Passive Data-Plane Telemetry to Mitigate Long-Distance BGP Hijacks

Satadal Sengupta
Department of Computer Science, Princeton University, USA

Hyojoon Kim
Department of Computer Science, University of Virginia, USA

Daniel Jubas[1]
Five Rings LLC., USA

Maria Apostolaki
Department of Electrical and Computer Engineering, Princeton University, USA

Jennifer Rexford
Department of Computer Science, Princeton University, USA

Abstract

Poor security of Internet routing enables adversaries to divert user data through unintended infrastructures in attacks known as hijacks. Of particular concern—and the focus of this paper—are cases where attackers reroute domestic traffic through foreign countries and still deliver it to the intended destination, exposing traffic to surveillance, bypassing legal privacy protections, and posing national security threats. Efforts to detect and mitigate such attacks have focused primarily on the control plane, while data-plane signals remain largely overlooked. In this paper, we argue that *passively-monitored round-trip time (RTT)*—and, in particular, changes in its *propagation-delay* component—offers a promising signal for detection: the increased propagation delay is unavoidable and directly observable from affected networks, enabling opportunities for faster detection and mitigation. We explore the practicality of using RTT variations for hijack detection, addressing two key questions: (1) What coverage can this provide, given its heavy dependence on the geolocations of the sender, receiver, and adversary? and (2) Can an always-on RTT-based detection system be deployed without disrupting normal network operations? Focusing on cross-country interception attacks, we find that coverage is high: 97% under ideal (i.e., data travels at the speed of light) conditions, and 91% and 86% with real traffic from two datasets. To demonstrate practicality, we design HiDe, which reliably detects delay surges from long-distance hijacks at line rate using commodity programmable hardware. We measure HiDes accuracy and false-positive rate on real-world data and validate it with ethically conducted hijacks.

2012 ACM Subject Classification Networks → Network properties; Networks → Network protocols; Networks → Network services; Networks → Routing protocols; Networks → Network security; Networks → Programmable networks; Networks → Network monitoring; Networks → In-network processing; Security and privacy → Network security

Keywords and phrases Network security, routing, Border Gateway Protocol, hijack, interception attack, programmable networks, in-network detection, in-network mitigation

Digital Object Identifier 10.4230/OASIcs.NINeS.2026.14

Related Version *Previous Version*: `https://arxiv.org/abs/2507.14842`

Supplementary Material *Software (Source Code)*: `https://github.com/Princeton-Cabernet/HiDe`
archived at `SoftwareHeritageIdentifier`

[1] Work done as a Master's student at Princeton University.

1 Introduction

BGP (Border Gateway Protocol) hijacks are a long-standing threat where attackers exploit the poor security of BGP—the Internet's default routing protocol—to redirect traffic through their own infrastructure. These attacks can be dangerous, allowing the theft of sensitive data and inflicting severe financial damage [27, 19, 15, 6, 13, 3, 1, 31]. Despite years of research, BGP hijacks remain a serious threat today [32, 7]. Major prior efforts have attempted to *proactively* neutralize this threat, but suffer from limited adoption or scope: Standards like BGPsec [5] and clean-slate alternatives such as SCION [29] require ubiquitous adoption to be truly effective, which is challenging given the decentralized nature of the Internet. Meanwhile, mechanisms like RPKI (Resource Public Key Infrastructure) [12] can prevent only a subset of attacks (i.e., forged-origin hijacks), leaving other attack classes unaddressed.

Consequently, network operators rely heavily on *reactive* approaches to defend against BGP hijacks. State-of-the-art reactive approaches primarily work in the control plane by monitoring BGP feeds from public monitors like RIS and RouteViews, private commercially-operated monitors, and/or the network's own routers. Such systems (e.g., ARTEMIS [36], DFOH [22]) detect suspicious announcements, and mitigate attacks using techniques such as announcing *more-specific* prefixes. These approaches detect attacks quickly and accurately when malicious announcements are immediately visible from their vantage points, but they fall short in two key scenarios. First, they may entirely miss attacks in which malicious announcements are specifically designed to evade control-plane-based detection [7, 9, 26, 41]. For example, attackers can manipulate BGP communities to surgically steer traffic while limiting visibility at public monitors [9]. Similarly, in *interception attacks*, the adversary diverts traffic through its own infrastructure but still forwards it to the intended destination, reducing the likelihood of detection [4, 20, 41]. Second, some attacks may eventually become visible at route monitors, but only after sufficient time has passed for the attacker to cause significant damage. For example, website fingerprinting attacks can succeed within a few seconds of the hijack, since they may require only the first few packets, bursts, or bytes of page content [39, 17, 37]. Similarly, in hijacks such as the one on Amazons DNS infrastructure in 2018, attackers could collect users credentials within the first few seconds, which later enabled them to steal cryptocurrency [30]. Other reactive approaches (e.g., iSPY [42]) implement *active probing* in the data plane (e.g., ping, traceroute), but require responsive IPs and sustained probing to maintain accuracy, resulting in challenges with scalability (due to probe traffic overhead) and coverage (since some IPs may not be responsive). Hybrid methods (e.g., Argus [38], HEAP [33]) combine control and data plane signals to improve accuracy, but do not alleviate challenges with scalability, attack visibility, or detection speed.

In this paper, we argue that *passively-monitored round-trip time (RTT)* is a highly useful yet underexplored data-plane signal for detecting BGP hijacks. In particular, hijacks induce a change in the *propagation-delay* component of RTT by changing the physical path of traffic. Unlike control-plane signals, RTT cannot be hidden from the victim. For instance, if an attacker from North Korea launches an interception attack on a connection between two hosts within the UK, the traffic must cover an additional $15,025$ kilometers at least, causing a minimum additional RTT of 75 ms[2]—an effect the victim will directly experience. The RTT change is also *immediate*, enabling an opportunity for faster detection and mitigation

[2] This example assumes the speed of data transmission to be the speed of light in optical fiber, given by $c_f = 2c/3$ (approx.), where c is the speed of light in vacuum [23]. Hereafter, in this paper, *speed of light* refers to c_f, which is approx. 200 km per millisecond (more precisely, 199.86 km/ms).

compared to existing reactive approaches. Additionally, contrary to active probing, passive monitoring does not require responsive IPs and incurs no probing overhead.

However, building an effective RTT-based BGP hijack detector is challenging. The expected increase in propagation delay—which is at the heart of an RTT-based approach—is highly *location-dependent*. To be detectable, this increase in propagation delay must be large enough to stand out from the natural RTT variation that occurs in real traffic. Such variation can be caused by many factors, including network congestion, host processing times, noise in access networks, and benign route changes (e.g., due to traffic engineering). As a result, to be reliably detected, the diversion must be geographically long enough to induce an RTT increase noticeably higher than the natural variation. For example, an RTT-based detector might successfully flag traffic between hosts in the US being diverted through the UK. In contrast, diversions over shorter distances may be difficult, and in some cases even impossible, to detect using RTT alone—for example, if US traffic is diverted through Canada, or in regions such as Europe where many countries are physically close and even cross-country detours may not induce a sufficiently large RTT increase. For this reason, RTT is not a silver bullet that enables a general-purpose solution for hijack detection.

Despite its limitations in scope, an RTT-based hijack defense is promising since it addresses key weaknesses of existing approaches in four main ways. First, it can detect the *long-distance* subset of stealthy attacks that control-plane-based approaches miss entirely. Second, for attacks that are not immediately visible in the control plane, it can detect the long-distance subset much more quickly (sometimes within milliseconds for high-data-rate flows), thereby reducing the amount of traffic exposed to the attacker before mitigation. Third, it can serve as an additional signal for existing hijack detectors, where combining control-plane and data-plane information can improve robustness and reduce false positives. Fourth, in such a combined setting, by virtue of its placement in the data plane, it enables finer-grained, operator-configurable mitigation strategies compared to control-plane-based mitigation that can only operate at the granularity of an entire prefix (e.g., a /24)—for example, traffic to sensitive IPs within the prefix could be temporarily rate-limited, or even blocked when the data plane suspects an attack, awaiting confirmation from the control plane.

Among BGP hijacks that cause substantial harm, and that lie within the scope of BGP hijacks where an RTT-based defense is most effective, one class is of particular concern and is the focus of this paper: hijacks that reroute domestic traffic through a foreign location and still deliver it to the intended destination. These *cross-country interception attacks* are especially troubling because they, unbeknownst to the user, expose the users traffic to different jurisdictions and thus to different privacy and surveillance laws [18, 20, 41]. However, as noted earlier, an RTT-based defense may not be able to detect *all* cross-country attacks (e.g., US via Canada, Germany via France, etc.) Our first major goal in this study is to understand the effectiveness of RTT-based hijack detection for cross-country attacks. In particular, we ask the research question (RQ1): *What fraction of possible cross-country interception attacks can an RTT-based approach reliably detect?* We find that cross-country attacks are highly detectable using RTT: overall, 97% can be detected reliably under ideal conditions (i.e., data travels at the speed of light and is not affected by real-world noise), and detection remains high at 91% and 86% even with real traffic from two datasets (§3).

Although cross-country interception attacks are highly detectable, doing so in a scalable, real-time manner remains challenging. Per-packet RTT calculation is expensive at line rate, and maintaining state while monitoring every flow is intractable. Thus, we pose the following research question (RQ2): *Can we design a practical, always-on monitoring system to detect* *and mitigate cross-country interception attacks in a scalable manner without excessive cost?*

To this end, we design HiDe, a practical RTT-based system for detecting hijacks. First, BGP hijacks occur at the IP-prefix level and affect all traffic routed to the targeted prefix, meaning that during a real hijack, no flow to the targeted prefix can have an RTT less than the minimum required for the attackers route. By passively measuring the RTTs experienced by as many packets as possible and relying on the minimum per prefix, HiDe can reliably and scalably detect spurious RTT surges. Second, we observe that a BGP hijack induces a distinct pattern in the *denoised* RTT over time, clearly differentiating it from other events, such as congestion. Concretely, a hijack causes a *sharp* surge with location-dependent but calculable *minimum height*, which one can detect using a changepoint detection algorithm. Third, implementing HiDe can be made practical by deploying it on high-speed programmable hardware (e.g., switches). This is possible, despite the rigid computation and memory constraints of such devices, thanks to our switch-native implementation of changepoint detection and scalable latency measurements.

Our evaluation demonstrates that HiDe is reliable (zero false negatives by design), minimally disruptive to real traffic, and implementable on commodity hardware. To assess HiDe's fidelity, we tested it against ethically-conducted real-world hijacks [3] and found that it detects them within 0.5 second. Additionally, to evaluate its impact on regular operations, we run HiDe on campus network traces (19 billion packets, 5.3 TB bytes). The results show that its combination of algorithms effectively minimizes false alarms ($< 0.012\%$), even in the presence of highly noisy real-world RTT signals. Furthermore, HiDe reduces the impact of such false alarms by identifying and correcting them within a median of 0.75 seconds without human intervention. We implement HiDe entirely on a programmable switch, showcasing its potential for seamless deployability on a network's border gateway with minimal hardware cost and no delay overhead to normal traffic.

2 Background and Threat Model

2.1 BGP-based attacks

BGP is the primary protocol that connects Autonomous Systems (ASes) by enabling them to exchange and forward route announcements for IP prefixes. Each AS advertises routes for the prefixes it owns, including an AS path indicating the sequence of ASes to traverse to reach it. Routers independently select the best route for each prefix based on attributes like path length and routing policies.

BGP hijack. A BGP hijack occurs when a malicious or compromised AS falsely advertises routes to IP prefixes it does not own or cannot reach, misleading other ASes into rerouting traffic through its infrastructure. Suppose $AS100$ legitimately owns the IP prefix 1.1.1.0/24. A malicious AS, $AS200$, falsely announces ownership of 1.1.1.0/24 to its BGP peers. These peers may accept the announcement as valid and propagate it to their own peers, spreading the false route across the network. As a result, traffic destined for 1.1.1.0/24—for instance, originating from another prefix like 2.2.2.0/24 owned by $AS300$—may get misrouted to $AS200$ instead of reaching $AS100$. This enables the attacker to eavesdrop on, fingerprint, manipulate, or drop this illegitimately-obtained traffic. In some cases, the attacker may even serve malicious content by impersonating the legitimate destination (e.g., by acquiring

[3] Ethical issues are discussed in our *Ethics* section (Appendix A).

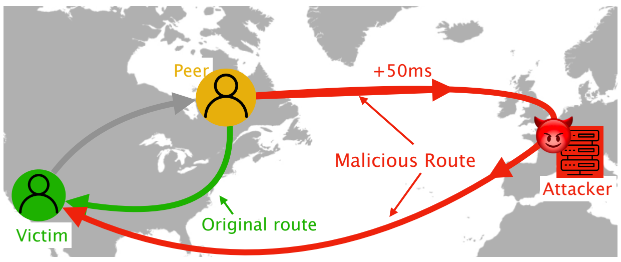


Figure 1 An attacker in the UK exploits the weakness of routing security to redirect traffic from a *peer* host in the US—originally destined for a *victim* host in the US—through the attackers own infrastructure. The *mid-attack* path (in red) from the peer to the victim is longer than the original *pre-attack* path (in green), adding an extra 50 ms of propagation delay to the RTT of the traffic.

a *valid* certificate first by exploiting weakness in the certificate issuance verification process [8]).

BGP interception attack. A BGP *interception attack* (Figure 1) is a specific type of BGP hijack where the attacker intercepts traffic but forwards it to the original destination, enabling analysis or manipulation while remaining undetected by the end hosts. Of particular concern is a stealthy subset of these attacks where a sophisticated attacker employs techniques like *AS-path poisoning* and the manipulation of *BGP communities* to limit the propagation of malicious announcements in a bid to evade detection by BGP monitors near the victim. For instance, in the example above, $AS200$ could manipulate its announcements to propagate only to routers near $AS300$ while suppressing those to routers near $AS100$. This could cause traffic from 2.2.2.0/24 to 1.1.1.0/24 to get misrouted via $AS200$, while $AS100$ remains unaware as BGP monitors near it never observe the malicious route. Such an attack has been demonstrated by Birge-Lee et al. [9], which we ethically reproduce in §7.2.

2.2 Threat model

We consider an adversary performing a *stealthy* BGP interception attack—such as the one described above—to reroute traffic destined for a victim through distant infrastructure before forwarding it back to the victim. This infrastructure is located in a different country, potentially under different privacy and security laws. The adversary is sophisticated, aware of detection systems, and employs evasion techniques to prevent forged announcements from reaching BGP monitors leveraged by the detection systems [9]. By rerouting traffic back to the victim, the attacker keeps connections alive, enabling traffic analysis while evading detection at the application layer. Such attacks can serve as tools in cyber warfare or surveillance. Real-world examples of long-distance interceptions include (among numerous others) the rerouting of US-based traffic via the UK to enable surveillance (Figure 1) [18], rerouting of US-based traffic managed by China Telecom via China undetected over 2.5 years [20], and rerouting of traffic between two hosts in Denver, USA via Iceland [41].

3 Feasibility Study

HiDe relies on the propagation-delay component of RTT for real-time BGP hijack detection. Propagation delay is the time it takes a packet to travel across the network path from the sender to the receiver, determined primarily by the physical distance and the transmission medium, rather than by network congestion or processing delays at the end hosts. In this section, we examine the *feasibility* of using propagation delay alone to defend against cross-country interception attacks. In such attacks, both the victim and its peer reside in a *victim country* C_V, while the attacker operates from a different *threat country* C_T. Such attacks are common in cyber warfare and surveillance carried out by nation states.

3.1 Key questions and observations

We ask the following questions about cross-country attacks:

1. Does traffic within a single country experience significantly lower propagation delay compared to traffic across countries? If so, can we use this difference to detect cross-country interception attacks?
2. Are all countries equally *defendable* using propagation delay-based detection? If not, and some countries are more defendable, what geographic factors drive this difference?
3. What fraction of cross-country attacks can, in principle, be detected by comparing propagation delays—both under idealized assumptions and real-world measurements?

Our analysis in this section reveals the following:

1. Across 258 countries, the maximum distance within a country's borders (*max. intra-country distance*) is typically far smaller than the minimum distance from that country to any other country (*min. inter-country distance*). The $25^{th}/50^{th}/75^{th}$ percentiles of max. intracountry distances are 49 km, 413 km, and 1,129 km, respectively; the corresponding percentiles for min. intercountry distances are 4,027 km, 7,689 km, and 11,420 km. Naturally, propagation delays follow a similar trend. (§3.2).
2. Some countries are more defendable using propagation delay-based detection than others. For example, Russia ranks among the least defendable, whereas New Zealand is one of the most defendable. More generally, larger countries with many nearby neighboring countries are less defendable, while smaller or more geographically isolated countries are more defendable. (§3.4).
3. Considering the worst-case (least defendable) attack paths between every pair of countries, we find that 97% cross-country attacks can be detected assuming speed-of-light RTTs, and 91% and 86% respectively, using realworld measurements from two production datasets. (§3.4, §3.5).

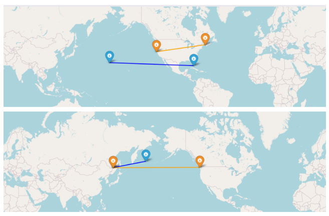
3.2 Intra-country vs. inter-country distances

Our first goal is to assess whether the greatcircle distance (shortest distance along Earth's curvature) between two hosts in the same country is significantly smaller than between hosts in different countries, so a crosscountry detour would induce a detectable increase in propagation delay.

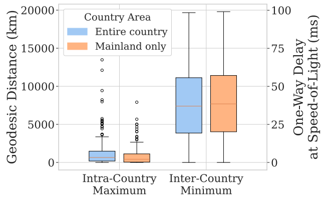
Dataset. We use the *Natural Earth Admin-0 Countries* dataset—one of the most popular boundary datasets in the Geographic Information System community. It provides highresolution boundary coordinates (avg. 2.5 km) for 258 countries (Figure 2a) [16]. Since some countries consist of multiple disconnected regions (e.g., contiguous US plus Alaska and islands), we distinguish each countrys *mainland* (largest contiguous landmass) from *entire country* (all regions combined).

Method. For each country (entire country and mainland), we calculate: (1) Max. intra-country distance: the largest pairwise great-circle distance among all boundary points of the country, and (2) Min. inter-country distance: the smallest great-circle distance from any

(a) Centroids (orange icons) of the mainlands of 258 countries in the Natural Earth Admin-0 dataset [16].

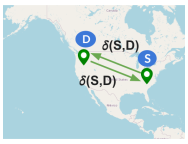


(b) Top: Max. distance within country. Bottom: Min. distance between countries. (Blue: Entire country, orange: mainland).

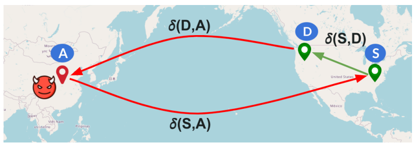


(c) Distribution of max. within and min. between countries: distance (left y-axis) and one-way delay at c_f (right y-axis).

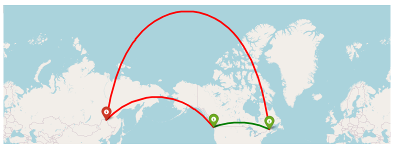
Figure 2 For all 258 countries (Figure a)—using both entire country areas and mainlands only (Figure b)—we compute each countrys maximum internal distance and its minimum distance to every other country, then plot both distributions (Figure c). Typically, a countrys foreign neighbors are more distant than its own farthest points.



(a) Distance before attack: $\delta_{pre} = 2\delta(S, D)$.



(b) Distance in the middle of the stealthy interception attack: $\delta_{mid} = \delta(S, D) + \delta(S, A) + \delta(D, A)$.



(c) Locations of source, destination in the US and attacker in China such that $\delta_{mid} - \delta_{pre}$ is minimized.

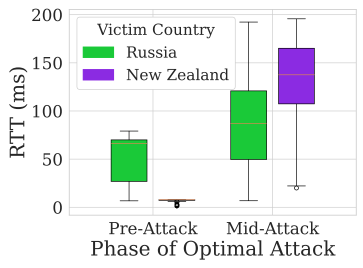
Figure 3 In this example, source S and destination D lie in mainland US and attacker A in mainland China. Figure (a) shows the *pre-attack* round-trip distance δ_{pre} and (b) the *mid-attack* round-trip distance δ_{mid}, leading to the deviation $\delta_{deviation} = \delta_{mid} - \delta_{pre} = \delta(S, A) + \delta(D, A) - \delta(S, D)$. Figure (c) shows the *most optimal attack* on the US from China, with curved lines indicating shortest great-circle paths. (Green: Original path, red: diversion due to attack.)

point in this country to any point in another. Figure 2b illustrates these distances within the US (top) and between the US and China (bottom).

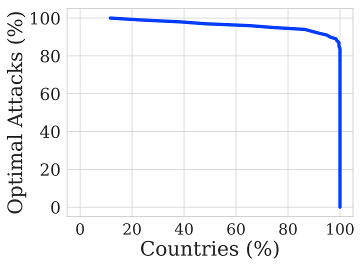
Observations. Figure 2c shows: (1) min. inter-country distances far exceed max. intra-country distances, and (2) these distances for entire countries vs. mainlands are similar: we proceed with mainlands in the rest of this paper for better interpretability. At the speed of light, the $25^{th}/50^{th}/75^{th}$ percentile max. intra-country one-way delay (OWD) are 0.2 ms, 2.1 ms, and 5.6 ms, respectively. The corresponding values for min. intercountry OWD are 20 ms, 38 ms, and 57 ms—at least an order of magnitude larger.

3.3 Identifying least defendable attacks

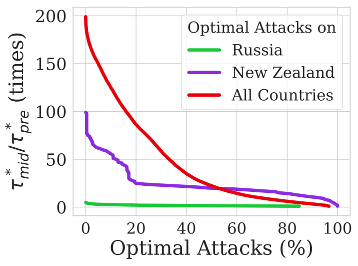
In this subsection, we describe how we identify the least defendable attack given a C_V and a C_T; later, we analyze defendability under ideal (§3.4) and realistic (§3.5) conditions. Figures 3a, 3b show the paths before and during the attack: with source S and destination D in C_V and attacker A in C_T, the round-trip distance changes from $\delta_{pre} = 2\delta(S, D)$ to $\delta_{mid} = \delta(S, D) + \delta(S, A) + \delta(D, A)$, resulting in a deviation of $\delta_{deviation} = \delta(S, A) + \delta(D, A) - \delta(S, D)$. In the worst-case (least defendable) attack scenario, $\delta_{deviation}$ is minimized by having S and D on C_V's border and as far apart from each other as possible, and A on C_T's border and as close to S and D as possible—we call this an *optimal attack*. Under such an attack, we denote the pre-, mid-attack distances, and deviation as δ^*_{pre}, δ^*_{mid}, and $\delta^*_{deviation}$, respectively, and



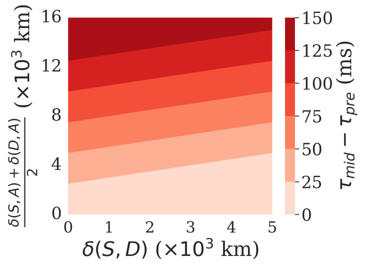
(a) Pre- and mid-attack RTTs during all possible optimal attacks on 2 example countries.



(b) Percentage of countries vs. percentage of defendable optimal attacks.



(c) Attack coverage vs. mid-attack RTT (as a multiple of pre-attack RTT).



(d) Deviation (z) as a function of dist. b/w hosts (x) and avg. dist. b/w hosts & attacker (y).

Figure 4 Defendability against optimal attacks assuming speed-of-light RTT: With Russia and New Zealand (NZ) as example victim countries, (a) shows mid-attack RTT is typically much higher than pre-attack RTT; size and proximity of victim country to other countries determine the extent. Figure (b) shows that for 86% countries can be defended against 94% optimal attacks. Figure (c) shows Russias post-attack RTT peaks at 4x its pre-attack RTT (corresponding to 110 ms absolute difference), NZ at 100x (190 ms), and all countries combined at 198x (200 ms). Figure (d) shows that, when the victim and peer are co-located, the attacker must be 2,500 km away to induce a deviation of 25 ms; as the victim and peer separate, the attacker must be more distant to induce the same deviation.

the corresponding RTTs as τ^*_{pre}, τ^*_{mid}, and $\tau^*_{deviation}$. Figure 3c shows the optimal attack from China on the US, as an example. We compute the optimal attack for each ordered pair of victim and threat countries (258 x 257 attack scenarios) and use these scenarios in our analysis in the next two subsections.

3.4 Defendability under ideal conditions

In this subsection, we analyze the feasibility of detecting cross-country optimal attacks under *ideal conditions* (defined below).

Assumptions. *Ideal conditions* include: (1) Ideal network, i.e., data travels at the speed of light, and (2) Ideal measurements, i.e., our measurements capture the actual distance-based propagation delay. Under these assumptions, propagation delay = minimum RTT (minRTT) = RTT. Therefore, in this subsection, *RTT* is synonymous with propagation delay. We relax these assumptions in the next subsection where we analyze real-world measurements (§3.5).

Most and least defendable countries. Optimal attacks determine the lower bound of our defense capabilities. To assess defendability under optimal attacks, we compute propagation delay as the round-trip distance (δ^*_{pre} or δ^*_{mid}) divided by the speed of light. To illustrate the difference in defendability across countries, we select two examples: Russia, which has among the smallest median $\tau^*_{deviation}$, and New Zealand (NZ), which has one of the largest median $\tau^*_{deviation}$. Figure 4a shows their pre-attack and mid-attack RTT distributions. NZs RTTs increase significantly mid-attack, making it more defendable; Russias RTT changes are smaller making it less defendable. In general, small countries with distant neighbors (low $\delta(S, D)$, high $\delta(S, A) + \delta(D, A)$) are the most defendable, while large countries with many close neighbors are the least defendable.

Attack coverage. To quantify defendability, we define *attack coverage* as the percentage of optimal attacks that can be detected under some given condition. To compute overall coverage, we set the condition $\tau^*_{deviation} \geq 5$ ms because: (1) 5 ms far exceeds typical noise in measurements (e.g., due to coarse-grained timestamps, rounding off errors, approximations in distance calculations, etc.), and (2) At speed of light, 5 ms corresponds to approx. 1,000 km of extra path length, so it captures significant geographic detours. The attack coverage for Russia (over 257 optimal attacks) is 85% (the minimum for any country), while the same for NZ is 100%. When expanded to all countries (i.e., 258 x 257 optimal attacks), the coverage is 96.6%. Figure 4b shows that 100% (i.e., all) countries can be defended against 84% attacks, 75% against 95%, 23% against 99%, and 11% against 100%. These results illustrate the promise and generality of propagation delay-based detection (in ideal conditions).

Attack coverage at given RTT deviations. To analyze the extent to which optimal attacks increase RTT in ideal conditions, we plot the attack coverage (x-axis) given the ratio between mid- and pre-attack RTT (Figure 4c). For NZ (purple), this ratio ranges from 1-100x (5-190 ms absolute difference), while for Russia (green), due to its large pre-attack RTTs, it is 1-4x (5-110 ms). For attacks on all countries (red), it is 1-198x (5-200 ms). The ratio is 2x (8 ms) for 95% attack coverage on all countries, and 4x (23 ms) for 85% coverage.

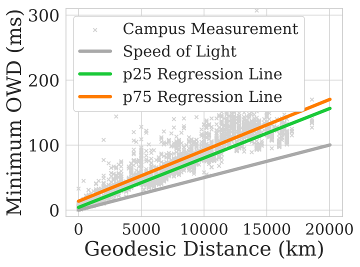
Deviation as a function of distances. The analysis of cross-country attacks does not inform us directly about the relationship between distance and RTT deviation. To bridge this gap, Figure 4d shows RTT deviation (z-axis)—as a function of pre-attack distance ($\delta(S, D)$) (x-axis), and average distance between S-A and D-A (y-axis)—in a heatmap. The x- and y-axis are capped at the maximum intra- and minimum inter-country distances. The $85^{th}/95^{th}$ percentile pre-attack distances are 1,090 km and 1,872 km; to induce an RTT deviation of 25 ms, the attacker needs to be at an average distance of 3,045 km and 3,436 km, respectively, from S and D.

3.5 Defendability *in the wild*

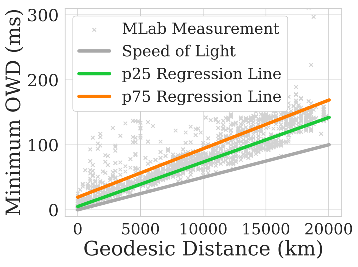
While our observations under ideal conditions are promising, defendability may differ in the real world because: (1) the actual propagation delay may be larger than the speed-of-light RTT due to longer physical paths, and (2) RTT measurements may not capture the actual propagation delay due to congestion or poor channel conditions (in wireless networks). In this subsection, we evaluate defendability in realistic conditions using two production datasets.

Datasets. Our datasets are outlined below:

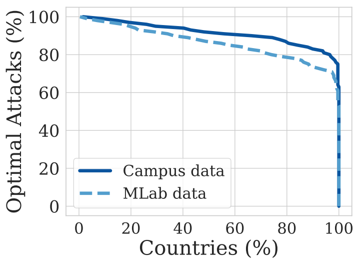
1. ***Campus dataset:*** We collect traffic from 7.5 M TCP flows on our campus over 12h on a weekday in May 2022, and compute RTTs by matching data packets with ACKs [14].
2. ***MLab dataset:*** We collect minRTTs of 4.3 M TCP flows from NDT7-based measurements over 5 days in Dec. 2024 [25].



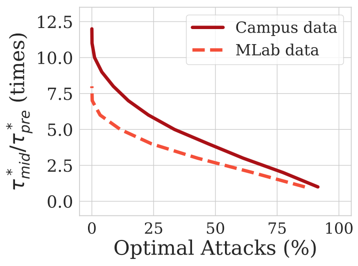
(a) Campus measurements over 12 hours in May ‘22.



(b) Google MLab measurements over 5 days in Dec. ‘24.



(c) % countries vs. % defendable optimal attacks.



(d) % attacks vs. ratio of mid-attack to pre-attack RTT.

Figure 5 Defendability against optimal attacks based on real measurements: We estimate (using linear regression) the $p25$ and $p75$ OWD for each 200 km distance bucket of our campus dataset and the Google MLab dataset, in (a) and (b) respectively. Using $p75$ OWD to estimate pre-attack and $p25$ to estimate mid-attack RTT, 85% countries can be defended against 85% attacks based on the campus dataset, and 85% against 78% based on MLab (Figure (c)). Figure (d) shows that the mid-attack RTT peaks at 12x pre-attack RTT in the campus dataset, and 7.5x in MLab.

For each flow in each dataset, we collect geolocations of the source and destination. We discard flows whose minRTT indicates shorter distances than those permitted by their reported geolocations, which is physically impossible. Finally, we group remaining measurements by source-destination /24 prefixes, because (1) a /24 prefix is the smallest unit on which a BGP hijack can be launched, and (2) aggregating by prefix improves the chance of measuring true minRTT (see §5.2 for a more detailed discussion on the advantage of aggregation by prefix).

Estimating propagation delay from distance. To quantify defendability in realistic settings, we estimate realworld propagation delay between any two hosts from their great-circle distance d. This real-world propagation delay may vary across host pairs separated by the same distance d depending on: (1) Geolocation: Some regions in the world have denser network connectivity than others, (2) Routing policies: Some providers make shorter paths available than others, (3) Consistent queues: Some paths experience consistent queuing delay due to deep buffers and consistent traffic, etc. Furthermore, even if the true propagation delay is same, we may measure different minRTTs due to transient congestion. It is infeasible to collect reliable minRTTs from all possible attack locations in all 258 countries. Instead, we apply the following method to both our campus dataset (215 countries) and Google MLab (234 countries) to capture variability:

1. ***Bin distances:*** Divide all distances up to 20,075 km (Earths diameter) into 200 km bins (≈1 ms at c_f).
2. ***Assign prefixes to bins:*** For each sourcedestination prefix pair, compute its greatcircle

distance, assign it to the appropriate bin, and record its minOWD (minRTT/2).

3. ***Percentile computation:*** Within each bin, compute the pth percentile of these minOWDs for $p = 1, \ldots, 100$.
4. ***Regression fitting:*** Fit one linear regression per percentile across all bins (e.g., a p=1 line through every bins 1percentile).

Figures 5a and 5b plot the p=25 and p=75 regression lines for the campus and MLab datasets, respectively. The campus data shows a narrower inter-quartile range—likely because the location of one end is fixed (on campus) and the entire variability is due to the remote host—whereas for MLab, the servers and clients are in different locations. With these delay estimates, we proceed to evaluate defendability against optimal attacks under realistic conditions.

Estimating pre-attack and mid-attack minRTTs. The variability of minOWDs for the same distance poses a challenge: if our detection is unlucky, it could measure a higher percentile minRTT before the attack and a lower percentile during the attack, causing the deviation in minRTT to be much lower than speed-of-light deviation. To model such a scenario, we use the upper quartile (75^{th} percentile) minOWDs to estimate pre-attack minRTTs, and the lower quartile (25^{th} percentile) minOWDs to estimate mid-attack minRTTs. Then, we evaluate defendability using the same metrics as before.

Observations. Using the condition $\tau^*_{deviation} \geq 5\,\text{ms}$, the overall attack coverage is 91% on the campus dataset and 86% on MLab. Figure 5c shows that in the campus data, 100% countries can be defended against 63% attacks, 75% against 89%, and 2% against 100%. In MLab, 100% countries can be defended against 53% attacks, 75% against 80%, and <1% against 100%. Real-world defendability is therefore less than in the ideal case—especially in MLab, where minRTT variability is higher. Figure 5d plots coverage versus the mid-/pre-attack minRTT ratio. The ratio ranges from 112x (5290 ms) for campus; 17.5x (5250 ms) for MLab.

3.6 Takeaways

Our analysis shows that propagationdelay measurements offer a highly effective signal to defend against crosscountry interception, primarily because diverted paths almost always incur substantially greater delays than pre-attack paths. Although realworld variability degrades detection coverage compared to ideal conditions, our focus on worstcase (optimal) attacks means these results are conservative—actual detection performance will often exceed our current estimates. We believe our findings are sufficiently strong to justify designing an interception-detection based solely on propagation delay. At the same time, a range of factors influences how accurately any given attack can be detected: the victim countrys size and distance from potential adversaries; the exact geolocations and separation of victim and peer hosts; the true lengths of the pre-attack and mid-attack network paths; transient or persistent congestion along those paths; and the precision of our measurement and aggregation techniques. With these insights in mind, in the next section, we present HiDe—a scalable, always-on, data-plane system for detecting and mitigating interception attacks.

4 HiDe: Overview

HiDe is a system to detect and mitigate BGP interception attacks. HiDe runs entirely on a programmable switch and uses real-time minRTT measurements for attack detection. In this section, we present the key insights that drive HiDe.

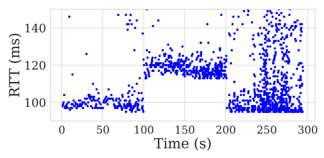


Figure 6 *Abrupt* and *significant* rise and fall in RTT due to interception attack launched (ethically) at 100th second and withdrawn at 200th.

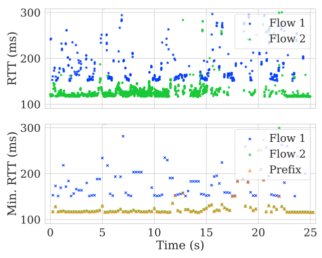


Figure 7 Top: Flow 1 (blue) with noisy RTTs and flow 2 (green) with stable RTTs. Bottom: Aggregating by prefix stabilizes minRTTs (orange).

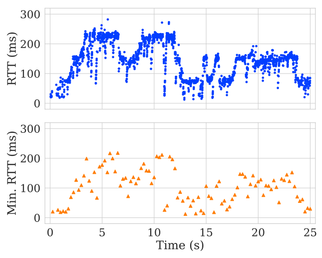


Figure 8 Prefix with noisy RTTs (top) that produce noisy minRTTs (bottom) despite windowing. Such prefixes are less defendable.

Converting noisy RTT into a reliable detection signal. During a hijack, all traffic to a victim prefix must traverse the longer path via the attacker, so no RTT sample can be shorter than the minimum propagation delay via the attacker. HiDe exploits this by passively collecting RTT samples for every TCP dataACK pair at the network border (thereby avoiding noise from the internal network), aggregating samples per prefix, and tracking the minRTT per time window. By monitoring these minRTTs, HiDe detects hijacks as *sudden, sustained spikes* in delay. For example (Figure 6), hijacking test traffic ethically from a Stockholm client via Amsterdam causes the minRTT to jump by about 20 ms at attack start and to fall back when the hijack ends. A changepoint detection algorithm can reliably identify such shifts.

Prioritize guaranteed protection over broad coverage. We posit that operators would prefer a system that reliably defends a welldefined subset of prefixes rather than a besteffort approach that *covers* everything but floods them with false alarms. Section 5.3 shows how HiDe restricts its scope to prefixes it can protect with high confidence. When false positives do occur, HiDe (optionally, determined by the operator) continues measuring RTT and automatically rolls back its mitigation if the spike proves transient.

Optimize for commodity hardware. HiDe stores only per-prefix state—the running minRTT and count of RTT samples per time window—instead of expensive per-flow or per-packet state. It employs a lightweight, two-window, threshold-based changepoint detector that is hardware-amenable. On the Intel Tofino2, HiDe uses native primitives (*mirror* and *packetgen*) to generate packet replicas for RTT measurement, and occasionally—optionally—for false-positive correction and user alerting, while forwarding all other traffic at line rate with zero additional latency.

5 HiDe: Methodology

5.1 Computing location-based lower bound

Translating user input into geographic locations. The user provides HiDe with the IP prefix of the home network and the threat regions they want to protect their data from, based on policy decisions or anticipation of threats. The threat regions are either names of countries or enclosed polygons of geographic coordinates. HiDe also obtains from its data plane the destination prefixes observed by it. The user can, *optionally*, set threat regions *per destination prefix*. Eventually, the control plane converts all the information into triplets of geolocation information: {source_coordinates, destination_coordinates, threat_coordinates_list} using public geolocation services (*IPinfo*, *MaxMind*) and public geographic datasets (*Natural Earth Admin-0*) [16, 24, 40].

Computing lower bound of mid-attack RTT. For each location triplet, we first identify the *optimal attack*, i.e., the attacker's location in the threat region that minimizes mid-attack round-trip distance. Note that this distance can be much higher than in the optimal attacks in §3 because there, source and destination were always on the victim country's border whereas here, they are almost always inland. Next, we compute the minimum possible mid-attack RTT for this optimal attack (τ^*_{mid}), based on the speed of light. We designate this *lower bound* RTT as the *absolute threshold* of our changepoint detector: HiDe flags an attack whenever the observed minRTT reaches τ^*_{mid}, guaranteeing *zero false negatives* (see §5.6 for more on false positives). While we could choose a less conservative bound—e.g., the 25th-percentile estimated latency in 200 km buckets (§3.5)—we opt for the most conservative threshold to *guarantee* protection, at the expense of coverage, as is our design goal (§4).

5.2 Reducing noise in the RTT signal

Aggregating by prefix to reduce impact of noisy flows. BGP attacks target prefixes, with a /24 prefix being the smallest possible target. All flows to an attacked prefix experience the same change in propagation delay, but noisy RTTs in individual flows can obscure this change. We aggregate RTT samples by prefix before computing the minimum RTT per window (discussed next), as at least one flow per window is likely to produce a sample representative of the true propagation delay. Figure 7 demonstrates this for a US-based destination prefix with one noisy and one stable flow. Also, prefix-level aggregation reduces switch memory requirements from per-flow to per-prefix, which is significant.

Windowing to discard short-term fluctuations. We divide streams of per-prefix RTT samples into non-overlapping time windows of a fixed size (i.e., *tumbling* windows) and compute the minRTT in each window. This filters out short-term RTT spikes due to benign confounding factors like queuing delay from short-lived congestion, end-host processing delays, and TCP oddities like *delayed ACKs* [35]. We select a sub-second (e.g., 100 ms) time window—while minRTT can be measured more reliably with longer time windows, it would delay mitigation allowing an attacker more time to complete their attack. Finally, tracking minRTTs in *non-overlapping* tumbling windows requires only per-flow state, as opposed to *overlapping* sliding windows, making it more suitable for a switch implementation.

5.3 Vulnerable and defendable prefixes

Identifying vulnerable prefixes. BGP interception attacks primarily target prefixes that host sensitive services—government sites, banking portals, cryptocurrency nodes, and the like—because such websites handle sensitive data from users around the world. The attacker places itself between the user and the server—intercepting *valuable* data. HiDe prioritizes these vulnerable server prefixes for protection. By default, it excludes prefixes used exclusively by WiFi or cellular access networks—since they rarely host critical services—unless the operator explicitly includes them.

Identifying defendable prefixes. Some destination prefixes have RTTs that are *consistently* noisy or high even under benign conditions, making it nearly impossible to detect interception attacks on them from certain threat regions without excessive false positives. For example, Figure 8 shows a prefix where the minRTT often exceeds 100 ms, making it impractical to defend against a threat region that causes a small deviation in comparison. Further analysis of such prefixes reveals that often, they tend to be associated with client-side access networks, such as cellular or WiFi, which HiDe does not defend by default anyway. Concretely, during a *profiling* phase independent of the detection phase, we monitor the *max. of min. RTTs* in tumbling time windows for each destination prefix. Later, we defend a prefix against a threat region only if $\tau^*_{mid} - max(RTT_{min}) > \lambda$, where λ is called the *surge threshold.* λ defines the min. increase in RTT_{min} required between two consecutive windows to flag an attack, and can be set to a constant (e.g., 10 ms) or a fraction of $max(RTT_{min})$ (e.g., 10%). A lower λ provides broader coverage but increases susceptibility to false positives, and vice-versa. Users can adjust λ based on their desired trade-offs. We evaluate the false positive rate for different values of λ in §8.

5.4 Hardware-amenable changepoint detection

We implement changepoint detection directly in switch hardware by combining the techniques described so far in an approach called the *two-window algorithm.* This involves tracking the per-prefix min. RTT (RTT^i_{min}) for each tumbling window i. Once the i^{th} window completes ($i > 0$), we compare RTT^{i-1}_{min} and RTT^i_{min} and mark the prefix as *attacked* if both the following *surge* conditions are met:

1. $RTT^{i-1}_{min} < \tau^*_{mid}$ and $RTT^i_{min} > \tau^*_{mid}$: The minRTT crosses the absolute threshold between two consecutive windows.
2. $RTT^i_{min} - RTT^{i-1}_{min} > \lambda$: The minimum RTT surges by at least the surge threshold in consecutive windows.

5.5 Adaptive windowing and speed of detection

Since HiDe relies on the change in minRTT across windows of RTT samples, its detection speed depends on the following factors:

1. ***Traffic-related factor:*** The RTT sample rate per second (R), which is a function of the number of ACK packets per second, and thus of the data rate (packets per second) and TCP's cumulative ACKing behavior.
2. ***HiDes detection parameters:*** In particular, the size of each time window (T), and the minimum number of RTT samples in a *valid* (defined below) window (S).

Only time windows containing at least S samples are considered valid and used for detection; windows with fewer samples do not necessarily benefit from min-filtering and could lead to false positives. In our dataset, we observe that S=5 works well and we use this value in our experiments (§8). The sample rate R is determined by the traffic, while T and S are set based on information learned during the profiling phase. In an ideal scenario, RTT samples arrive at a roughly uniform rate and $R = (1/T) \cdot S = S/T$, meaning that samples divide neatly into time windows that are each just large enough to contain S samples. In this case, the *time-to-detection* is approximately $2T = 2(S/R)$. For example, if the data rate is 1 MB/s and the cumulative ACK frequency is one ACK per five data packets, the expected time-to-detection is about 71 ms.

In the wild, the rate at which RTT samples are generated is not necessarily uniform, so the time-to-detection can be slightly higher and may vary across profiled prefixes, but it generally remains within a small fraction of a second for active flows. To mitigate cases where S is effectively too high for the prevailing traffic rate (e.g., when the data rate during an attack is significantly lower than during profiling), we implement an *adaptive time window.* Under this strategy, for each window, HiDe starts with size T, but if S samples are not

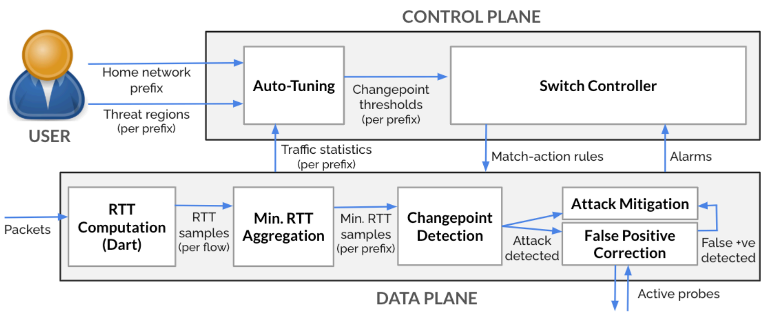


Figure 9 HiDe consists of a software control plane and a hardware data plane. The control plane auto-tunes per-prefix parameters for changepoint detection based on user inputs and traffic statistics from the data plane, and installs those parameters as match-action rules on the data plane. The data plane computes RTT samples, aggregates them by prefix, computes minRTT per window, and performs changepoint detection. Upon detecting an attack, the data plane blocks the corresponding prefix and triggers active probing to correct false positives.

accumulated by the end of the window, it increases the window to $2T$, $3T$, and so on until at least S samples are reached. This ensures that HiDe always has valid windows to work with, and that the time-to-detection is as low as possible while still remaining accurate.

5.6 Minimizing impact of false positives

Despite reducing false positives, our detection algorithm is not foolproof and may occasionally generate them. To minimize their impact and to eliminate the need for human intervention, HiDe optionally employs an automatic *false positive correction* mechanism. When an attack is detected, HiDe blocks the affected prefix and simultaneously initiates active probing by sending ICMP echo packets to the most recently active IP address in the prefix at each time window. It monitors the corresponding RTT, and if the RTT falls below τ^*_{mid}, the prefix is unblocked, and detection resumes, minimizing disruption to regular operations. To limit probe traffic, HiDe reduces the probe rate to one per minute after five minutes of attempts. These parameters are user-adjustable for flexibility.

6 HiDe: System

Figure 9 presents an overview of HiDe's end-to-end workflow. HiDe comprises a control plane, implemented in software on a server, and a data plane, operating in high-speed hardware on a programmable switch. HiDe is deployed at the edge of a production network, and can observe all or most of its traffic depending on the network topology (§6.4).

6.1 Control plane

User input. The user configures the control plane by providing the network prefix of their home network (source prefix) and specifying the threat regions (optionally, per prefix).

Auto-tuning. The auto-tuning component translates the user inputs and destination prefixes read from the data plane into corresponding geographic coordinates and computes the τ^*_{mid}. It also retrieves traffic statistics (specifically, $min(RTT_{min})$ and $max(RTT_{min})$) from the data plane for each prefix. Combining this information, the component identifies which prefixes can be effectively protected and generates changepoint detection parameters for those prefixes, which it then sends to the switch controller. For prefixes that cannot be protected, it provides the user with a summary listing each prefix and their corresponding RTT_{min} statistics.

Switch controller. The switch controller translates the received parameters into corresponding match-action rules and installs them on the data plane. It also receives attack alarms from the data plane and (optionally) notifies the user.

6.2 Data plane

RTT computation. We leverage Dart, an existing system, to generate accurate RTT measurements per flow from all the traffic observed by the switch. Dart achieves this at scale, handling a large number of flows without missing any RTT samples by efficiently managing switch resources [35].

Min. RTT aggregation. The next component aggregates RTT samples per destination prefix, breaks them down into time windows, and calculates the minimum RTT per window. Additionally, it computes traffic statistics like $min(RTT_{min}$ and $max(RTT_{min})$ per prefix to share with the control plane.

Changepoint. The data plane performs changepoint detection using our two-window algorithm, minRTTs per window per prefix, and parameters installed by the control plane.

Attack mitigation. When an attack is detected, the data plane rate limits or blocks the corresponding prefix—depending on operator configuration—and raises an alarm.

False positive correction. Optionally, if the operator seeks more confidence in the detection result, and HiDe is not being used in conjunction with control-plane-based approaches, it can craft and send active probes periodically to determine whether the detection was a false positive. If so, it unblocks the prefix.

6.3 Hardware prototype

We implement our prototype in $P4_{16}$, and deploy it on the Intel Tofino2 high-speed programmable switch, which supports up to 12.8 Tbps of traffic at line rate [2, 10]. Our prototype does not depend on any specific features available on the Tofino, and can be ported readily to other programmable packet-processing hardware including other switches (e.g., Juniper Trio) and SmartNICs (e.g., Nvidia BlueField3).

Switch control plane. The switch control plane installs per-prefix match-action rules specifying the window size (W), absolute threshold (τ^*_{mid}), and surge threshold (λ). If enabled, it listens on the CPU port for packets from the data plane containing information about either attack detections or non-coverage, and notifies the user. Additionally, it configures the Tofino's *packet generator* to send active probes during the *false positive correction* phase.

RTT computation. We leverage DART for continuous and accurate per-flow RTT computation [35], and utilize *packet mirroring*, a native feature that replicates packets, to enhance its functionality. First, the original packet is forwarded without added latency, with the mirrored copy used for RTT computation. Second, RTT samples generated by DART are passed to HiDe-specific data-plane components.

Per-prefix state. We maintain a *prefix table* in register memory to support changepoint detection. The table uses a *prefix signature*, derived by hashing the first 24 bits of the external IP, as the key. The stored values include the prefix's start timestamp, timestamp of

Resource Type	Compute RTT [35]	Track MinRTT	Detect Change	Mitigate Attack
Stages	7	2	4	3
TCAM	2.9%	0.0%	1.1%	0.0%
SRAM	4.5%	4.0%	2.4%	3.6%
Instructions	3.6%	2.4%	1.0%	1.1%
Hash Units	35.8%	12.5%	2.8%	5.6%
Input Crossbars	10.1%	3.0%	1.6%	1.9%

Table 1 Hardware resource usage of the Tofino2-based prototype, divided by functional component.

most recent RTT, start timestamp of current window, number of RTT samples in current window, minRTTs for the current and previous windows, attack status, $max(RTT_{min})$, and $min(RTT_{min})$. The table accommodates up to 65,536 active prefixes, significantly exceeding the peak observed in our 12h campus trace (approx. 5K assuming a 5-second timeout), minimizing hash collisions. For collisions, we use *cuckoo hashing* [28]: the new prefix replaces the old one, which is loaded into memory, checked for timeout, and recirculated to a new index using a different hash seed if still valid. Each insertion allows up to 3 recirculations.

Changepoint detection. When an RTT sample is generated for a prefix, HiDe checks the status of the corresponding time window. If the window is not full, it updates the most recent timestamp, increments the RTT count, and replaces the current minimum RTT if the new sample is smaller. If the window is full, the current minimum RTT replaces the previous window's minimum, and $max(RTT_{min})$ and $min(RTT_{min})$ are updated. If the window is valid (i.e., it has enough samples), HiDe evaluates the surge conditions and, if those are satisfied, starts the mitigation process by blocking all non-ICMP packets from/to the prefix by adding it to a *block table*.

Active probing. Simultaneously, we start crafting and sending ICMP echo packets to the latest IP seen from the prefix and listening to responses to monitor its RTT. If a false positive is detected, the prefix is removed from the block table.

Resource usage. We analyze the resource usage of our prototype by function and find that its low resource consumption leaves ample resources for other concurrent switch functions (Table 1).

6.4 Deployment

HiDe is deployed at the edge of a production network (Figure 10), protecting clients within the network by monitoring the external leg of RTTs (HiDe to external hosts) rather than the internal leg (HiDe to internal hosts) [35]. We denote connections with clients inside the defended network as *Client-In-Server-Out* (CISO) and connections with servers inside the network as *Server-In-Client-Out* (SICO). We observe that the primary source of noise in RTTs is typically the access link near the client. For CISO connections, which are associated with the vulnerable prefixes (described earlier in §5.3), the access link is part of the internal leg and does not affect the monitored RTTs, resulting in less noise. In contrast, SICO connections experience higher noise levels, as the access link is external. In our campus data, we apply a TCP port number-based heuristic to distinguish CISO connections from SICO connections: if the port number used by the campus-internal host is < 1024 and the one used by the external host is > 1024, we consider it a SICO connection, and vice-versa.

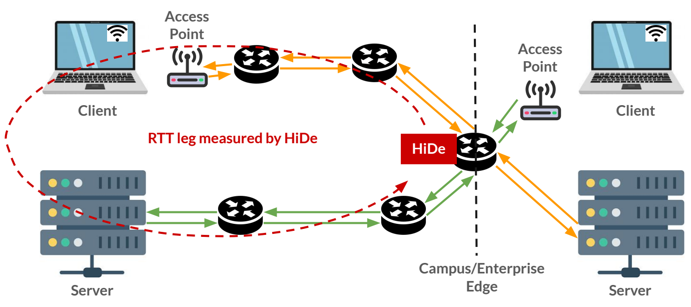


Figure 10 HiDe—deployed at the edge of a production network—defends servers (associated with vulnerable prefixes) and clients (optionally since associated with less vulnerable prefixes) inside it by measuring the *external leg* of RTT from itself to external hosts.

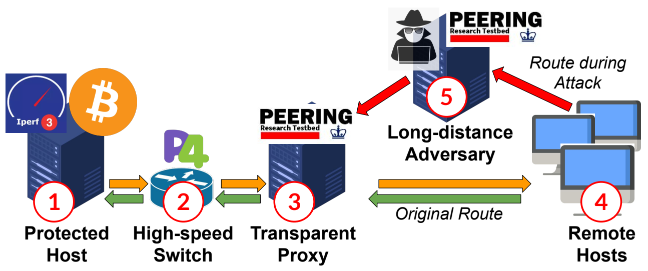


Figure 11 Experimental setup for our live experiments. The orange and green arrows indicate the original *protected host* to *remote hosts* route and back, respectively. The return path (green) is intercepted by the *long-distance adversary*—the diverted portion of the route is shown with red arrows.

7 Experimental Setup

We outline our experimental setup: first, to demonstrate live detection of ethically launched interception attacks on controlled *iperf* traffic; second, to collect a 12-hour campus trace highlighting HiDe's low false positive rate and minimal impact on regular operations.

7.1 Passive capture of campus traffic

Data collection. As outlined before, we captured 12 hours of production traffic—covering 1 pm to 1 am local time to include global working hours—at the edge of our US-based campus network using a TAP device near the gateway router. Packets—only TCP headers, anonymized at source in a prefix-preserved manner (discussed in Appendix A)—from selected subnets were mirrored and recorded on a collection server with *tcpdump*.

Dataset overview The dataset comprises 1.1 TB of trace data representing 5.32 TB of packet bytes, encompassing 19 billion packets, 7.5 million flows, and 238 million RTT samples. It includes 12K unique internal IPs and 324K unique external IPs, distributed across 183K external prefixes, 23.2K of which are based in the US.

7.2 Live experiments

Figure 11 shows the experimental setup for our live experiments involving active iperf3 traffic.

Deploying HiDe to protect experimental traffic. We set up our experiment using three key components: (1) a host on our campus running iperf3, (2) a high-speed programmable switch on campus where HiDe is deployed to monitor traffic, and (3) a transparent TCP proxy on an Amazon AWS instance. The proxy, which doubles as a *PEERING* node, advertises a /24 prefix allocated to our experiment. PEERING provides distributed ASes for controlled, real BGP announcements [34]. We use one IP address from the /24 pool, applying *iptables* rules on components 1 and 3 to masquerade it as the application's IP. This setup enables HiDe to monitor all experimental traffic while allowing external adversaries to ethically launch BGP interception attacks on the /24 prefix.

Setting up remote hosts. We deploy AWS instances in geographically diverse locations, including the US east and west coasts, Europe, and Asia. The *iperf3* server runs on the protected host across multiple ports, while clients on the distributed AWS instances connect to the PEERING IP of the transparent proxy, which forwards traffic to the server. The remote hosts are indicated as component 4.

Launching ethical routing attacks. The final step in our setup is launching ethical BGP interception attacks on the PEERING IP. We designate the PEERING node in Amsterdam as the attacker (component 5) and implement a stealthy interception attack using the technique by Birge-Lee et al., which employs BGP communities to control the blast radius of the attack [9]. The attacker advertises the same /24 prefix as the transparent proxy (an *equally-specific* attack), redirecting traffic from nearby nodes to Amsterdam. The attacker then forwards the intercepted traffic to the transparent proxy, leaving both sender and receiver unaware of the attack.

8 Evaluation

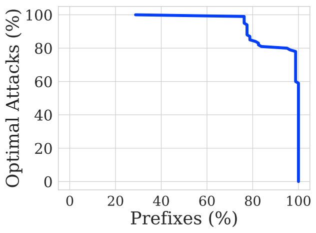
In this section, we present our evaluation results. In the first part (Section 8.1), we run simulations of HiDe on our campus dataset and report coverage, false positive rate, and downtime due to false positives. In the second part (Section 8.2), we present results from live experiments where the HiDe prototype defends the protected host (in Figure 11) when a subset of connections are impacted by an ethically conducted interception attack.

8.1 Trace-based evaluation

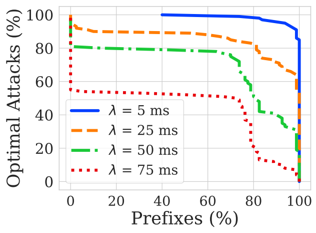
We evaluate HiDe using three metrics: (1) False positive rate or FPR (measures reliability/usability/practicality), (2) Coverage (measures the trade-off with low FNR and FPR), and (3) Downtime (measures impact of false positives on regular operation). We perform this evaluation using a faithful simulation of HiDe written in Python on real latency data obtained from production traffic on our campus (Section 7.1). We operate with the goal of protecting US-based prefixes from long-distance interception attacks from the mainlands of other countries in our dataset. For each prefix, we divide into two equal parts the total time during which the prefix was active: the first half is used for *profiling* while the second half is used for *detection.*

Vulnerable prefixes. In accordance with HiDe's coverage strategy, we only defend vulnerable (CISO) prefixes, i.e., external prefixes associated with a server. As described earlier, we identify such prefixes using a TCP port number-based heuristic. 16.8K US-based external prefixes match this condition in our campus dataset.

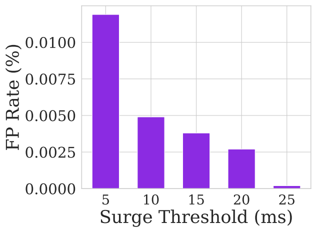
Profiled prefixes. From external server prefixes, we further select those that were active



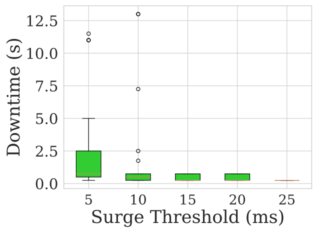
(a) Coverage (lower bound)



(b) Coverage (defendability)



(c) False positive rate



(d) Downtime

Figure 12 Faithful simulation on campus data illustrates that HiDe can defend most prefixes from optimal attacks from most countries, incurs low false positives (<=0.012%) and low downtime due to false positives (median<=0.75s).

for at least 10 minutes out of 12 hours (so we profile on at least 5 mins of data). We determine this by dividing the 12-hour period into buckets of 1 min, and checking which prefixes generated an RTT sample in at least 10 such buckets. 6K US-based external server prefixes are retained after this step. In a real network, the operator could profile prefixes for as long as needed to ensure it covers typical variation of RTT during benign operation—without any excess overhead since the profiling happens directly in the switch. For the following analysis, we make the assumption that the campus data captured during the 12-hour period did not experience any long-distance interception attacks (i.e., no true positives were present), so if HiDe detects a prefix it must be a false positive. We report our results based on optimal attacks from all 257 non-US threat countries.

Coverage impact of theoretical lower bound. Some US-based prefixes are not defendable against certain threat regions because during the profiling phase, they exhibit a $min(RTT_{min})$ larger than the corresponding τ^*_{mid} (i.e., measured delay without diversion is always higher than the lower bound with diversion). This could be because the threat region is geographically too close or because the network is always congested. Figure 12a shows that, based on this condition, HiDe can cover 99% prefixes against 78% attacks and 75% prefixes against 99% attacks. The covered prefix-threat country pairs are considered in subsequent experiments.

Coverage impact of defendability analysis. For different values of the surge threshold (λ), Figure 12b shows HiDe's coverage based on defendability—i.e., whether the mid-attack lower bound RTT clears the pre-attack $max(RTT_{min})$ by at least λ ms. At $\lambda = 5$ ms, 25 ms, 50 ms, and 75 ms respectively, HiDe can cover 99% prefixes against 91%, 64%, 31%, and 7% attacks, respectively. This illustrates the trade-off between surge threshold and coverage.

False positive rate. By focusing on defendable prefixes, we achieve a false positive rate of approximately 0.012% at worst, as shown in Figure 12c. For higher surge thresholds (i.e.,

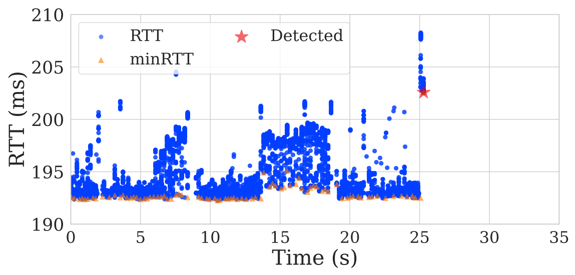


Figure 13 HiDe (immediately) detects interception attacks ethically launched by us on iperf3 traffic.

30 ms), it drops to zero.

Downtime due to false positives. We estimate the likely downtime from a false positive by measuring how long (in multiples of time window size) it takes for the minRTT to return to normal for a falsely detected prefix. Since our (optional) active probing sends one probe per window, we expect similar results in reality. The median downtime is only 0.75 seconds.

8.2 Live Interception Attack Detection

In this section, we evaluate HiDe on a *live* interception attack to demonstrate the *fidelity* of our hardware prototype, i.e., that HiDes in-switch RTT monitoring and changepoint detection behave as intended under real network conditions. This case study is not intended to establish that HiDe detects interception attacks in a fully general setting or to characterize coverage across all geographies. Instead, we focus on a long-distance scenario in which the adversary is sufficiently far away to induce a large RTT increase, yielding a clear delay signal for validating end-to-end correctness. Sections 3 and 8.1 analyze HiDes coverage more broadly, including settings where the detour is geographically short and the RTT change may be too small to detect reliably.

Setup. We run the iperf3 server on our campus and the transparent proxy in Ireland, who forwards all traffic it receives to our campus via our prototype. The iperf3 clients are in Virginia (2 flows from the same prefix), Ohio, and Mumbai. The prefix in Virginia is hijacked from Amsterdam, causing Ireland to send traffic to Amsterdam instead of Virginia. Amsterdam then forwards the traffic to Virginia. The traffic takes the following round-trip route before the attack: *Virginia (via PEERING infrastructure) to Ireland to our campus to Ireland to Virginia (via PEERING infrastructure)* and the following one during the attack: Virginia (via PEERING infrastructure) to Ireland to our campus to Ireland to Amsterdam (via. PEERING infrastructure) to Virginia (via PEERING). Due to limitations of where we can deploy a Tofino switch on live traffic and a lack of diversity in the PEERING topology, we are restricted to this complex setup. The attack takes effect at 25 seconds, as can be observed from the abrupt rise in RTT (blue dots) in Figure 13.

Interception attack detection. Using multiple runs of traceroute, we estimate the lower bound of RTT as approx. 190.5 ms before attack and 199 ms during attack (absolute threshold). We set the window size to 250 ms and the surge threshold to 5 ms. Based on the minimum RTTs (orange triangles), we detect the attack in 500 ms (red star).

9 Related Work

Prior work on defending against BGP hijacks broadly falls into *proactive* approaches that aim to prevent hijacks from succeeding in the first place, and *reactive* approaches that detect and mitigate attacks after they begin. Reactive defenses can be further divided into control-plane-based, data-plane-based (using active or passive measurements), and hybrid designs that combine both. In this paragraph, we summarize these lines of work and highlight how HiDe fits into, and complements, this landscape.

9.1 Proactive approaches

A long line of prior work seeks to proactively secure interdomain routing by preventing invalid routes from being accepted in the first place. One direction is cryptographic upgrades to BGP, such as BGPsec, which provide path validation but require widespread deployment and operational support to be effective [5]. Another direction is origin authentication and filtering via RPKI and route-origin validation, which can prevent a subset of attacks (e.g., forged-origin hijacks) but does not address many other attack classes (e.g., interception or path manipulation) [12]. Clean-slate architectures such as SCION provide stronger security properties by design, but similarly face the barrier of incremental deployability at Internet scale [29]. These proactive mechanisms can be highly effective where deployed, but their protection is limited by adoption and the subset of attacks they cover. HiDe is complementary: it is a reactive defense that does not rely on broad Internet-wide adoption, and is therefore readily deployable in today's ecosystem.

9.2 Reactive control-plane-based detection and mitigation

Reactive defenses commonly analyze BGP updates from public monitors (e.g., RIS and RouteViews), private commercially-operated monitors, and/or an operator's own routers to detect suspicious announcements and trigger mitigations such as prefix deaggregation (announcing more-specific routes) or filter-based actions. Systems such as ARTEMIS and DFOH illustrate this model, emphasizing fast identification of suspicious announcements and automated mitigation [22, 36]. A key limitation of this class is *visibility*: detection quality depends on what monitors can see and when they see it. Prior work shows that attackers can craft targeted hijacks that restrict propagation toward monitors and limit global visibility, including attacks that manipulate BGP attributes to achieve *stealthy, targeted* interception [7, 26, 41]. A concrete example is the manipulation of BGP communities to surgically steer traffic while limiting visibility at monitors [9].

Even when attacks are eventually detected, detection and mitigation can still be *slow*: updates may take time to reach monitors, for detectors to accumulate sufficient evidence, and for mitigation to kick in, increasing the exposure window during which damage can occur. In fact, meaningful harm can occur within seconds of a hijack taking effect: early-stage website fingerprinting based on statistical properties can succeed using only the first few packets (20 in one study, 100 in another), bursts (3 in one study) or a small fraction (22% in one study) of a page load [39, 17, 37]. Attackers can also mirror and store diverted traffic for offline analysis and potentially cause further harm by, for example, trying to break encryption of payloads. Real incidents likewise show that interception can be monetized immediately; for example, in the 2018 hijack involving Amazons DNS infrastructure, diverted cryptocurrency traffic enabled attackers to collect credentials and later steal cryptocurrency [30]. These results motivate complementary signals—such as RTT in the data plane (leveraged by HiDe)—that remain observable at the victim even when control-plane visibility is incomplete or slow.

9.3 Reactive data-plane-based detection using active measurements.

An alternative is to detect hijacks using *active probing* in the data plane (e.g., ping, traceroute, nmap) to infer unexpected path changes. iSPY is a representative approach that uses active probing to identify suspicious routing behavior [42]. Defenses based on active measurements can provide direct evidence of path changes and can sometimes localize anomalies, but they come with practical constraints: they require responsive targets, introduce probe traffic overhead that grows with scope, and often require sustained probing to maintain accuracy. These factors complicate always-on deployment at large scale and can limit coverage for unresponsive or rate-limited destinations. HiDe instead relies primarily on *passive* monitoring of RTT, avoiding probing overhead in the common case; it (optionally) uses active probing only as a targeted follow-up to correct potential false positives during mitigation.

9.4 Reactive data-plane-based detection using passive measurements

A smaller body of work leverages passively observed performance signals to flag routing anomalies. For example, Hiran et al. use crowd-sourced RTT measurements to detect routing anomalies [21], though their method only addresses detection, not mitigation. Oscilloscope [11] offers advanced hijack detection but relies on emulated data, suffers from high false-positive and false-negative rates, and lacks a hardware implementation, limiting its applicability and scalability. HiDe differs from these approaches in their operational goals and deployability: HiDe is a *real-time* defense for long-distance interception attacks, and is designed for *always-on, line-rate* operation on programmable switches.

9.5 Hybrid approaches combining control-plane and data-plane signals

Hybrid defenses combine BGP-derived signals with data-plane observations to improve detection confidence and reduce false positives. Argus and HEAP exemplify this approach by correlating control-plane anomalies with data-plane signals to strengthen detection [33, 38]. In general, hybrid methods inherit benefits and limitations from both sides: they can be more robust than either signal alone, but still face challenges with visibility at monitors, and measurement overhead (therefore, scalability) of active probing. HiDe is compatible with the hybrid model *and* strengthens it significantly by providing a fast data-plane signal that does not incur probing overhead and does not require responsive destination IPs.

10 Conclusion

We present HiDe, a system to detect and mitigate long-distance BGP interception attacks—where an adversary in another country diverts traffic through its own infrastructure to eavesdrop before forwarding it to the victim. By leveraging RTT measurements that attackers cannot conceal, HiDe delivers high-accuracy defense at line rate on a Tbps-scale programmable switch. Our analysis of worst-case attacks across 258 countries confirms its effectiveness, and we validate HiDe's fidelity and effectiveness through experiments with anonymized campus traces and ethically conducted real-world hijacks, achieving robust mitigation with low false-positive rates.

References

1 Aftab Siddiqui. Public DNS in Taiwan the latest victim to BGP hijack. https://manrs.org/2019/05/public-dns-in-taiwan-the-latest-victim-to-bgp-hijack/.

2 Anurag Agrawal and Changhoon Kim. Intel tofino2–a 12.9 tbps p4-programmable ethernet switch. In *2020 IEEE Hot Chips 32 Symposium (HCS)*, pages 1–32. IEEE Computer Society, 2020.

3 Andree Toonk. How Hacking Team Helped Italian Special Operations Group with Routing Hijack. https://bgpmon.net/how-hacking-team-helped-italian-special-operations-group-with-bgp-routing-hijack/.

4 Axel Arnbak and Sharon Goldberg. Loopholes for circumventing the constitution: Unrestrained bulk surveillance on americans by collecting network traffic abroad. *Michigan Telecommunications and Technology Law Review*, January 2015.

5 Rob Austein, Steven Bellovin, Russ Housley, Stephen Kent, Warren Kumari, Doug Montgomery, Chris Morrow, Sandy Murphy, Keyur Patel, John Scudder, Samuel Weiler, Matthew Lepinski, and Kotikalapudi Sriram. BGPsec Protocol Specification. RFC 8205, IETF, 2017.

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

7 Henry Birge-Lee, Maria Apostolaki, and Jennifer Rexford. Global bgp attacks that evade route monitoring. In *International Conference on Passive and Active Network Measurement*, pages 335–357. Springer, 2025.

8 Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. Bamboozling certificate authorities with {BGP}. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 833–849, 2018.

9 Henry Birge-Lee, Liang Wang, Jennifer Rexford, and Prateek Mittal. Sico: Surgical interception attacks by manipulating BGP communities. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 431–448, 2019.

10 Mihai Budiu and Chris Dodd. The p416 programming language. *ACM SIGOPS Operating Systems Review*, 51(1):5–14, 2017.

11 Tobias Bühler, Alexandros Milolidakis, Romain Jacob, Marco Chiesa, Stefano Vissicchio, and Laurent Vanbever. Oscilloscope: Detecting BGP Hijacks in the Data Plane. *arXiv:2301.12843*, 2023.

12 R. Bush and R. Austein. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810, IETF, 2013.

13 Celer bridge incident analysis. https://www.coinbase.com/blog/celer-bridge-incident-analysis.

14 Xiaoqi Chen, Hyojoon Kim, Javed M Aman, Willie Chang, Mack Lee, and Jennifer Rexford. Measuring TCP round-trip time in the data plane. In *ACM SIGCOMM Workshop on Secure Programmable Network Infrastructure*, pages 35–41, 2020.

15 Catalin Cimpanu. Klayswap crypto users lose funds after bgp hijack. https://therecord.media/klayswap-crypto-users-lose-funds-after-bgp-hijack/.

16 Natural Earth Data. Natural earth data-free vector and raster map data. *http://www. naturalearthdata. com (accessed 10.12.2024)*, 2011.

17 Xinhao Deng, Qi Li, and Ke Xu. Robust and reliable early-stage website fingerprinting attacks via spatial-temporal distribution analysis. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1997–2011, 2024.

18 Sharon Goldberg. Why is it taking so long to secure internet routing? *Communications of the ACM*, 57(10):56–63, 2014.

19 Dan Goodin. Russian-controlled telecom hijacks financial services internet traffic. https://arstechnica.com/information-technology/2017/04/russian-controlled-telecom-hijacks-financial-services-internet-traffic/.

20 Dan Goodin. Strange snafu misroutes domestic us internet traffic through china telecom. *Ars Technica*, 6, 2018.

21 Rahul Hiran, Niklas Carlsson, and Nahid Shahmehri. Crowd-based detection of routing anomalies on the internet. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 388–396. IEEE, 2015.

22 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.

23 Dave Levin, Youndo Lee, Luke Valenta, Zhihao Li, Victoria Lai, Cristian Lumezanu, Neil Spring, and Bobby Bhattacharjee. Alibi routing. In *ACM SIGCOMM*, pages 611–624, August 2015.

24 LLC MaxMind. Geoip, 2006.

25 Measurement Lab. The M-Lab NDT data set. `https://measurementlab.net/tests/ndt`, (2024-12-01 – 2024-12-05). Bigquery table `measurement-lab.ndt.download`.

26 Alexandros Milolidakis, Tobias Bühler, Kunyu Wang, Marco Chiesa, Laurent Vanbever, and Stefano Vissicchio. On the effectiveness of bgp hijackers that evade public route collectors. *IEEE Access*, 11:31092–31124, 2023.

27 Shaun Nichols. AWS DNS network hijack turns MyEtherWallet into ThievesEtherWallet. URL: `https://www.theregister.com/2018/04/24/myetherwallet_dns_hijack/`.

28 Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

29 Adrian Perrig, Pawel Szalachowski, Raphael M. Reischuk, and Laurent Chuat. *SCION: A Secure Internet Architecture*. Springer Verlag, 2017.

30 Louis Poinsignon. Bgp leaks and cryptocurrencies. *The Cloudflare Blog, April*, 2018.

31 RIPE NCC. YouTube Hijacking: A RIPE NCC RIS case study. `https://www.ripe.net/publications/news/youtube-hijacking-a-ripe-ncc-ris-case-study/`.

32 Andrei Robachevsky. routing security getting better, but no reason to rest!, 2019.

33 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.

34 Brandon Schlinker, Todd Arnold, Italo Cunha, and Ethan Katz-Bassett. PEERING: Virtualizing BGP at the Edge for Research. In *ACM CoNEXT*, Orlando, FL, December 2019.

35 Satadal Sengupta, Hyojoon Kim, and Jennifer Rexford. Continuous in-network round-trip time monitoring. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 473–485, 2022.

36 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.

37 Meng Shen, Yiting Liu, Siqi Chen, Liehuang Zhu, and Yuchao Zhang. Webpage fingerprinting using only packet length information. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

38 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.

39 Jean-Pierre Smith, Prateek Mittal, and Adrian Perrig. Website fingerprinting in the age of quic. *Proceedings on Privacy Enhancing Technologies*, 2021(2):48–69, 2021. URL: `https://doi.org/10.2478/popets-2021-0017`.

40 IP Info Team. Ip info, 2017.

41 Dan York. Bgp hijacking in iceland and belarus shows increased need for bgp security. `https://www.internetsociety.org/blog/2014/02/bgp-hijacking-in-iceland-and-belarus-shows-increased-need-for-bgp-security/`, Feb 2014.

42 Zheng Zhang, Ying Zhang, Y Charlie Hu, Z Morley Mao, and Randy Bush. ispy: Detecting ip prefix hijacking on my own. In *ACM SIGCOMM*, pages 327–338, 2008.

A Ethics

This research study was reviewed and approved by our Institutional Review Board (IRB). All packettrace data come from our university network and were anonymized at the point of collection by network engineers who are expressly authorized to handle private data. Anonymization followed the exact procedures laid down by the IRB—anonymizing all IP and MAC addresses, and stripping all payloads. Researchers never had access to any raw or deanonymized data. To validate HiDe s detection and mitigation capabilities in a live Internet environment, we performed two controlled BGP hijacks using prefixes assigned to us by the PEERING testbed [34]. We temporarily announced these prefixes from our own hosts under testbed guidelines, ensuring no impact on any external networks or clients. All BGP announcements and withdrawals adhered to PEERINGs guidelines, and only our own test prefixes were affected.