ROV++: Improved Deployable Defense against BGP Hijacking

Reynaldo Morillo
University of Connecticut
reynaldo.morillo@uconn.edu

Justin Furuness
University of Connecticut
jfuruness@gmail.com

Cameron Morris
University of Connecticut
cameron.morris@uconn.edu

James Breslin
University of Connecticut
james.breslin@uconn.edu

Amir Herzberg
University of Connecticut
amir.herzberg@uconn.edu

Bing Wang
University of Connecticut
bing@uconn.edu

***Abstract*—We study and extend Route Origin Validation (ROV), the basis for the IETF defenses of interdomain routing. We focus on two important hijack attacks: *subprefix hijacks* and *non-routed prefix hijacks*. For both attacks, we show that, with partial deployment, ROV provides disappointing security benefits. We also present a new attack, *superprefix hijacks*, which completely circumvent ROV's defense for non-routed prefix hijacks.**

We then present ROV++, a novel extension of ROV, with significantly improved security benefits even with partial adoption. For example, with uniform 5% adoption for edge ASes (ASes with no customers or peers), ROV prevents less than 5% of subprefix hijacks, while ROV++ prevents more than 90% of subprefix hijacks. ROV++ also defends well against non-routed prefix attacks and the novel superprefix attacks.

We evaluated several ROV++ variants, all sharing the improvements in defense; this includes "Lite", *software-only* variants, deployable with existing routers. Our evaluation is based on extensive simulations over the Internet topology.

We also expose an obscure yet important aspect of BGP, much amplified by ROV: *inconsistencies* between the observable BGP path (control-plane) and the actual traffic flows (data-plane). These inconsistencies are highly relevant for security, and often lead to a challenge we refer to as *hidden hijacks*.

I. Introduction

BGP, the Internet's inter-domain routing protocol, does not have authentication mechanisms, and suffers from frequent misconfigurations and attacks, often with wide-ranging impacts, e.g., [47], [58], [2], [60], [5]. By far, the most common and effective attacks are *prefix* and *subprefix hijacking*, where a rogue or misconfigured autonomous system (AS) announces an IP address block (prefix or subprefix) not assigned to it.

The standardized defense against prefix and subprefix hijacks is *Route Origin Validation (ROV)*, which is expected to be the first and most important application of the Resource Public Key Infrastructure (RPKI) [32], [7], [17], [61], [54]. ROV is also the prerequisite for BGPsec [37], the other standardized defense for BGP, and for other inter-domain routing security mechanisms, such as soBGP [63] and path-end validation [15].

Network and Distributed Systems Security (NDSS) Symposium 2021
21-24 February 2021
ISBN 1-891562-66-5
https://dx.doi.org/10.14722/ndss.2021.24438
www.ndss-symposium.org

Reports from NIST RPKI monitor [6] and a recent longitudinal study [14] show that, despite a slow start, adoption of RPKI—esp. issuing of routing certificates and Route Origin Authorizations (ROAs)—has finally taken off, mainly since 2018. Importantly, misconfigurations that were abundant in the past have become quite rare; there are also proposals for further supporting (correct) deployments, e.g., [24]. As a result, it appears that RPKI is coming of age, and can finally be deployed by ROV to filter and drop invalid announcements. Another recent study [57] shows that ROV has gone from being non-existent two years ago to being used by more ISPs (e.g., AT&T [1]) to filter BGP announcements, which has led to benefits for the ASes that use RPKI.

Deployment of ROV is, however, still slow, as shown in multiple works [29], [30], [62], [48], [19], [12], [25], [24]. Furthermore, Gilad et al. [19] have shown that, until it is widely deployed, ROV only leads to limited benefits.

The security benefits of partially-deployed ROV are especially meager against *subprefix* hijacks; even for *adopting* ASes, the benefits are meager, unless almost all top ASes adopt ROV [19]. Note that an adopting AS will never *use* an invalid announcement; however, its *traffic* may still be hijacked due to routing via a non-adopting transit AS, through which the packets will be routed to the attacker because of the preference of more-specific routes. We refer to this as *data-plane hijack*, since, although the routing table (control plane) indicates a path to the correct origin, the traffic itself (data plane) is hijacked (see §II). Analyzing BGP routing data provided by RouteViews [49], we see approximately 343K prefixes are shorter than /24 as of July 2020, which are susceptible to subprefix hijacks, endangering roughly 2B IP addresses, approximately half of the Internet. For ROV to be effective against data-plane subprefix hijacks, it requires an extremely widespread deployment, which is not expected in the foreseeable future.

We also study the security of ROV against another important threat: *non-routed prefix hijacking*. Non-routed prefixes are widely abused, esp. for spam, phishing and DDoS attacks; see [60]. In addition, we introduce a novel variant of non-routed prefix hijacks, involving *superprefix announcement*. We show that ROV is less effective against non-routed prefix hijacks, and completely fails against the superprefix variant of this attack.

Better defenses? Introducing ROV++! We present *ROV++*, an extension of ROV, with much improved security

against subprefix and non-routed hijacks, including the new superprefix hijacks. ROV++ defense is effective even with limited deployment, with improvements to both adopting and non-adopting ASes. We present several variants of ROV++, and evaluate them using large-scale simulations. The results show clear improved security of ROV++ (compared to ROV), and tradeoffs between the different ROV++ variants.

We refer to several variants as *Lite*; these can be deployed very easily—using *existing routers*. These Lite versions have nearly the same security guarantees as their non-Lite versions, all of which are certainly much more secure than ROV.

A. Main Contributions

• **Design of ROV++, an improved security extension to ROV (§III)**. We identify common scenarios, in which ROV fails to prevent (data-plane) hijacking, and present corresponding extensions, ROV++, versions v1, v2 and v3, which prevent the hijacks. **ROV++ v1** prevents interception for each prefix in one of two ways. The first is to select an announcement received from a neighbor who did not send a hijacked announcement for a subprefix. If no such announcement exists, the traffic sent to the hijacked subprefix is dropped. **ROV++ v2** announces a path to the hijacked subprefix when the AS cannot correctly route to it, in addition to blackholing this traffic (as in v1). This *blackhole announcement* competes with the subprefix hijack, and therefore may protect other ASes from being hijacked. Note, however, that blackhole announcements may also increase the number of ASes whose packets cannot reach the subprefix. **ROV++ v3** returns to the case handled by v1, where an AS receives from one neighbor a subprefix hijack, and from another neighbor, an announcement with the correct origin for the prefix, which it then announces. In such cases, to help other ASes reach the correct origin, the v3 AS will send *preventive subprefix announcement*, using the same path. Like the blackhole announcements of v2, the preventive announcement competes with the subprefix hijack. If these preventive announcements are successful, traffic would reach the correct origin, hopefully *improving reachability*.

• **ROV++ security analysis (§IV).** We show that an AS that adopts ROV++ does not introduce hijacks beyond what would have happened if the AS used BGP or ROV.

• **Detailed evaluation of ROV and ROV++ (§V):** To evaluate and fine-tune ROV++, we implemented a detailed BGP simulator, incorporating BGP announcement and withdrawal mechanisms. Using large-scale simulations over empirically-derived datasets, we performed extensive measurements of partial deployments of ROV and different variants of ROV++. Our experiments cover adopting ASes and non-adopting ASes ("collateral benefit"), results for subprefix hijacking and non-routed prefixes, data-plane and control-plane results, impact on different categories of ASes (edge, top-tier, others), and more. The results show dramatic benefits of ROV++, esp. for early adoption scenarios. In particular, for adoption at just 5%, about *95%* of the *adopting edge* ASes fall victim to subprefix hijacks due to "collateral damage" with ROV, compared to only *5%* with *all* variants of ROV++.

• **Non-routed prefixes: evaluation of ROV and ROV++, and the superprefix-attack (§V).** We present the first evaluation of the effectiveness of ROV and ROV++ against hijacking of non-routed prefixes, widely abused for malicious applications such as spam and DDoS [60]. We also present the *superprefix hijack attack*, a novel way to hijack a non-routed prefix, where the attacker announces a *superprefix* of the non-routed prefix, together with (or even instead of) sending a hijack announcement for the non-routed prefix. ROV is quite helpless against this attack; in contrast, all versions of ROV++ defend well against it, even with minimal deployment.

• **Lite ROV++: software-only, readily-deployable design (§VI).** To facilitate implementation and fast adoption, we designed the *ROV++ Lite* versions, which can be readily implemented and deployed using existing routers. Our extensive simulation results demonstrate that ROV++ Lite variants are almost as effective as the corresponding non-Lite ROV++ versions.

• **Recommendations (§VII).** Our results seem to indicate a clear recommendation for deployment: ROV++ v1 for routed prefixes, and ROV++ v2 (or a variant) for non-routed prefix hijacks. We recommend the Lite version for both cases since it is easier to deploy and has similar performance as the non-Lite version.

B. Related Work

Security of inter-domain routing. Security vulnerabilities of inter-domain routing have long been recognized (see summary in [45], [28], [8], [23], [42], [52], and common misconfigurations in [39]). A large number of security solutions have been proposed [34], [63], [21], [3], [26], [55], [64], [31], [9], [59], [15], [53]; the effectiveness of BGP security protocols are compared in [13], [20]. In addition, techniques have been proposed to detect security problems, e.g., invalid multiple origin AS [65] and BGP serial hijackers [56]. IETF has standardized approaches for secure inter-domain routing, namely, RPKI [32], [17], [61] and BGPsec [37], for origin and path authentication, respectively. RPKI is the prerequisite for more comprehensive security solutions, including BGPsec. The studies in [5], [20], [35] show that RPKI can already provide significant benefits in improving inter-domain routing security; path-security solutions (such as BGPsec) only provides moderate benefits over RPKI, especially under partial deployment [35], [15].

RPKI deployment and security. A number of studies [29], [30], [62], [48], [19], [12], [25], [24], [6] investigate the deployment status of RPKI, including the issuing of ROAs and the adoption of ROV, and investigate how to improve adoption. A recent study [14] uses 8 years' data to show that after a gradual start, RPKI has seen rapid adoption in the past two years and misconfigurations have become rare. As a result, the authors advocate using ROV to identify and drop invalid announcements to increase routing security. The study in [57] shows that ISPs (e.g., AT&T [1]) have started adopting ROV to filter BGP announcements recently.

Several works study security concerns with RPKI, e.g., the threat of misbehaving RPKI authorities [16], [22], [33]. These concerns are orthogonal to our study.

II. Motivating ROV++ by Use Cases

Brief background on RPKI and ROV. RPKI facilitates the issuing, distribution, and authentication of (signed) Route Origin Authorizations (ROAs). A ROA R is a signed tuple $R = (R.p, R.o, R.l)$, indicating that the AS $R.o$ is authorized to be the origin of BGP announcements for prefix $p \subseteq R.p$, unless p is longer than $R.l$ (in which case, we say that p is *too specific*). A BGP announcement A for prefix $A.p$ with origin $A.o$ is considered *valid* if there is a properly-signed ROA R, which indicates that $A.o$ is authorized to announce A, i.e., $A.o = R.o$, $A.p \subseteq R.p$ and $len(A.p) \leq R.l$. In contrast, an announcement A is *invalid* if a properly signed ROA R binds the announcement's prefix to some origin (i.e., $A.p \subseteq R.p$), but $A.o \neq R.o$ or $len(A.p) > R.l$. The status of an announcement is *unknown* if it is neither valid nor invalid. We focus on prefixes covered by correct ROAs [32], [36], [19], except for superprefix in §III-D for the reasons explained there.

ROV: Sub-optimal security benefits? While ROV provides origin authentication, it does not make the best use of the ROAs to improve security. In particular, ROV often fails to prevent data capture via subprefix hijacks; furthermore, it often *hides* the subprefix hijacks, resulting in *hidden hijacks* (see more details below), which are not visible from the routing tables, and may mislead network operators and hijack defenses to believe that the data is not hijacked. In this section, we discuss several *use cases* that illustrate the suboptimal benefits of ROV, and how ROV++ provides better security benefits given the same ROAs. The detailed design of ROV++ and deployment/implementation issues are deferred to §III and §VI.

Valley-free and other routing assumptions. We make the standard simplifying assumptions about the policy used by ASes. Most significantly, we assume *valley-free* (Gao-Rexford) routing is used by all ASes [18]. That is, for any given prefix, an AS forwards the best announcement that it receives from its customers to all the neighboring ASes (customers, providers and peers); if none was received from customers, then it forwards the best announcement that it receives from a peer—or, if none, from a provider—but only to its customers. If, for some prefix, there is more than one announcement with the "best" relationship, then the AS selects the one with a shorter AS-path; if the length is also the same, the AS uses some tie-breaking rule. While existing studies have shown that real-world routing is not always valley-free [44], [38], [41], [4], we adopt this simple policy as in most existing studies on routing security (e.g. [13], [20], [19]) due to lack of better models.

We also assume that BGP selects one announcement, if available, for *every* prefix, regardless of the existence of announcements for sub-prefixes and super-prefixes. In particular, we assume no prefix aggregation.

Note that as a result of these assumptions and the strong connectivity of the Internet AS topology, if a prefix is announced by an AS, then an announcement for it would reach almost every other AS. In particular, if a prefix p is announced only by one origin AS, say AS a, then almost all ASes would receive the announcement for p with origin a. This is the case for the announcements sent in the attacks we focus on: subprefix hijack, non-routed prefix hijack, and superprefix hijack.

Hidden hijacks occur when IP packets sent by an AS to some prefix are hijacked, even though all the BGP announcements received by the AS contain the correct origin. Hidden hijacks are possible with ROV, ROV++ and even BGP. Fig. 1(a) shows an example, where due to valley-free routing, AS 44 will fall victim to a subprefix hijacking without receiving the subprefix-hijack announcement. This is because AS 88 does not send the subprefix-hijack announcement to AS 44, since AS 88 received this announcement from a peer, while AS 44 is a provider of AS 88; on the other hand, traffic from AS 44 to the subprefix 1.2.3/24 will be hijacked anyway once reaching AS 77. ROV may also create hidden hijacks, e.g., as in Fig. 1(b). In this example, the ROV adopting AS 78 is hijacked and, because AS 78 drops the subprefix hijack announcement, the subprefix hijack will be hidden from AS 44, whose traffic to 1.2.3/24 will be routed to the attacker and it will not know the change in data flow. If AS 78 did not adopt ROV, the hijack would have been visible at AS 44. Hidden hijacks are a significant obstacle in securing BGP without perfect knowledge of the data plane. We formally define hidden hijacks and describe hidden hijacks in ROV++ in §IV.

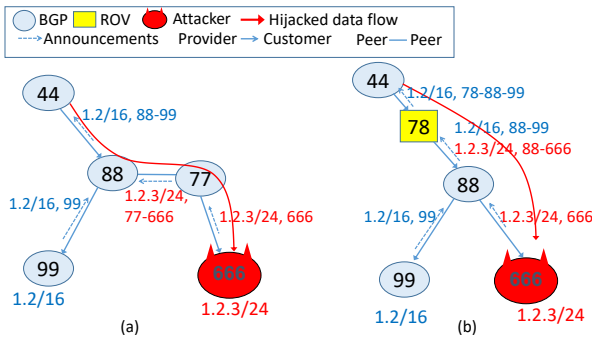
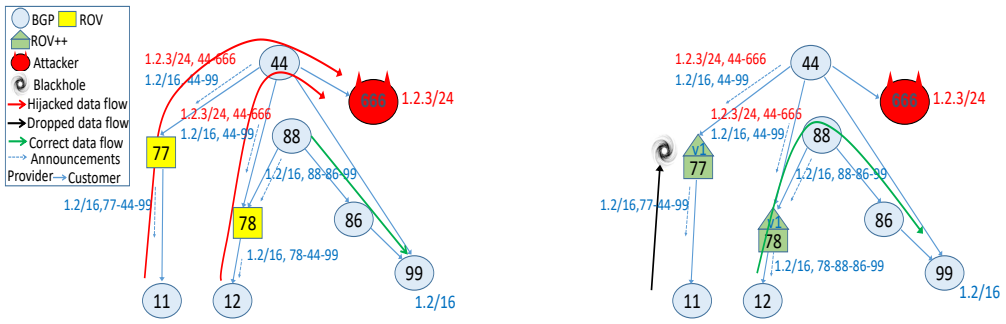


Figure 1. Illustration of hidden hijacks with (a) BGP and (b) ROV.

Three versions of ROV++. In the following, we present the three basic mechanisms of ROV++, which are incorporated, progressively, into three versions (ROV++ versions v1, v2 and v3). ROV++ v1 has the first mechanism, ROV++ v2 has the first two mechanisms, and ROV++ v3 has all three mechanisms. In the figures below, blue ovals and yellow rectangles represent the ASes that run BGP and ROV, respectively. An AS that adopts ROV++ is represented as a green rectangle, topped by a triangle containing the version number.

A. Preventing Visible Hijacks (ROV++ v1)

The simplest type of data-plane hijack that is not prevented by ROV occurs when a ROV-deploying AS receives, from the same neighbor, both a valid announcement for a prefix (say, 1.2/16) and an invalid announcement for a subprefix (say, 1.2.3/24); this is the scenario for both ASes 77 and 78 in Fig. 2(a). Suppose that this is either the *only* announcement received for the prefix (as is the case for AS 77) or the *best* announcement for the prefix (as is the case for AS 78). In both cases, ROV will discard the invalid subprefix announcement and use the prefix announcement—e.g., in Fig. 2(a), both ASes 77 and 78 will use the announcement received from AS 44. As a result, they will forward data packets with destination IP addresses in 1.2/16 to AS 44. Since AS 44 is using plain BGP, however, it will route these packets directly to the attacker, AS 666. This is a data-plane hijack; it applies to all traffic sent



(a) (b)

Figure 2. ROV++ v1 *prevents visible data-plane subprefix hijacks*. (a) ASes 77 and 78 adopt ROV, and yet their traffic to 1.2.3/24 is hijacked, (b) the hijack is prevented using ROV++ v1, by blackholing at AS 77 and by preferring a secure alternative path at AS 78. ASes 77 and 78's adoption of ROV++ also protect their customers.

from ASes 77 and 78, as well as from their customers (ASes 11 and 12), to the subprefix 1.2.3/24.

Notice that AS 77 and AS 78 could have deduced that any traffic to 1.2.3/24 routed to AS 44 would be hijacked, since they also received from AS 44 the subprefix hijack announcement, and hence they "knew" that traffic to the subprefix would be hijacked. In other words, the data-plane subprefix hijack was *visible* to the ROV-deploying AS. However, ROV fails to take this into account, and traffic to 1.2.3/24 is hijacked.

Alternative behaviors for AS 77 and AS 78 are depicted in Fig. 2(b), which show improved resilience to visible hijacks. AS 78 *prefers* a different path to the 1.2/16 prefix, which goes via AS 88, thereby avoiding the hijack. AS 77 can only route via AS 44—that is, it has no "safe route" that it could prefer and reach the origin, AS 99; instead, it *blackholes* the "doomed" traffic to 1.2.3/24. While this is not the most desirable outcome, as it results in the loss of this traffic, it is still much better than having the traffic hijacked. The blackholing and preference mechanisms by ASes 77 and 78 also benefit their customers (ASes 11 and 12, respectively).

We use the term *ROV++ v1* to refer to the extension of ROV that provides a preference to a secure path and blackholing when no secure path exists. The actual design also addresses more complex scenarios, explored in the next section. Note that ROV++ v1 differs from another route validation policy, Drop Invalid if Still Routable (DISR) [53]. In Fig. 2(b), if AS 78 deploys DISR, then it will drop the subprefix announcement 1.2.3/24 since it can route traffic (going to 1.2.3/24) using the less specific prefix, 1.2/16. However, in that case, the traffic from ASes 78 and 12 to 1.2.3/24 will be routed using a shorter path, i.e., via AS 44, and hence hijacked.

B. Blackhole Announcements (ROV++ v2)

As discussed above, subprefix hijack announcements—like every announcements of a prefix p sent only with a single origin AS a—typically reach almost every AS in the Internet. This holds even when a few of these announcements get dropped, as would happen with partial deployment of ROV or ROV++ v1. For example, in Fig. 3(a), AS 77 drops the subprefix-hijack announcement; however, its customers, e.g., AS 11, still receive the subprefix hijack announcement, since they are also connected via AS 55. And while AS 77 blackholes traffic sent to the hijacked subprefix 1.2.3/24, this will only apply to packets sent from AS 77 itself; packets from ASes 11, 32 and 33 will not reach AS 77 at all because AS 11 will forward them to AS 55, since routers always relays packets to the most specific (longest) prefix. As a result, their traffic will be hijacked.

ROV++ v2 introduces an additional mechanism, whose goal is to prevent hijack of traffic from customer ASes by "attracting" it and then blackholing it. This is done by having the AS send BGP *blackhole announcements*, i.e., announcing the blackholed prefix; these would "compete" with any subprefix hijack announcements received from non-adopting ASes.

For example, with ROV++ v2, AS 77 would send a subprefix announcement for the hijacked subprefix, thereby attracting such traffic away from the attacker and then blackholing it. Blackhole announcements are illustrated In Fig. 3(b), where AS 77 adopts ROV++v2, and sends blackhole announcement for subprefix 1.2.3/24 to AS 11. As a result, traffic to IP addresses in the subprefix 1.2.3/24 from ASes 11, 32 and 33, and of course from AS 77 itself, would be blackholed rather than hijacked (as it was in Fig. 3(a)).

Blackhole announcements against prefix hijack attacks? Should a ROV++ v2 AS send blackhole announcements upon receiving prefix-hijack announcements for a given prefix? For instance, in Fig. 3(b), suppose AS 77 received a hijack announcement for prefix 1.2/16, should it send blackhole announcements for 1.2/16? Such announcements would compete with prefix-hijack announcements, which may result in blackholing traffic that would otherwise be hijacked. However, such announcements may also compete with *legitimate* announcements for the prefix, which may result in blackholing traffic that would otherwise be routed successfully to the destination. It seems preferable, therefore, not to send the blackhole announcement for prefix hijacking and only send them for subprefix hijacking. In fact, even for subprefix hijacking, blackhole announcements can, rarely, cause blackholing of traffic that would otherwise be routed successfully to the legitimate destination, as we show in §V.

Note, however, that blackhole announcements *should* be used for *non-routed prefix hijacks*, i.e., announcements of prefixes which are known to never be legitimately announced. The RPKI specifications allow ROAs to identify non-routed prefixes by using the special AS number of zero [27]. In this case, there is no risk of blackholing traffic that will otherwise

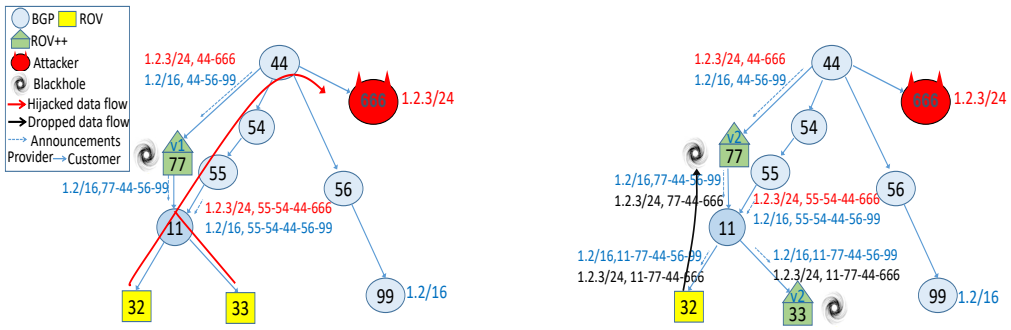


Figure 3. ROV++ v2: *Blackhole announcements protect downstream ASes*. (a) AS 77 runs ROV++ v1 and blackholes the traffic to 1.2.3/24. However, the data from ASes 11, 32 and 33 to 1.2.3/24 is routed via AS 55, and hijacked. (b) AS 77 runs ROV++ v2; it sends a blackhole announcement to AS 11, which protects downstream ASes: traffic from ASes 11 and 32 to 1.2.3/24 is attracted to AS 77 and dropped there; AS 33 directly blackholes the traffic going to 1.2.3/24.

reach the correct destination, and therefore blackholing should be applied. Hijacking non-routed prefixes is a common attack, allowing the use of such IP address blocks for unauthorized purposes such as sending spam, phishing, and Denial-of-Service traffic; see [60]. As we shall see in §V, ROV++ v2 is significantly more effective than ROV in mitigating non-routed prefix hijacks.

Contents of blackhole announcements. The goal of blackhole announcements is to "compete" with the subprefix hijack announcements by attracting and blackholing traffic that would otherwise be hijacked. To conform to BGP and avoid any unexpected side-effects, this is done by forwarding the subprefix hijack announcement, just as it would have been forwarded if the AS has simply used BGP without ROV or ROV++.

Blackhole announcements: when and to whom to send? In ROV++ v2, blackhole announcements are only sent to the customers, as per the valley-free rule [18], and only when receiving a subprefix hijack announcement from peers or providers, not from customers. The rationale is that customers are usually connected only to a few providers, and therefore, when a provider blocks a hijack announcement from a customer, there is a fair chance of blocking the hijacks for other customers, even without sending a blackhole announcement to them. Our evaluation (§V) confirms the advantage of the above restrictive rule.

ROV++ v2a: aggressive non-routed blackhole announcements. ROV++ v2a is a "more **a**ggressive" version of ROV++ v2 that, upon receiving non-routed prefix hijacks, *from any neighbor*, sends a corresponding blackhole announcement according to export policy. The reasoning for the "**a**ggressive" v2**a** is that non-routed prefixes should not be announced at all, and hence there is no danger of competing with the legitimate announcement. Our evaluation in §V confirms that ROV++ v2a is significantly more effective than ROV++ v2, and surely than ROV, in defeating non-routed prefix hijacks.

C. Preventive Announcements (ROV++ v3)

Consider an ROV++ v2 AS that receives only a legitimate prefix announcement from one provider and a subprefix hijack announcement from another provider; see, for example, AS 33 in Fig. 4(a). Since AS 33 has a valid path, via AS 87, for the entire 1.2/16 prefix, it will obviously *not* send a subprefix blackhole announcement. The lack of a blackhole announcement from AS 33, however, may cause AS 33's customers to fall victim to the attack. Specifically, consider AS 22, a customer of AS 33 as well as of AS 54. AS 22 receives from both ASes 54 and 33 a legitimate prefix announcement for 1.2/16. However, AS 22 also receives from AS 54 the hijacked subprefix announcement for 1.2.3/24, since all ASes along this path (44-53-54) use BGP. Following the longest prefix matching, AS 22 will therefore route traffic to the subprefix 1.2.3/24 via AS 54, causing this traffic to be hijacked.

In ROV++ v3, we introduce an additional mechanism, called *preventive announcements*, whose goal is to "compete" with the hijack announcement, much like the blackhole announcements of v2, but do it also when there seem to be a route to the legitimate origin (i.e., no need to blackhole). In a preventive announcement, an ROV++ v3 AS, e.g. AS 33, sends a BGP announcement *for the hijacked subprefix*; the AS-path in this announcement, however, will be a path to the legitimate origin of the prefix that contains the hijacked subprefix. Since both the preventive and hijacked announcements are for the same subprefix, the preventive announcements may *prevent* some traffic interception.

This is illustrated in Fig. 4(b), where AS 33 deploys ROV++ v3. As before, AS 33 routes correctly the entire 1.2/16 prefix, via AS 87; however, now AS 33 also sends a *preventive announcement* to AS 22, with the subprefix 1.2.3/24 and the path 33-87-99 routing back to the authorized origin of the prefix. AS 22 also receives the subprefix hijack announcement from AS 54, but prefers the route 33-87-99, since it is shorter. As a result, its traffic reaches the correct origin (AS 99), and is neither hijacked nor blackholed.

For similar reasons as explained for blackhole announcements, preventive announcements are only generated when receiving a subprefix hijack announcement from peers or providers, and are only sent to customers. Additional mechanisms are included to constrain the sending of preventive announcements to avoid routing loops; see more details in §III. Since a preventive announcement is created by a ROV++ v3 AS, without actually receiving such an announcement from the origin, it will not work when certain path authentication mechanisms (e.g., BGPsec [37]) are in place; we discuss the pros and cons of ROV++ v3 in more detail in §VII-A.

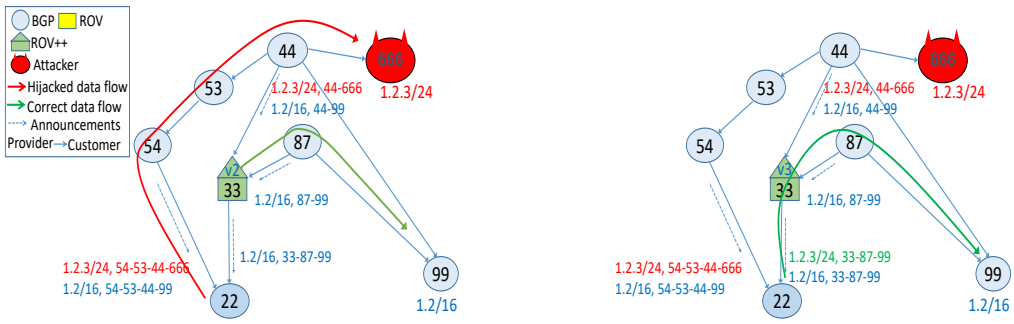


Figure 4. ROV++ v3: *Preventive announcements* help to route data to the legitimate origin. (a) Although AS 33 runs ROV++ v2, the traffic from AS 22 to 1.2.3/24 is still routed via AS 54, following the longest-prefix routing rule, and hence is hijacked. (b) When AS 33 runs ROV++ v3 and sends preventive announcements to AS 22, the traffic of AS 22 to 1.2.3/24 is routed via AS 33 (shorter path) and reaches the legitimate origin.

III. ROV++: Goals, Model and Design

In this section, we first describe the design goals of ROV++ and the attack model, then describe the detailed design of the ROV++ mechanisms.

A. Design Goals

The first goal is simple: the adoption of ROV++ by an AS should not cause hijack of traffic from that AS, beyond what would have happened if the AS used BGP or ROV.

Goal 1 (Do no harm 1: no hijack due to ROV++)**.** *If traffic from AS X to prefix p is hijacked when AS X deploys ROV++, then this traffic would also have been hijacked if X used plain BGP or ROV.*

The second goal considers the impact of one AS's adoption of ROV++ on hijacks of traffic from *other* ASes. It would have been great if we could extend the first goal, and require that adoption of ROV++ by any AS X would not cause hijack of traffic from *any* other AS. However, due the complexity of BGP relationships and Internet topology, this extended goal seems hard to achieve for any reasonable policy; indeed, it does not hold for ROV. Therefore, we set a more modest goal, which only requires improvement in the hijack *probability* for a random edge AS. We require this to hold for *any* rate of adoption, although we are especially interested in the impact in the early adoption phase.

Goal 2 (Prevent hijacks (on average))**.** *Let x be the percentage of ASes adopting ROV++, and let $P_{Hijack,SubP}(x)$ (resp. $P_{Hijack,NonR}(x)$) be the probability that traffic from a random edge AS is hijacked by a subprefix hijack (resp. non-routed prefix hijack). Then both probabilities should monotonically decrease in x. Furthermore, for any given x, $P_{Hijack,SubP}(x)$ and $P_{Hijack,NonR}(x)$ should be less than the corresponding values for ROV.*

The previous goal focuses on reducing the probability of hijacks. However, it does not necessarily translate to increase in the probability of successful connections to the correct origin, since some ASes may be disconnected from some prefixes. Indeed, ROV++ would often only be able to *prevent* hijack, but it may not be able to find a secure path to the correct origin. We therefore consider a more modest, no-harm goal: to have successful connection rates not much worse than those of ROV, and preferably better.

Goal 3 (Do no harm 2: maintain successful connections)**.** *Let $P_{ROV}(x)$ (resp. $P_{ROV++}(x)$) be the probability that traffic from a random edge AS reaches the correct origin during a subprefix hijack, where x is the percentage of the ASes that adopt ROV (resp. ROV++). Then, $P_{ROV}(x) \lessapprox P_{ROV++}(x)$ for all (or most) values of x.*

Finally, we mention two non-quantifiable yet important goals: simplicity and ease of deployment.

Goal 4 (Keep It Simple)**.** *ROV++ design should be as simple as possible.*

Goal 5 (Easy to Deploy)**.** *ROV++ design should be easy to implement and deploy.*

In the above, Goals 1 to 3 are security-oriented goals, while Goals 4 and 5 are system-oriented goals.

ROV++ meets the goals. In §IV, we prove that ROV++ achieves Goal 1 (see Corollary 1). Our simulation results in §V show that Goals 2 and 3 are also achieved. In §VI we present the Lite versions of ROV++, which meet the system goals. The "do no harm" property achieved by ROV++ is with respect to what is listed in Goals 1 and 3; ROV++ may lead to non-necessary disconnections (e.g., due to blackhole announcements in ROV++ v2).

B. The Prefix-hijack Attack Model

Many studies in cryptography and network security are designed against arbitrary attack strategies and often assume very powerful attack models such as MitM. ROV and ROV++ focus on the *prefix-hijack attack model*, a more specific attack model that allows only a restricted set of attacker operations. We next explain this attack model and justify why we adopt it for this study.

The prefix-hijack attack model considers an attacker who controls an AS. The attack method that the attacker exploits is to send misleading BGP announcements from this AS to other ASes with whom it has a connection (and relationship). The misleading BGP announcements incorrectly indicate the attacking AS as the origin of an IP prefix p, *not* assigned to

the attacker; the main evaluation metric is the amount of traffic to prefix p that the attacker succeeds in intercepting. We do not consider impersonation and MitM attacks since there are widely-deployed mechanisms to prevent them, e.g., IPsec and TLS.

The main reason that we do not consider other attacks such as *path manipulation* or *route leakage* is that, if not prevented, prefix hijacks are more effective than other attacks. Specifically, in other attacks, by definition, the attacker will *not* be the origin, and hence the announcement it sends will contain AS-path of length at least two, while a prefix hijack for the same prefix will send announcements with AS-path containing only the attacking AS. Since ASes give preference to announcements with shorter AS-path, the prefix-hijack announcements are more likely to attract the traffic to the prefix, compared to similar announcements with longer AS-path (e.g., due to path manipulation or route leakage). The fact that prefix hijack is a more effective attack was confirmed by simulations, e.g., in [15].

Within the prefix-hijack attack model, we focus on protecting prefixes protected by a ROA. Namely, we assume that an AS that deploys ROV or ROV++ can identify prefix/subprefix hijacks reliably and correctly through ROAs. This is a reasonable assumption given that misconfigurations of ROAs have become rare [14]; dealing with ROA failures is out of the scope of this paper.

C. Detailed Design

As illustrated in §II, a basic mechanism in ROV++ (all three variants) is to support data-plane blackholing and preference to safer paths. For this purpose, each ROV++ AS identifies and maintains *holes* for incoming announcements, defined as follows. Suppose an AS X receives a valid announcement *ann* and an invalid announcement *ann′*, both from the same neighbor, and *ann′* has a subprefix (i.e., *ann′*$.pre \subset$ *ann.pre*), then X marks that *ann* has a hole to indicate that if it uses the path in *ann* to route data to subprefix *ann′*$.pre$, then the data will be hijacked. Whenever possible, X chooses a path that has no hole (i.e., prefer safe path), and blockholes subprefix *ann′*$.pre$ if no safe path is available.

Specifically, X maintains a table of invalid announcements, denoted as $\mathcal{H}$. Each announcement *hole* $\in \mathcal{H}$ includes the prefix *hole.pre* and the neighbor AS, *hole.from*, from which *hole* was received. Let $\mathcal{A}$ denote the set of non-invalid announcements (i.e., classified as valid or unknown by ROV). Similarly, each announcement *ann* $\in \mathcal{A}$ has *ann.pre* and *ann.from*, with three additional attributes: *ann.rel* that represents the relationship between *ann.from* and X (i.e., peer, provider, or customer), *ann.path* that represents the AS-path, and *ann.hole* that represents the set of holes in *ann.pre*. Initially *ann.hole* is empty, and holes will be added to *ann.hole* once identified (see details below).

In ROV++, the incoming announcements can be standard BGP announcements (update, withdraw), blackhole announcements (in ROV++ v2 and v3), or preventive announcements (in ROV++ v3). In the following, we first describe the marking of blackhole and preventive announcements so that a ROV++ AS can identify them (If only v1 is used, this aspect is not needed). We then describe the handling of an incoming announcement.

Markings in blackhole announcements. A ROV++ AS *marks* a blackhole announcement, allowing subsequent ROV++ ASes to be aware that the traffic would be blackholed (dropped). Marking should be *transitive*, i.e., be propagated also by ASes which use ROV or plain BGP, to reach ROV++ ASes without requiring direct connections between ROV++ ASes. This can be done in different ways; the best may be to use an optional transitive BGP attribute [46]. A blackhole announcement is treated mostly like an invalid announcement (see details later).

Markings in preventive announcements. Preventive announcements are marked similarly so that they can be identified by ROV++ ASes, and not flagged as invalid and dropped by the ROV mechanism. For example, suppose AS 99 owns the IP address block 1.2/16, and the issued ROA has the maximum length of 16. Then a preventive announcement of 1.2.3/24 with the origin as AS 99 will be flagged as invalid (since 24 is larger than the maximum length in the ROA) if not marked; see the full version [43] on detailed examples that show the advantage of marking preventive announcements. A preventive announcement is treated mostly like a legitimate announcement (see details later).

Handling incoming announcements. Algorithm 1 summarizes how ROV++ handles incoming announcements. For clarity, it does not include the handling of withdrawal announcements, which are handled in the same manner as in BGP. Only the most essential procedures are defined in Algorithm 1; the others are defined in Appendix A. The description is for ROV++ v3, which includes all the three mechanisms in ROV++; the operations in ROV++ v1 and v2 are (mostly) a subset, as marked in Algorithm 1.

Consider an incoming announcement *ann*. We consider two cases. **(i)** The announcement *ann* is determined as *valid* or *unknown* by ROV, i.e., indicated by PASSROV(*ann*) as being true, or it is a preventive announcement. In this case, the router uses UPDATEHOLE($ann, \mathcal{H}$) to check whether any *hole* needs to be added to *ann*. After that, ann is added to $\mathcal{A}$, and the router uses DECIDEBESTROUTE to find the best route. If the best route is a new route (i.e., differs from the currently used best route), then the router needs to perform a sequence of actions, including removing blackhole, blackhole or preventive announcements that were set up or sent due to the old route, and actions for the new route. For the new route, if there is no alternative safe route for a subprefix, the router sets up a local blackhole for that subprefix; it further creates a blackhole announcement for that subprefix if it is received from a provider or peer, and sends it to customers (in ROV++ v2 or v3) or according to export policy (in ROV++ v2a). Otherwise, i.e., there is a safe route, the router decides whether to send a preventive announcement through SENDPREVENTIVEANN (to be described later). **(ii)** The announcement *ann* is an *invalid* or blackhole announcement. In this case, the router invokes UPDATEHOLE ($\mathcal{A}$, *ann*), which updates the holes for all the announcements in $\mathcal{A}$ based on *ann*. If the holes of an announcement $a \in \mathcal{A}$ have been updated, then the router uses DECIDEBESTROUTE to find the best route for prefix $a.pre$.

In the above, the procedure UPDATEHOLE takes two sets, one containing valid announcements whose holes need to be updated and the other containing one or more invalid announcements. In the procedure, each valid announcement a

Algorithm 1 ROV++ Handling Incoming Announcements.

```
1: procedure IncomingAnnoucement (ann)
2: if PassROV (ann) or IsPreventive (ann) then
3:    UpdateHole (ann, H)
4:    A = A ∪ ann
5:    DecideBestRoute (ann.pre)
6: else
7:    // invalid or blackhole announcement
8:    H = H ∪ ann
9:    UpdateHole (A, ann)
10:   if ∃a ∈ A s.t. a.hole has been updated then
11:      DecideBestRoute (a.pre)
12:   end if
13: end if
14: end procedure

15: procedure UpdateHole (annSet, holeSet)
16: for ∀a ∈ annSet and ∀b ∈ holeSet do
17:   if b.pre ⊂ a.pre and a.from = b.from then
18:      a.hole = a.hole ∪ b
19:   end if
20: end for
21: end procedure

22: procedure DecideBestRoute (pre)
23: Let u ∈ A be the currently used route for prefix pre; set
    to NULL if no route exists
24: a = FindBest (pre)
25: if a ≠ u then
26:   Use (a)
27:   if u.hole ≠ ∅ then
28:      for all hole ∈ u.hole do
29:         RemoveBlackhole (hole)
30:         // Only for ROV++ v2 and v3
31:         WithdrawBlackholeAnn (hole)
32:      end for
33:   else
34:      // Only for ROV++ v3
35:      for all hole ∈ H s.t. hole.pre ⊂ u.pre do
36:         WithdrawPreventiveAnn (u, hole.pre)
37:      end for
38:   end if
39:   if a.hole ≠ ∅ then
40:      for all hole ∈ a.hole do
41:         Blackhole (hole)
42:         // Only for ROV++ v2 and v3
43:         if IsParentOrPeer (hole.from) then
44:            SendBlackholeAnn (hole)
45:         end if
46:      end for
47:   else
48:      // Only for ROV++ v3
49:      for all hole ∈ H s.t. hole.pre ⊂ a.pre do
50:         SendPreventiveAnn (a, hole.pre)
51:      end for
52:   end if
53: end if
54: end procedure
```

is checked with each invalid announcement b. If $a.pre$ contains $b.pre$ (i.e., $b.pre \subset a.pre$), and the two announcements have the same upstream AS (i.e., $a.from = b.from$), then $a.hole = a.hole \cup b$.

The procedure FindBest applies a set of rules, in a priority order, to find the best route for a destination prefix. We use the following priority rules: first based on relationships (i.e., prefer the route from a customer over that from a peer over that from a provider), and then prefer the route with no (or least) holes. If there are still ties, prefer the route with shortest path. The rationale is that this order first considers the economic incentives of the ASes, and then security, and performance last.

The procedure SendPreventiveAnn decides whether to send a preventive announcement or not in ROV++ v3. This is a bit delicate; ROV++ v3 uses multiple measures to prevent unsafe paths from propagating in the network, which may harm convergence or cause loops. First, when a ROV++ v3 AS sends an announcement, it includes a transitive field to indicates whether the announcement contains holes or not, which can be recognized by other ROV++ v3 ASes. Second, an ROV++ v3 AS X only generates a preventive announcement about a subprefix *subp* if (i) X receives a subprefix hijack announcement *subp* from a provider or peer, and (ii) X receives a legitimate announcement *pre*, s.t. *subp* $\subset$ *pre*, that contains no holes (as indicated by the transitive field) from some neighbor AS Y that did not send *subp* to X. Third, X sends the preventive announcement only to customers from which it has not received *pre*, to prevent routing loops. Suppose a customer of AS X, say AS Z, receives a preventive announcement from X on *subp*, and later sends an announcement with prefix *pre* to X. Then X will withdraw the preventive announcement from Z, and Z will in turn withdraw it from other ASes.

D. Non-routed Prefix and Superprefix Hijacks

So far, the ROV++ design focuses on mitigation of subprefix hijacks. We now discuss minor extensions that have significant impact on other important hijack scenarios.

Non-routed prefix hijacks. These are hijacks on IP addresses that are normally *non-routed*, i.e., not part of any (legitimately) announced IP prefix. Somewhat surprising, there are plenty such IP addresses, typically owned by organizations that were allocated large address blocks in the early days of the Internet, and never got to utilize them. As shown in [60], non-routed prefixes are often hijacked, and then used to launch different attacks such as spam and DDoS. The main motivation of the attackers is to foil blacklists: once one such hijacked prefix is blacklisted, they just move to another one. Indeed, a significant fraction of the non-routed address blocks is blacklisted at any given moment, due to such attacks.

RPKI allows ROA to specify origin AS zero to signal a non-routed prefix so that ROV ASes will drop these announcements, and thereby not send packets to the non-routed prefixes, as desired. This completely prevents hijacking of packets sent by a ROV-deploying AS to non-routed prefixes. ROV also provides *some* defense to non-adopting ASes, but this defense is quite limited. We present the first evaluation of the defense that partial-adoption of ROV provides to non-adopting ASes against non-routed prefix hijacks (see §V), which shows a

significant improvement, almost linearly as adoption of ROV grows. However, we next explain how a simple adaptation of the ROV++ blackhole announcement mechanism provides much improved security (to non-adopting ASes).

ROV++: non-routed blackhole announcements. To better defend against non-routed hijacks, ROV++ sends *blackhole announcements* as follows. An AS that receives an announcement for a prefix with non-routed ROA (identified as origin AS being zero [27]) sends the corresponding blackhole announcement, following either the v2 or v2a rules. This announcement will now compete with the attacker's non-routable prefix hijack, preventing hijack of traffic from more non-adopting ASes. Our evaluation in §V confirms the benefits of this improvement.

In **superprefix hijack of non-routed prefix**, an attacker announces a *superprefix* of a non-routed prefix. Since the prefix is not announced, then packets whose destination IP address is in the non-routed prefix, will be routed following the superprefix announcement directly to the attacker. This attack circumvents ROV; if there is no ROA for the superprefix, which is the common case, then ROV will announce the superprefix, and the attack succeeds. For example, suppose prefix 1.2.3/24 is non-routed and protected by a ROA with origin AS 0. An attacker, AS 6, wants to hijack 1.2.3/24, and therefore announces superprefix 1.2.0/22. It is reasonable to assume that no other AS announces 1.2.0/22 and no corresponding ROA was published (due to concerns that issuing such a broad ROA may invalidate legit announcements of many prefixes that are not yet covered by ROAs [19]). Using ROV (and BGP), the announcement of 1.2.0/22 is likely to reach *almost all* ASes, including ROV ASes and their customers. As a result, AS 6 will successfully hijack *almost all* traffic sent to 1.2.3/24 or any other unannounced subprefix of 1.2.0/22.

ROV++ blackholing defeats superprefix hijack of non-routed prefix. ROV++ *always blackholes* all traffic sent to non-routed prefixes. This suffices to *completely foil* the superprefix hijack of non-routed prefixes. Further, this provides some benefits to non-adopting ASes; see evaluation in §V.

Extensions foiling reflection-DDoS. The fact that ROV++ blackholes traffic sent to non-routed prefixes foils many exploits of non-routed prefixes. In particular, it foils spam, since the spammer will not be able to establish the SMTP connection. However, this alone may not provide the best defense against the common threat of reflection-DDoS attacks. Improved defense can be achieved through another minor extension: the AS should also *blackhole* traffic whose *source IP* is in a non-routed prefix.

Superprefix+prefix hijack prefix. The superprefix attack may also be combined with a prefix-hijack attack, i.e., to hijack traffic sent to a prefix which was announced by its legitimate origin AS, and protected by a ROA. The goal would, again, be to reduce the effectiveness of ROV to defend against hijack—in this case, to protect against prefix hijack. Let us explain the attack using an example. As the name implies, the attacker sends a prefix hijack announcement, say for prefix 1.2.3/24, but *also* announces the superprefix, say 1.2.0/22. As before, we assume that 1.2.0/22 is not covered by a ROA. Consider, for simplicity, a stub AS that uses ROV, and assume this AS received the hijack announcement for 1.2.3/24; due to the use of ROV, this announcement is dropped. However, like almost all ASes, this AS will almost surely receive the superprefix (1.2.0/22) announcement. As a result, the AS will send packets with destination IP in 1.2.3/24 to its provider, and from there, these packets will be routed to the attacker, since the provider is using the hijacked announcement.

Defenses. The blackhole mechanism makes ROV++ adopting ASes immune to this improved attack. It is possible to foil the superprefix attacks by issuing non-routed ROAs (origin AS set to zero) for superprefixes of ROA-protected prefixes. However, this may cause any unknown prefix covered by the superprefix to be considered invalid, and hence needs to be applied carefully.

IV. Security Analysis

It is challenging to identify the exact properties of routing security mechanisms, including ROV++, especially under partial deployment. We present some preliminary results, and leave further analysis for future work.

As mentioned earlier, hidden hijacks have significant implications on ROV and ROV++. We next formally define *hidden hijacks* and *visible hijacks*, and then compare the extent of hidden/visible hijacks when deploying ROV++ versus ROV, and show that ROV++ achieves better security than ROV and BGP.

Definition (visible and hidden hijacks). Suppose that an AS A_0 selects the path $(A_1, \ldots, A_n)$ for routing packets for the entire prefix *pre* (no subprefixes). If A_0 sends a packet to an IP address $x \in$ *pre*, and this packet does not reach A_n, then we say that it was *hijacked*. If A_0 received from A_1 an announcement for a subprefix *subp* $\subset$ *pre* with an AS-path that does not terminate in A_n, we say this was a *visible hijack*; otherwise, we say this was a *hidden* hijack. See examples of hidden hijacks in Fig. 1. An example of visible hijack is in Fig. 2(a). There, AS 77 receives two announcements from AS 44: one with path $(44, 99)$ for prefix $1.2/16$, which if valid and selected by AS 77, and one with path $(44, 666)$ for subprefix $1.2.3/24$, which is invalid and dropped. However, the traffic from AS 77 to $1.2.3/24$ is still hijacked, which is a visible hijack since AS 77 received the subprefix hijack announcement from AS 44.

Hidden hijacks in ROV and ROV++. As shown in §II, ROV ASes are vulnerable to visible subprefix hijacks. In addition, since ROV ASes drop illegitimate subprefix announcements, it can *create* hidden hijacks, i.e., causing other ASes to fall victim to hidden hijacks; see one example in Fig. 1(b), where AS 44 will suffer from hidden hijacks because ROV AS 78 drops the subprefix hijack announcement. This is a significant weakness of ROV, and a concern for mixing ROV with ROV++ and other defense mechanisms, including BGP hijack detection services and the Artemis defense [51].

In contrast, ROV++ does not suffer from visible subprefix hijacks. This is by design—even in ROV++ v1—an ROV++ AS either blackholes the subprefix or uses an alternative safe path (if available) to prevent visible subprefix hijacks. Therefore, *all subprefix hijacks in ROV++ are hidden hijacks*. In addition, ROV++ ASes will never create hidden subprefix hijacks. These properties are formally stated below; we assume

the relevant RPKI ROAs are correct, i.e., if PASSROV(ann) is invalid, then ann is a prefix or subprefix hijack announcement

Lemma 1. *Consider a ROV++ AS X that receives a* visible *hijack announcement ann, i.e.,* PASSROV(ann) *is invalid, from neighbor ann.from. Then X will not send packets to ann.from for any packet whose destination IP is in ann.pre.*

The lemma follows from the design of ROV++ (all variants). Note that if a stub AS receives a visible hijack for some prefix *pre*, then it would not send any traffic to *pre*, i.e., become disconnected from it (rather than allow the traffic to be hijacked).

We next show that ROV++ achieves Goal 1, if we do not have two concurrent hijacks of different sub-prefixes of the same prefix.

Corollary 1 (Goal 1 is achieved, for a single hijack)**.** *If traffic from AS A_0 to prefix pre is not hijacked when A_0 is using BGP or ROV, then traffic from A_0 to pre will also not be hijacked if A_0 uses (any variant of) ROV++, provided that at most one subprefix of pre is hijacked.*

Proof: Consider any IP address $x \in$ *pre*. Let A_1 be the AS to which A_0 is forwarding packets with destination x, when A_0 uses BGP/ROV. If, when running ROV++, A_0 is also forwarding packets to x, or A_0 cannot route packets to x, the claim follows trivially. Hence, assume that when A_0 runs ROV++, it sends packets whose destination is x to another neighbor, $A_1' \neq A_1$. In all versions of ROV++, this happens only if A_0 detects hijacking of a subprefix *subp* $\subset$ *pre*. If $x \in$ *subp*, then x *would* be hijacked if A_0 used ROV/BGP. Otherwise, then x is not in any hijacked (sub)prefix, since we assumed at most one subprefix of *pre* is hijacked; hence, x must be routed correctly also when A_0 uses ROV++. □

The claim and proof also extend to the ROV++ Lite versions, presented in §VI, even without the restriction to a single hijack. The proof is omitted due to length restrictions.

Hidden hijacks in BGP and ROV++. Interestingly, hidden hijacks can also occur in BGP-only networks, where an AS that uses BGP does not receive a subprefix hijack announcement from any neighbor, yet the data traffic to the subprefix is hijacked; see one example in Fig. 1(a). In a BGP-only network, a subprefix hijack announcement will normally reach almost all the ASes, leading to visible subprefix hijacks; but some (usually few) ASes may not receive the subprefix hijack announcement (e.g., due to valley-free routing as in Fig. 1(a)), and suffer from hidden subprefix hijacks. Our analysis (see full version [43]) shows that BGP ASes generally make subprefix hijacks visible by forwarding the subprefix announcement, hence the hidden hijack rate in BGP-only network is low (the visible subprefix hijacks rate is, however, close to 100%). In contrast, ROV can create hidden hijacks, leading to higher hidden hijack rate.

ROV++, unlike ROV and BGP, never *creates* hidden hijacks. However, ROV++ cannot completely prevent hidden hijacks, since an AS A_0 cannot detect that traffic may be hijacked when routed via a neighbor A_1, if A_0 did not receive a (visible) hijack announcement for a subprefix *subp* $\subset$ *pre* from A_1. Furthermore, unlike BGP, ROV++ never outputs a visible hijack—and, as a result, may quite often *propagate hidden hijacks*, i.e., output a hidden hijack—when receiving a hidden hijack announcement. This happens when the hidden hijack announcement is selected, which may happen quite often, esp. considering ROV++ "prefers" it, hoping it does not result in hijack. This causes an interesting phenomenon in the evaluations, where hijacking from ROV++ ASes somewhat increases as adoption increases (up to about 40%), due to the higher proliferation of (BGP-induced) hidden hijacks. This is more pronounced in ROV++ v3, where an adopting AS that has selected the hidden hijack announcement will use the hijacked path to generate preventive announcements. Any neighbor that uses this preventive announcement will then be subject to hidden hijack.

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of ROV++ in comparison with ROV using simulation. We first describe the simulation setup, and then present the results.

A. Simulation Setup

We evaluate the performance of the multiple ROV++ variants and ROV using a simulator that we developed. This simulator provides detailed simulation of BGP announcement propagation, including both update and withdrawal of announcements. It further incorporates simulation of route validation, best path decision making and export policy.

All evaluation uses an empirically-derived Internet-scale AS topology (each edge marked with an AS relationship) from CAIDA [10] (using the serial-2 dataset for July, 2020). For an AS that adopts plain BGP or ROV, the best path decision making follows three standard rules in order: (i) local preference (i.e., prefer announcements from customers, over those from peers, and those from providers), (ii) shortest path, and (iii) breaking ties randomly. For a ROV++ adopting AS, preference to routes with no blackholes is placed right after local preference, so that the economic interests of the AS is still considered first. The routing follows the commonly assumed valley-free routing [18].

For each ROV or ROV++ policy, we assume a certain percentage of ASes adopts the policy, while the others run plain BGP. The percentage of adoption varies from 0%-100% for a given policy[1]. For each setting, we run over 2000 trials, each with a random attacker and victim from the edge ASes, and present the average results with 95% confidence intervals. We consider three categories of ASes: top 100 ASes (obtained from ASRank [11] on July 23rd, 2020), edge ASes (those without customers or peers), and the other ASes (those that are not among top 100 or edge ASes). The results we present use uniform adoption among these three categories of ASes, as in most previous works, although we also tested results for non-uniform adoption; results were quite similar.

We focus on two types of hijacks: (i) subprefix hijacks, where a prefix, say $1.2/16$, is announced by its legitimate origin, and an attacking AS announces itself as the origin of a subprefix of $1.2/16$, say $1.2.3/24$; attacker and victim pairs are randomly chosen from the edge ASes, and (ii) non-routed

[1] In 0% adoption, only the tested AS adopts the policy; in 100% adoption, all but the tested AS adopts the policy. This treatment allows us to present the trend for adopting and non-adoption ASes separately.

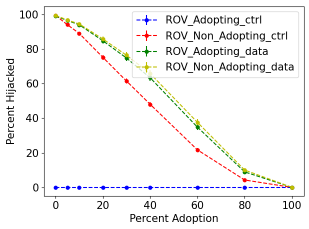


Figure 5. Subprefix hijack rates in control and data planes.

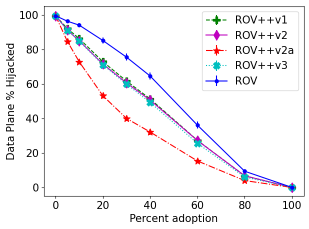


Figure 6. Subprefix hijack: overall hijack rate.

prefix hijacks, where we consider two cases, one assuming that an attacker announces a non-routed prefix, say $1.2.3/24$, that is protected by a ROA with origin AS 0, and the other assuming that the attacker announces a superprefix, say $1.2.0/22$ (i.e., superprefix hijack of non-routed prefix, see §III). No other AS announces the prefix/superprefix.

For subprefix hijacks, the performance metrics include (i) *hijack rate*, i.e., the percentage of ASes whose traffic to the subprefix is hijacked by the attacker, (ii) *successful connection rate*, i.e., the percentage of ASes whose traffic to the subprefix is routed successfully to the victim, and (iii) *disconnect rate*, i.e., the percentage of ASes whose traffic to the subprefix is dropped, and routed to neither the victim nor the attacker. For non-routed prefix hijacks, the performance metrics is only the hijack rate, since no victim is supposed to receive such prefixes; the disconnect rate is simply the complementary of the hijack rate.

B. Data Plane vs. Control Plane Results

Before presenting the results, we first highlight the importance of quantifying the various performance metrics based on the data plane (i.e., the actual paths for the data packets), instead of the control plane (i.e., the absence or presence of a subprefix/prefix in the routing table). Fig. 5 shows an example. It plots the hijack rate when 0 to 100% of the ASes adopt ROV under subprefix hijacks. For adopting ASes, the control-plane hijack rate is always zero, since all these ASes drop the hijacked subprefix; the actual data-plane hijack rate, however, begins at essentially 100% and then decreases with adoption rate; this is due to data-plane hijacks, as described in §II-A. Also for non-adopting ASes, data-plane hijack rates are significantly higher (worse) than the control plane hijack rate.

The above example demonstrates that results in the control plane can lead to significant underestimation of hijacks. Hence, *all the results* in the rest of this section are obtained from the *data plane*, which provide accurate characterization of the actual performance of ROV and various ROV++ policies.

C. Subprefix Hijacks

For subprefix hijacks, we first present the overall hijack rate for all ASes, with different percentages adopting (and the rest non-adopting). We then present the results for adopting and non-adopting ASes separately. For adopting ASes, the results quantify the security benefits from adopting the policy, while for non-adopting ASes, the results quantify the collateral benefits such ASes obtained from the adopting ASes. The results for both cases are important—a policy is more likely to see wider adoption when it provides higher security benefits to the adopting ASes and higher collateral benefits to the non-adopting ASes. For all the above, we obtain the results for the three categories of ASes, i.e., top 100 ASes, edge ASes and others. In the interests of space, we only present the results for the edge ASes, which account for over 75% of the ASes, and used by most hosts; the results for the other two categories show similar trends and are found in Appendix B.

Overall hijack rate. Fig. 6 plots the average hijack rate over all the edge ASes (including both adopting and non-adopting ASes) versus adoption rate. The results show that on average, the hijack rate for all variants of ROV++ decreases with the adoption rate, achieving Goal 2 for subprefix hijacks. We further observe that all three ROV++ variants (v1-v3) have similar hijack rates, all significantly lower than that of ROV for low to medium adoption rate. As expected, ROV++ v2a is more aggressive in mitigating hijacks (at the cost of significantly higher disconnection rate, as we show below). We next show the detailed results for adopting and non-adopting ASes, respectively.

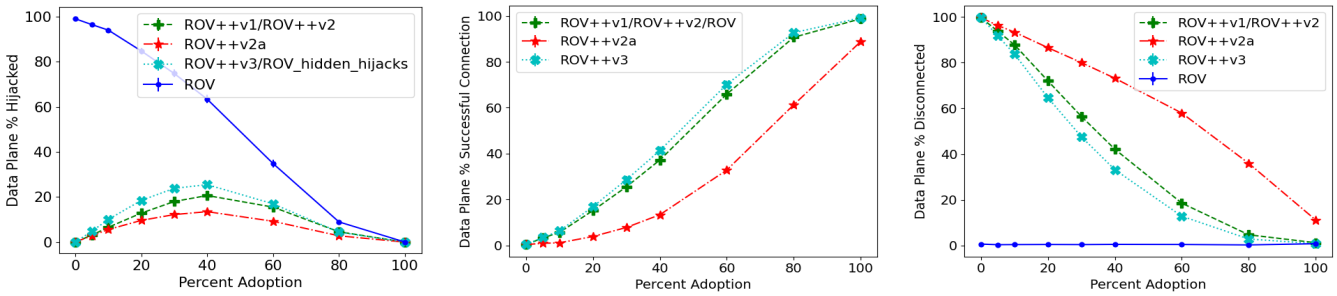
Results for adopting ASes. Fig. 7 presents the results for the adopting ASes. We see from Fig. 7(a) that the hijack rate of ROV is much higher than that of the ROV++ variants, particularly under low adoption rate: with 10% adoption, the hijack rate for edge ASes of ROV is still above 94%, vs. less than 10% for ROV++ variants. As expected, ROV++ v2a has the lowest hijack rate, followed by ROV++ v1 and v2 (which have very similar hijack rate). ROV++ v3 has slightly higher hijack rate than the other ROV++ variants.

Counter-intuitively, the hijack rates for ROV++ adopting nodes gradually *increase* until reaching a peak at about 40% adoption, and only then gradually decrease. This surprising "bump" is due to hidden hijacks, as explained in §IV.

Notice that hidden-hijacks behave in a similar manner also for ROV. This cannot be seen directly from the blue line in Fig. 7(a), representing hijack rate for ROV, since that line includes also many visible hijacks. However, we also measured separately the number of hidden hijacks in ROV, and the result is also shown in Fig. 7(a). In fact, the result was so close to that of ROV++v3 (not visually distinguishable) that we use the line for ROV++ v3 to represent also the hidden hijack rates for ROV.

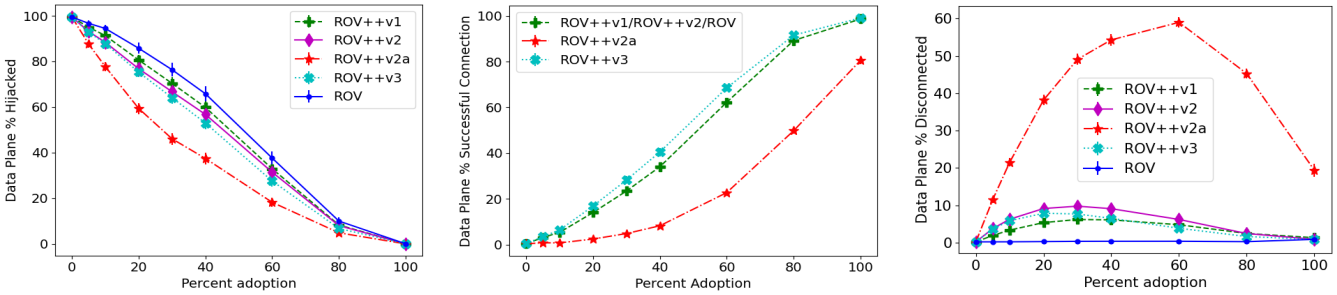
The successful connection rate of all ROV and ROV++ policies (see Fig. 7(b)) increases with the adoption rate. As expected, ROV++ v3 has the highest successful connection rates, due to its preventive announcement mechanism. However, the improvement is not that significant, and we doubt it would justify the use of v3, which is more complex and has somewhat higher hijack rates. Close behind are the successful connection rates of ROV, ROV++ v1, and ROV++ v2, all of which are so close that we present them by a single line (of v1, but one really cannot tell them apart anyway). ROV++ v2a definitely has significantly inferior successful connection rates, and therefore, does not appear to be a good choice to protect against subprefix hijacks.

Fig. 7(c) shows the disconnection rates, which are simply the complement of the hijack rates plus the successful connection rates. Considering that the goal is to have low hijack rates *and* high successful connection rates, it seems that the disconnection rate alone should not be a measure to compare



(a) Hijack rate (b) Successful connection rate (c) Disconnect rate

Figure 7. Subprefix hijacks: Data plane results for adopting edge ASes with 95% confidence intervals (a curve marked with multiple policies presents the result for the first policy; the others are visually indistinguishable and hence not plotted for clarity).



(a) Hijack rate (b) Successful connection rate (c) Disconnect rate (y-axis up to 60%)

Figure 8. Subprefix hijacks: Data plane results for non-adopting edge ASes with 95% confidence intervals (a curve marked with multiple policies presents the result for the first policy; the others are visually indistinguishable and hence not plotted for clarity).

the mechanisms. Not surprisingly, ROV++ v2a has the highest disconnection rates, and ROV has almost no disconnections (since it has many hijacks). The results of ROV++ v1 and v2 are very close (indeed visually indistinguishable), while v3 has slightly lower disconnection rates.

Results for non-adopting ASes. Fig. 8 presents the results for non-adopting ASes. For all ROV and ROV++ policies, the hijack rate decreases with increased adoption, showing collateral benefits, i.e., the non-adopting ASes benefit from the adopting ASes; however, the advantage is not very significant, definitely much less than for adopting ASes. The different ROV++ variants have quite close hijack rates, except v2a, which has significantly lower (better) hijack rates, but that comes at the cost of very significantly reduced successful connection rates. Excluding ROV++ v2a, it is clear that ROV++ v3 is the best for non-adopting ASes, having both the highest successful connection rates and the lowest hijack rates, but not by much, esp. when compared to ROV++ v1 and v2 (it has a bit more advantage over ROV).

We see from Figures 7(b) and 8(b) that ROV++ v2 has similar successful connection rates as v1, for both adopting and non-adopting ASes, indicating that although the blackhole announcements in ROV++ v2 can potentially compete with legitimate prefix announcements, causing failed delivery of traffic to the legitimate destination, this rarely happens based on our extensive evaluation. ROV++ v2a, on the other hand, leads to much lower successful connection rate than v1 due to its aggressive propagation of blackhole announcements.

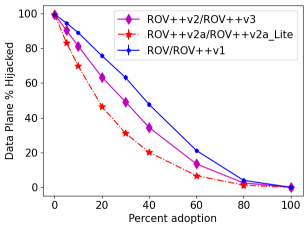


Figure 9. Prefix hijack on non-routed prefix: Data plane hijack rate for *non-adopting* edge ASes. The results for ROV and ROV++ v1 overlap; the results of ROV++ v2 and v3 overlap, and are the same for the Lite versions. The hijack rate for ROV and ROV++ (all variants) *adopting* ASes is zero (figure omitted).

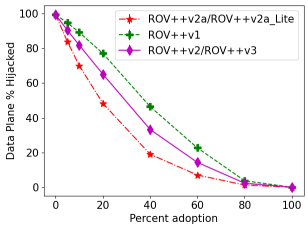


Figure 10. Superprefix attack on non-routed prefix: Data plane hijack rate for *non-adopting* edge ASes. Results are for the case where attacker announces *both* the non-routed prefix and its superprefix, and are the same for the Lite versions. ROV++ *adopting* ASes have *no hijacks*; for ROV, *all traffic is hijacked* for adopting and non-adopting ASes (figures omitted).

D. Hijack of Non-routed Prefixes

We now present the results for hijacking of non-routed prefixes. Our evaluation is in two cases: (i) the classical *prefix-hijack attack on a non-routed prefix*, where the attacker announces the non-routed prefix, and (ii) the novel *superprefix attack on a non-routed prefix*, introduced in §III-D, where the attacker announces a superprefix of the non-routed prefix. The results were essentially the same whether the attacker also announces the non-routed prefix itself of not.

Prefix-hijack attack on a non-routed prefix. In this type of hijack, the hijack rate is zero for all adopting ASes—for

both ROV and all ROV++ variants. This is because adopting ASes drop the non-routed prefix, and hence have no route to the (non-routed) prefix—i.e., the attack fails—adopting ASes are always disconnected (as desired to foil abuse of non-routed prefixes).

Hence, we focus on non-adopting ASes; Fig. 9 plots the hijack rate for the non-adopting edge ASes. For ROV, the results show roughly linear decrease in hijack rate with increased adoption; ROV++ v1 has *identical* results as ROV, since in this case, there is no need to blackhole packets as the adopting AS would not have any route to the (non-routed) prefix. ROV++ v2 has noticeably lower hijack rates than v1, due to the blackhole announcement mechanism, and v3 has identical results as v2, since there are no valid routes to the prefix to begin with—preventive announcements are irrelevant. As expected, ROV++ v2a has the best (lowest) hijack rates, due to its aggressive propagation of blackhole announcements. We also confirmed that the Lite version of ROV++ v2 and v2a, presented in §VI, has visually indistinguishable results as their non-lite version; we mention this explicitly (and in the legend of the figure), since this result is significant to our recommendations later on.

Summarizing the results for adopting and non-adopting ASes, the overall hijack rate for all variants of ROV++ decreases with the adoption rate (figure omitted), achieving Goal 2 for non-routed prefix hijacks.

Superprefix attack on a non-routed prefix. In this type of hijack, introduced in §III-D, the attacker announces a superprefix of the non-routed prefix, and the superprefix is not covered by a ROA. As explained in §III-D, ROV fails to block this attack, since almost all ASes would have the superprefix path and there is no blackholing of packets.

The result is completely the opposite for ROV++ adopting ASes, since they all *blackhole* traffic to (and from) non-routed prefixes, or, more precisely, prefixes covered by a ROA with origin AS zero (indicating being non-routed). Note that traffic to any subprefixes with valid ROA is not blackholed. Hence, there are absolutely *no hijacks* from ROV++ adopting ASes.

It remains to consider the results for non-adopting ASes, with different adoption rates of ROV++ (as explained, ROV adoption is completely useless). Fig. 10 shows the results for a variant of the attack, where the attacker also announces the non-routed prefix. The non-routed prefix announcement causes ROV++ ASes, except v1, to send blackhole announcements, which reduce hijack rates, most effectively for v2a. However, the attacker is better off sending *only* the superprefix, in which case the results for all ROV++ variants become as shown for v1. In both cases (i.e., whether the attacker sends or not sends the non-routed prefix), the Lite versions will have exactly the same results, since there are no "holes" as in subprefix hijacks (see §VI).

E. Summary of Main Results

To summarize, ROV++ improves dramatically upon ROV, for both subprefix and non-routed prefix hijacks, especially with low adoption. We observe tradeoff among the various ROV++ variants. For subprefix hijacks of routed prefixes, ROV++ v1 achieves similar performance as ROV++ v2/v3, while is simpler and easier to implement and deploy; ROV++ v2a has much lower hijack rate than the other variants at the cost of significantly lower successful connection rate. For hijacks involving non-routed prefixes, the blackhole announcement mechanism introduced in v2 leads to significantly lower hijack rate for non-adopting ASes, esp. for v2a.

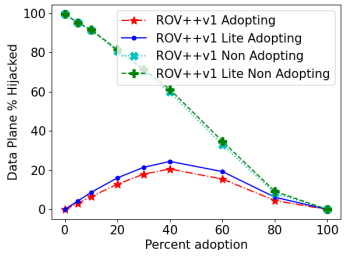


Figure 11. Subprefix hijacks: hijack rate of ROV++ v1 Lite and ROV++ v1: differences between Lite and "regular" are very small. We only show results for hijack rates, since for successful connection and disconnection rates, there is no visible difference between the Lite and "regular" versions.

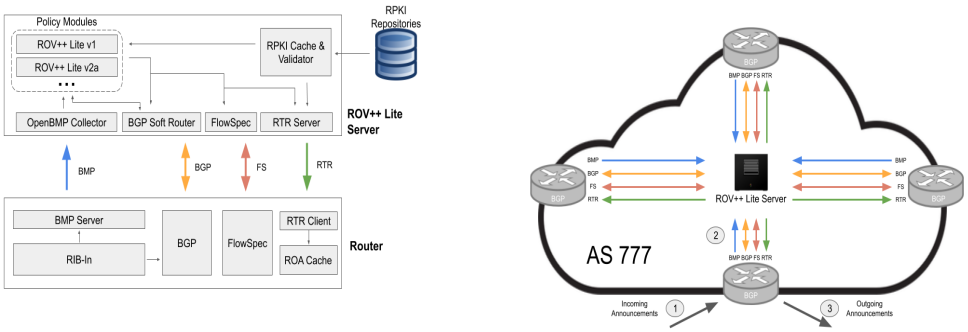
The above observations focus on the performance of the ROV++ variants. In §VII, we further discuss other issues (e.g., susceptibility to abuse, ease of deployment, incentive of deployment) and provide recommendations on which ROV++ variant(s) to choose for subprefix hijacking and non-routed prefix hijacking, respectively.

VI. Implementation, Deployment and ROV++ Lite

The blackholing, blackhole announcement and preventive announcement mechanisms in ROV++ are easy to implement on existing routers. The preference to safe paths (included in all ROV++ variants), however, requires changes to the BGP path selection process, which requires changes to BGP routers and possibly also the configuration parameters and scripts used by network operators. We design *ROV++ Lite* that removes this preference mechanism for easy implementation and deployment, while only leads to slightly degraded performance compared to its non-lite counterpart as we show below.

ROV++ Lite: a plug-and-play design. ROV++ Lite differs from ROV only in that it "gives up" the ROV++ safe-path preference mechanism. Specifically, it uses the standard BGP best path selection process. If the selected path contains holes, then it sets up data-plane blackholing, which is ROV++ Lite v1. ROV++ Lite v2 and v3 further include the blackhole announcement mechanism, and ROV++ Lite v3 further includes the preventive announcement mechanism. All Lite versions can be implemented on legacy BGP routers (see below).

Performance of ROV++ Lite. For non-routed prefix hijacking, the performance of the Lite version is identical to that of the non-Lite version (for all variants), since the safe-path preference mechanism is irrelevant for such hijacks. For subprefix hijacking, our evaluation shows that the Lite version is very much on par with the non-lite version. Fig. 11 shows one example by comparing the hijack rate of ROV++ v1 Lite and the non-lite version for both adopting and non-adopting ASes. We see that the Lite version only has slightly higher hijack rate than the non-Lite version for adopting ASes; for non-adopting ASes, their performance is almost identical. We observe similar results for successful connection rate and disconnect rate (figures omitted).



(a) ROV++ Lite server design

(b) Deployment of ROV++ Lite server

Figure 12. The design and deployment of ROV++ Lite server for implementing ROV++ Lite.

ROV++ Lite Implementation. ROV++ Lite can be deployed in any AS with legacy BGP routers, assuming only several standard mechanisms available in most BGP routers: ROV, BMP [50], and BGP FlowSpec [40]. We implement ROV++ Lite through a ROV++ Lite server as shown in Fig. 12. A ROV++ Lite server, deployed in an AS, can communicate with multiple routers in the AS.

The Lite server needs to be aware of any incoming hijack announcement to take the corresponding countermeasures, beginning with blackholing (already from v1). To achieve this, it obtains all the incoming BGP announcements to the routers through **BMP** [50]. Next, the Lite server needs to identify invalid announcements (i.e., suspected hijacks). This is achieved similarly to the regular ROV implementation[2]. Namely, the Lite server receives updated RPKI information (ROAs and RCs) from the standard RPKI repositories, and then caches and validates them. We use the RIPE implementation, and the **RTR** protocol to set the ROV rules in the routers.

After identifying a hijack, the Lite server applies the ROV++ policy and countermeasures: blackholing (from v1), blackhole announcements (from v2) and preventive announcements (from v3). In particular, to implement blackholing, it uses **BGP FlowSpec** [40]. Listing 1 shows an example of blackholing prefix 1.2.3/24 using FlowSpec on JunOS, by specifying the action for the matching prefix as **discard**. A similar set of commands can be used for Cisco IOS.

Listing 1. Blackholing with FlowSpec on JunOS

```
edit routing-options flow route BLK_PRE
edit match
set source 1.2.3.0/24
set then discard
commit
```

To issue blackhole announcements (from v2) and preventive announcements (v3), the Lite server uses a software BGP router that is part of the Lite ROV++ server package. Note that blackhole announcements are merely BGP announcements with additional flag information posted in the transitive attribute section of a BGP announcement. Reading the flag from the transitive attribute, the Lite server can execute the blackhole specified using the commands in Listing 1.

[2]Our implementation supports optional non-standard mechanisms to identify hijacks, but these are not related to ROV++ and therefore out of scope.

Table I. SUMMARY: PROS, CONS AND OUR RECOMMENDATION.

	Subprefix hijacking	**Non-routed prefix hijacking**
v1	**recommended:** performance on par with v2 and v3, not susceptible to abuse	**not recommended**
v2	**not recommended:** seems like propagating hijack, deployment more complex than v1, while performance comparable to v1	**recommended,** lower (better) hijack rate than v1, aligned with economic incentives
v2a	**not recommended:** has much lower successful connection rate, albeit lower hijack rate	**recommended**, but hard to deploy, albeit significantly better (lower) hijack rate
v3	**not recommended:** violates BGP ann. mechanism, has risk of abuse by attacker, while performance comparable to v1	**not recommended,** higher complexity, performance same as v2, risk of abuse
Lite versions recommended Comparable results, much easier to deploy.		

Feedback from Network Operators. We are currently working with network operators to get feedback about our design and implementation, to understand their concerns and desires from a system such as ROV++, and to coordinate deployment. The network operators with whom we communicated have shown keen interest in the potential of ROV++ to defend against DoS attacks, a major concern of many network operators. This seems to provide an incentive for operators to deploy ROV++, particularly the blackholing mechanism, that does not exist in ROV, while requires little effort to implement and deploy. Overall, the responses from the network operators are positive, and we hope that this would indeed translate to experimental (and later, hopefully, long-term) deployments. We will continue to work with operators to improve our design, install and deploy ROV++ Lite, and receive more feedback. We are very grateful for their cooperation.

VII. RECOMMENDATION AND DISCUSSION

A. ROV++ Variants: Pros, Cons and Recommendations

We next summarize the pros and cons of the various ROV++ variants (both non-Lite and Lite versions), and present our recommendation on which variant(s) to adopt for routed and non-routed prefix hijacks, respectively. The summary is in Table I.

• **Lite versions recommended.** We recommend the easier-to-deploy Lite versions (§VI), since in our experiments, they were on par with the "regular" versions; e.g., see Fig. 11.

• **Subprefix hijacking. (i)** For subprefix hijacking, we recommend *only* using ROV++ v1. This is because, as shown in Figures 7 and 8, ROV++ v1 provides similar performance

as ROV++ v2/v3, but is simpler and easier to implement. **(ii)** We do not recommend ROV++ v2 or v3, since their hijack rate is only slightly lower than that of v1, and they are harder to deploy and implement; also, the blackhole announcement in v2 can attract legitimate traffic, causing unnecessary disconnection. Furthermore, there may be concerns about abuses and unexpected side-effects of the blackhole and (even more) preventive announcements. **(iii)** We do not recommend ROV++ v2a because of its significantly lower successful connection rate than other variants.

• **Non-routed prefix hijacking. (i)** We recommend ROV++ v2a for non-routed prefix hijacking (including the novel superprefix hijacking). This is because it has the lowest hijack rate; its aggressive blackhole announcement propagation does not cause adverse impact since the prefixes are non-routed (i.e., should not be announced at all), and hence there is no risk of competing with legitimate traffic. On the other hand, a v2a AS will send blackhole announcements to its providers and peers, and then drop the attracted traffic, which will lead to deployment challenges (the v2a AS attracts traffic from its provider and peers to prevent them from being hijacked, while not being compensated for the traffic, since it has to drop the traffic instead of forwarding the traffic to its customers). **(ii)** Considering both performance and deployment incentives, we also recommend v2. This is because v2 outperforms v1 and ASes may have more economic incentives to adopt v2 than v2a (v2 ASes only send blackhole announcements to their customers, and will be compensated for attracting and dropping the traffic from their customers).

Our recommendation above considers multiple aspects, including ease of deployment, risk of abuse, performance and deployment incentives. In general, routing is a *social* operation: when an AS sends routing announcements, there is potential impact on other ASes. It is therefore necessary to carefully consider the potential impact of any proposed change in routing, and ensure that it meets the *do no harm* principle. We discuss ethics of routing security in more detail in the full version [43].

B. Mixed Deployments of ROV and ROV++ Variants

We next briefly discuss how ROV++ interacts with ROV, and how the three ROV++ variants interact with each other.

• ROV may often result in hidden hijacks, preventing ROV++ ASes (all versions) down the path from receiving hijack announcements and hence damaging their capability in mitigating the attacks. In contrast, doing *any* version of ROV++ and then ROV reduces the risk of hidden hijacks due to the ROV ASes (they may select announcements created by ROV++ v2 or v3, which will not create new hidden hijack), reducing the harm from ROV.

• Mixing higher versions of ROV++ with lower versions can reduce the impact of the higher versions. As an example, if the path contains a ROV++ v1 AS followed by a v3 AS, then the v3 AS may not be aware that the v1 AS blackholes traffic to the subprefix, and send a preventive announcement with the v1 AS in the path, and the traffic is blackholed (while otherwise there may have been a path allowing delivery); the v3 AS may also not be aware of the hijack at all, and hence does not send a preventive announcement.

• ROV++ never introduces hidden hijacks in all mixed deployment scenarios, including mixed deployment of ROV++ and ROV, and mixed deployment of multiple ROV++ variants.

VIII. Conclusions and Future Work

We presented and evaluated ROV++, an improvement of the ROV standard. ROV++ extends ROV by including several mechanisms with different impacts and tradeoffs. Extensive evaluation based on detailed simulations demonstrates that ROV++ significantly improves upon ROV, significantly reducing hijack rates for attacks on both subprefixes and non-routed prefixes, and especially with low adoption. We further developed and evaluated ROV++ Lite versions, which can be easily deployed on existing routers. Last, we discussed the pros and cons of the various ROV++ variants, and provided recommendations for deployment.

As future work, we are exploring in the following three directions: (i) develop techniques to further reduce the impact of hidden hijacks, (ii) explore scenarios with mixed deployments of ROV and ROV++, (iii) explore other ROV extensions that will provide defense even when adoption happens only, or almost only, at the edge ASes, and (iv) evaluate the performance of ROV++ under more realistic simulation settings, e.g., by taking account of real-world routing decisions, instead of only using valley-free routing, and considering the current deployment locations of ROV (e.g., as reported in [57]), instead of or in addition to random locations.

We hope that our study raises the awareness of the security limitations of ROV and motivates further studies addressing these limitations. A collective effort of the community can lead to a stronger foundation to continue building the momentum of RPKI adoption and improve inter-domain routing security.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful and constructive feedback, and our shepherd Dr. Samuel Jero for his helpful guidance. We also thank Joel Halpern, Shuai Hao, John Kristoff, Louis Poinsignon, Lars Prehn, Kotikalapudi Sriram, Celia Testart, and Russ White for their comments and suggestions on earlier drafts of the paper. Our collaborating partners, Connecticut Education Network (Rick Cheung, Ryan Kocsondy, and Michael Pennington) and UConn IT Services (Robert Kent, Michael Mundrane, and Michael Williams) provided us valuable feedback on deployment and operation aspects of this project. We are grateful for the great work by the project team at UConn, including Jack Aaron, Abhinna Adhikari, Matthew Jaccino, Sam Kasbawala, Shariq Khan, Pablo Rodriguez, Sam Secondo, Nicholas Shpetner, and Tony Zheng. This work was partially supported by NSF under award OAC-1840041 and by the Comcast Corporation. The opinions expressed in the paper are those of the researchers and not of their university or funding sources.

References

[1] AT&T/as7018 now drops invalid prefixes from peers. https://seclists.org/nanog/2019/Feb/140.

[2] Indosat a Quick Report. http://www.bgpmon.net/hijack-by-as4761-indosat-a-quick-report/.

[3] W. Aiello, J. Ioannidis, and P. McDaniel. Origin authentication in interdomain routing. In *Proc of CCS*, 2003.

[4] Ruwaifa Anwar, Haseeb Niaz, David Choffnes, Italo Cunha, Phillipa Gill, and Ethan Katz-Bassett. Investigating Interdomain Routing Policies in the Wild. In *Proc. of ACM IMC*, Oct. 2015.

[5] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A Study of Prefix Hijacking and Interception in the Internet. In *Proc. of ACM SIGCOMM*, pages 265–276, 2007.

[6] O. Borchert, O. Kim, L. Hannachi, D. Montgomery, and K. Sriram. NIST RPKI Monitor and Test System, NIST Test and Measurement Tool. https://rpki-monitor.antd.nist.gov/.

[7] R. Bush and R. Austein. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810 (Proposed Standard), January 2013. Updated by RFC 8210.

[8] Kevin Butler, Toni R. Farley, Patrick McDaniel, and Jennifer Rexford. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122, 2010.

[9] Kevin Butler, Patrick McDaniel, and William Aiello. Optimizing bgp security by exploiting path stability. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pages 298–310, New York, NY, USA, 2006. ACM.

[10] CAIDA. The CAIDA AS Relationships Dataset. http://www.caida.org/data/as-relationships/, January 2016.

[11] CAIDA. ASRank CAIDA's Ranking of Autonomous Systems. https://asrank.caida.org/, 2018.

[12] B. Cartwright-Cox. Measuring RPKI adoption via the data-plane. nlnog day 2018.

[13] Haowen Chan, Debabrata Dash, Adrian Perrig, and Hui Zhang. Modeling Adoptability of Secure BGP Protocols. In *Proc. of SIGCOMM*. ACM, 2006.

[14] Taejoong Chung, Emile Aben, Tim Bruijnzeels, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce Maggs, Alan Mislove, Roland van Rijswijk-Deij, John Rula, and Nick Sullivan. RPKI is Coming of Age: A Longitudinal Study of RPKI Deployment and Invalid Route Origins. In *Proc. of IMC*. ACM, 2019.

[15] Avichai Cohen, Yossi Gilad, Amir Herzberg, and Michael Schapira. Jumpstarting BGP security with path-end validation. In *Proc. of ACM SIGCOMM*, pages 342–355. ACM, 2016.

[16] D. Cooper, E. Heilman, K. Brogle, L. Reyzin, and S. Goldberg. On the risk of misbehaving RPKI authorities. In *Proc. of HotNets*. ACM, 2013.

[17] Remy de Boer and Javy de Koning. BGP Origin Validation (RPKI). Technical report, Univeristy of Amsterdam, Systems and Network Engineering Group, July 2013.

[18] Lixin Gao and Jennifer Rexford. Stable Internet Routing without Global Coordination. *IEEE/ACM Trans. Netw.*, 9(6):681–692, 2001.

[19] Yossi Gilad, Avichai Cohen, Amir Herzberg, Michael Schapira, and Haya Shulman. Are We There Yet? On RPKI's Deployment and Security. In *NDSS*. The Internet Society, 2017.

[20] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford. How Secure are Secure Interdomain Routing Protocols? In *Proc. of SIGCOMM*. ACM, 2010.

[21] Geoffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick Drew McDaniel, and Aviel D. Rubin. Working around BGP: An Incremental Approach to Improving Security and Accuracy in Interdomain Routing. In *NDSS*. The Internet Society, 2003.

[22] Ethan Heilman, Danny Cooper, Leonid Reyzin, and Sharon Goldberg. From the consent of the routed: improving the transparency of the RPKI. In *Proc. of ACM SIGCOMM*, pages 51–62, 2014.

[23] Amir Herzberg, Matthias Hollick, and Adrian Perrig. Secure Routing for Future Communication Networks (Dagstuhl Seminar 15102). *Dagstuhl Reports*, 5(3):28–40, 2015.

[24] Tomas Hlavacek, Italo Cunha, Yossi Gilad, Amir Herzberg, Ethan Katz-Bassett, Michael Schapira, and Haya Shulman. DISCO: Sidestepping RPKI's deployment barriers. In *Proceedings of the 2020 Network and Distributed System Security (NDSS) Symposium*, February 2020.

[25] Tomas Hlavacek, Amir Herzberg, Haya Shulman, and Michael Waidner. Practical Experience: Methodologies for Measuring Route Origin Validation. In *IEEE/IFIP International Conference on Dependable Systems and Networks - DSN*, June 2018.

[26] Y.-C. Hu, A. Perrig, and M. Sirbu. SPV: secure path vector routing for securing BGP. In *Proc. of ACM SIGCOMM*, September 2004.

[27] G. Huston and G. Michaelson. Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs). RFC 6483 (Informational), February 2012.

[28] G. Huston, M. Rossi, and G. Armitage. Securing BGP: A literature survey. *IEEE Communications Surveys & Tutorials*, 13(2):199–222, 2011.

[29] Daniele Iamartino. Study and Measurements of the RPKI Deployment, 2015.

[30] Daniele Iamartino, Cristel Pelsser, and Randy Bush. Measuring BGP Route Origin Registration and Validation. In Jelena Mirkovic and Yong Liu, editors, *PAM*, volume 8995 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 2015.

[31] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. Pretty Good BGP: Improving BGP by Cautiously Adopting Routes. In *ICNP*, pages 290–299. IEEE Computer Society, 2006.

[32] S. Kent and K.seo. An Infrastructure to Support Secure Internet Routing. RFC 6480, The Internet society, February 2012.

[33] S. Kent and D. Mandelberg. Suspenders: A Fail-safe Mechanism for the RPKI, 2013.

[34] Stephen Kent, Charles Lynn, and Karen Seo. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected areas in Communications*, 18(4):582–592, 2000.

[35] Robert Lychev, Sharon Goldberg, and Michael Schapira. BGP security in partial deployment: Is the juice worth the squeeze? *ACM SIGCOMM Computer Communication Review*, 43(4):171–182, 2013.

[36] Lepinski M., Kong D., and Kent S. A Profile for Route Origin Authorizations (ROAs), February 2012.

[37] M. Lepinski (Ed.) and K. Sriram (Ed.). BGPsec Protocol Specification. RFC 8205, September 2017.

[38] H. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane Nano: Path Prediction for Peer-to-Peer Applications. In *Proce of NSDI*, 2009.

[39] Ratul Mahajan, David Wetherall, and Thomas E. Anderson. Understanding BGP Misconfiguration. In *Proc. of SIGCOMM*, pages 3–16. ACM, 2002.

[40] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch, and D. McPherson. Dissemination of Flow Specification Rules, August 2009. `https://tools.ietf.org/html/rfc5575`.

[41] R. Mazloum, M. Buob, J. Auge, B. Baynat, D. Rossi, and T. Friedman. Violation of Interdomain Routing Assumptions. In *Proc. of Passive and Active Measurement Conference (PAM)*, March 2014.

[42] Asya Mitseva, Andriy Panchenko, and Thomas Engel. The state of affairs in BGP security: A survey of attacks and defenses. *Computer Communications*, 124:45–60, June 2018.

[43] Reynaldo Morillo, Justin Furuness, Cameron Morris, James Breslin, Amir Herzberg, and Bing Wang. ROV++: Improved deployable defense against BGP hijacking. https://sidr.engr.uconn.edu, 2021. Full version.

[44] W. Mühlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig. Building an AS-topology model that captures route diversity. In *Proc. of SIGCOMM*, 2006.

[45] S. Murphy. BGP Security Vulnerabilities Analysis, IETF draft-ietf-idr-bgp-vuln-00. February 2002.

[46] Y. Rekhter (Ed.), T. Li (Ed.), and S. Hares (Ed.). A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006. Updated by RFCs 6286, 6608, 6793, 7606, 7607, 7705, 8212, 8654.

[47] Renesys. The New Threat: Targeted Internet Traffic Misdirection. http://www.renesys.com/2013/11/mitm-internet-hijacking/.

[48] Andreas Reuter, Randy Bush, Ítalo Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wählisch. Towards a Rigorous Methodology for Measuring Adoption of Rpki Route Validation and Filtering. *CoRR*, abs/1706.04263, 2017.

[49] RouteViews. University of Oregon Route Views Project. http://www.routeviews.org/routeviews/, 2018.

[50] J. Scudder (Ed.), R. Fernando, and S. Stuart. BGP Monitoring Protocol (BMP). RFC 7854 (Proposed Standard), June 2016. Updated by RFC 8671.

[51] 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.

[52] Muhammad S. Siddiqui, Diego Montero, Rene Serral-Gracia, Xavi Masip-Bruin, and Marcelo Yannuzzi. A survey on the recent efforts of the Internet Standardization Body for securing inter-domain routing. *Computer Networks*, 80:1–26, April 2015.

[53] K. Sriram, O. Borchert, and D. Montgomery. Origin Validation Policy Considerations for Dropping Invalid Routes, IETF Internet Draft (Standards Track, SIDROPS Working Group), May 2020.

[54] K. Sriram and D. Montgomery. Resilient Interdomain Traffic Exchange: BGP security and DDoS Mitigation. December 2019. NIST SP 800-189.

[55] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and whisper: Security mechanisms for BGP. In *Proc. of NSDI*, 2004.

[56] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark. Profiling BGP Serial Hijackers: Capturing Persistent Misbehavior in the Global Routing Table. In *Proc. of IMC*. ACM, 2019.

[57] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark. To Filter or Not to Filter: Measuring the Benefits of Registering in the RPKI Today. In *Proc. of Passive and Active Measurement Conference (PAM)*, January 2020.

[58] Andree Toonk. Turkey Hijacking IP Addresses for Popular Global DNS Providers. BGPMon.

[59] Paul C van Oorschot, Tao Wan, and Evangelos Kranakis. On interdomain routing security and pretty secure BGP (psBGP). *ACM Transactions on Information and System Security (TISSEC)*, 10(3):11, 2007.

[60] Pierre-Antoine Vervier, Olivier Thonnard, and Marc Dacier. Mind Your Blocks: On the Stealthiness of Malicious BGP Hijacks. In *NDSS*, 2015.

[61] Matthias Wählisch, Olaf Maennel, and Thomas C. Schmidt. Towards detecting BGP route hijacking using the RPKI. In *Proc. of ACM SIGCOMM*, pages 103–104, 2012.

[62] Matthias Wählisch, Robert Schmidt, Thomas C. Schmidt, Olaf Maennel, Steve Uhlig, and Gareth Tyson. RiPKI: The Tragic Story of RPKI Deployment in the Web Ecosystem. In Jaudelice de Oliveira, Jonathan Smith, Katerina J. Argyraki, and Philip Levis, editors, *Proceedings of the 14th ACM Workshop on Hot Topics in Networks (HotNets)*, pages 11:1–11:7. ACM, November 2015.

[63] R. White. Deployment Considerations for Secure Origin BGP (soBGP)., June 2003.

[64] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. Aggregated path authentication for efficient BGP security. In *Proc. of CCS*, 2005.

[65] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Wu, and L. Zhang. Detection of invalid routing announcements in the Internet. In *Proc. of IEEE DSN*, 2002.

APPENDIX A
AUXILIARY PSEUDO-CODE IN ROV++

In Alg. 2, we present the pseudo-code of several procedures in ROV++ that are not described in Section III-C. Let $\mathcal{B}$ denote the set of blackholes that is set up at a router. Initially $\mathcal{B}$ is empty. For a $hole \in \mathcal{H}$, let $hole.path$ denote the AS path in the announcement. Similarly, for an announcement $ann \in \mathcal{A}$, let $ann.path$ denote the AS path in the announcement.

APPENDIX B
OTHER PERFORMANCE RESULTS

Figures 13 and 14 show the results for the adopting and non-adopting ASes, respectively, under subprefix hijacks in the category of top 100 ASes. Figures 15 and 16 show the corresponding results for the ASes in the category of other ASes (i.e., not in top 100 and not edge ASes).

Algorithm 2 Auxiliary procedures in ROV++

procedure FINDBEST (pre)
Set $S = \{a \mid a \in \mathcal{A}, a.pre \supseteq pre\}$
Find the best route $a \in S$ following priority rules in the order of relationship, minimize-holes, and path length
return a
end procedure

procedure BLACKHOLE ($hole$)
$\mathcal{B} = \mathcal{B} \cup hole.pre$
Set up routing table to drop all traffic to destination IP addresses in $hole.pre$
end procedure

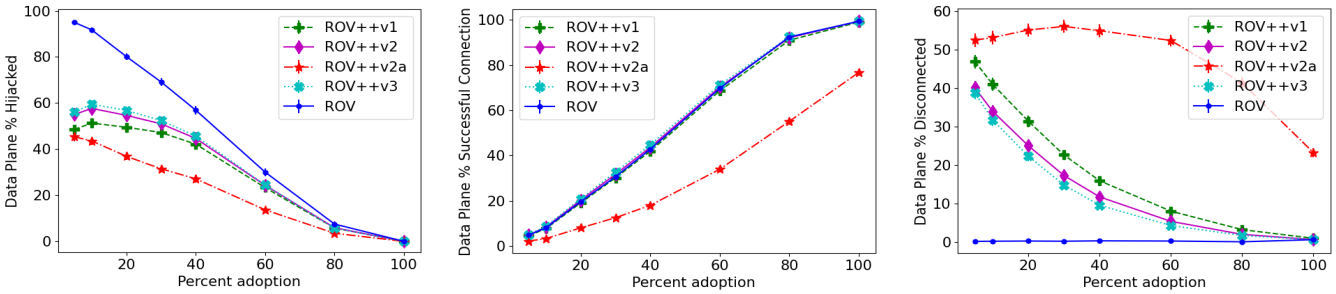
procedure REMOVEBLACKHOLE ($hole$)
$\mathcal{B} = \mathcal{B} \setminus hole.pre$
Remove blackhole on $hole.pre$ from the routing table
end procedure

procedure SENDBLACKHOLEANN ($hole$)
//only do so if $hole.from$ is a provider or peer
Create a blackhole announcement *ann* with prefix as $hole.pre$ and AS path as $hole.path$
Set transitive flag in *ann* to mark that it is a blackhole announcement
Send *ann* to customers in ROV++ v2 and according to export policy in ROV++ v2a
end procedure

procedure WITHDRAWBLACKHOLEANN ($hole$)
Withdraw blackhole announcements with prefix $hole.pre$ that were sent to other ASes
end procedure

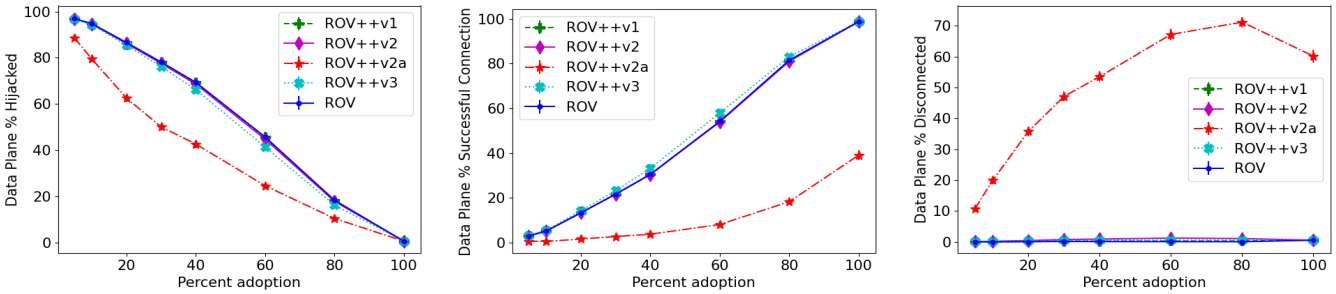
procedure SENDPREVENTIVEANN (a, *subp*)
if ISCUSTOMER ($a.from$) **then**
  return
end if
if a has a transitive field indicating hole **then**
  return
end if
Create a preventive announcement *ann*
ann.pre = *subp*
ann.path = $a.path$
Set transitive flag in *ann* to mark that it is a preventive announcement
Send *ann* only to customers that have not sent $a.pre$ to this AS
end procedure

procedure WITHDRAWPREVENTIVEANN ($a, subp$)
Withdraw preventive announcements with prefix $subp$ that was sent to other ASes due to announcement a
end procedure



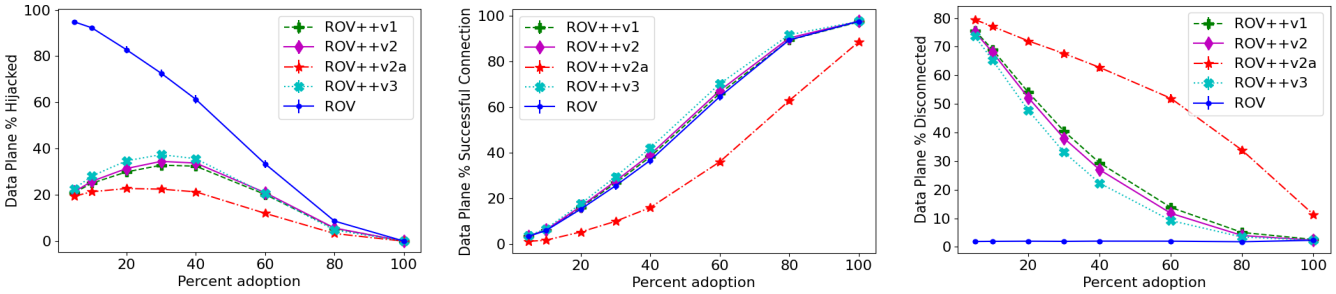
(a) Hijack rate. (b) Successful connection rate. (c) Disconnect rate.

Figure 13. Subprefix hijacks: Data plane results for adopting ASes in the category of top 100 ASes.



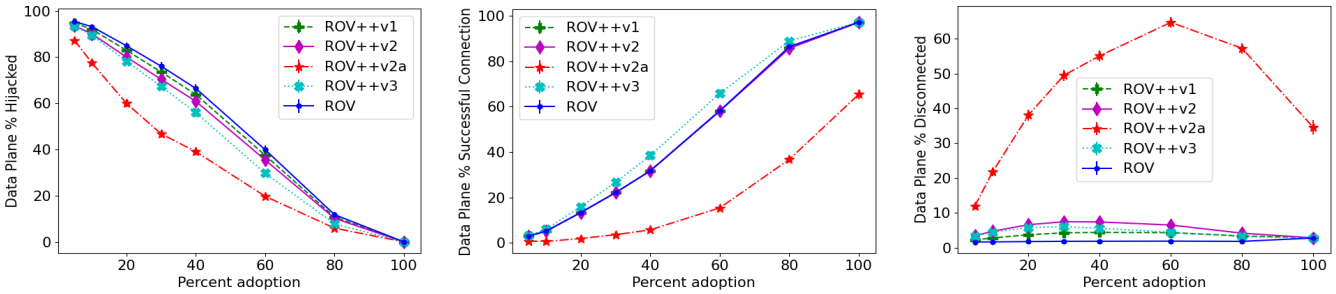
(a) Hijack rate. (b) Successful connection rate. (c) Disconnect rate.

Figure 14. Subprefix hijacks: Data plane results for non-adopting ASes in the category of top 100 ASes.



(a) Hijack rate. (b) Successful connection rate. (c) Disconnect rate.

Figure 15. Subprefix hijacks: Data plane results for adopting ASes in the category of other ASes.



(a) Hijack rate. (b) Successful connection rate. (c) Disconnect rate.

Figure 16. Subprefix hijacks: Data plane results for non-adopting ASes in the category of other ASes.