods, completes detection within an average of 23.9 seconds for updates in every five-minute volume. This indicates that all evaluated methods are capable of achieving real-time anomaly detection.

Cost comparison with existing methods. Finally, to compare the detection cost of different methods, we recorded the memory consumption, disk space consumption and preliminary works of different methods during the efficiency evaluation comparison. Memory and disk space consumption are listed in Table 9. Note that these results are estimated values due to the dynamic usage of memory and disk space. Additionally, the four methods all require preliminary work as listed below.

- *Ares.* (1) Less than one hour of preprocessing. (2) 1-2 hours of threshold counting per month.
- *Metis.* 7-9 hours of model training per month.
- *DFOH.* 2-3 hours of maintaining a local database per day.
- *BEAM.* 10-11 hours of model training per month.

Overall, Ares requires more memory but less disk space than the other three methods to achieve effective and efficient detection. Additionally, its preliminary processing is faster. In summary, each of the four methods has distinct trade-offs in terms of detection cost.

5.4 Real-World Deployment

We conduct detection experiments using Ares on the network in October 2024 and evaluate its alerting capabilities. We utilize routing data provided by one of the collectors from RIPE RIS (i.e., rrc03 [41]) as the input for detection. Specifically, we initialize Ares with the RIB file from rrc03 [41] at midnight on October 1st, 2024, and sequentially feed BGP updates obtained through BGPstream [15] into Ares for detection. In our experiments, we collect all routing paths provided

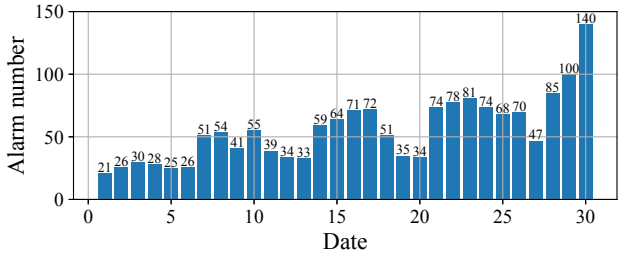


Figure 9: Daily alarm number of Ares' real-world deployment

by rrc03 [41] for 30 days, excluding routing loops and routes with reserved ASNs, and conduct detection on all origin ASes with a prefix count exceeding a certain amount. Overall, we process a total of 1,487,907,824 BGP updates. The specific detection results are shown in Figure 9. To reduce instances of repeated alarms, we merge alerts that occurred close together in time and have the same hijacker. In this experiment, we utilize the threshold calculation method mentioned in Section 5.1 to compute the threshold using the routing data from September 30, 2024. Overall, Ares demonstrates highly practical operational performance, generating only 2.31 alarms per hour on average - a manageable load for operators and dramatically lower than DFOH's 11.36 alarms during the same period. We manually analyzed all reported 1666 alarms and confirmed 751 out of 1182 are true alarms (detailed in Appendix C). While the inherent class imbalance between legitimate route changes and routing hijack anomalies fundamentally limits hijack detection systems' precision, Ares nevertheless achieves a remarkable 63.0% precision, which significantly reduces the workload for manual verification.

5.5 Case Study

In this section, we demonstrate Ares' **anomaly localization** capabilities through the analysis of different cases. We conduct specific analysis on three historical events representing common path hijacking, Defcon attacks, and route leaks, as well as an example alarm from our real-world deployment.

Case 1. The *facebook_1* event [50] illustrates a typical path hijacking without triggering MOAS. On March 22, 2011, Facebook's prefix 69.171.255.0/24 (AS32934) exhibited path changes from **VP-2914-32934** to **VP-4134-9318-32934** across two VPs. Although the OPRT structure showed minimal alterations, Ares reported an alarm due to the anomalous link (9318, 32934). Analysis suggested AS9318 likely manipulated AS32934's prepend, causing traffic diversion.

Case 2. The *defcon_1* event [7] demonstrates the pattern of Defcon attack targeting Sparkplug Las Vegas (AS20195), who had a sole provider AS23005. Before the attack, the AS paths to reach AS20195 only share the same suffix of **23005-20195** but during the attack, all VPs detected altered paths **26627-4436-22822-23005-20195**. This attack was more covert compared to other BGP hijacks because all AS links in the artificially crafted prepend existed in the network. Ares

assigned higher weights to the AS links (26627, 4436) and (4436, 22822) because they significantly impacted all VPs in the routing tree and differed greatly from the other routing tree structures of AS20195. Additionally, the hijack caused non-valley-free propagation in the routing tree of AS20195. Therefore, Ares successfully detected this hijacking incident.

Case 3. The *enzu_1* event [13] illustrates a route leak scenario where Ares flagged the anomalous propagation pattern **40633-18978-6939**. In reality, this incident was formerly categorized as a Type-U Subhijack, but it was actually caused by Enzu, Inc. (AS18978) accidentally leaking some sub-prefixes, generated by the BGP optimizer and not originally announced by the origin AS, to the Los Angeles Internet Exchange (AS40633). The detection results indicate that Ares can accurately locate routing anomalies and possesses some capability to discern BGP routing anomalies beyond path hijacking.

Case 4. During October 2024's first week, we noticed traffic towards Kalaam-Telecom (AS39273) occasionally passed through STORMWALL-AS (AS59796). AS59796 was absent in most routing trees of Kalaam-Telecom and had no existing business relationship information with it. Consequently, Ares suspects a potential hijacking of this link. Our investigation revealed that StormWall [47] uses BGP for DDoS protection. Specifically, AS59796 connects with clients requiring protection, announces their IP prefixes, filters traffic, and forwards sanitized traffic to prevent DDoS attacks. Generally speaking, StormWall hijacks and filters traffic destined for AS39273, separating potential DDoS traffic from legitimate traffic. Therefore, we deem that Ares made the correct hijacking assessment. However, given the security nature of this activity, we plan to enhance our handling of such scenarios in the future.

6 Related Works

Rule-Based BGP Hijacking Detection The rule-based BGP hijacking detection method uses predefined rules and heuristic algorithms to determine whether BGP route updates exhibit anomalous behavior. These methods [31, 33, 43, 46, 51, 53] are typically classified into control plane-based, data plane-based, and hybrid plane-based approaches (as shown in Table 2). Such algorithms offer good interpretability; however, most existing algorithms are primarily designed to detect origin hijacking, with only a few capable of detecting path hijacking. However, they are unable to detect all types of path hijacking like Ares can.

Data-driven BGP Anomaly Detection Data-driven BGP hijack detection methods leverage machine learning, deep learning, time series analysis, and statistical modeling. They typically treat detection as classification task [12, 23, 29, 39, 49], sequence analysis task [20, 35, 38], or embedding task [20, 35, 38]. Methods based on classifying legitimate MOAS [39] and link prediction [19, 49] face the challenge of lack of labeled data and imbalanced classes. They also could not cover certain scenarios of hijackings. Methods based on sequence analysis [20, 35, 38] are computationally expensive and lack sufficient ground truth for training. Furthermore, all Data-driven detection requires significant computational resources for training while offering limited interpretability.

Applications of Tree Edit Distance. Tree Edit Distance is commonly used to measure the similarity between two trees and has applications in fields like bioinformatics [11], XML document processing [26], etc. There are a lot of algorithms [14, 37] to calculate tree edit distance. In this paper, we design and implement a weighted edit distance-based tree clustering to group the OPRTs to detect path hijackings.

7 Discussion

Detection of Other Anomalies. In addition to detecting path hijacking, Ares is also capable of identifying other BGP anomalies, such as route leaks and route changes caused by outages. From a design perspective, Ares can capture anomalies that result in uncommon or suspicious routes. However, due to its reliance on routing tree statistics and clustering of a single origin AS, Ares finds it challenging to identify hijackings or anomalies that cause MOAS issues. Fortunately, existing technologies such as RPKI are effective in recognizing such anomalies. By integrating existing technologies with Ares, it's possible to detect the majority of routing anomalies.

Unused Origin AS Hijacking. In some cases, attackers may forge an unused origin AS when conducting path hijacking. This type of hijacking cannot be captured by Ares because an unused origin AS lacks any known routing tree information. This scenario can be divided into two categories. The first category involves hijacked prefixes (or sub-prefixes) being utilized by other origin ASes, which triggers MOAS. In this case, the hijacking is relatively easy to detect. The second category comprises hijacked prefixes that have never been used, significantly reducing the potential harm of the hijacking. Therefore, we believe that disregarding this scenario will not impact the effectiveness of Ares.

8 Conclusion

In this work, we proposed Ares, a novel system for detecting BGP path hijacking based on the observation of routing tree changes within ASes. By leveraging weighted edit distance, clustering, and heuristics, Ares achieves high accuracy and efficiency in detecting stealthy exact and sub-hijackings. Our evaluation across real-world hijackings, simulations, and deployments demonstrates its practicality and effectiveness. In future work, we plan to integrate Ares with ROA and other BGP security mechanisms to build a unified framework for defending against a wide range of hijacking attacks. Specifically, we will further improve Ares and integrate it into BGP-Watch [24].

Acknowledgments

We sincerely thank our shepherd and anonymous reviewers for their valuable feedback that significantly improved the final paper. This work was supported in part by Zhongguancun Laboratory, and in part by the National Key Research and Development Program of China under Grant 2020YFE0200500. Also, this work was supported by Tsinghua University - Beijing Qihu Technology Co., Ltd Joint Research Center for Cyberspace Surveying and Mapping.

9 Ethic Considerations

In conducting this research, we have carefully considered the ethical implications of our work to ensure alignment with responsible research practices. Our primary stakeholders include organizations such as RIPE and CAIDA, whose datasets form the foundation of our analysis. For RIPE RIS data, our research purposefully supports their mission to enhance Internet routing observability and security through BGP path hijacking detection, thereby creating no conflict with the data's original security-oriented purpose. Similarly, CAIDA's privacy-preserving data donation framework is fully respected, as our analysis exclusively employs anonymized datasets already sanitized for research use, ensuring no adverse impacts on data donors or the organization itself. Throughout the study, we implemented strict ethical safeguards: All experiments utilized passively collected BGP data without active network probing, eliminating risks of Internet disruption or unauthorized data exposure. Importantly, the research process does not involve any sensitive personal information or commercial interests, focusing solely on routing pattern analysis for security enhancement. We rigorously complied with both organizations' data usage policies, including restrictions on redistribution and commercial applications. To further strengthen ethical rigor, we conducted a harm assessment confirming that our methodology neither introduces new privacy risks nor enables potential misuse of the datasets. The research outcomes aim to benefit the broader networking community by advancing detection capabilities against routing threats while maintaining full transparency through method disclosure and reproducibility verification. This ethical framework ensures our work contributes to Internet security improvements without compromising the trust relationships between data providers and the research community.

10 Open Science

All the data used in our research are publicly available. The source code used in our experiments – including Ares and the methods compared with Ares – along with the raw experimental data, is publicly available at https://doi.org/10.5281/zenodo.15589806.

References

[1] Ripe ris. https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-data-access/.

[2] Routeviews. https://www.routeviews.org/routeviews/.

[3] As 7007 incident, 1997. https://en.wikipedia.org/wiki/AS_7007_incident.

[4] Turkey telecom leak of december 2004, 2004. https://archive.nanog.org/meetings/nanog34/presentations/underwood.pdf.

[5] How pakistan knocked youtube offline, 2008. https://www.cnet.com/culture/how-pakistan-knocked-youtube-offline-and-how-to-make-sure-it-never-happens-again/.

[6] $83k in bitcoins 'stolen' through bgp hijack, 2014. https://www.virusbulletin.com/blog/2014/08/83k-bitcoins-stolen-through-bgp-hijack.

[7] Stealing the internet: An internet-scale man in the middle attack, 2016. https://we.riseup.net/assets/43591/defcon-16-pilosov-kapela.pdf.

[8] What happened? the amazon route 53 bgp hijack to take over ethereum cryptocurrency wallets, 2018. https://www.internetsociety.org/blog/2018/04/amazons-route-53-bgp-hijack/.

[9] How 3 hours of inaction from amazon cost cryptocurrency holders $235,000, 2022. https://arstechnica.com/information-technology/2022/09/how-3-hours-of-inaction-from-amazon-cost-cryptocurrency-holders-235000/.

[10] Klayswap – another bgp hijack targeting crypto wallets, 2022. https://manrs.org/2022/02/klayswap-another-bgp-hijack-targeting-crypto-wallets/.

[11] Tatsuya Akutsu. Tree edit distance problems: Algorithms and applications to bioinformatics. *IEICE transactions on information and systems*, 93(2):208–218, 2010.

[12] Nabil M Al-Rousan and Ljiljana Trajković. Machine learning models for classification of bgp anomalies. In *2012 IEEE 13th International Conference on High Performance Switching and Routing*, pages 103–108. IEEE, 2012.

[13] BGPMon. Bgp optimizer causes thousands of fake routes, 2015. https://www.bgpmon.net/bgp-optimizer-causes-thousands-of-fake-routes/.

[14] Philip Bille. A survey on tree edit distance and related problems. *Theoretical computer science*, 337(1-3):217–239, 2005.

[15] CAIDA. Bgpstream, 2024. https://bgpstream.caida.org/.

[16] CAIDA. The caida as classification dataset, 2024. https://publicdata.caida.org/datasets/as-classification_restricted/.

[17] CAIDA. The caida as relationships dataset, 2024.

[18] catchpoint. Bgp multipath, 2024. https://www.catchpoint.com/bgp-monitoring/bgp-multipath.

[19] Yihao Chen, Qilei Yin, Qi Li, Zhuotao Liu, Ke Xu, Yi Xu, Mingwei Xu, Ziqian Liu, and Jianping Wu. Learning with semantics: Towards a semantics-aware routing anomaly detection system. *arXiv preprint arXiv:2402.16025*, 2024.

[20] Min Cheng, Qian Xu, LV Jianming, Wenyin Liu, Qing Li, and Jianping Wang. Ms-lstm: A multi-scale lstm model for bgp anomaly detection. In *2016 IEEE 24th international conference on network protocols (ICNP)*, pages 1–6. IEEE, 2016.

[21] Shinyoung Cho, Romain Fontugne, Kenjiro Cho, Alberto Dainotti, and Phillipa Gill. Bgp hijacking classification. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pages 25–32. IEEE, 2019.

[22] Derya Dinler, Mustafa Kemal Tural, and Nur Evin Ozdemirel. Centroid based tree-structured data clustering using vertex/edge overlap and graph edit distance. *Annals of Operations Research*, 289:85–122, 2020.

[23] Yutao Dong, Qing Li, Richard O Sinnott, Yong Jiang, and Shutao Xia. Isp self-operated bgp anomaly detection based on weakly supervised learning. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE, 2021.

[24] DragonLab. Bgpwatch, 2024. https://bgpwatch.cgtf.net/#/.

[25] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions on networking*, 9(6):681–692, 2001.

[26] Minos Garofalakis and Amit Kumar. Xml stream processing using tree-edit distance embeddings. *ACM Transactions on Database Systems (TODS)*, 30(1):279–332, 2005.

[27] Phillipa Gill, Michael Schapira, and Sharon Goldberg. Modeling on quicksand: Dealing with the scarcity of ground truth in interdomain routing data. *ACM SIGCOMM Computer Communication Review*, 42(1):40–46, 2012.

[28] Shay Gueron, Simon Johnson, and Jesse Walker. Sha-512/256. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 354–358. IEEE, 2011.

[29] Thomas Holterbach, Thomas Alfroy, Amreesh Phokeer, Alberto Dainotti, and Cristel Pelsser. A system to detect forged-origin bgp hijacks. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1751–1770, 2024.

[30] David C Howell. Median absolute deviation. *Encyclopedia of statistics in behavioral science*, 2005.

[31] Xin Hu and Z Morley Mao. Accurate real-time identification of ip prefix hijacking. In *2007 IEEE Symposium on Security and Privacy (SP'07)*, pages 3–17. IEEE, 2007.

[32] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed internet routing convergence. *ACM SIGCOMM Computer Communication Review*, 30(4):175–187, 2000.

[33] Mohit Lad, Daniel Massey, Dan Pei, Yiguo Wu, Beichuan Zhang, and Lixia Zhang. Phas: A prefix hijack alert system. In *USENIX Security symposium*, volume 1, page 3, 2006.

[34] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and KC Claffy. As relationships, customer cones, and validation. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 243–256, 2013.

[35] Jianning Mai, Lihua Yuan, and Chen-Nee Chuah. Detecting bgp anomalies with wavelet. In *NOMS 2008-2008 IEEE Network Operations and Management Symposium*, pages 465–472. IEEE, 2008.

[36] Prodosh Mohapatra, John Scudder, David Ward, Randy Bush, and Rob Austein. BGP Prefix Origin Validation. RFC 6811, January 2013.

[37] Mateusz Pawlik and Nikolaus Augsten. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56:157–173, 2016.

[38] B Aditya Prakash, Nicholas Valler, David Andersen, Michalis Faloutsos, and Christos Faloutsos. Bgp-lens: Patterns and anomalies in internet routing updates. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1315–1324, 2009.

[39] Lancheng Qin, Dan Li, Ruifeng Li, and Kang Wang. Themis: Accelerating the detection of route origin hijacking by distinguishing legitimate and illegitimate moas. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4509–4524, 2022.

[40] Yakov Rekhter, Tony Li, and Susan Hares. A border gateway protocol 4 (bgp-4). Technical report, 2006.

[41] RIPE RIS. Route collector, 2024. https://ris.ripe.net/docs/route-collectors/.

[42] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[43] Johann Schlamp, Ralph Holz, Quentin Jacquemart, Georg Carle, and Ernst W Biersack. Heap: reliable assessment of bgp hijacking attacks. *IEEE Journal on Selected Areas in Communications*, 34(6):1849–1861, 2016.

[44] Pavlos Sermpezis and Vasileios Kotronis. Inferring catchment in internet routing. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(2):1–31, 2019.

[45] Pavlos Sermpezis, Vasileios Kotronis, Petros Gigis, Xenofontas Dimitropoulos, Danilo Cicalese, Alistair King, and Alberto Dainotti. Artemis: Neutralizing bgp hijacking within a minute. *IEEE/ACM transactions on networking*, 26(6):2471–2486, 2018.

[46] Xingang Shi, Yang Xiang, Zhiliang Wang, Xia Yin, and Jianping Wu. Detecting prefix hijackings in the internet with argus. In *Proceedings of the 2012 Internet Measurement Conference*, pages 15–28, 2012.

[47] StormWall. Stormwall, 2024. https://stormwall.network/.

[48] Curtis Villamizar, Ravi Chandra, and Dr. Ramesh Govindan. BGP Route Flap Damping. RFC 2439, November 1998.

[49] Chengwan Zhang, Congcong Miao, Changqing An, Anlun Hong, Ning Wang, Zhiquan Wang, and Jilong Wang. Metis: Detecting fake as-paths based on link prediction. In *2023 IEEE Symposium on Computers and Communications (ISCC)*, pages 656–662. IEEE, 2023.

[50] Ying Zhang and Makan Pourzandi. Studying impacts of prefix interception attack by exploring bgp as-path prepending. In *2012 IEEE 32nd International Conference on Distributed Computing Systems*, pages 667–677. IEEE, 2012.

[51] Zheng Zhang, Ying Zhang, Y Charlie Hu, Z Morley Mao, and Randy Bush. ispy: Detecting ip prefix hijacking on my own. In *Proceedings of the ACM SIGCOMM 2008 conference on Data Communication*, pages 327–338, 2008.

[52] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S Felix Wu, and Lixia Zhang. An analysis of bgp multiple origin as (moas) conflicts. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 31–35, 2001.

[53] Changxi Zheng, Lusheng Ji, Dan Pei, Jia Wang, and Paul Francis. A light-weight distributed scheme for detecting ip prefix hijacks in real-time. *ACM SIGCOMM Computer Communication Review*, 37(4):277–288, 2007.

A Pseudocode of the Clustering Algorithm

The pseudocode of the clustering algorithm is shown in Algorithm 1.

Algorithm 1 Clustering for a specific AS

Input: List of OPRTs L, threshold th
Output: List of clusters RL
{**Step 1**}
for each OPRT t_i in L **do**
  $edgeset \leftarrow$ collection of edges in t_i
  $hash\_value(t_i) \leftarrow$ Hash value of $edgeset$
end for
{**Step 2**}
for each $hash\_value$ **do**
  if $hash\_value$ contains more than 10 prefixes **then**
    Mark $hash\_value$ as main cluster
  else
    Mark $hash\_value$ as sub cluster
  end if
end for
{**Step 3**}
for each sub cluster SC_i and main cluster MC_j **do**
  if GetEditDistance$(MC_j, SC_i) \leq th$ **then**
    Mark SC_i can be merged into MC_j
  end if
end for
Merge each SC_i if possible, otherwise skip it
{**Step 4**}
for each sub cluster pair (SC_i, SC_j) in RL **do**
  if GetEditDistance$(SC_i, SC_j) \leq th$ **then**
    Merge SC_i and SC_j
    UPDATE RL
  end if
end for
return RL

Table 10: Simplified Alarm Classification

Types	Anti-DDoS/Security Protect	Possible Leaks	External-factor Induced False Alarms	New/Backup Links	Unconfirmed Alarms	Configuration Error	Possible Hijacks	Experiment Usage	All
Alarms	553	161	148	293	467	17	20	7	1666

B Recall Rate of Different Methods on Simulation Dataset

Recall rates for Type-1 and Type-U hijacks targeting Enterprise and T/A Ares are illustrated in Table 11 and 12 respectively. Furthermore, Recall rates for Type-2 and Type-3 hijacks are illustrated in Tables 13-16.

C Alarm Analysis

The results of our alarm analysis are presented in Table 10. The alarms are divided into seven types.

• *Anti-DDoS/Security Protect.* This type of alarm includes a 'hijacker' that provides DDoS protection or other security services similar to case 4 discussed in Case Study. As explained in Section 5.5, we regard this type of alarms as true positives.

• *Possible Leaks.* This type of alarm includes route changes that violate the valley-free principle and has a fairly big influence (e.g., observed by a certain amount of VPs). We regard them also as true positives.

• *External-factor Induced False Alarms.* This type of alarm includes false alarms caused by external factors. For example, misclassified business relationships may result in false valley-free violations. Furthermore, the absence of sibling-to-sibling relationships in the business relationship dataset also induced some false alarms. We regard this type of alarm as false positives.

• *New/Backup Links.* This type of alarm includes alarms triggered by missing business relationships yet we deduce they are false positives. For example, the business relationship dataset [17] may miss a certain number of links for one month and include it in the next and last month. Expanding the source of BGP topology information could reduce such false positives.

• *Unconfirmed Alarms.* This type of alarm includes alarms that are triggered due to huge route changes and missing business relationships yet we do not have sufficient evidence to determine whether they are normal events or abnormal events. We exclude these alarms from our analysis.

• *Configuration Error.* This type of alarm includes several alarms that are deemed caused by misconfiguration. We regard them as true positives.

• *Possible Hijacks.* This type of alarm includes several alarms that cannot be explained by other reasons. Therefore we regard them as possible hijacks and true positives.

• *Experiment Usage.* This type of alarm includes several special ASes noted its purpose is to conduct experiments. We exclude these alarms from our analysis.

D Discussion of Certain Problems

Hijacks Performed by Tier-1 ASes. As shown in Section 5, Ares struggles to detect hijacks performed by Tier-1 ASes. However, we do not consider this as a major limitation due to two reasons. On the one hand, although Tier-1 ASes are valuable targets for attackers, they usually have more secure infrastructures, making them harder to be controlled by attackers. On the other hand, Tier-1 ASes are less likely to perform hijacks themselves. The purpose of a hijack is to attract traffic to the hijacker. However, Tier-1 ASes already handle a large amount of traffic in the Internet.

Bias of vantage points. Like all control-plane detection methods, Ares relies on AS paths visible through monitored VPs to detect BGP hijack events, whose detection capability inherently correlates with the visibility of these VPs. Nevertheless, our system demonstrates effective detection capabilities even with a limited number and distribution of VPs (primarily RIPE RIS rrc00/rrc03). This is because, if a hijacking event occurs, it is not likely none of the VPs observe corresponding route changes unless the hijacking event has a relatively small scope of influence. As long as any VP observes the route change, the corresponding routing tree will change, and thus trigger our detection mechanism. Of course, increasing the number of VPs would improve the chances of observing attacks, but also raise processing overhead. Thus, VP selection requires balancing detection accuracy and efficiency. However, if attackers deliberately evade these VPs, all control-plane-dependent detection methods, including Ares, would likely fail to detect.

Threshold re-evaluating. The threshold of Ares is automatically set near the knee point of Weighted Edit Distance distribution (Section 5.1). Due to the dynamic nature of the Internet, the threshold should be periodically reevaluated in practice (e.g., once a month).

Will anycast affect Ares' performance? OPRTs created by Ares are origin-AS-specific. Specifically, Ares takes all OPRTs that share the same origin AS into consideration in order to detect hijacks targeting that origin AS. When the anycast prefix sits on different origin ASes, the OPRTs of this prefix are separated. No matter which origin AS an attacker path hijacks, Ares could detect it within that origin AS. When the anycast prefix sits on a single AS, it appears the same as unicast prefixes in terms of OPRTs.

Could Ares detect DDoS mitigation? Currently, Ares could not detect DDoS mitigation events and would regard them as hijacks. However, simply creating a whitelist of DDoS mitigation providers will effectively filter such events.

Monitoring evasion. If an attacker gets to know specific

Table 11: Recall of different methods on simulation dataset (**Type-1 Hijacks**; Victim Types: Enterprise & T/A)

Victim Attacker Affected prefix	Enterprise Tier-1 Exact	Enterprise Tier-1 Sub	Enterprise Content Exact	Enterprise Content Sub	Enterprise Enterprise Exact	Enterprise Enterprise Sub	Enterprise T/A Exact	Enterprise T/A Sub	Enterprise All Exact	Enterprise All Sub	T/A Tier-1 Exact	T/A Tier-1 Sub	T/A Content Exact	T/A Content Sub	T/A Enterprise Exact	T/A Enterprise Sub	T/A T/A Exact	T/A T/A Sub	T/A All Exact	T/A All Sub
Ares	**94.8%**	**90.5%**	**98.0%**	**95.2%**	**100.0%**	**96.2%**	**98.9%**	**95.2%**	**97.9%**	**94.3%**	**87.4%**	**85.6%**	**100.0%**	**99.0%**	**99.0%**	**98.1%**	**98.9%**	**98.1%**	**96.4%**	**95.2%**
Metis	0.0%	0.0%	61.6%	60.0%	81.9%	78.1%	71.1%	68.6%	53.3%	51.7%	0.0%	0.0%	51.5%	52.4%	70.4%	71.2%	66.0%	63.5%	47.2%	46.7%
DFOH	24.0%	22.9%	53.5%	51.4%	78.7%	75.2%	71.1%	70.2%	56.5%	54.9%	27.4%	25.0%	53.5%	53.4%	81.6%	81.7%	65.6%	67.0%	57.1%	56.8%
BEAM-auto	1.0%	4.8%	5.1%	8.6%	6.4%	12.4%	4.4%	13.5%	4.2%	9.8%	1.1%	4.8%	4.0%	6.8%	8.2%	16.3%	6.5%	16.5%	4.9%	11.1%
BEAM-tweaked	5.2%	36.2%	65.7%	67.6%	60.6%	75.2%	56.7%	73.1%	47.0%	63.0%	10.5%	46.2%	70.3%	77.7%	56.1%	77.9%	65.6%	78.6%	50.9%	70.0%

Table 12: Recall of different methods on simulation dataset (**Type-U Hijacks**; Victim Types: Enterprise & T/A)

Victim Attacker Affected prefix	Enterprise Tier-1 Exact	Enterprise Tier-1 Sub	Enterprise Content Exact	Enterprise Content Sub	Enterprise Enterprise Exact	Enterprise Enterprise Sub	Enterprise T/A Exact	Enterprise T/A Sub	Enterprise All Exact	Enterprise All Sub	T/A Tier-1 Exact	T/A Tier-1 Sub	T/A Content Exact	T/A Content Sub	T/A Enterprise Exact	T/A Enterprise Sub	T/A T/A Exact	T/A T/A Sub	T/A All Exact	T/A All Sub
Ares	**32.3%**	40.0%	**100.0%**	**95.0%**	**98.2%**	**96.6%**	**96.6%**	**95.3%**	**81.5%**	**79.4%**	**50.0%**	**60.6%**	**100.0%**	**98.8%**	**100.0%**	**98.8%**	**98.3%**	**98.8%**	**87.4%**	**87.6%**
Metis	0.0%	0.0%	0.0%	0.0%	1.8%	1.1%	1.7%	1.2%	0.8%	0.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	1.7%	1.2%	0.4%	0.3%
DFOH	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
BEAM-auto	1.6%	9.5%	3.0%	10.0%	3.6%	23.6%	6.9%	22.4%	3.7%	16.2%	0.0%	11.5%	3.9%	11.2%	3.6%	23.3%	1.7%	16.5%	2.4%	15.5%
BEAM-tweaked	4.8%	**58.1%**	35.8%	57.5%	28.6%	70.8%	43.1%	69.4%	28.0%	63.8%	4.8%	53.8%	38.2%	70.0%	37.5%	77.9%	42.4%	69.4%	30.8%	67.0%

Table 13: Recall of different methods on simulation dataset (**Type-2 Hijacks**; Victim Types: Tier-1 & Content)

Victim Attacker Affected prefix	Tier-1 Tier-1 Exact	Tier-1 Tier-1 Sub	Tier-1 Content Exact	Tier-1 Content Sub	Tier-1 Enterprise Exact	Tier-1 Enterprise Sub	Tier-1 T/A Exact	Tier-1 T/A Sub	Tier-1 All Exact	Tier-1 All Sub	Content Tier-1 Exact	Content Tier-1 Sub	Content Content Exact	Content Content Sub	Content Enterprise Exact	Content Enterprise Sub	Content T/A Exact	Content T/A Sub	Content All Exact	Content All Sub
Ares	**32.9%**	41.1%	**100.0%**	**100.0%**	**97.6%**	**100.0%**	**96.5%**	**98.0%**	**85.1%**	84.3%	**48.1%**	**58.7%**	**98.0%**	**98.1%**	**96.1%**	**100.0%**	**96.0%**	**98.0%**	**88.2%**	**87.4%**
Metis	0.0%	0.0%	7.4%	10.7%	4.8%	5.0%	11.8%	16.0%	6.3%	7.9%	0.0%	0.0%	10.2%	11.1%	23.4%	23.0%	18.7%	17.0%	13.8%	12.1%
DFOH	17.8%	16.1%	46.2%	46.2%	70.2%	74.0%	76.5%	77.8%	53.2%	52.1%	7.4%	10.7%	38.8%	38.9%	72.7%	72.0%	61.3%	64.6%	47.4%	44.6%
BEAM-auto	0.0%	1.8%	3.3%	45.5%	1.2%	61.0%	3.5%	57.6%	2.2%	40.5%	0.0%	13.2%	0.0%	0.9%	0.0%	11.0%	0.0%	9.1%	0.0%	8.6%
BEAM-tweaked	2.7%	**67.9%**	72.7%	98.3%	54.8%	98.0%	50.6%	96.0%	49.3%	**89.8%**	0.0%	29.8%	11.2%	27.8%	19.5%	58.0%	5.3%	47.5%	9.9%	40.0%

Table 14: Recall of different methods on simulation dataset (**Type-3 Hijacks**; Victim Types: Tier-1 & Content)

Victim Attacker Affected prefix	Tier-1 Tier-1 Exact	Tier-1 Tier-1 Sub	Tier-1 Content Exact	Tier-1 Content Sub	Tier-1 Enterprise Exact	Tier-1 Enterprise Sub	Tier-1 T/A Exact	Tier-1 T/A Sub	Tier-1 All Exact	Tier-1 All Sub	Content Tier-1 Exact	Content Tier-1 Sub	Content Content Exact	Content Content Sub	Content Enterprise Exact	Content Enterprise Sub	Content T/A Exact	Content T/A Sub	Content All Exact	Content All Sub
Ares	**58.5%**	72.3%	**93.4%**	**100.0%**	**97.3%**	**100.0%**	**96.2%**	**100.0%**	**85.7%**	92.8%	**50.0%**	**65.3%**	**98.9%**	**100.0%**	**95.5%**	**100.0%**	**94.4%**	**100.0%**	**91.4%**	**90.2%**
Metis	0.0%	0.0%	11.5%	19.8%	8.1%	20.0%	9.6%	17.2%	7.4%	14.1%	0.0%	0.0%	12.5%	12.0%	18.2%	27.0%	7.4%	12.1%	11.0%	12.1%
DFOH	32.1%	44.6%	27.1%	54.6%	73.0%	85.0%	65.4%	81.6%	46.8%	65.3%	25.0%	29.8%	31.8%	38.0%	56.8%	64.0%	50.0%	61.2%	41.0%	47.1%
BEAM-auto	0.0%	7.1%	0.0%	73.6%	0.0%	85.0%	0.0%	77.6%	0.0%	59.9%	0.0%	11.6%	0.0%	2.8%	0.0%	14.0%	0.0%	9.2%	0.0%	9.4%
BEAM-tweaked	5.7%	**90.2%**	26.2%	**100.0%**	5.4%	**100.0%**	15.4%	**100.0%**	14.3%	**97.4%**	0.0%	42.1%	4.5%	43.5%	2.3%	64.0%	1.9%	56.1%	2.9%	50.8%

Table 15: Recall of different methods on simulation dataset (**Type-2 Hijacks**; Victim Types: Enterprise & T/A)

Victim Attacker Affected prefix	Enterprise Tier-1 Exact	Enterprise Tier-1 Sub	Enterprise Content Exact	Enterprise Content Sub	Enterprise Enterprise Exact	Enterprise Enterprise Sub	Enterprise T/A Exact	Enterprise T/A Sub	Enterprise All Exact	Enterprise All Sub	T/A Tier-1 Exact	T/A Tier-1 Sub	T/A Content Exact	T/A Content Sub	T/A Enterprise Exact	T/A Enterprise Sub	T/A T/A Exact	T/A T/A Sub	T/A All Exact	T/A All Sub
Ares	**52.1%**	**56.2%**	**97.9%**	**95.2%**	**97.6%**	**96.2%**	**97.5%**	**93.3%**	**87.9%**	**85.2%**	**54.5%**	**66.3%**	**100.0%**	**99.0%**	**100.0%**	**99.0%**	**98.8%**	**99.0%**	**90.9%**	**90.8%**
Metis	0.0%	0.0%	15.6%	16.2%	23.5%	25.7%	20.3%	19.0%	15.4%	15.2%	0.0%	0.0%	19.8%	20.4%	23.1%	25.0%	16.9%	17.3%	16.1%	15.7%
DFOH	11.3%	14.3%	37.5%	36.2%	60.0%	59.0%	55.7%	59.6%	42.0%	42.2%	15.2%	17.3%	34.7%	34.0%	65.9%	68.3%	54.9%	56.3%	44.1%	44.0%
BEAM-auto	1.4%	7.6%	1.0%	2.9%	0.0%	5.7%	1.3%	10.6%	0.9%	6.7%	0.0%	10.6%	2.0%	3.9%	2.2%	12.5%	1.2%	9.7%	1.5%	9.2%
BEAM-tweaked	1.4%	29.5%	16.7%	37.1%	23.5%	46.7%	13.9%	43.3%	14.5%	39.1%	4.5%	47.1%	32.7%	49.5%	27.5%	58.7%	22.0%	56.3%	23.2%	52.9%

Table 16: Recall of different methods on simulation dataset (**Type-3 Hijacks**; Victim Types: Enterprise & T/A)

Victim Attacker Affected prefix	Enterprise Tier-1 Exact	Enterprise Tier-1 Sub	Enterprise Content Exact	Enterprise Content Sub	Enterprise Enterprise Exact	Enterprise Enterprise Sub	Enterprise T/A Exact	Enterprise T/A Sub	Enterprise All Exact	Enterprise All Sub	T/A Tier-1 Exact	T/A Tier-1 Sub	T/A Content Exact	T/A Content Sub	T/A Enterprise Exact	T/A Enterprise Sub	T/A T/A Exact	T/A T/A Sub	T/A All Exact	T/A All Sub
Ares	**47.4%**	**55.2%**	**95.6%**	**94.3%**	**97.3%**	**96.2%**	**97.1%**	**94.2%**	**89.7%**	**85.0%**	**55.8%**	**58.7%**	**99.0%**	**99.0%**	**98.6%**	**98.1%**	**96.7%**	**98.1%**	**91.4%**	**88.4%**
Metis	0.0%	0.0%	4.4%	8.6%	14.7%	19.0%	5.9%	9.6%	7.0%	9.3%	0.0%	0.0%	14.6%	9.7%	10.0%	14.4%	11.7%	14.6%	10.4%	9.7%
DFOH	5.3%	17.1%	22.0%	22.9%	50.7%	53.3%	38.2%	45.6%	31.6%	34.7%	20.9%	22.1%	26.0%	26.2%	50.0%	54.8%	51.7%	50.0%	37.2%	38.3%
BEAM-auto	0.0%	4.8%	0.0%	4.8%	0.0%	13.3%	0.0%	12.6%	0.0%	8.9%	0.0%	10.6%	0.0%	10.7%	0.0%	18.3%	0.0%	15.7%	0.0%	13.8%
BEAM-tweaked	0.0%	40.0%	11.0%	41.0%	5.3%	52.4%	10.3%	50.5%	7.7%	45.9%	0.0%	50.0%	17.7%	52.4%	8.6%	57.7%	8.3%	57.8%	10.4%	54.5%

methods applied by Ares, it may try to repeatedly announcing and withdrawing a hijacked route to evade Ares' detection. However, such operations are likely to cause the flapping route to be damped according to Route Flap Damping [48], making the hijack hard to succeed.