iSPY: Detecting IP Prefix Hijacking on My Own

Zheng Zhang, Ying Zhang, Y. Charlie Hu, *Senior Member, IEEE, Senior Member, ACM*, Z. Morley Mao, and Randy Bush

***Abstract*—IP prefix hijacking remains a major threat to the security of the Internet routing system due to a lack of authoritative prefix ownership information. Despite many efforts in designing IP prefix hijack detection schemes, no existing design can satisfy all the critical requirements of a truly effective system: real-time, accurate, lightweight, easily and incrementally deployable, as well as robust in victim notification. In this paper, we present a novel approach that fulfills all these goals by monitoring network reachability from key external transit networks to one's own network through lightweight prefix-owner-based active probing. Using the prefix-owner's view of reachability, our detection system, iSPY, can differentiate between IP prefix hijacking and network failures based on the observation that hijacking is likely to result in topologically more diverse polluted networks and unreachability. Through detailed simulations of Internet routing, 25-day deployment in 88 autonomous systems (ASs) (108 prefixes), and experiments with hijacking events of our own prefix from multiple locations, we demonstrate that iSPY is accurate with false negative ratio below 0.45% and false positive ratio below 0.17%. Furthermore, iSPY is truly real-time; it can detect hijacking events within a few minutes.**

***Index Terms*—BGP, detection, Internet, prefix hijacking, routing.**

I. Introduction

IP PREFIX hijacking poses a serious threat to the robustness and security of the Internet routing system. Any network whose prefix is hijacked may experience reachability problems and cannot easily identify the actual cause. IP prefix hijacking is essentially a special form of denial-of-service attack. Hijacked prefixes can also be used for carrying out malicious activities, raising the challenge of identifying the actual perpetrator. Eliminating IP prefix hijacking is close to impossible given today's routing design, partly due to the lack of authoritative information on prefix ownerships. Even with such information, topology can still be spoofed without modifying prefix-owners, resulting in intercepted traffic. Thus, we believe there is a critical need to design an effective IP prefix hijacking detection system to inform the mitigation response and help locate the responsible autonomous system (AS) for the attack. Such a detection system should satisfy all of the following critical requirements.

1) *Real-time*: Detection should be real-time to identify short-lived attacks and minimize potential damage.
2) *Accurate*: The detection accuracy must be high, with both low false positive and false negative ratios.
3) *Lightweight*: Detection should be lightweight and scale well with the number of protected IP prefixes and networks without sacrificing the detection accuracy.
4) *Easy to deploy*: The detection system can be easily deployed incrementally without requiring privileged access to data such as live BGP feeds from many ASs.
5) *Incentive to deploy*: The system is designed to tie the deployment effort to the direct benefits of the deploying organization and hence creates strong incentives for widespread deployment.
6) *Robust in victim notification*: The system is able to notify the victim (owner) of the hijacked IP prefix in a robust fashion.

In addition to these six critical requirements, it is desirable that the detection system accurately *identifies the attacker* and *identifies the networks polluted by the attack* to faciliate the mitigation of the attack.

Most existing proposals on prefix hijack detection fall into three categories, as summarized in Table I, based on the type of information used. The first category of control-plane-based approaches (e.g., [1]–[3]) perform passive monitoring of BGP routing information to detect anomalous behavior and hence are easily deployable, but can be fairly inaccurate due to limited BGP data [4] and legitimate reasons for anomalous updates [5]. The timeliness of such an approach heavily relies on access to real-time BGP feeds.

The second category of detection systems (e.g., [5]) collects real-time information from both the control plane and the data plane to perform joint analysis and hence is real-time, but it also requires privileged access to live BGP feeds and the detection accuracy is still limited by the vantage point locations of both data sources. Such limitations can allow attackers to evade detection. Control-plane-based systems can potentially identify the attacker that does not deliberately hide its identity by falsifying the AS path. The third category (e.g., [6] and [7]) only relies on real-time data plane information and hence is more easily deployable via active probing, but also suffers from the same vantage point limitations. Note that existing schemes using data plane information so far have taken an infrastructure-based approach, relying on a restricted set of network locations to probe prefixes in the entire Internet and hence suffer from poor scalability. Finally, somewhat ironically, none of the above proposals has devised a way that guarantees to notify the victim (owner) of the hijacked IP prefix, which is the final, but also a crucial, step of the prefix hijack detection process.

Manuscript received April 09, 2009; revised October 23, 2009 and February 10, 2010; accepted April 22, 2010; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor O. Bonaventure. Date of publication September 23, 2010; date of current version December 17, 2010.This work was supported in part by NSF CAREER-0238379, CyberTrust-0430204, and CAREER-0643612. An earlier version of this paper was presented at ACM SIGCOMM 2008.

Z. Zhang and Y. C. Hu are with Purdue University, West Lafayette, IN 47907 USA (e-mail: zhang97@purdue.edu; ychu@purdue.edu).

Y. Zhang and Z. M. Mao are with the University of Michigan, Ann Arbor, MI 48109 USA (e-mail: wingying@umich.edu; zmao@umich.edu).

R. Bush is with the Internet Initiative Japan, Tokyo 101-0051, Japan (e-mail: randy@psg.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2010.2066284

1063-6692/$26.00 © 2010 IEEE

TABLE I
COMPARISON AMONG PREFIX HIJACKING DETECTION SYSTEMS

Requirement	Control-plane-based (passive) [1], [2], [3]	Control-plane & data-plane-based [5]	Data-plane-based [6], [7]	iSPY (this paper)
Real-time	depending on data sources	√	√	√
Accurate	X	limited by vantage points	limited by vantage points	√
Light-weight	√	√	√	√
Easily deployable	√	X	√	√
Incentive to deploy	X	X	X	√
Robust notification	probabilistic	X	X	√
Identifying the attacker	somewhat	somewhat	√	X (future work)
Identifying polluted networks	X	X	X	√

In this paper, we present an IP prefix hijacking detection system that satisfies all of the mentioned requirements. Our proposed system, iSPY, exploits a key observation about IP prefix hijacking: Due to the rich connectivity of the ASs in the Internet, a prefix hijack typically pollutes a significant percentage of the ASs, i.e., those ASs will route any packet destined to the hijacked prefix to the attacker's network, as opposed to the victim's network. In other words, when a prefix hijack is ongoing, the victim's network will experience failure in probing a large number of networks, as the probe reply will be routed to the attacker's network. This observation motivates our prefix-owner-centric data-plane-based hijacking detection system. Essentially, each network deploys iSPY to detect hijacking of its own prefixes, and iSPY simply performs continuous probing to transit ASs and detects hijacking events based on the observed reachability to these ASs.

A fundamental difference between iSPY and previous approaches using data-plane information [6], [7] is that iSPY is *prefix-owner-centric* in that each network performs real-time probing in the data plane to detect potential hijacking of its own prefix(es). This approach makes the detection system not only *real-time* and *easy to deploy*, the same as previous proposals, but also exhibits the following additional properties: 1) *accurate* as the detection accuracy is not limited by the placement of any vantage points; 2) creating *strong incentives* to deploy as deployment by each prefix-owner directly benefits itself; 3) *lightweight* as it is fully decentralized among the prefix-owner networks, and each prefix-owner just needs to continuously probe the over 3000 transit ASs; and 4) intrinsically *robust in victim notification* as the prefix-owner makes the hijacking detection decision locally. Lastly, upon detecting a hijacking event, the victim network has also identified the set of polluted networks and can notify them of the event, e.g., using a different prefix, so that these networks can help to react by filtering out the attacker's route before the attacker is finally removed [8], [9].

The design and implementation of iSPY face several challenges. First, it needs to be able to effectively distinguish unreachability due to a hijacking event from other disruptive routing events such as link failures, congestion, and misconfigurations. To overcome this challenge, we propose a prefix-owner's view of the reachability from its network to the rest of the Internet. Such a view consists of forward AS-level paths taken from the network to reach all the transit ASs (collected from real-time probing in the data plane). Using such a prefix-owner's view of Internet reachability, we show that unreachability due to hijacking exhibits a very different pattern in terms of the *cuts* in the AS-level paths. The number of cuts in the AS-level paths is then used by iSPY as a unique unreachability signature to distinguish hijacking from other disruptive routing events such as link failures.

As a second challenge, the probing mechanism of iSPY needs to be carefully engineered as its performance directly affects the effectiveness of iSPY. In particular, the continuous probing performed needs to be *lightweight* to ensure low probing traffic. The probing must be *efficient* to guarantee low detection latency. The probing must also be *robust* to overcome effects caused by probing-unfriendly events such as ICMP rate limiting, link congestion, and traceroute blocking. A fundamental difference between iSPY and distributed vantage-point-based monitoring systems (e.g., [6], [10], and [11]) is that iSPY is prefix-owner-centric while prefix hijacking is at the AS-level, and hence a prefix-owner deploying iSPY only needs to monitor the reachability to the about 3000 transit ASs.[1] This low number of monitoring targets directly contributes to iSPY's efficiency and low detection latency.

This paper makes the following contributions. First, we propose the first prefix hijack detection system that satisfies all six critical requirements for an effective detection system (see Table I). Second, we present the key distinguishing signature of prefix hijacking from other routing failures from the victim network's point of view, which forms the underlying foundation for iSPY (Section III). Third, we present the detailed design and implementation of the prefix-owner-centric probing mechanism demonstrating an effective working system (Section IV). Fourth, we conduct analysis and Internet experiments to validate iSPY's effectiveness in action (Section VI). We demonstrate that iSPY is lightweight and can accurately detect prefix hijacking in real time with 0.45% false negative ratio and 0.17% false positive ratio.

II. BGP PREFIX HIJACKING

In this section, we briefly review IP prefix hijacking targeted at the interdomain routing protocol. IP prefix hijacking occurs when a misconfigured or malicious BGP router in a network N either originates or announces a route to traverse its network for an IP prefix not owned by itself. Due to a lack of widely deployed security mechanisms to ensure the correctness of BGP routing updates, forwarding tables of other networks may be polluted from adopting and propagating the bogus route. As a result, some of the traffic destined to the victim prefix is misrouted to the attacker BGP router, which can perform any malicious activities pretending to be the owner of the victim prefix

[1]Note that vantage-point-based monitoring systems can often make use of low overhead probes such as pings (e.g., [6] and [10]).

or may even choose to selectively forward the traffic back to the victim [12], [13].

For each AS n, it either receives the bogus route or does not observe it at all. In the former case, it may choose the bogus route if the route is more preferred and thus become *polluted*. In the latter case, n's neighbors must not be polluted, thus preventing n from observing the bogus route.

IP prefix hijacking can be performed in several ways. We describe the three main types to facilitate our subsequent discussion of detection schemes. A more detailed classification can be found in a recent study [5].

1) *Regular prefix hijacking* occurs when the attack router originates a route to an existing IP prefix of the victim network. As a result, the Internet is partially polluted, depending on how preferable the bogus route is compared to the valid route from the perspective of various networks.
2) *Subprefix hijacking* results from stealing a subnet of an existing prefix in the routing tables by announcing a route for the subnet originating from the attacker network. Due to longest-prefix-matching-based forwarding, most networks are polluted.
3) *Interception-based hijacking* is a special case of the regular prefix hijack in that the attacker network uses one of its unpolluted neighbors to forward the intercepted traffic back to the victim.

Our detection system iSPY addresses the basic type of hijacking attacks, namely the regular prefix hijacking. We will discuss subprefix hijacking and interception-based hijacking in Section VII.

III. Key Observation

The design of iSPY exploits a key observation about IP prefix hijacking: A prefix hijack almost always pollutes a significant percentage of ASs in the Internet, and hence during a hijacking event, probes initiated from the victim's network are expected to witness unreachability to a large number of ASs. More importantly, this unreachability to many ASs has a different signature from that due to a few link failures near the victim's network, which can also result in unreachability to many ASs. The unique unreachability signature of hijacking is used by iSPY to distinguish hijacking from other disruptive routing events such as link failures and congestion.

In this section, we present a reachability framework that formalizes this observation. We first define a prefix-owner's view of the Internet reachability and motivate the unreachability signature of hijacking. We then validate the unreachability signature of hijacking via simulations on the AS-level topology of the current Internet.

A. Prefix-Owner's View of Internet Reachability

To facilitate prefix-owner-centric monitoring of potential prefix hijacking, we need a way to capture the prefix-owner's view of Internet reachability that can also be easily implemented using existing probing tools supported in the Internet such as traceroute.

We propose to capture the prefix-owner's view of Internet reachability as a set of paths called *vPath* (victim's path). vPath is simply the set of AS-level *forward paths* from a potential victim IP prefix (i.e., the network that owns the prefix, i.e., the victim network) to all the ASs on the Internet. We resort to the *forward* paths instead of the backward paths because the former can be easily obtained from traceroute issued by the prefix-owner.

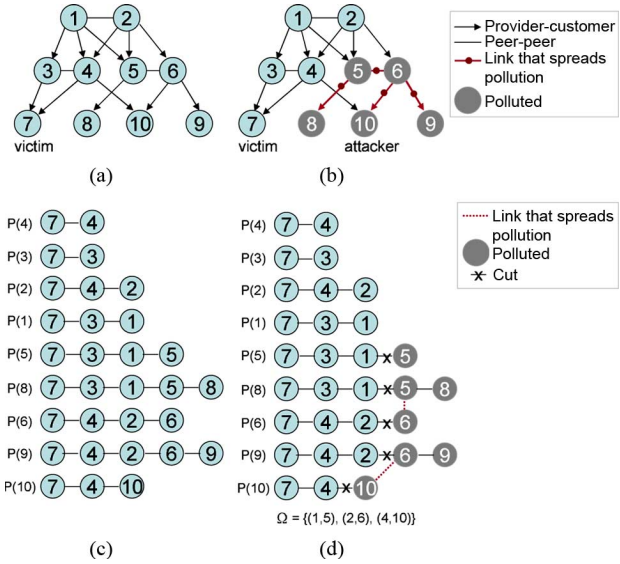


Fig. 1. Example of prefix hijacking, vPath, and cuts. (a) Topology before hijacking. (b) Topology after hijacking. (c) vPath before hijacking. (d) vPath after hijacking.

The intuition behind vPath is clear. If an AS X is polluted by a prefix hijack, replies to traceroute probes originated from the victim network to any prefix owned by X will not correctly reach the victim network. Therefore, vPath indirectly captures reachability from external networks to a potential victim network. For external networks with multiple prefixes, although the AS-level forward paths to these prefixes can differ, we just need to select any such prefix and regard the path to it as the path to this destination AS to capture the reachability to the victim network from this AS.

Fig. 1(a) shows an example AS topology, with AS relationships (e.g., provider–customer and peer–peer) implied by the relative position of ASs, and Fig. 1(c) shows the vPath representation of AS 7's reachability to other ASs.

Before choosing vPath as the reachability representation, we attempted to compress vPath into more concise presentations such as trees or DAGs. However, such compressions are not possible due to policy-based routing in the Internet. If the set of forward paths from an AS is collapsed into a directed graph, the graph may contain loops, and hence the information of individual forward paths is lost. For this reason, we resorted to the vPath representation.

Due to potential route asymmetry between two networks in BGP routing, it is possible that a destination AS is still reachable though certain ASs along the forward path to it are polluted, e.g., two nodes c and d along the forward AS-level path $[a, b, c, d, e]$ are polluted while node e is not. In this case, a probing tool such as traceroute may proceed to subsequent hops when probing to the hops in c and d returns "*" and finally reach e. Thus, the returned path may contain one or more "*"s followed by more IP hops reached further down the path. After resolving an IP path to an AS path and collapsing adjacent unresolved AS hops

to a symbol "#," which is used to represent the uncertain part of AS path, the AS-level path may contain one or more "#"s, e.g., $[a, b, \#, e]$.[2]

Monitoring Reachability to Transit ASs Only: In practice, obtaining paths to the large number of ASs in the Internet can be costly. We monitor reachability to the much smaller number of transit ASs and restrict the notion of vPath to the paths to these transit ASs. Transit ASs are those ASs that forward traffic for other ASs. Nontransit ASs are also known as stub ASs. However, this sampling mechanism will not cause iSPY to miss any attack. This is because for a hijack from a stub AS to be successful, it will always pollute its provider transit AS(s) and possibly other transit ASs.

B. Prefix Hijack Detection Problem

We now formulate the prefix hijack detection problem using the vPath reachability framework. A network that deploys iSPY for prefix hijack detection performs continuous rounds of probing from its prefix to all transit ASs to take periodic snapshots of vPath. Whenever a snapshot from a new round of probing, denoted as T_{new}, is obtained, the detection module of iSPY compares T_{new} with a previous snapshot T_{old} and searches for hints of a potential prefix hijack when T_{new} is found to be incomplete, indicating partial unreachability. Since in practice, there may always be some limited reachability problems due to other routing anomalies, our goal is to identify changes in reachability patterns due to hijacks. The hijack detection problem can be formulated as *given* T_{old}, *which indicates full reachability*, and T_{new}, *which indicates partial reachability, how to infer whether there is a hijacking event*. To solve this problem, we analyze the unique characteristics of the gap between T_{new} and T_{old} that is created by a prefix hijack. To facilitate the analysis, we first define the notion of *cuts* in vPath to capture the change in consecutive snapshots of the vPath representations.

Cuts in vPath: When there is no route change in the forward path from the prefix-owner s to destination AS d in between T_{old} and T_{new}, there are two main reasons that a hop (u_i, u_{i+1}) along an AS-level forward path $P(d) = [s, u_1, u_2, \ldots, u_n, d]$ in T_{old} becomes unreachable in T_{new}. First, AS u_{i+1} is polluted with a path that routes to the attacker's AS. Second, link (u_i, u_{i+1}) or, in the case of path asymmetry, some link along the return path from u_{i+1} back to s suffers from a physical link failure, though the latter is unlikely because the forward path is also a legitimate return path. Therefore, the most likely failing link is (u_i, u_{i+1}) on the forward path. Hence, for both reasons, if we define (u_i, u_{i+1}) as a cut, it generally accurately reflects the location of the cause of the unreachability. We note there could also be other reasons such as transient link congestion resulting in probing packet loss that can cause (u_i, u_{i+1}) to be unreachable.

It is possible that in reacting to a link failure, the path from s to destination AS d has changed between T_{old} and T_{new}. In this case, in T_{new}, the new path $P'(d) = [s, v_1, v_2, \ldots, v_n, d]$ (where $P(d)$ and $P'(d)$ differ by at least one AS hop) may be complete or may be partial, i.e., a link (v_i, v_{i+1}) is unreachable and the status of the remaining links is unknown. Together, there are four possible scenarios in terms of the reachability to each destination AS in T_{old} and T_{new}, and we define a *cut* to the path accordingly.

[2]Another cause for "*" is that some routers may not respond to traceroute, and we discuss how we complement traceroute with other probes in Section IV.

TABLE II
EXAMPLES OF CUTS UNDER THE CUT DEFINITION

Cut		Current path $P'(d)$		
		abcd	ab#d	ab#
Previous path $P(d)$	abcd	no cut	no cut	bc
	ab#d	no cut	no cut	b#

Definition 1 (Cuts in vPath T_{new}): We define a cut in the path to each destination AS d by comparing the paths to AS d in T_{old} and T_{new} as follows.

- Case 1: $P(d)$ remains complete and identical in T_{old} and T_{new}. There is no cut in this case.
- Case 2: $P(d)$ becomes partial in T_{new}. We define link (u_i, u_{i+1}) as the cut, where u_i is the last AS along the path for which traceroute obtained a reply.
- Case 3: $P(d)$ has changed to $P'(d)$ in T_{new}, but $P'(d)$ is complete. There is no cut in this case since clearly the destination is not polluted by the hijack; the route change was due to some legitimate reason, e.g., to recover from some link failure.
- Case 4: $P(d)$ has changed to $P'(d)$ in T_{new}, but $P'(d)$ is partial, and the last hop for which traceroute obtained a reply is v_i. If v_i also appears in $P(d)$, we define link (v_i, v_{i+1}) as the cut, where v_{i+1} is the hop after v_i in $P(d)$. If v_i does not appear in $P(d)$, we conservatively record that there is a cut $(v_i, *)$. The justification is that disregarding the reason for the route change, the partial path indicates that either there is a new link failure or recovery from a link failure is not yet completed.

We compute the cuts thus defined for all unreachable ASs in snapshot T_{new}, and denote the resulting set of distinct cuts as Ω.

We note that our definition of cuts also handles the cases when the AS paths in vPath contain uncertain subpaths "#" due to route asymmetry, as discussed in Section III-A. Table II shows the cuts defined under a few example scenarios.

Fig. 1(b) shows a prefix hijack in the example AS topology, where AS 10 hijacks the prefix of AS 7. The pollution caused by the hijack spreads through links (10, 6) and (6, 5) to pollute ASs 5, 8, 6, and 9. Note that the spreading of polluted paths for reaching back to the attacker respects AS relationship and valley-free routing. Fig. 1(d) shows the vPath observed by the victim after the hijack, and the hijack creates three cuts to the vPath.

Continuous Decision Making: In practice, a hijacking event can occur at any point in the middle of a probing round. Hence, making detection decisions at the finish of a probing round can lead to long detection latency. To overcome this restriction, we (re)define vPath snapshots T_{old} and T_{new} to be based on the notion of *logical rounds*, which are continuously updated. Assume each round probes the N transit ASs in some fixed order. At any point in the middle of a round, taking out the past N consecutive probes would also make up a complete *logical round* of probing. We refer to this logical round as the *current* logical round, and the N probes before the current

logical round as the *previous* logical round. We redefine T_{new} and T_{old} as the snapshots from the current and the previous logical rounds and adjust the definition of cuts accordingly. Thus, we are able to make detection decisions continuously.

C. Unreachability Signature of Hijacking

We conjecture that Ω is almost always large when there is an ongoing prefix hijack, and is typically small otherwise, i.e., due to link failures and congestion. Consequently, the large size of Ω can be used as a distinguishing *signature* of the unreachability pattern of prefix hijacking that is witnessed by the prefix-owner.

We first justify our hypothesis in the case of hijacking. The fundamental reason that hijacking will result in a large number of cuts in vPath is that the Internet topology is not a tree. A tree topology would have implied the pollution is always confined in one subtree and there is only one cut in trying to reach the polluted ASs from outside the polluted region. The actual Internet topology significantly deviates from such a simple tree topology due to the large number of peering links and multihomed links. As an evidence, the AS-level Internet topology that was inferred using routing table dumps in September 2007 from RouteViews [14] contains only 3742 transit ASs, but 18384 links between the transit ASs.

The large number of peering links and multihomed links has two implications. First, from the attacker's point of view, the pollution due to its hijacking is likely to propagate far into the Internet, reaching many other networks, first along multihomed links (recursively), and then the peer networks and customer networks of such polluted networks. In other words, the pollution can reach far beyond a cone region rooted at some polluted ancestor AS, e.g., some AS that is the attacker's provider's provider. A cone rooted at an AS is defined as the AS along its customers and its customers' customers, etc. As an example, in Fig. 1(b), the pollution is not confined to the cone rooted at AS 6; it propagates to AS 5 via a peering link, thereby polluting the cone rooted at AS 5.

Second, conversely, from the victim's point of view, because of the rich connectivity of the ASs due to peering links and multihomed links, the forward paths from the victim AS to the polluted ASs are likely to go through many diverse paths, as opposed to traversing only the roots of the polluted cones. Such diverse paths going from the region of unpolluted ASs to the region of polluted ASs results in many cuts in vPath. For example, in Fig. 1(b), the multihomed link from AS 4 to AS 10 results in the forward path from victim AS 7 to AS 10 to cut into the polluted region from the side, as opposed to going through the roots of the polluted cones AS 5 and AS 6. As a result, the forward path experiences a cut at that cut-through link.

In contrast to prefix hijacking, conventional disruptive routing events such as link failures and congestion typically result in few cuts in vPath. Note that link failures near the prefix-owner network, e.g., at its provider links, may cause unreachability to a large number of ASs. However, such link failures will result in few distinct cuts in vPath, i.e., near the victim AS. In general, we expect network events resulting in large-scale reachability loss to be very rare due to the following reasons. First, there are usually multiple physical links between adjacent ASs in the Internet, which can quickly recover from a single physical link failure or congestion by redirecting traffic to alternative egress points. Second, many ASs are multihomed (even more so for transit ASs) and have several routes to the Internet. Although not always [10], such redundancy usually helps these ASs stay connected in case of link failures. Third, and importantly, iSPY monitors the reachability to transit ASs, which are generally more stable than stub ASs. The latter are commonly considered in previous large-scale reachability studies such as [10].

Given the scale of the Internet, simultaneous events that affect the reachability from a given network's perspective to a large number of topologically uncorrelated networks are likely to be rare. Even with events such as 9/11 or the Northeast blackout, the impacted networks are found to be limited by geographic locations, likely to result in few cuts [15]. Finally, our own Internet measurement study of prefix-owner's view of Internet reachability (Section VI-A) from over 100 network locations (using nodes on PlanetLab) further confirms that the number of cuts witnessed by the individual networks consistently stays below 10.

D. Simulation Validation

Since prefix hijacking is a rare event in the Internet, we resort to simulations to validate our hypothesis that prefix hijacking creates a large set Ω.

Methodology: We simulate 2450 hijacking instances on a realistic AS-level Internet topology. The AS-level Internet topology is obtained by combining six-month routing table snapshots and updates collected from more than 100 vantage points in RouteViews [14], RIPE RIS [16], and Abilene in 2007 and running Gao's algorithm [17] on them to obtain the AS relationship. The obtained topology consists of 23 195 ASs. The ASs are classified into five categories by tier and transit/stub, namely, tier-1, tier-2 transit, tier-2 stub, tier-3+ (i.e., $\text{tier} \geq 3$) transit and tier 3+ stub. We define tier-1 ASs as those ASs that do not have providers and peering with all other tier-1 ASs [18], and the tier of an AS as one plus the minimum number of providers that connect this AS to a tier-1 AS [15]. Eleven tier-1 ASs are inferred from the topology. Stub ASs are recognized as those ASs that always appear in the last AS hop in routing tables. Nonstub ASs are transit ASs. We select 10 ASs from each of the five categories. Each hijacking instance selects a single attacker and a single victim from the total of 50 ASs.

Each instance is simulated via the following steps. 1) We compute the forward path $P(d)$ from the victim to each transit AS d before the hijack. To do that, we simulate that each AS d originates its own prefix, and all ASs eventually converge on the routes regarding all prefixes. Our simulator computes each AS's route selection upon route convergence by emulating BGP routing update propagation and the BGP decision process including relationship-based route export and customer-prefer route selection. The simulator however does not emulate the timing aspects of route convergence or complex interactions between the data plane and control plane before convergence. 2) We then simulate false-origin prefix hijacking by the attacker. After the attacker originates the victim's prefix and routing has converged, the ASs that select routes destined to the attacker are polluted. 3) We next compute the forward (partial) path $P'(d)$ from the victim to each transit AS d by emulating traceroute probing to AS d. As discussed in Section III-A,

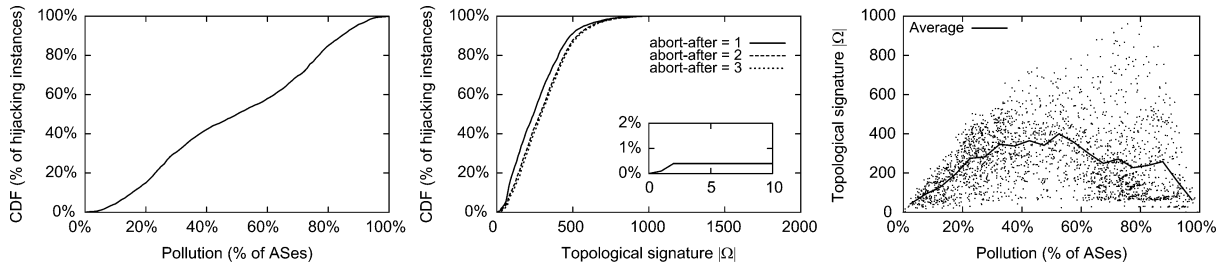


Fig. 2. (a) Distribution of pollution among the hijacking instances. (b) Distribution of the number of cuts $|\Omega|$ among the hijacking instances. (c) Correlation between pollution and $|\Omega|$, $\mathrm{abort\text{-}after} = 1$.

TABLE III
PERCENTAGE OF SMALL $|\Omega|$ INSTANCES

Victim category	Total inst.	Small $\|\Omega\|$ instances		
		$\|\Omega\| \leq 5$	$\|\Omega\| \leq 10$	$\|\Omega\| \leq 20$
Tier-1	490	3 (0.61%)	3 (0.61%)	4 (0.82%)
Tier-2 transit	490	1 (0.20%)	1 (0.20%)	1 (0.20%)
Tier-2 stub	490	4 (0.82%)	4 (0.82%)	5 (1.02%)
Tier-3+ transit	490	3 (0.61%)	3 (0.61%)	4 (0.82%)
Tier-3+ stub	490	0 (0.00%)	0 (0.00%)	0 (0.00%)
Any	2450	11 (0.45%)	11 (0.45%)	14 (0.57%)

due to route asymmetry, a path $P'(d)$ may contain uncertain nontrailing subpaths "#." Whether such nontrailing subpaths are collected in vPath depends on the traceroute configuration, i.e., how many consecutive unreachable IP hops traceroute tolerates before aborting. We approximate this configuration by assuming traceroute aborts after seeing a fixed number of consecutive unreachable AS hops. We denote this number as *abort-after* and simulate three scenarios: $\mathrm{abort\text{-}after} = 1$, 2, and 3.[3] 4) Finally, we calculate the set of distinct cuts Ω using $P(d)$ and $P'(d)$ for all d.

Note the simulation here focuses on the number of cuts caused by hijacking and ignores the timing aspects of detection. The cuts are thus identified by assuming a complete round of probing is performed after the hijacking event. We will study the detection delay by incorporating timing factors into the simulation in Section III-F and further study the detection delay using Internet hijacking experiments in Section VI.

Results: We first present the results assuming $\mathrm{abort\text{-}after} = 1$. Table III shows the percentage of hijack instances that result in a small number of cuts for each victim category and for all victims overall. Using a detection threshold cut number of 5, 10, and 20, the percentages of missed instances are 0.45%, 0.45%, and 0.57%, respectively.

To gain insight into these hijacking instances, we show the percentage distribution of the polluted ASs and the distribution of the topological signature $|\Omega|$ of these instances in Fig. 2(a) and (b). We see that 99.5%, 99.5%, and 99.4% of the instances have $|\Omega|$ more than 5, 10, and 20, respectively, which confirms our conjecture that $|\Omega|$ due to hijacking is typically large. Fig. 2(c) further shows the correlation between the pollution caused by a hijack and its resulting cut number $|\Omega|$. It confirms

[3]Configuring traceroute to tolerate more than three unreachable AS hops, which translates into many more unreachable IP hops, can incur high probing overhead.

TABLE IV
CUTS IN HISTORICAL HIJACKING EVENTS

Victim prefix	Victim prefix owner		Attacker	Pollu. (%)	$\|\Omega\|$
64.233.161.0/24	Google	15169	Cogent	31.6	492
63.165.71.0/24	Folksamerica	26913	ConEd.	65.7	458
64.132.55.0/24	OverseasMedia	33477	ConEd.	33.1	176
65.115.240.0/24	ViewTrade	23004	ConEd.	49.4	369
65.209.93.0/24	LavaTrading	35967	ConEd.	16.4	221
66.194.137.0/24	MacKayShields	31860	ConEd.	32.3	261
66.207.32.0/20	ADI	23011	ConEd.	83.0	594
69.64.209.0/24	TheStreet.Com	14732	ConEd.	78.0	658
160.79.45.0/24	RhodesASN	33313	ConEd.	27.5	380
192.251.16.0/24	T&TForex	20179	ConEd.	14.7	170
198.15.10.0/24	TigerFund	5703	ConEd.	86.0	707
204.13.72.0/24	FTENNY	33584	ConEd.	34.6	205
216.223.46.0/24	SDSNY	12265	ConEd.	77.6	606

the intuition that the number of cuts, which largely corresponds to the boundary between the region of polluted ASs and that of unpolluted ASs, is high when close to half of the ASs are polluted, and low when the pollution is either very high or very low.

We found that the results in Table III stay the same when changing *abort-after* to 2 or 3. This suggests that when a hijack results in a small number of cuts in vPath, the cut number varies little under different traceroute configurations. When a hijack results in a large number of cuts, we found that assuming $\mathrm{abort\text{-}after} = 1$ actually estimates fewer cuts, compared to assuming larger *abort-after*, as shown in Fig. 2(b). The intuition is as follows. When traceroute is configured to tolerate more consecutive unreachable AS hops, the cuts discovered based on our cut definition tend to be further away from the prefix-owner, and hence the cuts discovered by probing different destinations are more likely to be distinct. Hence, the total number of distinct cuts is larger.

E. Detecting Known Hijacking Events

We now validate our observation on the cut size using known hijacking events. We simulate the list of known hijacking events studied in [12]. As in Section III-D, for each hijack, we first reconstruct the forward path from the victim AS to every transit AS, and then calculate the set of unique cuts observed by the victim. Table IV shows the percentage of polluted transit ASs and the number of cuts for these hijacking events. Although the pollution varies from 14.7% to 86.0%, the number of cuts is above 170 for all hijacks.

F. Probing Strategies for Early Detection

In addition to serving as the unreachability signature of prefix hijacking, a large value of $|\Omega|$ compared to the detection threshold also suggests that the threshold number of cuts is likely to be observed long before completing a full probing round, triggering early detection of a hijacking event. As the probing order of the ASs potentially affects how soon the threshold number of cuts can be observed, we study several probing strategies in the following.

We state the probing order problem as follows. Given a probing round that starts at time t_0, a hijacking event that happens at a prior unknown time t_1 after t_0, and a detection threshold H, the probing order problem is to design a strategy of arranging the order of the ASs being probed, so that the detection latency $(t_2 - t_1)$ is minimized, where t_2 is the time when the number of cuts observed exceeds H. Since our goal is to compare different strategies, we assume in each strategy the ASs are probed one after another. Since the time it takes to probe different ASs may be different in practice, we make the following assumption on the probing cost. We assume traceroute is used, and use the distance in AS hops as the *virtual time* needed to probe an AS, which is roughly proportional to the number of IP hops needed in probing the AS.

Probing Strategies: Because vPath can be collapsed into a graph that resembles a tree except for a few loops, the most natural order to visit the nodes on this graph is to perform depth-first search (DFS) and breadth-first search (BFS). We denote the two corresponding probing strategies as DFS and BFS, i.e., to probe the ASs in DFS and BFS order on T_{old}, respectively. Since it is not obvious whether and how the cuts are clustered when ASs are probed in the DFS or BFS order, we also consider the *random* strategy that probes the ASs in a random order. Intuitively, following the random order effectively achieves randomly sampling, which is known to be resilient to any biased distribution.

We further study two hybrid strategies, namely $\mathrm{RandSibling_{all}}$ and $\mathrm{RandSibling_1}$. In $\mathrm{RandSibling_{all}}$, we probe ASs in a random order if no cuts are found, and once a cut XY is found, the siblings of Y, i.e., all other children of X, are probed next. Similarly, in $\mathrm{RandSibling_1}$, we probe ASs in a random order if no cuts are found, and once a cut XY is found, a sibling of Y is selected randomly and probed next. In both strategies, we make sure that siblings that have been probed before will not be probed again. These two strategies are based on the intuition that pollution and cuts tend to cluster at topologically close regions, and probing the siblings of a cut may have a higher probability of discovering new cuts than elsewhere.

A technique that can be incorporated into the search strategies is *pruning*. Since the goal of early detection is to quickly find a certain number of distinct cuts, probing an AS that will generate a known cut would not help to increase the number of distinct cuts $|\Omega|$. Thus, an AS X is pruned, i.e., not probed, if the path from the victim to AS X in vPath T_{old} contains a cut discovered while working on the current vPath T_{new}.

Simulation Methodology: Since it is difficult to predict the performance of the above strategies without knowing the distribution characteristics of the cuts on the graph, we use simulation to evaluate the goodness of these strategies.

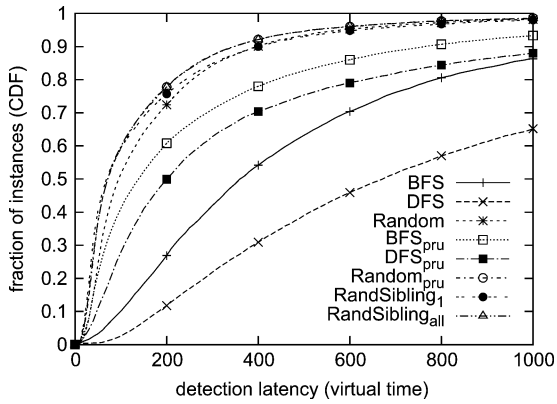


Fig. 3. Comparison of detection latencies of different probing strategies.

Our simulation follows the same methodology as the simulation in Section III-D. We study the same 2450 hijacking instances. The vPath snapshots before and after a hijacking event are computed as discussed in Section III-D. Because the detection latency depends on the start time of a hijack relative to the start time of a probing round, we simulate various start time of a hijack. For each hijacking instance, we start the hijacking after a fraction α of a probing round is finished, where $\alpha = 0, 0.1, 0.2, \ldots, 0.9$. The cuts are calculated based on continuously updated snapshots T_{old} and T_{new} defined in Section III-B. Note that if a hijack is started close to the end of a probing round, e.g., $\alpha = 0.9$, the number of distinct cuts seen by the end of the round may not be large enough to trigger detection. In this case, a second round will follow since the probing is continuous. One limitation of simulation is that we cannot easily simulate how fast the pollution spreads. We assume that the spread of pollution is instantaneous, i.e., after a hijacking starts, subsequent probes would start to witness polluted ASs. As we show later in Sections IV-B and VI-B, pollution spreads in less than 2 min due to the fast convergence of new route announcement, and this latency is relatively short compared to the duration of a probing round, typically 15–20 min.

Results: We simulated all of the eight strategies discussed. Each strategy is evaluated on the 2450 hijacking instances, each of which is repeated 10 times using 10 α-values. Detection is triggered when the number of cuts reaches threshold 10. Fig. 3 shows the cumulative distribution function (CDF) of detection latency of each strategy over the 24 500 hijacking instances. First, we see that pruning improves all three strategies, especially for DFS, because in DFS, offsprings of an AS X in the vPath-collapsed graph are probed immediately after X is probed, and these offsprings generate no new cuts if X already generates one. Second, we see that the strategies in the random family Random, $\mathrm{Random_{pru}}$, $\mathrm{RandSibling_1}$, and $\mathrm{RandSibling_{all}}$ are the best strategies. Compared to the 80th-percentile detection latency of $\mathrm{Random_{pru}}$, the 80th-percentile detection latency of $\mathrm{BFS_{pru}}$ and $\mathrm{DFS_{pru}}$ are respectively 2.0 and 2.9 times as long.

Fig. 4 explains why $\mathrm{Random_{pru}}$ has shorter detection latency than $\mathrm{BFS_{pru}}$ and $\mathrm{DFS_{pru}}$. This figure shows the number of cuts

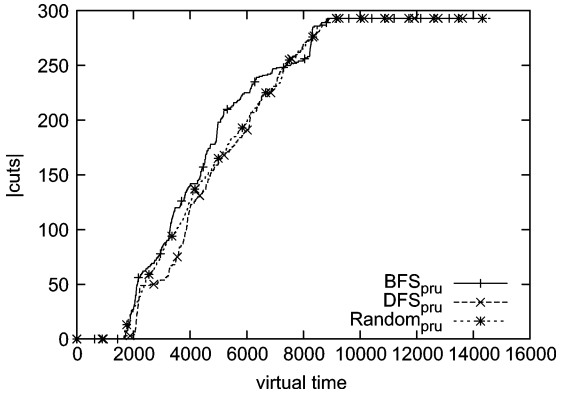


Fig. 4. The value of $|\Omega|$ as the probing round progresses in a particular hijacking instance.

as the probing round progresses by following three strategies. The hijacking event starts after 0.1 fraction of the round is finished. The $\mathrm{Random}_{\mathrm{pru}}$ curve maintains relatively stable slope. In contrast, $\mathrm{BFS}_{\mathrm{pru}}$ and $\mathrm{DFS}_{\mathrm{pru}}$ curves both show zigzag shape in some part of the curves. This observation indicates that the cuts discovered have some clustering when ASs are probed in the DFS or BFS order. Apparently, this clustering property has negative impact on the overall performance of the two strategies. We closely examined the places where cuts are clustered and found no strong correlation of these places and siblings. This observation explains why hybrid strategies $\mathrm{RandSibling}_1$ and $\mathrm{RandSibling}_{\mathrm{all}}$ are no better than $\mathrm{Random}_{\mathrm{pru}}$.

Summary: We studied the probing order problem for early detection. Our simulation results show that probing ASs in a random order achieves shorter detection latency than in the DFS or BFS order. Adding pruning further improves the 80th-percentile detection latency of *random* strategy by around 18%. For our detection scheme, we adopt the *random* strategy without pruning due to its simplicity.

IV. Prefix-Owner-Initiated Probing

In this section, we present the design of the probing module of iSPY. The probing module performs continuous rounds of probing to other ASs in the Internet to obtain the reachability and the AS-level paths to these ASs, which are used to construct the vPath used by iSPY to perform hijacking detection.

A. Design

The probing module faces several design challenges. First, the probing needs to be *lightweight* and *efficient* in order to scan the large number of ASs in the Internet in short intervals. Second, the probing mechanism has to be carefully designed to minimize the impact of probing unfriendly configurations and events in the Internet such as firewall, ICMP rate limiting, and congestion.

The probing module of iSPY aims to successfully probe at least one live IP per AS, via a combination of traceroute, ping, and TCP ping. Each network that deploys iSPY collects in advance and continuously maintains a database of live IPs using active probing and an IP-to-AS mapping. The live IPs respond either to ICMP ping (these IPs are called pingable IPs) or TCP connect at port 80 (these IPs are called Web IPs). iSPY launches traceroute to a single live pingable IP of each selected AS, but also retries traceroute to a different live pingable IP if the first traceroute fails to reach that AS. The traceroute probing is further complemented with ICMP ping probing if traceroute fails. Like traceroute, ICMP ping also tries at most two live pingable IPs. If ICMP ping still cannot reach that AS, TCP ping to port 80 of at most two Web IPs is attempted. In the following, we present more details on the major components of the probing module.

Probing Only Transit ASs: As discussed in Section III-A, to reduce the probing cost, iSPY probes only transit ASs, not stub ASs. This optimization reduces the total number of ASs to be probed from 23 191 to 3742 (transit ASs).

Live IPs: We created a database of live IPs as follows. We collected a set of probing candidate IPs from three sources: 1) x.x.x.1 of each announced prefix seen in RouteViews routing tables; 2) IPs found in a university department DNS server log; 3) Web client IPs found in a university department Web server log. We then expanded this set by performing traceroute to each candidate IP; the IPs that appear along the traceroute paths are added into the candidate set. Since these IPs may not be globally routable, we used ping to test the liveness of these added candidate IPs and filter out unresponsive IPs. Altogether, the set of live IPs we collected has a high coverage of all transit ASs: 3470 (92.7%) of the total 3742 transit ASs have at least one live IP, and 3464 (92.6%) of transit ASs have at least two live IPs. Throughout this paper, we use these 3470 transit ASs as the probing targets. We also incorporate Web IPs from the Web server list in [19], which covers 2997 transit ASs.

Resolving IP-Level Paths Into AS-Level Paths: Accurate IP-to-AS mapping is a challenging problem by itself [20] due to the lack of a uniform way of numbering router interfaces. Fortunately, iSPY can tolerate a certain amount of inaccuracy in IP-to-AS mapping as the cuts on the vPath are derived by comparing the current snapshot of the Internet reachability with a previous snapshot. Second, as shown in our Internet experiments in Section VI, there is a significant gap between the sizes of the cuts due to hijacking and link failures, and hence iSPY's detection module can tolerate some error in the cut size calculation. For simplicity, we used the BGP routing tables in RouteViews to generate IP-to-AS mapping. An IP prefix is mapped to the origin AS of its route announcement. Prefixes with multiple origins account for only 0.6% of the total prefixes. Such prefixes are marked as unmapped. The IP-level paths are resolved to AS-level paths by applying the IP-to-AS mapping. Some of the IP hops do not respond to traceroute, and appear as "*" in traceroute output. Such "*" hops as well as unmapped IP hops can still be resolved if both the previous hop and the next hop map to the same AS. For example, if "1239 * 1239" appears in the IP path, the "*" must belong to AS 1239. The remaining unmapped hops are translated into unresolved ASs.

Next, we collapse consecutive hops mapped to the same AS to produce the AS-level traceroute path. As in Section III-A, adjacent unresolved hops are collapsed together to a symbol "#," which is used to represent the uncertain part of an AS path. We also incorporate the results of ping probing into the AS path. If an AS d is not reached by traceroute but reached by ping, we append d to the end of the AS path obtained by traceroute. For example, if the AS path obtained by traceroute is $[sabc\#]$, but d is reached by ping, then we record the AS path as $P(d) =$

TABLE V
EFFICIENCY OF iSPY'S PROBING MODULE

	Among 108 sources		
	min	max	median
Avg hops per traceroute	10.7	19.9	15.5
Probing traffic per round (MB)	1.1	2.1	1.6
Time per traceroute (sec)	9.6	19.5	11.4
Probing time per round (min)	15	29	18
Bandwidth (KB/s)	0.8	2.2	1.5

$[sabc\#d]$. Such (maybe partially) resolved AS paths produce a snapshot of the vPath, i.e., the output of the probing module.

Increasing the Efficiency and Robustness of traceroute: We developed our own traceroute tool, iTraceroute [21], to perform probing. iTraceroute has the following features that enhance the efficiency and robustness over traditional traceroute. First, it can probe multiple destinations concurrently. Second, it performs IP-to-AS translation on the fly, i.e., hop IPs are translated into AS numbers when the TTL-expiration packets are received, which enables the traceroute to make intelligent early termination decision and avoid unnecessary work; whenever the current hop belongs to the destination AS, the traceroute is terminated. Third, since the goal of our probing is not to measure the per-hop delay, instead of sending three probes for each hop (per TTL), iTraceroute serially sends up to three probes, i.e., to stop sending more probes if one probe successfully replies. We believe that these functionalities are applicable to general Internet-wide lightweight probing. Existing traceroute implementations such as Paris traceroute [22] and NANOG traceroute fall short because none of them supports all of the above three functionalities.

B. Evaluation

We evaluate the performance of our probing design in the scenarios when there is no prefix hijacking. Such scenarios constitute the common case since hijacking is a rare event. In particular, we evaluate the probing efficiency, which determines the *real-time* and *lightweight* properties of iSPY, and the probing coverage, which affects the *accuracy* of iSPY.

Efficiency: We study the probing efficiency using a deployment of the iSPY probing module on 108 geographically diversely located PlanetLab nodes. The module on each node probes the 3470 transit ASs and is configured to maintain 50 concurrent traceroute threads. Table V shows the performance statistics on these 108 nodes. The probing module achieves high efficiency—finishing one probing round in 15–29 min, with low overhead—consuming only 0.8–2.2 kB/s bandwidth.

Coverage: We next evaluate the coverage of our probing module. The purpose of probing is to test the reachability of other ASs and discover the AS-level paths to them. Accordingly, we measure what fraction of the ASs can be reached by the probing and how complete the collected AS-level paths are. As before, we ran the probing module on the 108 PlanetLab nodes. Each node probes the set of 3470 transit ASs.

First, we focus on one node at Princeton University, Princeton, NJ. Table VI lists the coverage in terms of ASs by traceroute probing, complementary ICMP and TCP ping probing, and the overall probing. traceroute probing successfully reaches 91.4% of the 3470 transit ASs, ICMP ping reaches 7.5%, and TCP ping reaches the remaining 1.1%. The failed traceroutes are due to traceroute filtering since the associated destinations are mostly reachable by ping. Overall, 3468 (99.9%) ASs are reached. Therefore, our probing module design achieves high coverage in measuring reachability. In terms of AS-level path discovery, complete AS-level paths are obtained for 76.7% of the ASs. The incompletely resolved paths are mainly due to unmapped "*" hops in traceroute. This imperfection of path discovery poses a challenge for locating cuts in the vPath by iSPY. We will show in Section V how to tackle this problem.

TABLE VI
PROBING COVERAGE AND BREAKDOWN OF DIFFERENT METHODS

	Transit ASes	
	Number	Percent
Traceroute stat		
Probed	3470	100.0%
Reached	3170	91.4%
AS-path completely resolved	2663	76.7%
AS-path incompletely resolved	807	23.3%
Has at least 1 unmapped IP hop	155	4.5%
Has at least 1 unmapped * hop	680	19.6%
Complementary ping stat		
Probed	300	8.6%
Reached	261	7.5%
Complementary TCP stat		
Probed	39	1.1%
Reached	37	1.1%
Traceroute + ping + TCP stat		
Reached	3468	99.9%
AS-path completely resolved	2663	76.7%

Second, we study the coverage on all 108 nodes. Across all nodes, the reachable AS coverage is 95.6%–100% with a median of 99.9%, and the portion of ASs having complete paths ranges between 69.7%–85.9% with a median of 81.0%. This latter property depends on how frequently the probing traverses traceroute-filtering networks, which depends on the location of the probing source network.

V. iSPY: PREFIX-OWNER-CENTRIC HIJACK DETECTION

The architecture of our prefix-owner-based hijacking detection system is simple: It integrates our observation of the unreachability signature of prefix hijacking with the carefully engineered probing module. The probing module continuously probes the transit ASs in the Internet, and the reachability and AS-level paths are streamed into the detection decision-making module to scan for the unreachability signature of prefix hijacking. As discussed in Section III-F, the transit ASs are probed in a random order to minimize detection latency. The probing module itself alone is a useful tool named iTraceroute, as discussed in Section IV-A. The implementation of iSPY [23] has around 5000 lines of C code for iTraceroute, and around 1000 lines of perl code for the decision-making module. In the following, we discuss two details in the decision-making module of iSPY.

Handling Uncertain Subpaths: Accommodating the uncertainty in resolving IP-level paths into AS-level paths as discussed in Section IV-A makes calculating the exact number of distinct cuts $|\Omega|$ difficult since we do not know if two cuts both containing the same starting node going into "#" are actually the same cut. However, our goal is not to calculate the exact value

of $|\Omega|$, but to compare the value with a threshold. We can calculate the lower bound and the upper bound of $|\Omega|$ and use both bounds to aid decision making. We calculate the lower bound by simply assuming that all uncertain cuts sharing the same starting node are the same, and calculate the upper bound by assuming that each uncertain cut is a different cut. To be conservative, we will use the lower bound to perform hijack detection, although, as we will show in Section VI via Internet experiments, the gap between the two bounds is small.

Decision Making: Although iSPY performs probing round by round, iSPY uses the vPath snapshots from continuously updated logical rounds to calculate the cuts on the fly. Whenever the lower bound of the number of cuts exceeds a predetermined threshold C, the decision-making module reports the occurrence of a hijack. As with any threshold-based decision system (e.g., [6]), the choice for the threshold cut value used in hijacking detection in iSPY represents a tradeoff between acceptable false negative and false positive ratios. Lowering the threshold value is likely to lower the false negative ratio but also increase the false positive ratio, while increasing the threshold value is likely to increase the false negative ratio but reduce the false positive ratio. Our analysis in Section III-D shows that using a threshold value of 10 leads to a low false negative ratio of 0.45%. In Section VI-A, we measure the false positive ratios from Internet experiments and analyze the tradeoffs between the two ratios with varying threshold values.

VI. Internet Experiments

In this section, we evaluate the detection *accuracy* of the iSPY system via a deployment of iSPY in the Internet, and evaluate its performance in action when there is no prefix hijacking and also when there is a hijacking.

A. PlanetLab Experiment

iSPY has been continuously running on over 100 PlanetLab sites since November 2007. Since hijacking is a rare event in the Internet, in this section, we use this deployment to evaluate the detection false positive ratio of iSPY. For this evaluation, we report on iSPY's deployment on 108 PlanetLab hosts (distinct prefixes) located in 88 ASs starting from December 1, 2007 for 25 days to validate our design hypothesis that non-hijacking events do not cause large numbers of cuts.

Detection Accuracy: Over the 25-day period, iSPY reported hijacking alarms for 0.17% of all the probing rounds across all 108 hosts, using a detection threshold of 10 cuts. This alarm rate would be 0.44% and 0.05% when assuming a detection threshold of 5 and 20 cuts, respectively.

We believe that the alarms reported were all false positive, i.e., all cases with the number of cuts larger than five were not due to hijacking. This is based on two pieces of evidence. First, these cuts never lasted for more than one round, whereas hijacking would generally last longer. Second, we did not find any MOAS announcement in RouteViews for these hosts' prefixes during the 25-day period.

Fig. 5 shows the false positive ratios of the 108 PlanetLab hosts when the detection threshold is 10. Of the hosts, 70% have false positive ratios below 0.2%, and 95% have false positive ratios below 0.5%. Only three hosts have false positive ratios higher than 0.5%.

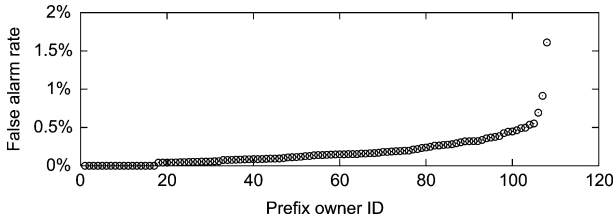


Fig. 5. False positive ratios of the 108 PlanetLab hosts during the 25-day period. Detection threshold was set to 10.

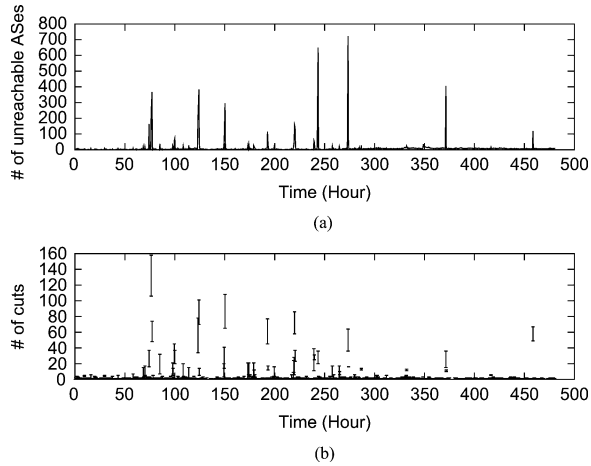


Fig. 6. Unreachable ASs and cuts witnessed by iSPY during its 500-h running on PlanetLab host stella.planetlab.ntua.gr. (a) Unreachable ASs. (b) Cuts (with lower and upper bounds).

To understand the cause for the cases with high false positive ratios, we closely examined the three hosts that triggered the most false alarms. Fig. 6 shows the number of unreachable transit ASs and the number of cuts witnessed by iSPY at stella.planetlab.ntua.gr that triggered the most alarms, a total of 22 alarms out of 1363 probing rounds, as indicated by the spikes in the figure. The total probing time was 500 h after removing the time when the host was down. We analyzed the rounds that triggered the alarms at the three hosts and found a common abnormal pattern in the probing outcomes. During these rounds, traceroute probes suffered more "*" hops in the middle of paths, and some of them aborted at random ASs, which were traversed by other probes in the same round. Some complementary ping and TCP ping probes also failed. This confused our detection system to generate cuts for a few inter-AS links. Another observation is that the round-trip times (RTTs) returned by successful traceroute, ping, and TCP ping probes are higher than normal. Moreover, such abnormal behavior never lasted for longer than one probing round. We suspect that the problem was caused by short-lived machine overload or congestion close to the probing machines, during which probes were dropped probabilistically.

Choice of Detection Threshold: The detection threshold used by iSPY affects both the false positive ratio and false negative ratio. To study the tradeoff between the two ratios, we plot the false positive ratios from this deployment study and the false negative ratios from Section III-D in Fig. 7, varying the detection threshold. The figure shows that the threshold of 10 cuts

TABLE VII
STATISTICS OF THE 15 HIJACKING EVENTS AND iSPY'S DETECTION PERFORMANCE

Event #	Scenario					iSPY Performance			
	Victim	Attacker	Hijack start time (GMT)	Pollution (%)	Cuts ([LB, UB])	Detected?	1st alarming round start time (min)	Detection latency (min)	Pollu. at alarm (%)
1	Seattle	London	Jan 22 14:00	35.6	[376, 409]	yes	-2.1	2.1	0.4
2	"	"	Jan 23 20:00	36.0	[383, 415]	yes	-4.0	2.1	0.4
3	"	"	Jan 25 02:00	36.0	[384, 417]	yes	-7.0	2.1	0.3
4	"	"	Jan 26 08:00	36.4	[382, 417]	yes	-4.7	2.7	0.4
5	"	"	Jan 27 14:00	36.1	[376, 409]	yes	-8.3	2.3	0.4
6	"	"	Jan 28 20:00	36.3	[379, 413]	yes	-2.6	2.7	0.5
7	Seattle	Tokyo	Jan 22 20:00	31.4	[205, 231]	yes	-2.7	2.7	0.5
8	"	"	Jan 24 02:00	31.0	[201, 226]	yes	-4.5	2.1	0.3
9	"	"	June 04 02:00	34.4	[219, 246]	yes	0.2	3.1	1.0
10	London	Seattle	Jan 27 02:00	51.3	[331, 376]	yes	-2.9	1.4	0.3
11	"	"	Jan 28 02:00	51.1	[328, 372]	yes	-2.7	1.4	0.4
12	Tokyo	Seattle	June 02 02:00	51.0	[788, 839]	yes	-0.4	3.1	0.4
13	"	"	June 02 06:00	52.3	[805, 855]	yes	-10.9	1.8	0.4
14	"	"	June 03 08:00	51.4	[785, 833]	yes	-5.5	2.3	0.4
15	"	"	June 03 14:00	51.2	[793, 841]	yes	-6.2	2.4	0.3

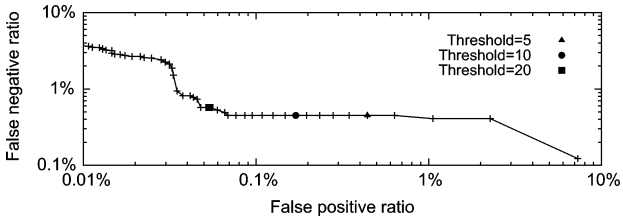


Fig. 7. False positive ratio versus false negative ratio.

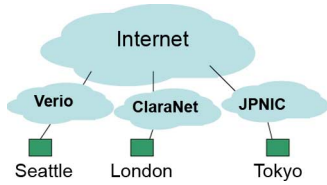


Fig. 8. Setup of our prefix hijacking test bed.

appears to strike a good balance between the two detection accuracy measures, giving a false positive ratio of 0.17% and a false negative ratio of 0.45%.

B. Hijacking Experiment

While our analysis in Section III has provided strong indication of iSPY's accuracy in detecting real hijacking events, we would like to validate its performance in real action. In addition to validating its detection accuracy, such Internet hijacking experiments allow us to study the detection latency of a deployed system in reporting real hijacks.

There have been historical cases of prefix hijacking in the Internet. However, our deployment of iSPY at over 100 PlanetLab sites has not witnessed any hijacking events to those networks so far. A further challenge with demonstrating iSPY in action is that it needs to be deployed by the prefix-owner network. To overcome these challenges, we set up a controlled hijacking test bed that launched a total of 15 hijacking attacks of our own prefix, and we deployed iSPY in this test bed and observed its live action during the hijacking events.

Experiment Setup: Our prefix hijacking test bed consists of three hosts located at three sites—Seattle, WA; London, U.K.; and Tokyo, Japan—running software BGP router to maintain BGP peering sessions with their upstream provider ISPs Verio (AS 2914), Clara.net (AS 8426), and JPNIC (AS 2497), respectively. This setup, shown in Fig. 8, allows us to inject an anycast prefix (198.180.153.0/24) from any of the three hosts to the Internet and hence emulate a few different hijacking scenarios.

For each hijacking event, we picked one site as the victim and another as the attacker. Initially, the victim injected the target prefix. Two hours later, the attacker also injected the prefix and hence started the hijacking event. Another 2 h later, the attacker withdrew the prefix. We performed a total of 15 hijackings during two one-week periods in January and June 2008. The times of hijacking events and the corresponding attackers and victims are listed in Table VII.[4] During these days, iSPY was continuously running on the hosts at the victim sites, with the detection threshold set to 10 cuts.

We note that while the hijacking events were controlled to start at some specific time, from iSPY's point of view, these hijacks happened at "random time" as its probing module was simply continuously running, i.e., the start time of probing rounds was not aligned with the start time of hijacks in any deliberate way.

Details of Detecting One Hijack: We first discuss one hijacking event in detail and, in doing so, introduce all the entries shown in Table VII. This hijack is event 1 in Table VII. It started on January 22, 2008, at 14:00 GMT by the London site and, upon convergence, rendered 35.6% of the transit ASs unreachable from the Seattle site, which resulted in cuts in the vPath in the range of $[376, 409]$. Fig. 9(a) and (b) show the number of unreachable ASs and the number of cuts computed at the boundary of probing rounds (marked with dots) during the 2 h before and the 2 h after the hijack injection moment of 14:00 GMT. Note that the probing rounds have shorter durations before the hijack than during the hijack because, during the hijack, the probing module had to retry alternative IPs for unreachable ASs and hence proceeded slower than before hijacking. We observe that before the hijack happened, the numbers of unreachable ASs and of cuts were both insignificant. At the hijack injection moment, the current probing round had been running for

[4]We could not enumerate all six hijacking scenarios due to machine unavailability.

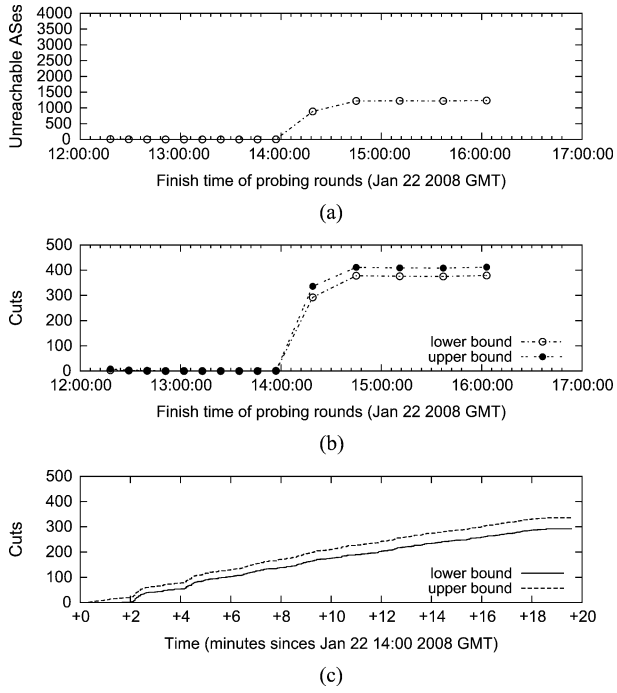


Fig. 9. Number of unreachable ASs and the number of cuts as the probing progresses in event 1. (a) Unreachable ASs throughout the event. (b) Cuts throughout the event. (c) Cuts in the first alarming round.

2.1 min. As the probing progressed, iSPY updated the observed cuts continuously as shown in Fig. 9(c). At 2.1 min after the hijack injection moment, the detected number of cuts reached the detection threshold and triggered the alarm. Hence, the detection latency was 2.1 min, well before the round finished 19 min later.

Overall Detection Performance: Table VII shows the details of all 15 hijacking events, all of which were detected definitively by iSPY. Upon convergence, the pollution ranges from 31.0% to 52.3%, and the number of cuts are all above 200, which again confirms our observation made in Section III that hijacking typically creates a large number of cuts in vPath. We call the first probing round in which the alarm is triggered "first alarming round" in Table VII. For events other than 9, the alarm was triggered by the probing round that was in progress when hijacking happened, and hence their start time relative to the hijacking event is negative. For event 9, such a round did not collect enough cuts to trigger the alarm, but due to the continuous way cut is computed, these cuts are accumulated to the round immediately after, and the alarm was triggered quickly in that round.

In summary, Table VII shows that despite the start time of the first alarming round varying from 10.9 min before to 0.2 min after the hijack injection, the detection latency remains consistently low, in the range of 1.4–3.1 min. We attribute this low detection latency and low sensitivity to the timings of hijacks and probing rounds to the fast BGP convergence of new route announcement [24] and iSPY's continuous decision-making mechanism. The last column of Table VII shows iSPY only witnessed 0.3%–1.0% of ASs were polluted when the alarm was raised. We note that the large pollution and number of cuts caused by these hijacking events contributed to the quick discovery of enough polluted ASs and cuts and hence the low detection latency. For hijacks that lead to fewer cuts, the detection latency is likely to be longer. Finally, during the hours when there was no hijacking, iSPY did not report any false alarm at the three sites.

VII. Discussion and Future Work

Countermeasures: One potential countermeasure by attackers to data-plane-based techniques is to modify the information contained in the probing replies (e.g., [6]). Doing so under a prefix-owner-centric scheme such as iSPY, where the victim AS probes a large number of ASs, requires the attacker AS to manipulate replies to all traceroute probes that are drawn to itself from polluted ASs. To forward probe replies back to the victim network, the attacker needs to maintain a valid route back to the victim, i.e., it needs to perform an interception attack [12], [13].

Another potential countermeasure is pollution shaping (e.g., [13]), i.e., to launch an attack with controlled pollution that causes few cuts. Such a controlled pollution may be achieved by manipulating the bogus route announcement, e.g., adding certain AS numbers to the bogus route when it is initially announced, which prevents the route from being adopted by those ASs. However, such a countermeasure is unlikely to pose a threat for the following reasons. First, shaping a small-cut pollution is difficult. Evading pollution at a few specified ASs may not affect the number of cuts due to pollution, as the large number of peering and multihomed links of other polluted ASs will still lead to many cuts. On the other hand, if the bogus route is padded with too many ASs, its competitiveness and hence the impact of the hijack become limited. Second, to calculate the exact set of ASs to add to the initial bogus route is quite challenging. It requires predicting the pollution, which is difficult without knowing the exact AS topology and routing policies. We will investigate these and other countermeasures that attackers may launch against iSPY as part of our future work.

Other Hijacking Attacks: Our scheme is designed to address the basic type of hijacking attacks, namely the regular prefix hijacking. Other types of hijack attacks include subprefix hijack and interception-based hijack. In a subprefix hijack, unless the attacker deliberately shapes the pollution, the bogus route for the hijacked subnet will be propagated globally, including to the prefix-owner, and hence subprefix hijacking can be easily detected via simple control-plane techniques such as examining the BGP updates. Note that subprefix hijacking coupled with pollution shaping can potentially create a type of stealthy hijacking. For example, by using a fake AS path that includes the set of ASs an attacker wants to avoid polluting, the attacker can effectively spread pollution to all other ASs on the Internet using bogus announcement of a subprefix. Such a type of attacks poses threat to not only our scheme, but also all vantage-point-based schemes, as the attacker can evade the detection by avoiding polluting all the vantage points. We also do not address interception-based hijack [12], [13], as the victim will not observe any changes in reachability. Interception-based hijack can be partially addressed by solutions such as encryption [25] for ensuring data confidentiality and integrity. We plan to investigate performance-based approaches

(e.g., hop count [6] and delay) in our victim-centric detection framework as future work.

Design Optimizations: Since iSPY will be deployed by the operators of individual networks, its probing module can passively leverage incoming data traffic into the network to reduce the active probing cost. We expect there is significant temporal diversity in the incoming data traffic to an AS. A similar idea was exploited in PlanetSeer [26].

When a network owns multiple prefixes, it can potentially improve iSPY's detection accuracy of hijacking events by comparing vPaths generated using probes from different prefixes. Dissimilar cuts, e.g., with some above the threshold and others below, in these vPaths potentially provide further evidence of hijacking of some of its prefixes. However, a detailed design exploiting this idea needs to take into consideration the possibility of all of the network's prefixes being hijacked, as well as legitimate reasons for different vPaths. Another interesting direction to pursue is whether selecting a "personalized" threshold by each network can improve its detection accuracy.

Finally, although the identity of the attacker can potentially be discovered from RouteViews [2], it remains interesting, and we are currently exploring ways to extend iSPY to also identify the attacker in real time.

VIII. Related Work

iSPY is closely related to previous work on IP prefix hijack detection [2], [3], [5], [6]. As summarized in the introduction, none of the existing detection systems effectively detect IP prefix hijacks while satisfying all six requirements. The prefix-owner-based probing in iSPY is intuitive and demonstrated to be effective. Unlike all previous data-plane-based approaches, the coverage of iSPY is less limited by probing-unfriendly configurations and events in the Internet such as traceroute blocking, as we only need to reach transit networks. Since iSPY is designed to be deployed by prefix-owner networks to protect their own prefixes, it does not detect hijacks of the unused portion of the prefix address space [27]. Note that the focus of our work on distinguishing reachability loss patterns due to hijacking from those due to other failures is a simpler problem than BGP root cause analysis [28], [29].

When prefix hijacking events are detected, it is helpful to identify the attackers. Achieving that is nontrivial for data-plane-based detection schemes because they do not directly observe the bogus route announcement. Recently, the authors of [6] proposed LOCK to narrow down the set of possible attackers [7]. The natural next step of action is to mitigate the impact of hijacking. Many existing mitigation schemes include manual response to install filters, ACR [25], MIRO [30], route purge-promotion [9], and overlay routing. Detection followed by reactive mitigation falls into the category of passive countermeasures against prefix hijacking. A number of proactive prevention solutions have also been proposed [8], [31]–[34]. Another area related to prefix hijacking is on impact analysis [4], [35].

Finally, the probing module design of iSPY builds on previous work of lightweight end-host-based monitoring systems such as Rocketfuel [36], PlanetSeer [26], iPlane [11], and Hubble [10]. Our work also relates to previous work on using data-plane information to troubleshoot routing problems such as missing routes [37] and bogon prefixes [38].

IX. Conclusion

This paper proposes a prefix-owner-based IP prefix hijacking detection system iSPY. The design of iSPY is based on a key observation we have made on prefix hijacking: Due to the rich connectivity of the ASs in the Internet, a prefix hijack almost always pollutes a significant percentage of the ASs. More importantly, this unreachability has a different signature from that due to a few link failures near the victim's network, which can also result in unreachability to a large number of ASs. iSPY is designed to be readily deployed by a prefix-owner network. It continuously monitors network reachability from external transit networks to its own network through lightweight probing and scans for the hijacking signature as the trigger for hijacking alarms.

The prefix-owner-centric design enables iSPY to satisfy all the requirements of a highly effectively prefix hijacking detection system: 1) it is highly accurate with both low false positive and negative ratios, and its detection accuracy is not limited by the placement of any vantage points; 2) it is real-time, showing a detection latency of 1.4–3.1 min in our hijacking experiments; 3) it is lightweight with the probing traffic rate between 0.8–2.2 kB/s, as it is fully decentralized among the prefix-owner networks, each of which only needs to monitor reachability to the transit networks; 4) it is readily deployable by any prefix-owner network; 5) it creates strong incentive for deployment as deployment by each prefix-owner network directly benefits itself; and 6) it is intrinsically robust in victim notification as the prefix-owner makes hijacking detection decision locally. Our work departs significantly from the existing infrastructure-based designs of hijacking attack detection and demonstrates the effectiveness of end-host-based probing and analysis.

References

[1] P. Boothe, J. Hiebert, and R. Bush, "How prevalent is prefix hijacking on the Internet," in *Proc. NANOG36 Talk*, Feb. 2006.

[2] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "PHAS: A prefix hijack alert system," in *Proc. USENIX Security Symp.*, 2006, Article no. 11.

[3] J. Qiu, L. Gao, S. Ranjan, and A. Nucci, "Detecting bogus BGP route information: Going beyond prefix hijacking," in *Proc. SECURECOMM*, 2007, pp. 381–390.

[4] Y. Zhang, Z. Zhang, Z. M. Mao, Y. C. Hu, and B. Maggs, "On the impact of route monitor selection," in *Proc. ACM SIGCOMM IMC*, 2007, pp. 215–220.

[5] X. Hu and Z. M. Mao, "Accurate real-time identification of IP prefix hijacking," in *Proc. IEEE Security Privacy*, 2007, pp. 3–17.

[6] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A light-weight distributed scheme for detecting IP prefix hijacks in real-time," in *Proc. ACM SIGCOMM*, 2007, pp. 277–288.

[7] T. Qiu, L. Ji, D. Pei, J. Wang, J. Xu, and H. Ballani, "Locating prefix hijackers using LOCK," in *Proc. USENIX Security Symp.*, 2009.

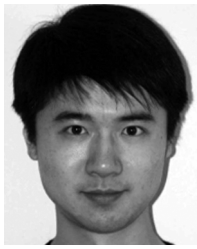
[8] J. Karlin, J. Karlin, S. Forrest, and J. Rexford, "Pretty good BGP: Improving BGP by cautiously adopting routes," in *Proc. ICNP*, 2006, pp. 290–299.

[9] Z. Zhang, Y. Zhang, Y. C. Hu, and Z. M. Mao, "Practical defenses against BGP prefix hijacking," in *Proc. ACM CoNEXT*, 2007, Article no. 3.

[10] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson, "Studying black holes in the Internet with Hubble," in *Proc. NSDI*, 2008, pp. 247–262.

[11] H. V. Madhyastha, T. Anderson, A. Krishnamurthy, N. Spring, and A. Venkataramani, "iPlane: An information plane for distributed services," in *Proc. OSDI*, Nov. 2006, pp. 367–380.

[12] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the Internet," in *Proc. ACM SIGCOMM*, Aug. 2007, pp. 265–276.
[13] A. Pilosov and T. Kapela, "Stealing the Internet: An Internet-scale man in the middle attack," in *Proc. 16th Defcon*, 2008.
[14] "University of Oregon Route Views archive project," Advanced Network Technology Center, Univ. Oregon, Eugene, OR, 2005 [Online]. Available: http://www.routeviews.org
[15] J. Wu, Y. Zhang, Z. M. Mao, and K. Shin, "Internet routing resilience to failures: Analysis and implications," in *Proc. ACM CoNEXT*, 2007, Article no. 25.
[16] "RIPE routing information service (RIS)," RIPE NCC, Amsterdam, The Netherlands [Online]. Available: http://www.ripe.net/ris/
[17] L. Gao, "On inferring autonomous system relationships in the Internet," in *Proc. IEEE GLOBECOM*, 2000, vol. 1, pp. 387–396.
[18] B. Huffaker, "CAIDA AS ranking project," CAIDA, La Jolla, CA, Jul. 2006 [Online]. Available: http://ww.caida.org/analysis/topology/rank_as/
[19] C. A. Shue, A. Kalafut, and M. Gupta, "The Web is smaller than it seems," in *Proc. ACM SIGCOMM IMC*, 2007, pp. 123–128.
[20] Z. M. Mao, J. Rexford, J. Wang, and R. Katz, "Towards an accurate AS-level traceroute tool," in *Proc. ACM SIGCOMM*, 2003, pp. 365–378.
[21] Y. C. Hu, "iTraceroute," DSNL, Purdue Univ., West Lafayette, IN, 2009 [Online]. Available: http://www.ece.purdue.edu/~ychu/itraceroute
[22] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute," in *Proc. ACM SIGCOMM IMC*, 2006, pp. 153–158.
[23] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush , "iSPY," [Online]. Available: http://www.ece.purdue.edu/~ychu/ispy
[24] R. Oliveira, B. Zhang, D. Pei, R. Izhak-Ratzin, and L. Zhang, "Quantifying path exploration in the Internet," in *Proc. ACM SIGCOMM IMC*, 2006, pp. 269–282.
[25] D. Wendlandt, I. Avramopoulos, D. Andersen, and J. Rexford, "Don't secure routing protocols, secure data delivery," in *Proc. ACM HotNets*, 2006.
[26] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang, "PlanetSeer: Internet path failure monitoring and characterization in wide-area services," in *Proc. OSDI*, Dec. 2004, vol. 6, p. 12.
[27] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," in *Proc. ACM SIGCOMM*, 2006, pp. 291–302.
[28] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating Internet routing instabilities," in *Proc. ACM SIGCOMM*, 2004, pp. 205–218.
[29] R. Teixeira and J. Rexford, "A measurement framework for pin-pointing routing changes," in *Proc. ACM Workshop Netw. Troubleshooting*, 2004, pp. 313–318.
[30] W. Xu and J. Rexford, "MIRO: Multi-path interdomain routing," in *Proc. ACM SIGCOMM*, 2006, pp. 171–182.
[31] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (Secure-BGP)," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 4, pp. 582–592, Apr. 2000.
[32] J. Ng, "Extensions to BGP to support secure origin BGP (soBGP)" IETF Draft: draft-ng-sobgp-bgp-extensions-01.txt, Nov. 2002.
[33] Y.-C. Hu, A. Perrig, and M. Sirbu, "SPV: A secure path vector scheme for securing BGP," in *Proc. ACM SIGCOMM*, 2004, pp. 179–192.
[34] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz, "Listen and whisper: Security mechanisms for BGP," in *Proc. NSDI*, 2004, pp. 127–140.
[35] M. Lad, R. Oliveira, B. Zhang, and L. Zhang, "Understanding resiliency of Internet topology against prefix hijack attacks," in *Proc. DSN*, 2007, pp. 368–377.
[36] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.
[37] D.-F. Chang, R. Govindan, and J. Heidemann, "Exploring the ability of locating BGP missing routes from multiple looking glasses," in *Proc. ACM Workshop Netw. Troubleshooting*, 2004.
[38] R. Bush, J. Hiebert, O. Maennel, M. Roughan, and S. Uhlig, "Testing the reachability of (new) address space," in *Proc. ACM SIGCOMM INM*, 2007, pp. 236–241.



Zheng Zhang received the Bachelor's degree in computer science from the University of Science and Technology of China, Hefei, China, in 2003, and is currently a fifth-year Ph.D. candidate at the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN.

He has interned at the IBM Almaden Research Center, San Jose, CA, and Microsoft Research, Redmond, WA. His research interests include Internet routing and measurement, peer-to-peer and overlay networks, and file caching and prefetching.

Ying Zhang received the B.S. degree in computer science from Beijing University, Beijing, China, in 2004. and is currently a fifth-year Ph.D. candidate with the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor.

She has interned at AT&T Labs-Research, Florham Park, NJ, and Microsoft Research, Redmond, CA. She has a broad research interest in the area of network monitoring, Internet routing, network measurement, and network security.

Y. Charlie Hu (S'91–M'98–SM'07) received the Ph.D. degree in computer science from Harvard University, Cambridge, MA, in 1997.

He is an Associate Professor of electrical and computer engineering and computer science (by courtesy) with Purdue University, West Lafayette, IN. From 1997 to 2001, he was a Research Scientist with Rice University, Houston, TX. His research interests include operating systems, distributed systems, Internet measurement and routing analysis, and wireless networking.

Dr. Hu is a Senior Member of the Association for Computing Machinery (ACM). He received the NSF CAREER Award in 2003.

Z. Morley Mao received the B.S. degree in electrical engineering and computer science and the M.S. and Ph.D. degrees in computer science from the University of California, Berkeley.

She is an Assistant Professor of electrical engineering and computer science with the University of Michigan, Ann Arbor. Her research interests include network measurements, routing protocols, distributed systems, and network security.

Dr. Mao is a recipient of the NSF CAREER Award and the IBM Faculty Partnership Award.

Randy Bush is a Senior Researcher and Network Operator with Internet Initiative Japan, Tokyo, Japan. He specializes in network measurement, especially routing, network security, routing protocols, and IPv6 deployment. He has been in computing for 45 years and has a few decades of Internet operations experience. He was the engineering founder of Verio, which is now NTT/Verio. He has been heavily involved in transferring Internet technologies to developing economies for over 20 years.