Poster: Adaptive Evasion of Tree-Based BGP Path Hijacking Detectors

Ilias Chanis
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
chanis@chalmers.se

Shahrooz Pouryousef
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
shahrooz.pouryousef@chalmers.se

Muoi Tran✉
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
muoi@chalmers.se

Abstract

Ares, the state of the art in BGP path-hijacking detection, aggregates AS_PATHs for each prefix seen by route collectors into Observed Prefix Routing Trees (OPRTs), clusters the trees of the same AS based on weighted tree-edit distances, and raises an alarm if a new tree fails to merge into an existing cluster and survives a set of anomaly-suppressing heuristics. We show that an adaptive adversary who knows Ares's pipeline can evade detection by (1) forging AS-PATHs that keep the post-attack OPRT within Ares's merge threshold, or (2) composing announcements from real, valley-free edges that bypass its filtering heuristic. On 13 historical hijacking events, these two attacks evade Ares in 76.9% and 53.8% of cases, and the largest reroutes traffic from 2,441 ASes (4.88% of the CAIDA AS graph) through the attacker without raising an alarm.

CCS Concepts

• **Security and privacy → Network security**.

Keywords

routing security; routing trees; tree edit distance; adaptive adversary

ACM Reference Format:
Ilias Chanis, Shahrooz Pouryousef, and Muoi Tran. 2026. Poster: Adaptive Evasion of Tree-Based BGP Path Hijacking Detectors. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3830454.3846444

1 Introduction

The Border Gateway Protocol (BGP) [9] lets tens of thousands of Autonomous Systems (ASes) exchange reachability information, and each AS re-advertises its neighbors' routes without verifying that the advertised AS_PATH was traversed [2]. Adversaries exploit this in *path hijacking attacks*, announcing a forged AS_PATH that inserts them into the route toward a victim prefix to attract and intercept traffic [1]. In Figure 1, origin O announces prefix 1.0.0.0/8 and adversary A hijacks it with the fake link A-O, rerouting victim traffic that formerly reached O via Q (green arrows) so that it now flows through A (red arrows). Because the origin AS is unchanged, origin authentication (e.g., ROV [8]) cannot prevent the attack.

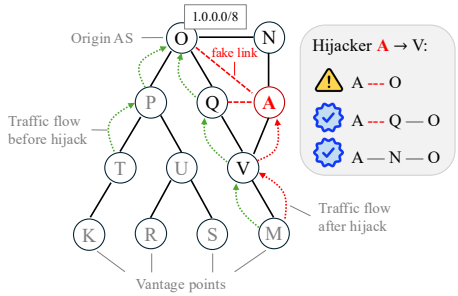

Figure 1: Ares [11] clusters the OPRT for prefix 1.0.0.0/8 against other trees of origin AS O. Ares detects a hijack from A when the fake link A-O is used as it leads to a new cluster. We devise two evasion strategies: (1) forging a link adaptively (e.g., A-Q) so the resulting tree still merges, and (2) using only real links (e.g., A-N-O) to bypass Ares's filtering heuristic.

Ares [11], the state-of-the-art path-hijacking detector, aggregates for each origin prefix the AS_PATHs from public route collectors (RouteViews [12] and RIPE RIS [10]) into an *Observed Prefix Routing Tree (OPRT)*. Ares clusters the OPRTs of each origin under a weighted tree-edit distance and raises an alarm when a new tree fails to merge into any cluster and survives a cascade of heuristics that suppress benign structural change. In Figure 1, the fake link A-O attaches a branch directly to the origin, producing a tree that does not merge into an existing cluster, so Ares raises an alarm.

Crucially, Ares depends entirely on the AS_PATHs that route collectors publish, data that the hijacking adversary can *fundamentally influence* through their own announcements. An *adaptive* adversary who understands Ares's detection pipeline, including OPRTs of targeted origin ASes, merge thresholds, and filtering heuristics, can therefore craft those announcements to evade detection.

We develop two such evasion strategies. The first strategy manipulates the OPRT directly, forging paths whose weighted contribution keeps the new tree below the merge threshold so that it folds into an existing cluster. In Figure 1, forging A-Q rather than A-O stays within the threshold because Q already lies near O, and the hijack merges undetected. The second strategy composes announcements from real, valley-free [5] edges recorded in CAIDA's AS-relationship dataset [7], bypassing Ares's filtering heuristic. Here the path A-N-O uses only genuine links, so it evades the heuristic while still routing traffic through A. The preliminary results with 13 historical events show that bypassing merge thresholds and heuristic respectively evades Ares in 76.9% and 53.8% of cases, hijacking traffic of up to 2,441 ASes (4.88% of all ASes).

This work is licensed under a Creative Commons Attribution 4.0 International License.
CCS '26, The Hague, Netherlands
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2871-6/2026/11
https://doi.org/10.1145/3830454.3846444

2 Two Evasion Attacks against Ares

2.1 Threat Model

Goals. The adversary hijacks traffic destined for a prefix of origin AS O, attracting routes through its own AS while evading Ares. It leaves the origin AS unchanged, so the announcement is a path hijacking that also evades RPKI validation.

Capabilities. The adversary controls a single AS A that originates BGP announcements with arbitrarily forged AS_PATH. Its announcements propagate under the same Gao–Rexford policies [5] as every other AS, so a forged route's reach is bounded by the adversary's business relationships and valley-free export.

Assumptions. The adversary is adaptive and knows all public information and Ares's published design. It reconstructs the pre-attack OPRT $\mathcal{T}$ of the victim prefix from the AS_PATHs published by RouteViews [12] and RIPE RIS [10] collectors. It reproduces the merge threshold τ for AS O's clusters by rerunning Ares's calibration over that same routing data,[1] and computes the AS-relationship graph $\mathcal{G}$ from CAIDA's public dataset [7].

2.2 Bypassing the Merge Threshold

This attack searches for a hijacking announcement whose post-attack OPRT $\mathcal{T}'$ stays within Ares's merge threshold of the pre-attack OPRT $\mathcal{T}$ (i.e., $|\mathcal{T}' - \mathcal{T}| \leq \tau$) so that Ares merges the two rather than raising an alarm. The key idea is that a fake link grafted deeper in the tree (e.g., $A - Q$ in Figure 1) diverts fewer ASes and leaves most of the tree unchanged. The adversary thus trades impact for stealth as each announcement hijacks only a small, targeted set of ASes, and repeating it extends the attack to further targets.

The adversary performs this search offline before advertising any route; see Algorithm 1. It scans the graft nodes Q of $\mathcal{T}$ from the root outward, testing the highest-impact hijacks first (line 1). For each Q, it reuses the real segment $Q - \cdots - O$ from $\mathcal{T}$ and prepends the attacker through the single fake link $A - Q$, forming the forged announcement π (lines 2–3). It then simulates Gao–Rexford propagation of π from A to $\mathcal{N}_A$, a subset of A's neighbors, and reconstructs the resulting $\mathcal{T}'$ (line 4). As a deeper graft perturbs less of the tree, $|\mathcal{T}' - \mathcal{T}|$ shrinks as the scan moves outward. The first Q satisfying $|\mathcal{T}' - \mathcal{T}| \leq \tau$ (lines 5–6) thus yields the most impactful hijack Ares still merges into $\mathcal{T}$'s cluster, returned as π. If no such Q exists, the algorithm reports so (line 9).

In practice, Ares measures $\mathcal{T}'$'s distance to existing clusters instead of only $\mathcal{T}$, so Algorithm 1 serves as a candidate generator rather than an exact oracle. The adversary confirms a candidate by replaying π against Ares, and reruns the search if Ares detects it.

2.3 Bypassing Path-Validity Heuristic

This attack evades Ares by satisfying its path-validity heuristic, which suppresses alarms for any announcement whose AS-PATH uses only CAIDA-recorded AS relationships and respects valley-free routing. The key idea is to forge the AS-PATH entirely from real relationship edges from unrelated prefixes, so that Ares deems the hijack legitimate even though the route it induces is fake.

[1] Ares sets τ from the knee point of the observed merge distances plus a fixed multiple of their median absolute deviation [11].

Algorithm 1 Offline search for an evasive hijack announcement

Require: Attacker A, receiver set $\mathcal{N}_A$ (a subset of A's neighbors), origin O, pre-attack OPRT $\mathcal{T}$, AS graph $\mathcal{G}$, threshold τ
Ensure: Evasive announcement π, or $\varnothing$ if none exists
1: **for all** graft node $Q \in \mathcal{T}$, by increasing depth **do**
2: $s \leftarrow \textsc{SegmentToRoot}(\mathcal{T}, Q)$ ▷ Real segment $Q - \cdots - O$ from $\mathcal{T}$
3: $\pi \leftarrow \langle A \rangle \| s$ ▷ Prepend A via fake link $A - Q$
4: $\mathcal{T}' \leftarrow \textsc{SimulateGR}(\mathcal{G}, A, \pi, \mathcal{N}_A)$ ▷ A advertises π to $\mathcal{N}_A$
5: **if** $|\mathcal{T}' - \mathcal{T}| \leq \tau$ **then**
6: **return** π ▷ Evasive candidate: $\mathcal{T}'$ likely stays in $\mathcal{T}$'s cluster
7: **end if**
8: **end for**
9: **return** $\varnothing$

To construct such a path, the adversary searches the valley-free product graph induced by the CAIDA relationships $\mathcal{G}$ for routes from origin O to attacker A within a chosen length range. Reversing each route yields a forged path $\pi = \langle A, X, \ldots, O \rangle$ that A can announce, where X is A's first hop toward O. Attacker A then announces π as widely as possible by exporting it to all neighbors when X is a customer and to its customers only otherwise, excluding the origin and any AS already on π to keep the propagation valley-free. Since every edge of π is a real relationship and π is valley-free by construction, the hijacking announcements satisfy the path-validity heuristic and thus bypass Ares's detection.

3 Evaluation

3.1 Methodology

Dataset. Like Ares's own evaluation, we assess our attacks on historical path hijacking events. Our dataset consists of 13 events reported by Cho et al. [4], each of which pairs the reported hijacker, victim, and affected prefix with the corresponding RIPE RIS data. For each event, we reconstruct the pre-attack OPRT $\mathcal{T}$ from the AS_PATHs in the closest snapshot before the hijack, and we take the CAIDA AS-relationship snapshot from the same month as the graph $\mathcal{G}$. We leave experiments on synthetic attacks to future work.

Simulation. We cannot observe how ASes would have responded to our forged announcements during these events, so we simulate their propagation with the Gao–Rexford model on inferred CAIDA data, inheriting the limitations of both. Following each attack's export policy (§2.2, §2.3), we propagate π over $\mathcal{G}$ with a preference-aware breadth-first search. The simulation then determines which ASes adopt the forged route from the attacker, and in turn whether the attack evades Ares and how much traffic it disrupts.

Platform. We run our experiments on an HPC cluster with nodes of 2–4 AMD EPYC 9354 (Zen 4) CPU cores and 768 GB of RAM. This hardware is needed only to run Ares. Our attack search and simulation are lightweight enough to run on a commodity laptop.

Metrics. We consider an attack successful when it both evades Ares and redirects traffic through the attacker. For each successful announcement π, we record the switched ASes $\mathcal{S}$, which adopt the forged route, and the disrupted ASes $\mathcal{D}$, whose traffic traverses the attacker, with $\mathcal{S} \subseteq \mathcal{D}$. We measure impact by the number of disrupted ASes $|\mathcal{D}|$ and by the disrupted fraction $|\mathcal{D}|/N$, where N is the total number of ASes in the CAIDA AS-relationship graph.

Table 1: Historical BGP path hijacking events (2008–2016) and our evasion results. *Evasion* reports whether each method evades Ares (✓) or not (✗). *Impact* of the best attack gives the number of switched and disrupted ASes, and the disrupted fraction.

		Evasion		Impact of best attack			
Event	**Year**	**Threshold**	**Heuristic**	**Switched**	**Disrupted**	**% of Net**	**#ASes (N)**
defcon_1	2008	✓	✓	503	1,399	4.79	29,178
facebook_1	2011	✓	✓	4	1,203	3.23	37,233
france_1	2012	✓	✓	62	1,760	4.08	43,109
torg_1	2014	✗	✗	–	–	–	46,063
torg_2	2014	✗	✗	–	–	–	46,063
petersburg_1	2015	✓	✓	453	562	1.22	46,172
petersburg_2	2015	✓	✓	453	562	1.22	46,172
bitcanal_3	2015	✓	✓	1	622	1.35	46,172
bitcanal_4	2015	✓	✓	1	622	1.35	46,172
enzu_1	2015	✓	✗	2	2,441	**4.88**	50,014
backconnect_3	2016	✓	✗	1	161	0.30	53,792
backconnect_6	2016	✓	✗	1	9	0.02	53,792
typo_1	2016	✗	✗	–	–	–	54,140
Total		**10/13**	**7/13**				

3.2 Results

Table 1 summarizes the results for the 13 historical hijacking events. Tree manipulation succeeds in 10 of the 13 events (76.9%), and heuristic bypass in 7 (53.8%). Every event that heuristic bypass evades, tree manipulation also evades, so tree manipulation strictly dominates on our dataset. Heuristic bypass finds evasive attacks in many events, but each disrupts few ASes. Tree manipulation instead produces far larger disruptions, reaching up to 2,441 ASes (4.88% of the CAIDA AS graph) and exceeding 4% in three events.

The number of switched ASes $|\mathcal{S}|$ does not determine the resulting disruption. In `enzu_1`, only two ASes adopt the forged route, yet one is AS174 (Cogent Communications), a Tier-1 transit provider, so traffic from 2,441 ASes flows through the attacker. We also observe this amplification in other events, e.g., `facebook_1` and `bitcanal_3`. Attack impact thus depends more on *which* ASes switch than on *how many*, and a stealthy attack that flips one well-placed AS can still hijack a large number of ASes.

Our experiments expose two factors that limit attack success. First, both methods depend strongly on the attacker's location in the AS graph. The path-validity heuristic requires valley-free export, so the attacker's customer cone bounds how widely a forged route spreads. Even under tree manipulation, disruption remains small when the chosen targets reach no large downstream region. Second, in the three events where tree manipulation fails, visibility rather than propagation is the limiting factor. The pre-attack OPRTs are too shallow, so collectors observe the forged route often enough that the post-attack tree exceeds Ares's threshold ($|\mathcal{T}' - \mathcal{T}| > \tau$).

4 Conclusion

We demonstrate that an adaptive adversary can evade Ares, the state of the art in BGP path-hijacking detection, by manipulating the public data that it relies on. This attack raises broader questions about the robustness of data-driven hijacking detectors:

How far does this weakness generalize? A stronger adversary could search jointly over attacker placement, victim set, and forged path, and detectors such as DFOH [6] and BEAM [3] draw on similarly public data that may expose the same weaknesses.

Can a detector remain robust when the adversary observes and manipulates its input? This may call for detection signals an adversary cannot fully control, such as data-plane measurements.

Acknowledgments

We would like to thank our reviewers for their valuable feedback and suggestions. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Hitesh Ballani, Paul Francis, and Xinyang Zhang. 2007. A Study of Prefix Hijacking and Interception in the Internet. In *Proc. ACM SIGCOMM.*
[2] Kevin Butler, Toni R. Farley, Patrick McDaniel, and Jennifer Rexford. 2010. A Survey of BGP Security Issues and Solutions. *Proc. of the IEEE.*
[3] Yihao Chen, Qilei Yin, Qi Li, Zhuotao Liu, Ke Xu, Yi Xu, Mingwei Xu, Ziqian Liu, and Jianping Wu. 2024. Towards a Semantics-Aware Routing Anomaly Detection System. In *Proc. USENIX Security.*
[4] Shinyoung Cho, Romain Fontugne, Kenjiro Cho, Alberto Dainotti, and Phillipa Gill. 2019. BGP hijacking classification. In *Proc. IEEE TMA.*
[5] Lixin Gao. 2001. On inferring autonomous system relationships in the Internet. *IEEE/ACM ToN.*
[6] Thomas Holterbach, Thomas Alfroy, Amreesh Phokeer, Alberto Dainotti, and Cristel Pelsser. 2024. A System to Detect Forged-Origin BGP Hijacks. In *Proc. USENIX NSDI.*
[7] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and kc claffy. 2013. AS Relationships, Customer Cones, and Validation. In *Proc. ACM IMC.*
[8] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein. 2013. RFC 6811: BGP Prefix Origin Validation. (2013).
[9] Y. Rekhter, T. Li, and S. Hares. 2006. RFC 4271 - A Border Gateway Protocol 4 (BGP-4). (2006).
[10] RIPE NCC. 2026. Routing Information Service (RIS). (2026).
[11] Yinxiang Tao, Chengwan Zhang, Changqing An, Shuying Zhuang, Jilong Wang, and Congcong Miao. 2025. Comprehensive Path Hijacking Detection via Routing Tree. In *Proc. USENIX Security.*
[12] University of Oregon. 2026. Route Views Project. (2026).