ImpROV: Measurement and Practical Mitigation of Collateral Damage in RPKI Route Origin Validation

Weitong Li
Virginia Tech

Yuze Li
Virginia Tech

Taejoong Chung
Virginia Tech

Abstract

The Resource Public Key Infrastructure (RPKI) enhances Internet routing security. RPKI are effective only when routers employ them to validate and filter invalid BGP announcements, a process known as Route Origin Validation (ROV). However, the partial deployment of ROV has led to the phenomenon of *collateral damage*, where even ROV-enabled ASes can inadvertently direct traffic to incorrect origins if subsequent hops fail to perform proper validation.

In this paper, we conduct the first comprehensive study to measure the extent of collateral damage in the real world. Our analysis reveals that a staggering 85.6% of RPKI-invalid announcements are vulnerable to collateral damage attacks and 34% of ROV-enabled ASes are still susceptible to collateral damage attacks. To address this critical issue, we introduce ImpROV, which detects and avoids next hops that are likely to cause collateral damage for a specific RPKI-invalid prefix; our approach operates without affecting other IP address spaces on the data plane that are not impacted by this collateral damage.

Our extensive evaluations show that ImpROV can reduce the hijack success ratio for most ASes that deployed ROV, while only introduce less than 3% and 4% of Memory and CPU overhead.

1 Introduction

The Border Gateway Protocol (BGP) plays a crucial role in the fabric of the Internet, guiding data through a complex maze of paths to reach its destination. Despite its foundational importance, BGP's initial design did not account for stringent security measures, notably lacking mechanisms to authenticate and authorize IP prefix announcements. This oversight left the protocol susceptible to various threats, including the notorious prefix hijacking [7, 9, 10, 48] and route leaks [7, 31, 45], posing significant risks to global Internet security and reliability.

In response to these vulnerabilities, the Internet Engineering Task Force (IETF) initiated the development of the Resource Public Key Infrastructure (RPKI) in April 2008 [28], under the Secure Inter-Domain Routing (SIDR) Working Group [44]. At the heart of RPKI's approach are two main components: Route Origin Authorizations (ROAs), digital certificates that attest to the authenticity of prefix announcements and Route Origin Validation (ROV), which performs the validation of incoming BGP announcements against the verified ROAs to ensure that the origin AS is authorized to announce the IP prefix. The implementation of ROA and ROV has seen rapid acceleration, with 40% of the total IPv4 address space now verifiable through ROAs [27, 38] and 18% ASes implementing ROV to eliminate invalid announcements [30].

Despite these advancements, ROV is not without its limitations. Specifically, ROV lacks path validation capabilities, meaning that even ASes implementing ROV could inadvertently direct traffic to an incorrect origin if the subsequent hop fails to perform validation, leading to what is termed as *collateral damage* [22]. In other words, even if an AS that performs ROV forwards traffic with the intention of reaching the legitimate origin, there is still a risk that the packet will be misdirected to the wrong destination if the next hop does not perform ROV. This limitation highlights a critical challenge: the difficulty in achieving comprehensive visibility into the routing path, making it challenging to (1) assess the extent of collateral damage and (2) devise strategies for mitigation.

While past studies have examined collateral damage through simulations [22] and traceroutes using a small number of vantage points [30], little is known about how collateral damage has adversely impacted routing security. Furthermore, there is a notable gap in the literature regarding practically deployable strategies to mitigate this collateral damage. In this paper, we make the following contribution:

- We conduct active measurements to assess collateral damage in RPKI deployments in real-world scenarios by announcing carefully chosen RPKI-invalid prefixes alongside RPKI-valid prefixes under our control.then perform data-plane measurements using traceroutes. Additionally, we perform extensive simulations to analyze how collateral damage might have affected real hijacking attacks thus far.

- We introduce ImpROV, a system that detects and avoids next hops likely to cause collateral damage without affecting other IP address spaces on the data plane that are not impacted by this collateral damage. Furthermore, we implement ImpROV on two popular BGP platforms, GoBGP and BIRD, and conduct comprehensive evaluations of its effectiveness and performance.

Our findings underscore the necessity of a lightweight, deployable ROV add-on for routers to mitigate collateral damage. We will make all our code for measurement and ImpROV implementation, along with descriptions of how we utilize public datasets, available to the research community at

```
https://improv.netsecurelab.org
```

for network operators, administrators, and researchers to benefit from our work.

2 Background

2.1 BGP

Internet routers build their routing tables using the Border Gateway Protocol (BGP), which enables them to exchange routing information and determine the most efficient paths for data transmission. BGP speakers announce routes to IP prefixes, specifying the sequence of Autonomous Systems (ASes) that data packets must traverse to reach their destination. For instance, a BGP route might look like this:

```
86.38.220.0/24, AS_PATH: AS3320 AS9121 AS48678
```

In this example, AS 48678 is the originator of the route for the IP prefix 86.38.220.0/24. Neighboring routers receive this route, incorporate it into their routing tables, and propagate it to their own neighbors based on their routing policies. When multiple announcements for the same IP prefix are received, the BGP route selection process determines the optimal path based on factors such as cost-efficiency and path length. During packet forwarding, routers use the most specific prefix match in their routing table to ensure accurate delivery.

However, the original BGP protocol lacks security features, making interdomain routing susceptible to various security threats. One such threat is prefix hijacking, where an attacker illegitimately announces an IP prefix to intercept traffic intended for that prefix. Another threat is sub-prefix hijacking, in which an attacker announces a more specific IP prefix than the legitimate one, exploiting routers' preference for the most specific prefix. These attacks have occurred frequently in practice, leading to significant consequences for the rightful owners of the affected IP prefixes [6,7,12]. For example, more than 1,400 sub-prefix hijacking attacks were detected during the first month of the Russia-Ukraine war [29].

2.2 RPKI

The Resource Public Key Infrastructure (RPKI) is a security framework designed to prevent IP prefix hijacking and sub-prefix hijacking attacks in BGP. RPKI provides a cryptographically verifiable method for mapping IP prefixes to their legitimate origin ASes. To ensure the effectiveness of RPKI, network resource owners must (1) register RPKI objects to protect their IP prefixes, and (2) network operators must validate BGP announcements using these objects to filter out RPKI-invalid routes.

2.2.1 Registering IP prefixes with ROA

Network resource owners can authorize their IP prefixes by creating two essential objects:

(a) a CA certificate, which associates Internet Number Resources (INRs) (e.g., ASNs or IP prefixes) with a public key,

(b) a Route Origin Authorization (ROA), which authorizes an AS to announce specific IP prefixes; this object is signed by the CA certificate.

These objects must be published in public RPKI repositories managed by the five Regional Internet Registries (RIRs), each maintaining its own trust anchor; each of them has a separate hierarchy starting at its own *trust anchor* and certificate [23, 28], similar to root certificates in other PKIs such as root stores in web PKI.

2.2.2 Validating BGP announcements against ROA

To validate BGP announcements against ROAs, RPKI validation software, known as Relying Party (RP) software, such as Routinator [41], fetches RPKI objects from the repositories and performs cryptographic validation. The software produces a list of validated tuples (ASN, ROA prefix, prefix length) called Validated ROA Payloads (VRPs). These VRPs are then sent to the AS's routers using the RP protocol, allowing them to perform Route Origin Validation (ROV) on incoming BGP announcements [32].

When an RPKI-validating router receives a BGP announcement, it uses the set of VRPs to validate the announcement. The router first checks if the BGP announcement is *covered* by a VRP, which occurs when the IP prefix address and the VRP IP prefix address are identical for all bits specified by the VRP IP prefix length.

If a covering VRP is found, the router then determines if the announcement exactly matches the VRP based on three criteria:

(a) The VRP IP prefix covers the announced IP prefix.

(b) The VRP AS matches the announced AS.

(c) The prefix length in the VRP is greater than or equal to the announced prefix.

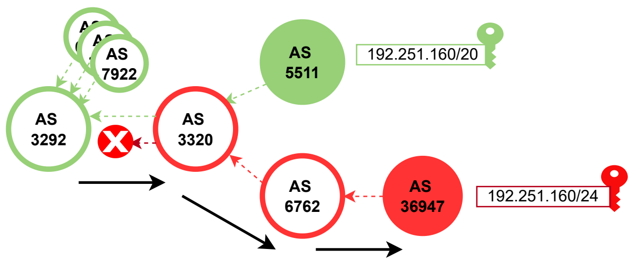

Figure 1: A real-world example of collateral damage occurred on April 4th, 2022, due to more specific RPKI-invalid announcements (red dotted lines). AS 3292 filtered this RPKI-invalid prefix (/24); however, for the /20 route, AS 3320 was still chosen, which eventually forwards /24 traffic to the invalid origin (black solid lines).

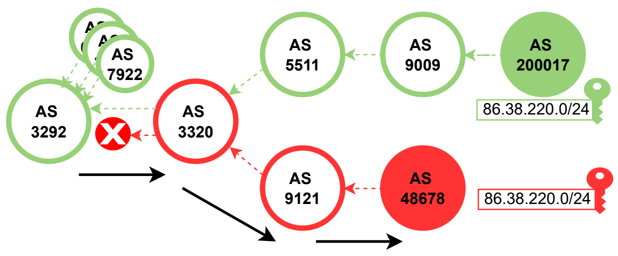

Figure 2: A real-world example of collateral damage occurred on April 4th, 2022: AS 3320 (non-ROV) forwarded traffic for a valid /20 prefix to an RPKI-invalid origin (AS 48678) based on an invalid /24 announcement (red dotted lines). AS 3292 (ROV) filtered the invalid /24 but still sent /20 traffic to AS 3320, resulting in misrouted traffic (black solid lines).

A BGP announcement is considered *valid* if it matches a VRP, *invalid* if it is covered but does not match any VRP, and *unknown* if it is not covered by any VRP. By rejecting RPKI-invalid announcements, RPKI-validating routers can prevent potential hijacking attempts and ensure the integrity of the routing system.

2.3 Collateral Damage

An AS that performs ROV does not accept RPKI-invalid prefixes. Since the AS also does not propagate the RPKI-invalid prefixes to their neighbors (e.g., customers), they do not learn the path towards the invalid origin, thus gaining collateral benefit.

However, less obviously, *even an AS that performs ROV can have its traffic susceptible to invalid origins*. This is because an AS typically can control where to forward the next hop, but not the entire path. As a result, even if an AS forwards traffic with the intention of reaching the correct origin, the presence of next hops that do not implement ROV introduces the risk of packets being forwarded to an invalid destination. We define this situation as *collateral damage*: packets are forwarded to an invalid origin even though the originating AS has deployed ROV. In this section, we explain two types of collateral damage:

RPKI-invalid prefixes with more specific length: This can happen when the RPKI-invalid announcement has a more specific prefix than the competing original prefix, causing the next hop to choose the path towards the more specific prefix. Figure 1 shows a real-world example that we observed on April 4th, 2022 using traceroute results from RIPE Atlas probes [37]. AS 36947 announces an RPKI-invalid prefix, 192.251.160.0/24, which is more specific than the RPKI-valid prefix (192.251.160.0/20) originated by AS 5511. However, when we perform a traceroute from the RIPE Atlas node in AS 3292 (TDC/AC), which is known to perform ROV, to an IP address within the /24 prefix, we notice that the route traverses towards the invalid origin; this is because AS 3320 (Deutsche Telekom) did not perform ROV. As a result, due to the longest-prefix-match routing, the more specific route was preferred, and the traffic was delivered to AS 36947, which is the invalid origin.

RPKI-invalid prefixes with the same prefix length: Collateral damage can also occur when the RPKI-invalid announcement has *the same prefix length as the valid announcement* but originates from an invalid origin. The next hop that does not perform ROV may choose the path towards the invalid origin if it is considered a shorter path or depending on other local preferences such as the AS relationship or the presence of a customer-provider agreement.

Figure 2 shows a real-world example observed on April 4th, 2022. AS 48678 announces an RPKI-invalid prefix that shares the exact same prefix (86.38.22.0/24) as the valid announcement. After AS 3292 forwards the announcement to AS 3320, we find that AS 3320 chooses AS 9121 as the next hop, which causes packets to be forwarded to the invalid origin even though AS 3292 enforces ROV.

A part of the reason that it seems challenging is that ASes typically do not have any control over the next hop; thus, the current ROV is designed simply filtering the invalid announcements without leveraging them to update their routing decision, even though they can *infer* what the next hop would do. In this paper, we propose ImpROV, which aims to identify and prioritize paths that do not propagate RPKI-invalid routes, which will be presented in §5.

3 Related works

Measurement of ROV deployment: Since the introduction of RPKI, numerous studies have investigated not only its deployment and misconfigurations of ROAs [15,26,46], but also focused on the deployment of RPKI validation (i.e., ROV) [16,25,30,36]. Some studies [22] have used simulations to understand the protection offered by ROV. However, these simulations did not consider the collateral damage caused by sub-prefix hijacking, nor did they incorporate real-world ROV deployment status in their models. Previous works also focused on ROV deployment [25, 36] by announcing invalid prefixes; then, they used traceroutes to infer whether ASes on the path filter invalid prefixes and estimate ROV status. However, none have designed real-world experiments to measure *collateral damage*. Such experiments require announcing RPKI-invalid prefixes along with RPKI-valid prefixes that *cover the invalid ones, crucially from different origins*.

Collateral damage in ROV: Through simulation, Gilad et al. [22] showed a possible scenario when collateral damage happens; however, since the ROV deployment when the paper was published in 2017 was very low, they could not analyze the actual collateral damage with real-world ROV deployment status. Recent works have shown real-world examples of collateral damage; Li et al. [30] showed a couple of real-world collateral damage using traceroute, and Du et al. [19] focused on collateral damage for a couple of top-tier ISPs. While these works prove that collateral damage does exist with the partial deployment of ROV nowadays, they only focus on one or several causes of collateral damage and do not further discuss mitigation strategies.

Recently, there have been proposals to mitigate the collateral damage or even validating the forwarding path. For example, Morillo et al. [33] proposed ROV++, which *penalizes* the next hop who forwards RPKI-invalid prefixes, which shares a similar goal with our paper. However, it *penalizes the next hop* not only for the received RPKI-invalid prefix but also for any less-specific prefixes that have not been hijacked (as detailed in §A.3). This approach might cause unexpected consequences by preventing *benign prefixes* (those unaffected by the subprefix hijack) from being forwarded to the next hop, even when the next hop is not vulnerable to the specific hijacked subprefix. For instance, we find that 66.0% of RPKI-invalid prefixes unintentionally create such benign prefixes (encompassing 84 times more IPv4 addresses than the actual hijacked IP addresses) when deploying ROV++.

Autonomous System Provider Authorizations (ASPA) in RPKI [5] was also recently proposed to let network operators create cryptographic certificates for all their peers and upstreams to validate their path; however, it has not been widely deployed yet [43].

4 Collateral Damage in the Wild

Obviously, the collateral damage happens when the RPKI-invalid announcement has been made. We first understand how many prefixes are potentially impacted; to this end, we collect BGP update datasets from all vantage points of the publicly available BGP collectors of RouteViews [40] from January 1st, 2023 to March 21st, 2024, and validate the announcements against ROAs.

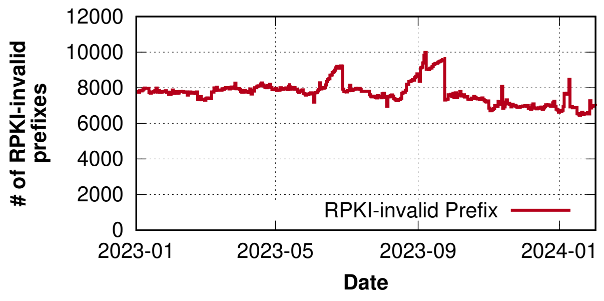

Figure 3: The number of RPKI-invalid announcement; in our latest snapshot, we observed 6.6K RPKI-invalid prefixes.

4.1 RPKI-invalid Prefixes

We now validate the announcements using an RPKI validation tool, Ziggy [35]; Figure 3 shows the number of RPKI-invalid prefixes announced during our measurement period. Our analysis reveals that, on average, 7,934 unique invalid prefixes were announced each day during the measurement period; in our latest snapshot, we find that 6,609 unique prefixes, covering *3.2 M IPv4 addresses, are RPKI invalid.* Interestingly, we notice two spikes in the number of RPKI invalid announcements during our measurement period. These spikes can be attributed to AS 39891 (Saudi Telecom) and AS 35913 (DediPath), which made 646 and 921 invalid announcements, respectively, during each of these two periods.

4.2 RPKI-invalid Prefixes Causing Collateral Damage

In this section, we focus on the potential RPKI-invalid announcements that may lead to collateral damage attacks. These RPKI-invalid prefixes can be classified into three distinct categories, each with its own implications for collateral damage:

(a) Exact match: There exist both RPKI-valid and RPKI-invalid announcements for the same prefix. Even if an ROV router filters out the RPKI-invalid announcement, if the next hop does not perform ROV, it may still choose the path towards the invalid announcer based on its local routing policy, potentially leading to collateral damage.

(b) Covered by RPKI-valid: RPKI-invalid announcements are covered by an RPKI-valid announcement with a less specific prefix. Due to the longest-prefix-match routing principle, a non-ROV next hop will likely prefer the path towards the invalid origin, as it advertises a more specific prefix, which result in collateral damage.

(c) No RPKI-valid coverage: In some cases, there are no RPKI-valid announcements that either cover or ex-

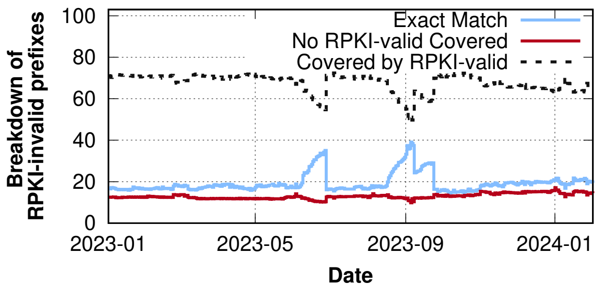

Figure 4: Ratio of the three categories of RPKI-invalid BGP announcements; note that the majority of RPKI-invalid announcements (85.6%), potentially cause collateral damage.

actly match the RPKI-invalid announcements; ROV-performing routers will drop these invalid announcements, making them unable to select a next hop; thus, it does not cause collateral damage.

Figure 4 shows the results. First, we immediately notice that *the vast majority of the RPKI-invalid announcements are covered by or exact matched with by RPKI-valid announcements, which are vulnerable to creating collateral damage attacks.* Even worse, in our latest snapshot, we find that 66.1% are more-specific RPKI announcements; this is particularly alarming as attackers can launch hijacking attacks with a high likelihood of success due to the longest-prefix-match routing principle. We also find that 19.5% are same-length RPKI-invalid announcements, still possibly causing collateral damage. In contrast, only 14.4% of RPKI-invalid announcements belong to the "no RPKI-valid covered" category, meaning they do not have any matching or covering RPKI-valid announcements; these exclusive RPKI-invalid announcements do not contribute to collateral damage, as ROV-enforcing routers will drop them, preventing their propagation.

The dominance of the "covered by RPKI-valid" category emphasizes the importance of addressing the collateral damage problem; attackers can easily exploit the longest-prefix-match routing principle by announcing more specific RPKI-invalid prefixes, even if some ASes along the path perform ROV.

4.3 ASes Vulnerable to Collateral Damage

In the previous section, we examined the prevalence of RPKI-invalid prefixes that can cause collateral damage in the wild. Now, we move on to understanding the *ASes in the real world that are potentially vulnerable to collateral damage attacks.* It has been known to be challenging to measure collateral damage in the real world [22]. This is mainly because (1) identifying the attackers who are announcing RPKI-invalid prefixes that might cause collateral damage is difficult, and (2) even if we do so, we cannot know who is actually vulnerable to this attack. One might argue that we can use an active measurement platform like RIPE Atlas and launch traceroutes to measure where they reach; however, since traceroutes do not reveal the AS information, it is infeasible to measure which AS is actually being reached, thus we have to rely on their AS path. To overcome these challenges and measure the impact of collateral damage, we use the PEERING testbed [42].

4.3.1 Methodology

We use the PEERING testbed to announce *both RPKI-invalid and RPKI-valid announcements* from two different ASNs under our control. However, to identify who is potentially vulnerable to collateral damage, *we announce those prefixes in a different order.*

Then, we use RIPE Atlas probes to launch traceroutes to assess the potential impact of collateral damage attacks. More specifically, we proceed our experiments as follows:

(a) We first register a valid ROA object, which authorizes AS 47065 (PEERING) to announce an IP prefix, `184.164.240.0/23`.
(b) We then announce an RPKI-invalid BGP announcement, `184.164.240.0/24`, from a different AS, AS 61574 (PEERING).
(c) For each RIPE Atlas probe, we run traceroute towards an IP address in `184.164.240.0/24` and measure its reachability. In this step, we only consider ASes that have at least 3 probes and remove those with inconsistent results across the probes to try to eliminate client-side issues such as multi-homing, which results in 6,672 RIPE Atlas probes that cover 3,048 ASes.
(d) As a next step, while announcing the RPKI-invalid BGP announcement from AS 61574, we now also announce `184.164.240.0/23` from AS 47065 (PEERING), which is the authorized AS, thus RPKI-valid.
(e) We repeat the same process as step 3. In this stage, if we observe an AS that was previously unable to reach the invalid AS but is now able to reach it, it indicates that some ASes in the middle of the path have chosen the more specific prefix, thus revealing a victim of collateral damage.

4.3.2 Experiment Results

We run traceroute for 6,672 RIPE Atlas probes and find that 28 (0.4%) probes in 12 (0.4%) ASes show inconsistent behavior in traceroute, so we remove them from the analysis. Finally, we consider the traceroute results from 6,660 (99.8%) RIPE Atlas probes for analysis that cover 3,036 ASes. Among them, we find that 902 (29.7%) of the 3,036 ASes were unable to reach the /24 RPKI-invalid prefix when only the invalid announcement (/24) was made. However, we find that *307 (34.0%) of these ASes are able to reach the invalid AS when we also announce the /23 RPKI-valid prefix from the authorized*

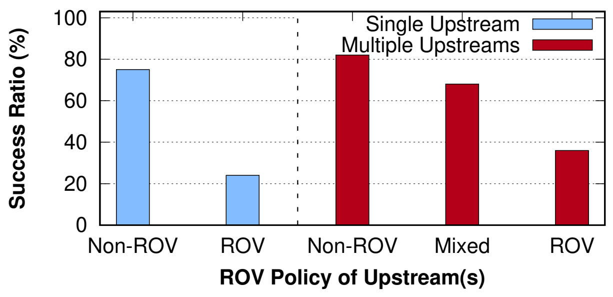

Figure 5: Likelihood of an AS being susceptible to collateral damage attacks based on the number of upstreams and their ROV policy. The chance of collateral damage decreases as more upstreams perform ROV. 'Mixed' indicates that only some upstreams perform ROV.

origin. This highlights the prevalence of collateral damage caused by upstream ASes not performing ROV, effectively undermining the security benefits of ROV deployment by downstream ASes.

Now, we take a closer look at how their upstreams impacts the susceptibility to collateral damage of a given AS; for example, the chance that an AS with ROV is susceptible to collateral damage attacks decreases as more of its upstreams perform ROV. To investigate this, we analyze the chance of being under collateral damage depending on (1) their number of upstreams and (2) whether they perform ROV or not. First, in order to estimate the number of upstreams for the ASes, we use CAIDA AS relationship datasets [17] to construct a topology, which covers all the ASes we measured and their 3,126 neighbor ASes; we find that 653 (72.4%) ASes have multiple upstreams or peers and 249 (27.6%) ASes have only one upstream or peers.

When it comes to identifying whether an AS performs ROV or not, we face a significant challenge: Identifying whether an AS performs ROV is known to be a challenging problem. Thus, as a proxy, we use RoVista datasets [30], which measures the ROV protection score of ASes—the ROV score has a higher correlation with ROV deployment. Since RoVista has the most coverage (around 32K), we classify an AS's upstreams as likely to perform ROV if they have 100% scores and as not performing ROV when their scores are 0.[1]

Among the 902 ASes, we find 792 (87.8%) ASes where we can identify all of their upstreams' ROV scores; 226 (28.5%) ASes that have a single upstream, and 566 (71.5%) ASes that have multiple upstreams. Next, we calculate the likelihood of an AS (out of the 902 ASes) being susceptible to collateral damage attacks depending on the number of their upstreams and their ROV policy. Figure 5 presents the results.

First of all, it demonstrates a clear relationship between the likelihood of an AS being susceptible to collateral damage attacks and the ROV of their upstreams; for example, when there is only one upstream, we find that 58 (75%) ASes are susceptible to collateral damage when their sole upstream does not perform ROV, while this number drops to 39 (24%) when their upstream performs ROV. *It is worth noting that collateral damage happens even when the primary upstream does ROV*; this is because some of the upstream providers on the path may not perform ROV, which underscores the importance of collaboration and coordination among ASes to ensure the widespread adoption of ROV.

[1]An AS that deploys ROV has a 100% ROV score. However, the reverse is not always true; an non-ROV AS can still have a 100% ROV score when all of their upstreams perform ROV.

4.4 Impact of Collateral Damage

Through experimentation with the PEERING testbed, we assess the real-world implications of collateral damage attacks and gauge the scope of potential victims. Nonetheless, our methodology exhibits constraints, notably the singular deployment of an RPKI-invalid IP prefix from a solitary AS, compounded by the restricted reach (i.e., potential victims) attributed to the deployment of RIPE Atlas probes. In this subsection, we delve into the frequency of BGP hijacking incidents facilitated by collateral damage.

4.4.1 RPKI-Covered Hijacking Incidents

We use BGPMon [11], a tool that monitors real-time BGP announcements from multiple data sources, including RouteViews and RIPE-RIS, to detect hijacking events—we documented 1,376 hijacking incidents from January 1st 2023 to September 18th 2023. These incident reports detail (1) the detection time of the hijack, (2) the compromised IP prefix, (3) the authorized AS, and (4) the perpetrating AS. This information enables us to find the RPKI coverage of the affected prefixes and to verify the authenticity of the BGP announcement from the originating AS by referencing RouteViews BGP announcements corresponding to the dates of each incident.

Our analysis reveals that 729 (53.0%) of these incidents were protected by RPKI. At first glance, this might suggest that ASes deploying ROV were not vulnerable to these hijacking attempts; however, as demonstrated earlier, even ASes enforcing ROV remain susceptible to collateral damage attacks. Accordingly, we now attempt to estimate the number of ASes at risk of such hijacking incidents.

By aligning the date of each incident with the corresponding RouteViews BGP table, we are able to identify the original BGP announcement. Subsequently, we categorize these hijacking attempts into two groups: (1) attacks employing a more specific prefix and (2) attacks utilizing a prefix of identical length. Our findings indicate that 291 (40.0%) of the attacks introduced a more specific prefix, whereas 438 (60.0%) employed a prefix of the same length as the legitimate one.

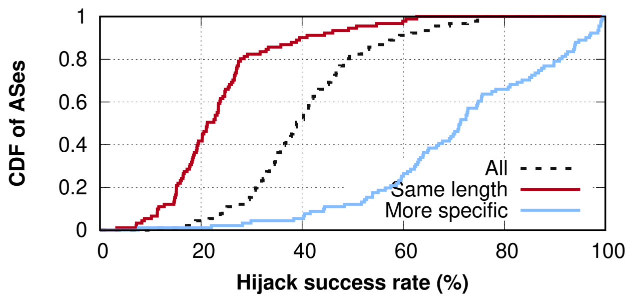

Figure 6: The CDF of hijack success ratio across all ROV ASes

4.4.2 Simulation settings

To measure the vulnerability of ASes to hijacking attacks, we perform simulations based on the BGP route computation methodology as outlined in [20–22, 33]. We start our simulation by employing the CAIDA AS relationships dataset to construct an AS-connectivity graph.[2] Subsequently, we adopt the path selection criteria as described in [22, 24], which stipulates 1) local preference, 2) shortest AS_Path first, 3) then randomly tie-breaking.

The primary aim of our study is to understand the impact of real-world hijacking incidents on ASes that have implemented ROV. To this end, we first classify ASes with a 100% ROV implementation rate, according to RoVista data, as ROV ASes, while designating the remainder as Non-ROV ASes by referring the RoVista's datasets captured on the same date of hijack incidents. Through this methodology, we identify 32,875 ASes within the topology graph, of which 4,730 were categorized as ROV ASes.

For each ROV AS, we assess whether the routing path leads to the invalid origin or the legitimate origin; ASes routed to the invalid origin are considered to have fallen victim to collateral damage and are labeled as victim ASes. The success ratio of hijacks is calculated by dividing the number of victim ASes by the total number of RPKI-validating ASes. Figure 6 illustrates the distribution of outcomes by attack type.

4.4.3 Simulation Results

Firstly, we can find that ROV ASes are still susceptible to these attacks. *A significant proportion of ROV-ASes (4,328, 91.5%, labeled as 'All') remain vulnerable to collateral damage attacks; for instance, more than 50% of these ASes could become victims of at least 535 attacks.* Focusing on the type of attack reveals that the hijacking success ratio notably increases *when attackers announce more specific prefixes*; for example, 87.0% of ASes are not protected by ROV against

[2]The CAIDA AS relationship datasets are released on a monthly basis; accordingly, we choose the most proximate snapshot to the occurrence date of each hijack for our simulations.

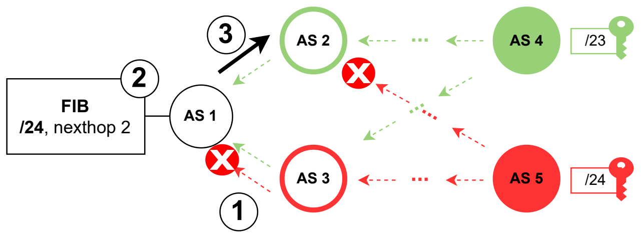

Figure 7: How ImpROV avoids collateral damage due to more specific RPKI-invaild prefix: when ImpROV receive an RPKI-invalid announcement ①, it adds a corresponding iRoute to the FIB ②, directing traffic towards a safer upstream ③ (black solid line) .

over half of the attacks involving more specific prefixes, while only 4.7% of ASes are vulnerable to 50% of the attacks with the same prefix length. This indicates that attackers can examine ROA objects first and intentionally announce smaller prefixes to usurp the prefix, leveraging the longest-prefix-match routing principle; thus, ROV alone is not sufficient to protect ASes from collateral damage attacks, particularly when attackers announce more specific prefixes.

5 ImpROV: System Design

We now introduce ImpROV, a lightweight extension on BGP implementation that can identify and mitigate the potential collateral damage. We design ImpROV based on the following goals:

- ImpROV should select safe next hops to forward traffic for IP spaces with potential collateral damage.
- ImpROV should not introduce any data-plane impact for IP spaces that are not under hijack.
- ImpROV need to be easily deployed with minimal performance downgrade and will not introduce additional attack surface of original RPKI.

5.1 Threat Model

ImpROV focuses on an attack model where an adversary, with access to or control over an AS, sends false BGP announcements to the neighbors of that AS. These announcements involve prefixes that are RPKI-covered but originate from IP prefixes not owned by the adversary. Such RPKI-invalid BGP announcements can result in traffic misdirection, interception, or blackholing, compromising data integrity and availability.

Although there are many different kinds of BGP hijacking, in ImpROV, we focus on BGP hijacking that 1) adversary uses an ASN that is not the legitimate owner of the IP prefixes, 2) the hijacking BGP announcement will cause multiple origin

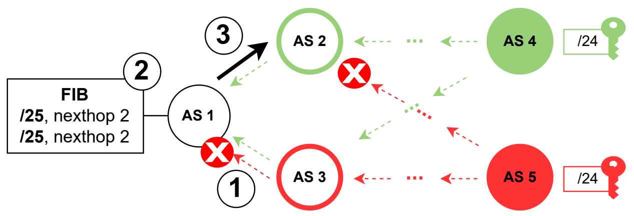

Figure 8: How ImpROV avoids collateral damage due to the same-length RPKI-invalid prefix: when ImpROV receives an RPKI-invalid announcement ①, it adds two more specific routes (one bit longer) to the FIB ②, directing traffic towards a safer upstream ③ (black solid line).

ASes (MOAS) conflicts. We do not consider AS path prepending attacks or route leaks that do not cause MOAS, as they are not within the original protection scope of ROV.

5.2 Basic Design

ImpROV is designed to *minimize* the chance of collateral damage by avoiding routes that could potentially cause collateral damage while minimizing modifications to existing router implementations and keeping performance intact. It is worth noting that we do not aim to achieve zero collateral damage, as this is almost infeasible. Collateral damage can happen on any AS along the path that is not performing ROV. Since we have no control over the path, the only mitigation we can do is to *avoid* the next hop that forwarded the RPKI-invalid route by adding a mitigation route, which we call *ImpROV route*, or iRoute, that we generate. For each RPKI-invalid announcement, we first examine our Routing Information Base (RIB) to determine if there are any covered RPKI-valid prefixes. These prefixes could be of the same length or less specific than the RPKI-invalid announcement. We give them a higher priority in the route selection process.

(a) RPKI-invalid announcement with more-specific prefix: When we find that the RPKI-invalid prefix is more specific than existing routes, we create a iRoute with the more specific length to a safer next hop. This allows the router to avoid the vulnerable next hop for the announced more-specific RPKI-invalid prefix. In Figure 7, for example, we create a /24 iRoute toward AS 2 to avoid sending traffic to AS 3 for the /24 prefix.

(b) RPKI-invalid announcement with the same-length prefix: When we find an exact match for the RPKI-invalid announcement prefix length in the RIB, we intentionally deaggregate the prefix to steer the traffic toward a safer next hop. For example, in Figure 8, upon receiving a /24 RPKI-invalid announcement, we add two /25 iRoutes that cover the entire /24 IP space towards the safer upstream.

5.3 Overall Architecture

Figure 9 shows the overall operation of ImpROV. For each RPKI-invalid announcement received, we perform an extra process to determine whether it causes collateral damage, find the next hop that has a lower chance of being susceptible to collateral damage, and update the Forwarding Information Base (FIB) to steer the traffic accordingly.

5.3.1 Identifying Collateral Damage

When receiving an RPKI-invalid announcement, we first check whether this invalid route can cause collateral damage; this happens when there are other existing routes that cover the announced prefix. By referring to the `adj-RIB-In`, we retrieve all routes that cover the RPKI-invalid route. If such routes exist, we move on to the next step. If not, it means that the announcement does not trigger collateral damage or is one that we cannot handle, so we stop here.

5.3.2 Finding the Safe Next Hop

Among the available alternative routes, we should identify the safest next hop. Ideally, network operators would manage a list of next hops protected by ROV to give them preference. However, it is challenging to measure the correct ROV status of their upstreams or peers. Even though there is some public data on ROV deployment, it is known that the status may change over time and not be consistent across all routes [30]. Thus, instead of managing a list, ImpROV manages the RPKI status of their neighbors by measuring whether they have propagated RPKI-invalid announcements or not, and simply prioritize those who have never forwarded these RPKI-invalid announcements. If there are still many available routes that have never propagated invalid routes after removing such candidates, we choose the next hop for the iRoute based on the original best route selection process, like `bgp_best_selection()` for BIRD [13].

5.3.3 Creating iRoutes

After choosing the next hop, we create the iRoutes. As described in §5.2, for RPKI-invalid prefixes with the same length prefix, we create two iRoutes that are more specific than the existing route by one bit. For RPKI-invalid prefixes that are more specific, we only need to create one iRoute toward the selected next hop. Since ImpROV mitigates collateral damage within the AS that deployed it, we do not need to propagate the iRoutes to other peers or downstream ASes. Thus, the route will not be installed in the `adj-RIB-out`; we provide our rationale for this decision in §5.5.

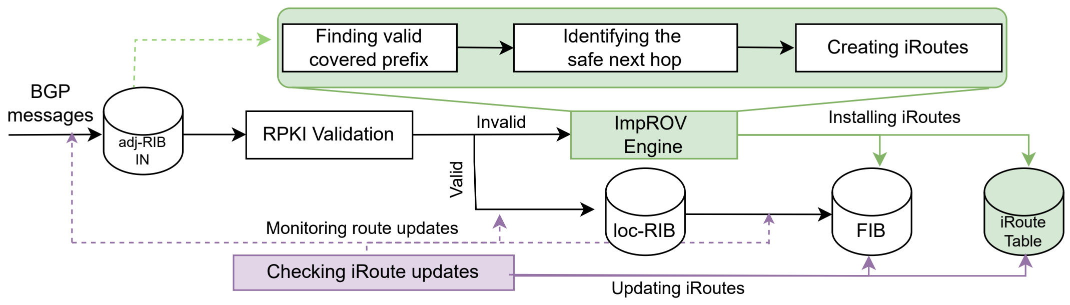

Figure 9: The overview of ImpROV. After a BGP message is stored in the `Adj-RIB-in` [47], RPKI validation happens. The validated entries will be forwarded to the `Loc-RIB` (Local RIB) [47] where the best path selection process takes place after applying the local filter (i.e., import filter). For RPKI-invalid entries, ImpROV evaluates whether the RPKI-invalid prefix causes collateral damage; if so, it creates the iRoutes and pushes them to the FIB. The process for updating iRoute is detailed in §5.4. It is worth noting that ImpROV does not involve the `adj-RIB-out` [47], which means that the iRoutes are not advertised to other neighbors (thus, omitted in the figure).

5.4 Updating iRoutes

After creating and installing iRoutes in the FIB, it is crucial to update them when the RPKI-invalid announcements stop or the valid route used to create iRoutes is withdrawn. Failing to do so can result in zombie routes in the FIB, which can harm performance and even cause blackholing. To address this issue, ImpROV includes additional logic to manage iRoutes accordingly. There are two main situations when these routes need to be updated or removed to ensure optimal routing and prevent stale entries: (1) when the RPKI-invalid prefix announcement is no longer advertised, or (2) when there are changes in the best route.

In such events, iRoute should be updated to prevent traffic from being forwarded to a suboptimal next hop or to avoid leaving stale routes in the FIB, which can harm router performance.

5.4.1 RPKI-Invalid Updates

There are two indicators that an RPKI-invalid prefix announcement has concluded: (a) the RPKI-invalid route is withdrawn or (b) the next hop of the RPKI-invalid route updates the route to a valid origin AS.

To handle these scenarios, ImpROV maintains an additional routing information table that stores iRoutes along with their corresponding invalid and valid routes and all next hops, which is called iRoute Table. This table can be implemented as two hash tables indexed by the invalid and valid prefixes used to create the iRoutes.

- When ImpROV receives a BGP withdrawal message, it checks the iRoute Table for a matching prefix and next hop; if found, ImpROV also withdraws the corresponding iRoute from the FIB and the iRoute Table.
- Similarly, when ImpROV receives a new valid or unknown route, it checks the iRoute Table for a match with the RPKI-invalid route's prefix and next hop. If there is a match, it indicates that the original invalid announcement from that next hop has been updated to a valid origin AS; otherwise, the RPKI-invalid announcement is considered over, thus ImpROV removes the iRoute from the FIB and the iRoute Table.

5.4.2 Best Route Updates

Even when RPKI-valid or unknown messages from other hops are received, iRoute changes may be required to refelect optimal routing.

- When ImpROV receives a new BGP update and chosen as the best route for a certain prefix, ImpROV also checks if the prefix matches any valid route in the iRoute Table. If matched, ImpROV updates the iRoute with the new best next hop.
- When the next hop of a iRoute withdraws the route, ImpROV must also update the iRoute to prevent traffic from being forwarded to a non-available next hop. Thus, when receiving a withdrawal message, ImpROV checks if the prefix matches any valid route in the iRoute Table and updates the iRoute accordingly; if no other RPKI valid or unknown route exists, ImpROV removes the iRoute from the FIB.

5.5 Propagation of iRoutes

Propagating the iRoutes to neighbors could potentially provide additional protection against RPKI-invalid prefix announcements, regardless of whether the neighbors deploy ROV or not; for example, iRoutes are more specific than the original valid announcements, neighbors receiving the iRoutes would always select them to forward traffic to the IP space covered by the RPKI-invalid announcement. However, in the current implementation, we do not propagate the iRoutes to neighbors. The primary reason is that forwarding iRoutes to downstream ASes or peers could lead to unintended consequences, as these routes might be further propagated to other networks. Rather, it might be possible to share iRoutes with neighbors in a more controlled manner, such as by using a specific BGP community [14] to signal that a route is a iRoute; this would allow neighboring ASes to make informed decisions about whether to accept and use these routes.

6 Evaluation

To understand the performance overhead of ImpROV, we implement it on two widely-used open-source BGP implementations, BIRD v.2.14 [13] and GoBGP v.3.19.0 [2].

6.1 Implementation

Creating iRoute: Three processes are involved when creating the iRoutes: (1) identifying the collateral damage, (2) finding the next hop for iRoute, and (3) installing the iRoute to FIB. One concern might be ImpROV would require all routes to be stored in `adj-RIB-In` with the RPKI validation results. Thus, if the router software drops the invalid routes from the `adj-RIB-In` upon finding that the route is RPKI invalid, then we would need to make more changes to the code and affect the original pipeline.

However, in practice, we find that BIRD and GoBGP already maintain the RPKI invalid routes and validation results in the `adj-RIB-In`, even though the BGP message turns out to be invalid. This is mainly due to two reasons; first, if the router does not keep the dropped RPKI-invalid routes, it would need to issue a route refresh when ROA changes, which can cause unnecessary performance degradation. Thus, RFC 9324 [8] recommends retaining at least the routes dropped due to ROV for a certain period in order to avoid frequent route refreshes. Second, in practice, some routers also keep the invalidated results to provide the option as an alternative path, de-prioritize them, or make other BGP attribute modifications, such as marking them with an RPKI-invalid community number to let other routers know the validation status when propagating them [1]. Thus, we re-use these codes to identify the collateral damage.

When choosing the next hop for the iRoute, we need to retrieve all routes that cover that invalid prefix; we also note that there are existing functions in the BGP software (e.g., `fib_get_chain()` in BIRD), which is to retrieve all covered routes for a certain prefix. Thus, we also do not introduce any other data structure or search function, since we believe the existing algorithms in the BGP software are already optimized.

Finally, we deploy the iRoute to FIB by calling an existing function (e.g., `fib_insert()` in BIRD).

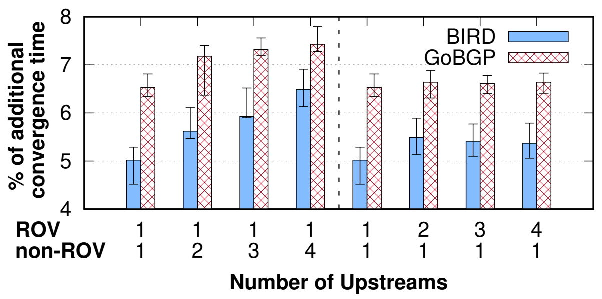

Figure 10: The percentage of additional convergence time with multiple ROV and non-ROV upstreams.

Updating iRoutes: We have introduced a new table called iRoute Table to store the iRoute that we installed. Typically, the BGP software manages a hash table combined with a tree structure for RIBs, where the hash table is used to search for the exact matched prefix while the tree structure is used to search for all covering prefixes and pick the most specific one.

In ImpROV, however, we only need to search for the exact matched prefix to update iRoute, thus we only introduced an extra hash table, which is more efficient. Implementing ImpROV in both GoBGP and BIRD induced a total of 365 and 283 additional lines of code, respectively.

6.2 Performance Evaluation

We evaluate the performance considering two perspectives: First, when a router establishes a new BGP session, typically with another router, BGP peer, or IXP, it begins exchanging routing tables. Typically, the entire BGP table exchange happens once a new BGP session is first established between two peers. Thus, we measure the time to complete the FIB and compare them with the original versions.

Second, after establishing a new BGP session, as new BGP messages arrive, ImpROV deploys and updates iRoutes; thus, we also measure the CPU and memory usage.

For all evaluations, ImpROV as well as BIRD and GoBGP run on an Intel® Xeon® Silver 4210R @ 2.40GHz with 187 GB of RAM, Linux Kernel 5.4.0-54, the upstreams BGP are running in another server with the exact same hardware, connected with a 100Gps Ethernet within the same physical location.

6.2.1 BGP Table Convergence Time

For evaluation, we use a simple network where one router has multiple upstreams, some of which perform ROV. The upstream routers send a full routing table taken at five different vantage points captured in RouteViews (i.e., `route-views[2-6].routeviews.org`) between January 1st, 2023 and February 1st, 2024. Since ImpROV introduces a new iRoute when it receives an RPKI-invalid route, we also control the number of upstreams that do not perform ROV. As the number of non-ROV routers increases, which send more RPKI-invalid routes, we expect that it may take more time to construct the full routing table. Starting from a simple topology with only one ROV-performing router and one non-ROV-performing router, we increment the number of non-ROV-performing routers to four, each sending a different routing table captured from a different vantage point in RouteViews. Considering that only 11.8% of ASes on the Internet have more than 5 neighbors [17], we decide to test the performance with 2 to 5 upstreams with different ROV policies. We measure the relative performance impact of running ImpROV compared to their native implementation in both BIRD and GoBGP. For each experiment, we run 10 times and obtain the daily average across 387 days. Figure 10 (left half) shows the results.

First, with only two upstreams, ImpROV introduces an extra 5.0% (BIRD) and 6.5% (GoBGP) of time for convergence when we use two upstreams with one non-ROV-performing router; the average time to finish is 47.2 and 103.4 seconds in the native implementation while it takes 49.6 and 110.0 seconds for ImpROV versions in GoBGP and BIRD implementation. As expected, as the number of non-ROV performing upstreams increases, we see more overhead. For example, we see that the overhead increases to 6.5% (BIRD) and 7.4% (GoBGP) as we introduce up to four unsafe upstreams. However, we believe the performance overhead of ImpROV still remains within acceptable bounds especially considering that exchanging the entire BGP table typically happens once a new BGP session is first established between two peers.

We also conduct a similar experiment, but now increasing the number of ROV upstreams (Figure 10, right half); this is to observe how ImpROV performance degradation happens even when RPKI-valid or RPKI-unknown announcements increase. We observe an interesting, non-linear relationship between the number of ROV upstreams and the overhead of ImpROV. As we increase the number of ROV upstreams, the additional performance overhead does not increase proportionally; for example, we find the overhead does not keep increasing as we increase the number of ROV upstreams when using BIRD, from 5.0% with 2 upstreams to 5.4% with 5 upstreams. We expect this behavior is driven by a distinct factor of ImpROV; since we add iRoutes when we receive RPKI-invalid announcements, more RPKI-valid and RPKI-unknown announcements cause negligible performance harm. Additionally, as the number of ROV upstreams increases, the percentage of RPKI-invalid announcements received by ImpROV decreases. This is an encouraging finding, as it suggests that as more ROV routers will be deployed, both the likelihood of being collaterally damaged and experiencing performance degradation decreases. These findings demonstrate the long-term viability of ImpROV as a collateral damage mitigation mechanism; its ability to handle increasing numbers of ROV upstreams without significant performance degradation, combined with the reduced likelihood of collateral damage as ROV adoption expands.

6.2.2 CPU and Memory Overhead

After the initial exchange of the BGP table, more subsequent BGP update messages come in, which involve the creation and management of iRoutes. Thus, these operations have to be efficient. To evaluate ImpROV's performance impact on BGP updates, we create a topology with our router connected to two upstreams: one non-ROV and one ROV-performing router. We then use *complete daily BGP updates* from RouteViews2 and RouteView3 collectors, forwarding them to ImpROV*while preserving original message timing*. This setup reflects real-world conditions. We run this measurement by using 1,233,692,735 BGP update messages from the period of January 1st to February 1st, 2024. This allows us to assess the performance of ImpROV with real-world BGP update messages and its impact on the router's resource utilization and processing efficiency.

Through the experiment, we find that the memory consumption is elevated by consuming 2.2% (BIRD) or 2.7% (GoBGP) more additional memory when we run our experiment throughout 32 days.[3] This is because ImpROV adds more iRoutes to FIB, which typically account for less than 1% [27] of the total RPKI-valid announcements, which is detailed in §A.4.

Furthermore, when we focus on CPU usage, we find that it only adds 2.1% (BIRD) or 3.8% (GoBGP) extra usage throughout our measurement period. Since it only involves a couple of hashmap lookup functions, which are known to be efficient, it does not cause heavy extra cycles. It is also worth pointing out that we do not cause any *burst* CPU usage, ensuring that it does not lead to any CPU throttling.

To further understand the performance overhead introduced by ImpROV, we analyze the execution time of each bytecode with code profilers. In ImpROV, we find two major parts causing extra overhead: (1) creating mitigation routes and (2) updating mitigation routes. For the creation process, the major performance overhead lies in querying the adj-RIB-In to fetch all covered prefixes of an invalid announcement; this step takes 51.4% of the total creation process time for the BIRD implementation and 43.2% for the GoBGP implementation. Interestingly, across all of our experiments, we notice that GoBGP takes more time to insert the mitigation route into the FIB compared to BIRD. This is because GoBGP does not have a native implementation for FIB manipulation but relies on zebra APIs [3], which require extra API calls. For updating the mitigation routes, the major overhead lies in checking the BGP update messages. Although we implement the mitigation route tables as hash tables to minimize the searching time, this searching process is triggered every time

[3]The actual CPU and memory usages of ImpROV and native versions are shown in §A.5.

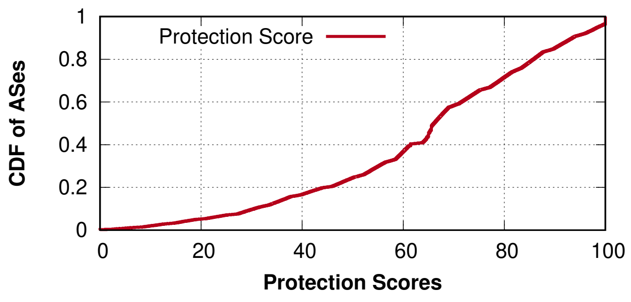

Figure 11: The effectiveness of ImpROV in preventing collateral damage: 50% of ASes with ImpROV are able to reduce more than 72.6% of hijacking attacks.

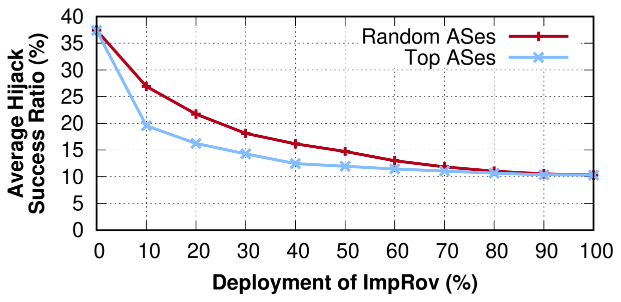

Figure 12: 10% of ImpROV deployment, the average hijacking success rate drops to 19% as focusing on higher ranked ASes.

a new BGP update arrives, accounting for a significant portion of the performance overhead. However, we believe that the performance overhead of ImpROV remains in acceptable bounds (2.2% or 2.7% in memory and 2.1% or 3.8% in CPU overhead with BIRD and GoBGP implementations).

6.3 Collateral Damage Mitigation

6.3.1 Effectiveness of ImpROV against Hijacking

Now, we evaluate how ImpROV helps mitigate the collateral damage of ROV when hijacking happens. We revisit the hijacking incident reports outlined in §4.4.1 to explore how ImpROV could reduce collateral damage to the ROV-enabled ASes if they deploy ImpROV. Initially, we focus on the ASes that deployed ROV but still remained susceptible to collateral damage attacks as detailed in §4.4.1; out of the 4,730 ROV ASes considered, we identify 2,815 (59.5%) that have multiple upstream connections including both non-ROV and ROV ASes, where ImpROV could provide additional protection. We then implement ImpROV in these 2,815 ASes and re-run our simulation to calculate the proportion of hijacking attacks reduced by ImpROV, called Protection Score; for instance, an AS that was vulnerable to collateral damage but successfully prevents all hijacking attempts after deploying ImpROV would achieve a 100% Protection Score. Figure 11 shows the results.

Firstly, we discover that a mere 0.3% of ASes failed to gain any protective benefits from deploying ImpROV. This lack of protection may stem from all of their upstream paths not performing ROV at all, leaving these paths exposed to vulnerabilities.

Secondly, more than half of ASes successfully can prevent 72.6% of collateral damage attacks through ImpROV, which shows that *merely circumventing the right upstreams that propagate RPKI-invalid announcements—without analyzing all possible paths—can still offer substantial protection.* Furthermore, the fact that only a small percentage of ASes failed to benefit from ImpROV suggests that the approach is widely applicable and can provide protection to a large portion of the network. In the next section, we also focus on other ASes who do not install ImpROV and how they can benefit from their upstreams who deployed ImpROV.

6.3.2 Collateral Benefits of ImpROV

An AS that deploys ROV can yield *collateral benefits*, protecting other ASes routing through it from receiving RPKI-invalid announcements. Similarly, the deployment of ImpROV not only benefits the deploying AS itself but also benefits the connected ASes by forwarding traffic to legitimate origins along the path, even though ImpROV only considers the next hop.

To evaluate the additional protection that ImpROV provides to other ROV ASes, we measure the average hijack success ratio *across all ROV ASes in the topology* when *k* ROV ASes have also deployed ImpROV. As a baseline, we first measure the average hijack success ratio among the ASes that perform ROV but without deploying ImpROV and analyze how this ratio decreases as more ROV ASes deploy ImpROV. We gradually increase the number of ASes that deploy ImpROV. While we can choose random ASes to deploy ImpROV, we also focus on selecting higher-ranked ASes since larger networks have a higher probability of encountering hijacking attempts, thus contributing more significantly to reducing the overall hijacking success ratio. When we choose random ASes, we calculate the average success ratio by running 10 different experiments and obtain their average.

Figure 12 shows the average hijack success ratio across ASes depending on the percentage of ROV ASes that have also deployed ImpROV. Interestingly, we find that the average successful hijack ratio drops from 37.4% to 26.92% when we deploy ImpROV on only 10% of ROV ASes randomly; however, this ratio further drops to 19.54% when we choose the top 10% ROV ASes for ImpROV deployment. This emphasizes the importance of higher-ranked ASes; they not only provide a larger global impact through the collateral benefit of ROV deployment [22], but they can also significantly contribute to reducing the *overall* collateral damage by deploying ImpROV.

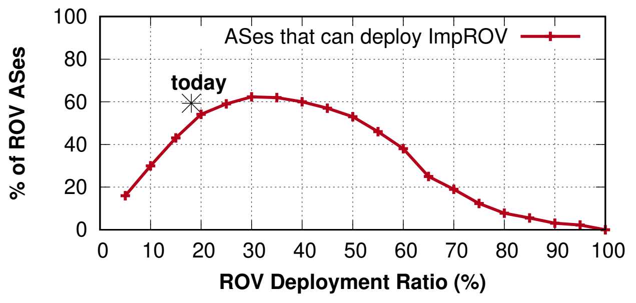

Figure 13: As ROV deployment grows, the number of ASes that can deploy and benefit from ImpROV also increases initially; however, after a certain point, this becomes marginal as collateral damage and hijacking vulnerability decrease with widespread ROV adoption.

The substantial difference in the hijack success ratio between random and top-ranked AS deployment highlights the critical role that influential ASes play in enhancing global routing security. By prioritizing the adoption of ImpROV among these key players, we can effectively amplify the collateral benefits and minimize the impact of hijacking attempts on the entire network.

6.4 Effectiveness of ImpROV vs. ROV Deployment

To deploy ImpROV, an AS must have multiple upstreams, with at least one dropping RPKI-invalid announcements; otherwise, there are no safe upstreams available to establish iRoutes. As ROV adoption increases, more ASes become capable of implementing ImpROV. However, if an AS only has ROV upstreams, ImpROV may not provide additional benefits. This raises the question: *to what extent can ImpROV be deployed as more ASes adopt ROV?* To address this, we use CAIDA's AS relationship dataset to construct a topology graph of 76,436 ASes. We randomly select different portions of ASes to deploy ROV and calculate the percentage of ROV ASes with both ROV and non-ROV neighbors. Based on the current topology, we assess how many ASes can deploy ImpROV by avoiding routes toward ASes that forward RPKI-invalid prefixes. We repeat this simulation 10 times and take the average. Figure 13 shows the results. Note that no ASes can deploy ImpROV without ROV-performing upstreams. As ROV deployment increases, the number of ASes capable of deploying ImpROV grows, peaking at 26.5% of total ASes when 50% have deployed ROV. Beyond 50% ROV deployment, the additional benefit of ImpROV decreases as more ASes have all ROV upstreams; ImpROV becomes less significant when it no longer introduces any iRoute due to receiving no RPKI-invalid BGP announcements from ROV-performing upstreams. Although ImpROV offers no additional benefits with 100% perfect ROV deployment, we believe this scenario is unlikely. With current ROV deployment below 18% [39] and many ASes implementing it partially due to business [30] or technical [4] considerations, ImpROV remains crucial.

In summary, ImpROV enhances protection as ROV adoption grows, maintaining relevance until ROV deployment reaches about 90%, highlighting its importance during the transition to widespread ROV adoption.

7 Potential Security Concerns

ImpROV creates and stores iRoutes to mitigate collateral damage. Specifically, for each invalid route, ImpROV generates two one-bit longer routes. For instance, a /20 route produces two /21 iRoutes, introducing $2 \times N$ iRoutes when N RPKI-invalid routes are received. This prompts concern over its possible use as a DoS attack vector by intentionally announcing RPKI-invalid prefixes, but its memory overhead is minimal, as elaborated below.

Memory overhead in real-world prefix hijacks: First, we consider the additional memory usage introduced by ImpROV in real-world prefix hijacking incidents. Although hijacking ROA-covered prefixes increases the number of RPKI-invalid routes, the resulting iRoutes are still small compared to the size of global routing tables; as a reference point, one of the largest prefix hijacking events occurred in April 2020, when AS 12389 hijacked more than 8,800 prefixes [11]. We simulated the memory overhead imposed by ImpROV in that scenario, assuming all hijacked prefixes were ROA-covered and thus triggered iRoutes creation. The results show only 34.6 MB of additional memory usage, which is negligible for modern routers.

Influxes of RPKI-invalid prefixes: In extreme cases—such as peering with malicious routers without defenses like route limits—the adversary could flood ImpROV with invalid prefixes, leading to increased memory consumption. However, even under the unlikely condition where every prefix in the global IPv4 routing table is both ROA-covered and hijacked, the memory usage of routing software (e.g., BIRD or GoBGP) with ImpROV remains under 3 GB. Since entry-level routers, such as Cisco's 4000 series, typically have 4 GB–16 GB of control-plane memory [18], this growth is unlikely to overwhelm router resources or introduce new security concerns. Moreover, large-scale flooding is already recognized as a threat [34], and standard mitigation techniques—such as rate limiting and prefix count limits—remain effective regardless of ImpROV.

8 Conclusion

In this paper, we presented the first comprehensive study of collateral damage in RPKI, revealing that 85.6% of RPKI-invalid announcements are vulnerable to such attacks, affecting 34% of ROV-enabled ASes. To address this issue, we introduced ImpROV, a lightweight system that mitigates collateral damage with minimal overhead. With only 10% deployment among top-tier ROV ASes, ImpROV reduces the average successful hijack ratio from 37.4% to 19.54%, while increasing memory usage by 2.2-2.7%, CPU usage by 2.1-3.8%, and FIB entries by 1.1% on average. After deployment, 50% of ASes can thwart approximately 72.6% of potential attacks. Despite some limitations, ImpROV offers a practical solution for enhancing inter-domain routing security in the context of RPKI deployment.

Acknowledgments

We thank anonymous reviewers for their helpful comments. This research was supported in part by NSF grant CNS-2323137, CNS-2339378, and the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [RS-2023-00215700, Trustworthy Metaverse: blockchain-enabled convergence research].

References

[1] Blackhole Communities. https://isp.anexia-it.net/communities/#rpki-communities.

[2] GoBGP: BGP implementation in Go. https://github.com/osrg/gobgp.

[3] GoBGP: FIB manipulation. https://github.com/osrg/gobgp/blob/master/docs/sources/zebra.md.

[4] 16.2R2-S9: Software Release Notification for Junos Software Service Release version 16.2R2-S9. https://supportportal.juniper.net/s/article/16-2R2-S9-Software-Release-Notification-for-Junos-Software-Service-Release-version-16-2R2-S9?language=en_US.

[5] A. Azimov, E. Uskov, R. Bush, K. Patel, J. Snijders, and R. Housley. A Profile for Autonomous System Provider Authorization. IETF, 2018. https://tools.ietf.org/html/draft-azimov-sidrops-aspa-profile-00.

[6] H. Ballani, P. Francis, and X. Zhang. A study of prefix hijacking and interception in the internet. *SIGCOMM*, 2007.

[7] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98(1), IEEE, 2010.

[8] R. Bush, K. Patel, P. F. Smith, and M. Tinka. Policy Based on the Resource Public Key Infrastructure (RPKI) without Route Refresh. RFC 9324, IETF, 2022.

[9] R. Brandom. Hackers emptied Ethereum wallets by breaking the basic infrastructure of the internet. 2018. https://www.theverge.com/2018/4/24/17275982/myetherwallet-hack-bgp-dns-hijacking-stolen-ethereum.

[10] M. A. Brown. Pakistan hijacks YouTube. 2008. https://dyn.com/blog/pakistan-hijacks-youtube-1/.

[11] BGPStream. https://bgpstream.com/.

[12] J. Cowie. China's 18-Minute Mystery. 2010. https://dyn.com/blog/chinas-18-minute-mystery/.

[13] N. CZ. BIRD Internet Routing Daemon. https://bird.network.cz/.

[14] R. Chandra, P. Traina, and T. Li. BGP Communities Attribute. RFC 1997, IETF, 1996. https://www.rfc-editor.org/info/rfc1997.

[15] T. Chung, E. Aben, T. Bruijnzeels, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, R. van Rijswijk-Deij, J. P. Rula, and N. Sullivan. RPKI is Coming of Age: A Longitudinal Study of RPKI Deployment and Invalid Route Origins. *IMC*, 2019.

[16] B. Cartwright-Cox. Measuring RPKI Adoption via the data-plane. NLNOG Day 2018. https://nlnog.net/static/nlnogday2018/8_Measuring_RPKI_ben_NLNOG_2018.pdf.

[17] CAIDA ASRelationships Dataset. http://www.caida.org/data/as-relationships/.

[18] Cisco 4000 Family Integrated Services Router Data Sheet. https://www.cisco.com/c/en/us/products/collateral/routers/4000-series-integrated-services-routers-isr/data_sheet-c78-732542.html.

[19] B. Du, C. Testart, R. Fontugne, A. C. Snoeren, and K. Claffy. Poster: Taking the Low Road: How RPKI Invalids Propagate. *SIGCOMM*, 2023.

[20] P. Gill, M. Schapira, and S. Goldberg. Let the market drive deployment: A strategy for transitioning to BGP security. *ACM SIGCOMM computer communication review*, 41(4), ACM New York, NY, USA, 2011.

[21] P. Gill, M. Schapira, and Goldberg. Modeling on quicksand: dealing with the scarcity of ground truth in interdomain routing data. *ACM SIGCOMM Computer Communication Review*, 42(1), ACM New York, NY, USA, 2012.

[22] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman. Are We There Yet? On RPKI's Deployment and Security. *NDSS*, 2017.

[23] G. Huston, R. Loomans, and G. Michaelson. A Profile for Resource Certificate Repository Structure. RFC 6481, IETF, 2012.

[24] T. Hlavacek, P. Jeitner, D. Mirdita, H. Shulman, and M. Waidner. Behind the scenes of RPKI. *CCS*, 2022.

[25] T. Hlavacek, H. Shulman, and M. Waidner. ”Keep Your Friends Close, but Your Routeservers Closer: Insights into RPKI Validation in the Internet. *USENIX Security*, 2023.

[26] D. Iamartino. Study and measurements of the RPKI deployment. Master’s Thesis, Politecnico di Milano, 2015.

[27] M. Kang, W. Li, R. van Rijswijk-Deij, T. T. Kwon, and T. Chung. IRRedicator: Pruning IRR with RPKI-Valid BGP Insights. *NDSS*, 2024.

[28] M. Lepinski and S. Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480, IETF, 2012.

[29] V. Luconi and A. Vecchio. Impact of the first months of war on routing and latency in Ukraine. *compnet*, 224, Elsevier, 2023.

[30] W. Li, Z. Lin, M. I. A. Khan, E. Aben, R. Fontugne, A. Phokeer, and T. Chung. RoVista: Measuring and Understanding the Route Origin Validation (ROV) in RPKI. *IMC*, 2023.

[31] A. Medina. CenturyLink / Level 3 Outage Analysis. 2020. https://www.thousandeyes.com/blog/centurylink-level-3-outage-analysis.

[32] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein. BGP Prefix Origin Validation. RFC 6811, IETF, 2013.

[33] R. Morillo, J. Furuness, C. Morris, J. Breslin, A. Herzberg, and B. Wang. ROV++: Improved Deployable Defense against BGP Hijacking. *NDSS*, 2021.

[34] S. Murphy. BGP security vulnerabilities analysis. RFC 4272, IETF, 2006.

[35] N. L. Ziggy: the RPKI Wayback Machine. https://github.com/NLnetLabs/ziggy.

[36] A. Reuter, R. Bush, I. Cunha, E. Katz-Bassett, T. C. Schmidt, and M. Whlisch. Towards a Rigorous Methodology for Measuring Adoption of RPKI Route Validation and Filtering. *CCR*, 48(1), 2018.

[37] RIPE Atlas. https://atlas.ripe.net/.

[38] RPKI Deployment Monitor. https://rpki-monitor.antd.nist.gov.

[39] RPKI I-Rov Filtering Rate. https://stats.labs.apnic.net.

[40] University of Oregon RouteViews project. http://www.routeviews.org/.

[41] Routinator. https://nlnetlabs.nl/projects/rpki/routinator/.

[42] B. Schlinker, T. Arnold, I. Cunha, and E. Katz-Bassett. PEERING: Virtualizing BGP at the Edge for Research. *CoNEXT*, 2019.

[43] J. Snijders. Estimating the Timeline for ASPA Deployment. https://https://manrs.org/2023/05/estimating-the-timeline-for-aspa-deployment/.

[44] Secure Inter-Domain Routing (sidr). https://datatracker.ietf.org/wg/sidr/about/.

[45] L. Tung. iCloud goes down: Apple joins the Google, Facebook, Cloudflare cloud outage club. 2019. https://www.zdnet.com/article/icloud-goes-down-apple-joins-the-google-facebook-cloudflare-cloud-outage-club/.

[46] M. Wählisch, R. Schmidt, T. C. Schmidt, O. Maennel, S. Uhlig, and G. Tyson. RiPKI: The Tragic Story of RPKI Deployment in the Web Ecosystem. *HotNets*, 2015.

[47] R. Yakov, L. Tony, H. Susan, and others. A border gateway protocol 4 (BGP-4). RFC 4271, IETF, 1994.

[48] YouTube Hijacking: A RIPE NCC RIS case study. 2008. https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study.

A Appendix

A.1 Ethics

This work does not raise any ethical issues.

A.2 Open Science Policy

The authors will make publicly available all source code for data collection and analyses used in this study, as well as the detailed descriptions of how public datasets are collected [4].

A.3 De-prioritizing Less-specific Route

To mitigate collateral damage, one can de-prioritize the next hop that forwards RPKI-invalid prefixes for existing routes in the routing table instead of generating additional routes. With this way, there's no need to generate new routes for each RPKI-invalid prefix. However, when sub-prefix hijacks occur, the existing routes in the routing table will be less specific than the actual hijacked IP blocks. De-prioritizing these less-specific routes will result in changing the next hop for benign IP spaces that are not under hijack (benign prefixes).

Figure 14 shows how to mitigate the collateral damage by de-prioritizing less-specific routes in AS 1, where AS4 announces 1.2.0.0/16 and AS5 hijacks AS4 with a more specific prefix 1.2.3.0/24. Before the hijack happens, AS1 selects AS3 as the best route to forward all traffic towards 1.2.0.0/16.

When AS1 receives the invalid announcement of 1.2.3.0/24, it will re-select the best route for *all less specific prefixes* with the modified best route selection algorithms.

Thus, AS1 re-runs the best route selection process for 1.2.0.0/16 and chooses the safer upstream AS2, which looks similar to the outcome of ImpROV. However, this will not only impact the IP space that is under hijack (1.2.3.0/24), but also impact other IPs that are under the less specific prefix 1.2.0.0/16; for example, 1.2.1.0/24 (not impacted by the hijack) should not be affected on the data-plane and could be forwarded to either AS2 or AS3 since these are not vulnerable to hijack. Since prefixes that are less specific than the announced RPKI-invalid prefix are de-prioritized, the next hop for benign IP spaces like 1.2.1.0/24 will also change to AS2, which may not be the desired outcome depending on AS1's routing policy (e.g., traffic engineering). We call these IP prefixes that are not impacted by the RPKI-invalid announcement, but forced to be de-prioritized, benign prefixes.

To estimate the impact on creating such benign prefixes, we first collect all RPKI-invalid prefixes that are covered by valid less specific prefixes from the dataset we used in §4 (dotted line in Figure 4). We then calculate the size of IP spaces impacted by de-prioritizing less-specific route versus the IP spaces that are under hijack; Figure 15 shows the number of impacted prefixes that each RPKI-Invalid prefixes create after de-prioritizing shorter routes under sub-prefix hijack. On average, to mitigate collateral damage of one RPKI-Invalid prefix, we find that de-prioritizing less-specific route will impacts 84 times more benign IPs than the actual prefixes under hijack, potentially leading to unexpected outcomes.

In contrast, ImpROV only prioritizes the next hop (which does not forward RPKI-invalid prefixes) by adding iRoutes without modifying the best route selection algorithm, thus avoiding the creation of such benign prefixes.

[4] https://doi.org/10.5281/zenodo.14643447

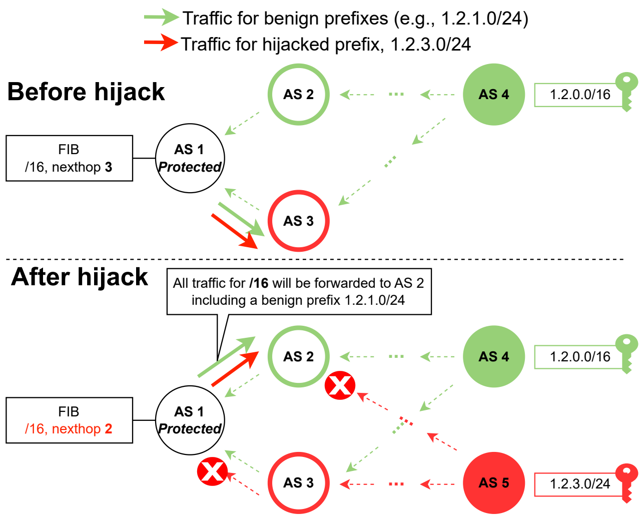
Traffic for benign prefixes (e.g., 1.2.1.0/24)
Traffic for hijacked prefix, 1.2.3.0/24
Before hijack
AS 2
AS 4
1.2.0.0/16
FIB
/16, nexthop 3
AS 1
Protected
AS 3
After hijack
All traffic for /16 will be forwarded to AS 2 including a benign prefix 1.2.1.0/24
AS 2
AS 4
1.2.0.0/16
FIB
/16, nexthop 2
AS 1
Protected
AS 3
AS 5
1.2.3.0/24

Figure 14: The data-plane impact of de-prioritizing less-specific route. To mitigate collateral damage of a hijacked /24 prefix, it impacts the data-plane of the entire /16 IP space, including other 255 /24 prefixes (e.g., 1.2.1.0/24) that are not being hijacked.

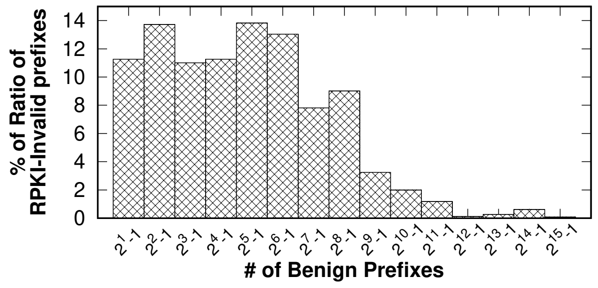

Figure 15: The distribution of the number of benign prefixes been impacted for each RPKI-invalid announcement.

A.4 The Number of Increased Entries in FIB

Another concern regarding the performance impact of ImpROV is the potential performance downgrade in the data-plane due to the additional routes installed in the routing

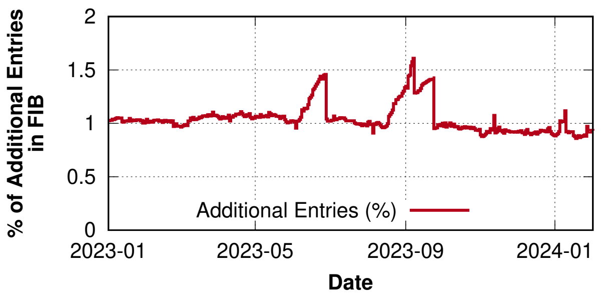

Figure 16: The percentage of additional entries in FIB.

tables; however, it is important to note that ImpROV does not install any additional routes in the `adj-RIB-In`, `Loc-RIB`, or `adj-RIB-Out`. The only routing table affected by ImpROV is the FIB, which is used for the actual forwarding of packets. Now, we calculate the actual number of increased entries in FIB due to the iRoute. To this end, we take each day's Route-Views RIB from January 1st, to February 1st, 2024 and run ImpROV and measure the number of FIB entries and compare them with the ones when we run the same experiment with the native implementation. Figure 16 shows the result; we can confirm that ImpROV only increases the number of entries in FIB by 1.1% additional routes on average, while having the maximum increase of 1.5% when there are the most RPKI-invalid routes during that day, as discussed in §4.1. This result demonstrates that the impact of ImpROV on the FIB size is relatively small, even during periods with a higher number of RPKI-invalid routes; the average increase of 1.1% in FIB entries suggests that ImpROV can effectively reduce collateral damage without causing a significant burden on the size of the FIB.

A.5 CPU and Memory Overhead

The CPU and memory usage of ImpROV and native versions are shown in Figure 17 and Figure 18, respective.

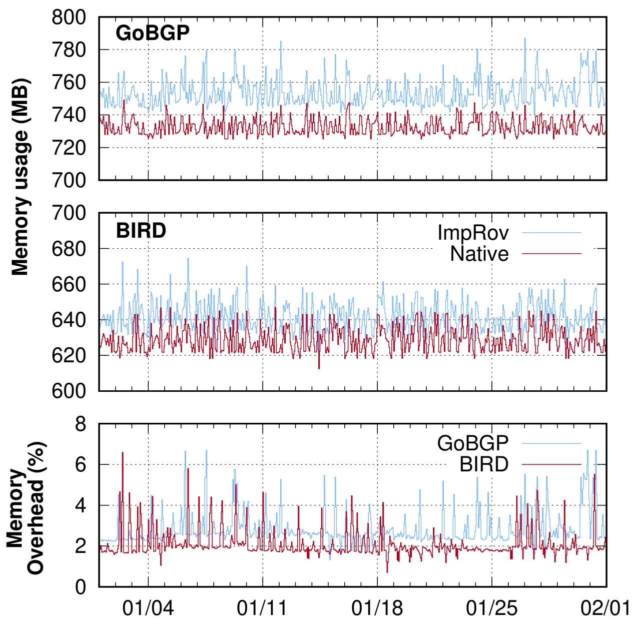

Figure 17: The memory usage of ImpROV and native versions (top and middle) and memory overheads of ImpROV are shown; on average, we find that ImpROV consumes 2.7% (GoBGP) and 2.2% (BIRD) additional memories.

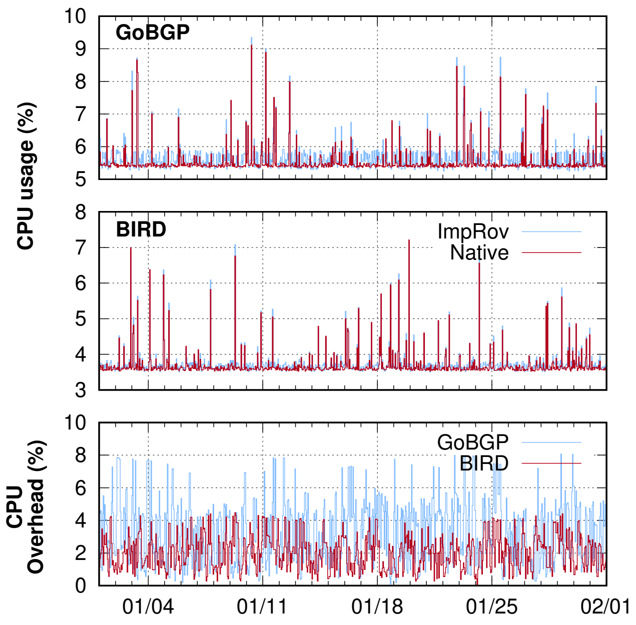

Figure 18: The CPU usage of ImpROV and native versions (top and middle) and the overheads of ImpROV are shown; on average, we find that ImpROV consumes 3.8% (GoBGP) and 2.1% (BIRD) more CPU cycles. Note that the overhead is calculated as the percentage of increased CPU usage relative to the native implementation.