

# iSPY: Detecting IP Prefix Hijacking on My Own

Zheng Zhang  
Purdue University

Ying Zhang  
University of Michigan

Y. Charlie Hu  
Purdue University

Z. Morley Mao  
University of Michigan

Randy Bush  
IIJ

## 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, light-weight, 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 ASes (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.

## Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General—*Security and protection (e.g., firewalls)*; C.2.2 [Computer-Communication Networks]: Network Protocol—*Routing Protocols*; C.2.3 [Computer-Communication Networks]: Network Operation—*Network Monitoring*

## General Terms

Measurement, Security

## Keywords

Routing, BGP, Hijacking, Detection

## 1. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.  
Copyright 2008 ACM 978-1-60558-175-0/08/08 ...\$5.00.

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 AS for the attack. Such a detection system should satisfy all of the following critical requirements:

**Real-time:** Detection should be real-time to identify short-lived attacks and minimize potential damage.

**Accurate:** The detection accuracy must be high, with both low false positive and false negative ratios.

**Light-weight:** Detection should be light-weight and scale well with the number of protected IP prefixes and networks without sacrificing the detection accuracy.

**Easy to deploy:** The detection system can be easily deployed incrementally without requiring privileged access to data such as live BGP feeds from many ASes.

**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 wide-spread deployment.

**Robust in victim notification:** The system is able to notify the victim (owner) of the hijacked IP prefix in a robust fashion. In addition, it is desirable that the system accurately identifies and notifies the polluted networks.

Most existing proposals on prefix hijack detection fall into three categories, as summarized in Table 1, based on the type of information used. The first category of control-plane-based approaches (e.g., [5, 18, 25]) 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 [35] and legitimate reasons for anomalous updates [12]. The timeliness of such an approach heavily relies on access to real-time BGP feeds.

The second category of detection systems (e.g., [12]) collect real-time information from both the control plane and the data plane to perform joint analysis and hence are real-time, but they also require 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. The third

**Table 1: Comparison among prefix hijacking detection systems.**

Requirement	Control-plane-based (passive) [5, 18, 25]	Control-plane & data-plane-based [12]	Data-plane-based [37]	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	X	X	X	✓

category (e.g., [37]) 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 have devised a robust way to notify the victim (owner) of the hijacked IP prefix nor polluted networks, which is the final but also a crucial step of the prefix hijack detection process.

In this paper, we present an IP prefix hijacking detection system that satisfies all of the above requirements. Our proposed system, iSPY, exploits a key observation about IP prefix hijacking: due to the rich connectivity of the ASes in the Internet, a prefix hijack almost always pollutes a significant percentage of the ASes, i.e., those ASes 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 ASes and detects hijacking events based on the observed reachability to these ASes.

A fundamental difference between iSPY and previous approaches using data-plane information [37] 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*, same as previous proposals, but also exhibit 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) *light-weight* as it is fully decentralized among the prefix owner networks, and each prefix owner just needs to continuously probe the over 3000 transit ASes, and (4) intrinsically *robust in victim notification* as the prefix owner makes hijacking detection decision locally. Furthermore, the prefix-owner-initiated probing for AS-level paths in iSPY avoids the firewall problem in previous vantage-point-based probing: since the probing is initiated from inside the network, the probe packets can usually exit the prefix owner’s network. Most transit networks enable ICMP replies; thus, the probes can effectively test reachability to such networks. 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.

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 ASes (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. Despite the high reachability problems with over 10% of the prefixes in the edge networks as recently reported by the Hubble system [16], our deployment of iSPY in 88 ASes over a 25-day period has shown that iSPY incurs a very low false positive ratio, from using the unique unreachability signature of prefix hijacking, and benefiting from the need to monitor only reachability to the prefixes of transit ASes which are generally much more stable than those of edge networks.

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 *light-weight* to ensure low probing traffic. It must be *efficient* so that each probing round can finish quickly 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., [16, 37, 20]) 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 ASes.<sup>1</sup> 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 1). 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 3). Third, we present the detailed design and implementation of the prefix-owner-centric probing mechanism demonstrating an effective working system (Section 4). Fourth, we conduct analysis and Internet experiments to validate iSPY’s effectiveness in action (Section 6). We demonstrate that iSPY is light-weight and can accurately detect prefix hijacking in real time with 0.45% false negative ratio and 0.17% false positive ratio.

## 2. 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

<sup>1</sup>Note that vantage-point-based monitoring systems can often make use of low overhead probes such as pings (e.g., [16, 37]).

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 or may even choose to selectively forward the traffic back to the victim [4].

For each AS  $n$ , it either receives the bogus route or does not at all observe it. 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 advertising the route 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 [12].

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 attack, namely the regular prefix hijacking. In a subprefix hijack, the bogus route for the hijacked subnet will likely 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. We also do not address interception-based hijack [4], as the victim will not observe any changes in reachability. Interception-based hijack can be partially addressed by solutions such as encryption [31]. We plan to investigate performance-based approaches (*e.g.*, hop count [37] and delay) in our victim-centric detection framework as future work.

### 3. KEY OBSERVATION

The design of ISPY exploits a key observation about IP prefix hijacking: a prefix hijack almost always pollutes a significant percentage of ASes 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 ASes. More importantly, this unreachability to many ASes 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 ASes. 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.

#### 3.1 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.

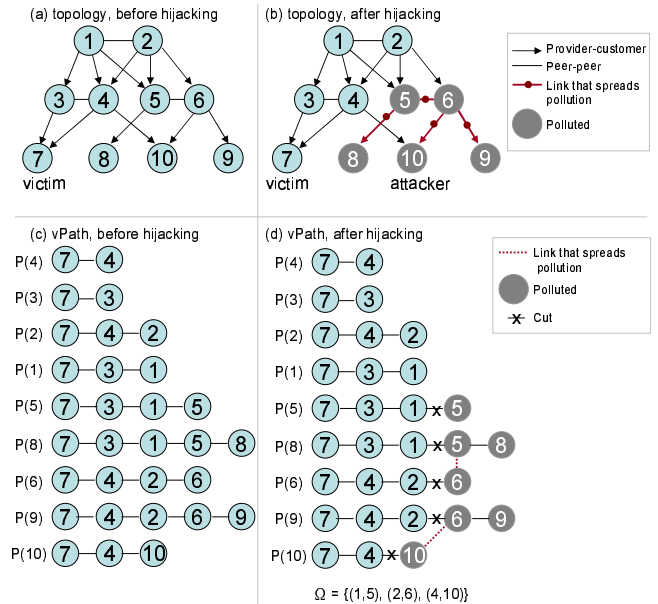


Figure 1: An example of prefix hijacking, vPath, and cuts.

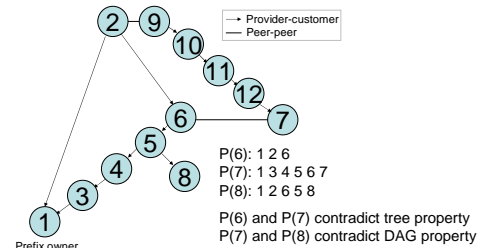


Figure 2: An example AS topology where collapsing the victim AS 1's paths in its vPath results in a graph containing a loop.

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, or simply, the victim network) to all the ASes 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.

The intuition behind *vPath* is clear. If an AS  $X$  is polluted by a prefix hijacking, replies to traceroute probes originated from the victim network to any prefix owned by  $X$  will not correctly reach back to the victim network. Therefore *vPath* captures the reachability from external networks to a potential victim network. If a destination AS has multiple prefixes, although the AS-level forward paths to these prefixes might be different, we just need to select any such prefix as a candidate and use the path to it as the path to the destination AS.

Figure 1(a) shows an example AS topology, with AS relationships (*e.g.*, provide-customer and peer-peer) implied by the relative position of ASes, and Figure 1(c) shows the *vPath* representation of AS 7's reachability to the rest of the ASes.

Before deciding on *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. Figure 2 shows an example where if the set of forward paths from an AS are collapsed into a directed graph, the graph contains loops, and hence the information of individual forward paths are 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 ASes 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.*,  $[a, b, \#, e]$ .<sup>2</sup>

**Monitoring Reachability to Only Transit ASes.** In practice, obtaining paths to the large number of ASes in the Internet can be costly. We monitor reachability to the much smaller number of transit ASes and restrict the notion of vPath to the paths to these transit ASes. Transit ASes are those ASes that forward traffic for other ASes. Non-transit ASes are called stub ASes. However, this sampling mechanism will not cause ISPY to miss any attack. This is because a hijacking originated at a stub AS will always result in polluting its provider transit AS and other transit ASes; otherwise, the attacker pollutes itself only.

### 3.2 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 ASes to take periodic snapshots of the 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 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<sup>3</sup> 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  $P(d) = [s, u_1, u_2, \dots, u_n, d]$  in  $T_{old}$  becomes unreachable in  $T_{new}$ . First, AS  $u_{i+1}$  is polluted with a path that goes to the prefix hijacker’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 a physical link failure, though the latter case of link failure is unlikely because the forward path can be used as backup when the backward path is broken. 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 well reflects the location of the cause of the reachability. We note that there could also be other reasons such as transient link congestion causing loss of probing packets 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,

<sup>2</sup>Another cause for “\*” is that some routers may not respond to traceroute and we discuss how we complement traceroute with other probes in Section 4.

<sup>3</sup>In practice,  $T_{old}$  is continuously updated to maintain recent paths to all monitored ASes that are complete.

**Table 2: 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#

in  $T_{new}$ , the new path  $P'(d) = [s, v_1, v_2, \dots, v_n, d]$  (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.

**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 completed yet.*

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

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 3.1. Table 2 shows the cuts defined under a few example scenarios.

Figure 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 ASes 5, 8, 6, and 9. Note the spreading of polluted paths for reaching back to the attacker respects AS relationship and valley-free routing. Figure 1(d) shows the vPath observed by the victim after the hijack, and the hijack creates three cuts to the vPath.

### 3.3 Unreachability Signature of Hijacking

We conjecture that  $\Omega$  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  $\Omega$  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 ASes 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 multi-homing links. As an evidence, the AS-level Internet topology that was inferred using routing table dumps in September 2007 from RouteViews [2] has 3742 transit ASes but 18384 links between the transit ASes.

The large number of peering links and multi-homing links have 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 multi-homing links (recursively), and then the peer networks and customer networks of such polluted networks. In other words, the pollution can be far beyond being confined to some 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 plus its customers plus its customers’ customers, *etc.*. As an example, in Figure 1(b), the pollution is not confined to the cone rooted at AS 6; it propagates to AS 5 via a peering link and then pollutes the cone rooted at AS 5.

Second, conversely, from the victim’s point of view, because of the rich connectivity of the ASes due to peering links and multi-homing links, the forward paths from the victim AS to the polluted ASes are likely to go through many diverse paths, as opposed to going through only the roots of the polluted cones. Such diverse paths going from the region of unpolluted ASes to the region of polluted ASes are what lead to many cuts in vPath. For example, in Figure 1(b), the multi-homing 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 side, as opposed to going through the roots of the polluted cones, AS 5 and AS 6, and 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 small cuts in vPath. Note link failures near the prefix-owner network, *e.g.*, at its provider links, may cause unreachability to a large number of ASes. 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 ASes in the Internet, and they can quickly recover from a single physical link failure or congestion by redirecting traffic to alternative egress points. Second, many ASes are multi-homed (even more so for transit ASes) and have several routes to the Internet. Although not always [16], such redundancy generally helps these ASes to stay connected in case of link failures. Third, and importantly, ISPY monitors the reachability to the transit ASes, which are generally more stable than stub ASes. The latter are generally considered in previous large-scale reachability studies such as [16].

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 small cuts [32]. Finally, our own Internet measurement study of prefix-owner’s view of Internet reachability (Section 6.1) 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.

### 3.4 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  $\Omega$ .

**Methodology.** We simulate 2450 hijacking instances on a realistic AS-level Internet topology. The AS-level Internet topology is ob-

**Table 3: The percentage of small  $|\Omega|$  instances.**

Victim category	Total instances	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%)

tained by combining six-month routing table snapshots and updates collected from more than 100 vantage points in RouteViews [2], RIPE RIS [1] and Abilene, and running Gao’s algorithm [10] on them to obtain the AS relationship. The obtained topology consists of 23,195 ASes. The ASes are classified into 5 categories by tier and transit/stub, namely, tier-1, tier-2 transit, tier-2 stub, tier-3+ (*i.e.*, tier $\geq$ 3) transit and tier 3+ stub. We define tier-1 ASes as those ASes that do not have providers and peering with all other tier-1 ASes [14], and the tier of an AS as the minimum number of providers that connect this AS to a tier-1 AS. 11 tier-1 ASes are inferred from the topology. Stub ASes are recognized as those ASes that always appear in the last AS hop in routing tables. Non-stub ASes are transit ASes. We select 10 ASes from each of the 5 categories. Each hijacking instance selects a single attacker and a single victim from the total 50 ASes.

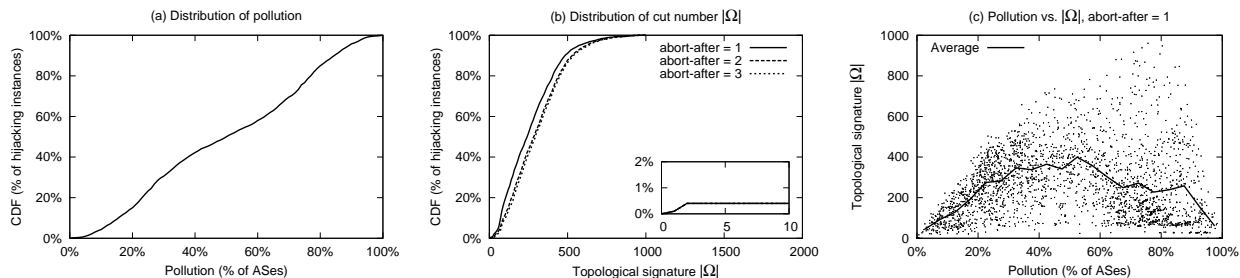
Each instance is simulated via the following steps. (1) We compute the forward path  $P(d)$  from the victim to each *transit* AS  $d$  before the hijacking. To do that, we simulate that each AS  $d$  originates its own prefix, and all ASes eventually converge on the routes regarding all prefixes. Our simulator emulates BGP routing update propagation and the BGP decision process including relationship-based route export and route preference. (2) We then simulate false origin prefix hijacking by the attacker. After the attacker originates the victim’s prefix and routing has converged, the ASes 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 3.1, due to route asymmetry, a path  $P'(d)$  may contain uncertain non-trailing subpaths “#”. Whether such non-trailing 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: *abort-after* = 1, 2 and 3.<sup>4</sup> (4) Finally, we calculate the set of distinct cuts  $\Omega$  using  $P(d)$  and  $P'(d)$  for all  $d$ .

A limitation with the simulation study is that we cannot easily characterize the timing aspects of hijacking and detection. For example, if the detection is triggered when a threshold number of cuts are observed, the detection delay depends on the snapshot durations as well as the start time of a hijack relative to the start time of the probing rounds. We will study the detection delay of ISPY using Internet hijacking experiments in Section 6.

**Results.** We first present the results assuming *abort-after*=1. Table 3 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 per-

<sup>4</sup>Configuring traceroute to tolerate more than 3 unreachable AS hops which translates into many more unreachable IP hops can incur high probing overhead.



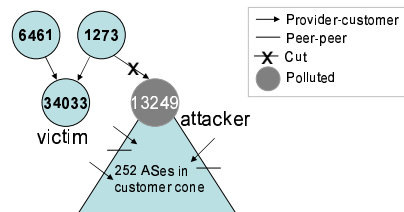
**Figure 3: (a) Distribution of pollution among the hijacking instances. (b) Distribution of the number of cuts  $|\Omega|$  among the hijacking instances. (c) Correlation between pollution and  $|\Omega|$ .**

centage distribution of the polluted ASes and the distribution of the topological signature  $|\Omega|$  of these instances in Figure 3(a) and Figure 3(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. Figure 3(c) further shows the correlation between the pollution caused by a hijack and its resulting cut number  $|\Omega|$ . It confirms the intuition that the number of cuts, which largely corresponds to the boundary between the region of polluted ASes and that of unpolluted ASes, is high when close to half of the ASes are polluted, and low when the pollution is either very high or very low.

We found that the results in Table 3 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 *abort-after*=1 actually estimates fewer cuts, compared to assuming larger *abort-after*, as shown in Figure 3(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.

**Analyzing Hijacking Instances with Small  $|\Omega|$ .** To gain insight into hijacking instances with small cut numbers, we first analyzed a hijacking instance simulated above that has a small cut number  $|\Omega|$  of 3. Figure 4 sketches the part of the AS topology relevant to the hijack. This hijack pollutes 144 ASes. The victim is AS 34033, a tier-3 transit AS multi-homed to AS 6461 and AS 1273. The attacker is AS 13249, a single-homed tier-3 transit AS that is a customer of AS 1273, one of the victim’s providers. AS 1273 is not polluted, and hence the pollution is restricted to the attacker’s customer cone. The cone consists of 252 ASes, 144 of which are polluted. Note not all ASes in the attacker’s customer cone are polluted, as some of them prefer victim’s valid routes announced by their possible multi-home providers or peers that are outside the cone. Out of the 51 transit ASes in the cone, the forward paths from the victim AS 34033 to 40 of them traverse the link 1273 - 13249; these forward paths will witness only one cut. The remaining 11 transit ASes contribute to two other cuts in vPath; a peering link and a link to a provider outside the cone both cut across the cone boundary.

We also analyzed the remaining hijacking instances that exhibit small cuts and found that two key conditions together often contribute to a small number of cuts. (1) None of the attacker’s provider(s) is polluted. This makes the pollution restricted to the attacker’s customer cone. If this condition is not met, the pollution would spread out and likely cause a larger number of cuts. (2) The attacker’s customer cone heavily relies on the attacker’s transit service in order to be reached from outside the cone, *i.e.*, the ASes



**Figure 4: An example hijacking instance with small  $|\Omega|$ .**

outside the cone are likely to use the attacker as transit to reach into the cone. This condition makes the victim’s path to most ASes in the cone share a few cuts, which are the links from the attacker’s providers to the attacker.

However, hijacking scenarios like above are rare because the two conditions are rarely satisfied for randomly picked victim and attacker pairs. For the first condition, since the attacker’s provider  $P$  is only one AS hop away and learns a customer route from the attacker, in order for  $P$  not to be polluted, the victim must also be  $P$ ’s customer in order to make its valid route more preferred by  $P$ . The second condition is not easily met unless the cone is small, because a large cone often consists of ASes with large degrees and are often multi-homed and/or have peers outside the attacker’s customer cone. We note that our simulation is based on AS-level Internet topology inferred from public BGP data which are known to miss many non-Tier-1’s peering links (*e.g.*, [8, 11, 23]). Hence the second condition is even less likely to hold in the real Internet.

The inverse case of the above hijacking instance, *i.e.*, AS 34033 hijacks AS 13249, also has a small number of cuts. In this case, ASes outside AS 13249’s customer cone are all polluted. For the same argument above (that the two conditions with the roles of victim and attacker switched are rarely satisfied), this case of hijacking is also rare.<sup>5</sup> Each of the remaining 9 hijacking instances in our simulation study that have a small number of cuts is similar to either of above two cases.

Finally, an adversary can deliberately exploit the above analysis to launch hijack attacks that potentially evade the detection by ISPY. However, either of the above two instances creates little incentive for the attacker to attempt. The instance where AS 13249 hijacks AS 34033 does not provide enough incentive, because without using hijacking, the attacker can also easily blackhole the victim for its customers. The instance of AS 34033 hijacking AS 13249, on the other hand, can be detected by just observing the bogus route at the victim, because the victim’s provider is polluted and announces bogus route to the victim.

<sup>5</sup>We note that the set of cuts in an instance and its inverse instance are not necessarily identical, because the vPaths for the two instances contain paths from different source ASes.

**Table 4: 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

### 3.5 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 [4]. As in Section 3.4, 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 4 shows the percentage of polluted transit ASes 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.

## 4. 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 ASes in the Internet to obtain the reachability and the AS-level paths to these ASes, which are used to construct the vPath used by ISPY to perform hijacking detection.

### 4.1 Design

The probing module faces several design challenges. First, the probing needs to be *light-weight* and *efficient*, in order to scan the large number of ASes 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 ASes.** As discussed in Section 3.1, to reduce the probing cost, ISPY probes only transit ASes, not stub ASes. This optimization reduces the total number of ASes to be probed from 23191 to 3742 (transit ASes).

**Live IPs.** We collected 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 ASes: 3470 (92.7%) of the total 3742 transit ASes have at least one live IP, and 3464 (92.6%) of transit ASes have at least two live IPs. Throughout this paper, we use these 3470 transit ASes as the probing targets. We also incorporate web IPs from the web server list in [27], which covers 2997 transit ASes.

**Resolving IP-Level Paths into AS-Level Paths.** Accurate IP-to-AS mapping is a challenging problem by itself [21] due to the lack of a uniform way of numbering router interfaces. 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 an unmapped “\*” hop as well as an unmapped IP hop 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 ASes.

Next, we collapse hops with the same AS mapping to produce the AS-level traceroute path; these hops always appear consecutive in the path. As in Section 3.1, 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  $P(d) = [sabc\#d]$ . 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 used the Paris-traceroute [3] tool, but modified it to improve its efficiency and robustness. First, we modified it to perform IP-to-AS translation on the fly. The on-the-fly translation enables the probing to make intelligent early termination decision; whenever the current hop belongs to the destination AS, the traceroute is terminated, since the AS-level path has already been discovered. Second, since the goal of our probing is not to measure the per-hop delay, instead of sending three probes for each hop (per TTL), we modified Paris-traceroute to serially send up to three probes, *i.e.*, to stop sending more probes if one probe successfully comes back.

### 4.2 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 *light-weight* 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 ASes, and is configured to maintain 50 concurrent traceroutes. We measure the average time it takes to finish a round of probing, *i.e.*,

**Table 5: Efficiency of iSPY’s probing module.**

	Five sample sources (by location)					Overall (108 sources)		
	UK	Pitts,US	LA,US	Norway	Japan	min	max	median
Avg hops per traceroute	16.6	13.5	17.2	16.7	16.1	10.7	19.9	15.5
Probing traffic per round (MB)	1.7	1.4	1.8	1.7	1.7	1.1	2.1	1.6
Time per traceroute (sec)	11.3	10.9	11.7	11.0	11.4	9.6	19.5	11.4
Probing time per round (min)	17	17	19	17	17	15	29	18
Bandwidth (KB/s)	1.7	1.4	1.6	1.7	1.6	0.8	2.2	1.5

to all probing destination ASes, and the amount of probing traffic. Table 5 shows the statistics of five sampled nodes among the 108 nodes, as well as the overall statistics.

First, the average number of IP hops per traceroute including tailing \*’s is 10 ~ 20. This number depends on the network location of the probing source. In general, networks higher up in the AS hierarchy have fewer IP hops per traceroute. This hops-per-traceroute metric of traceroute probing primarily determines the efficiency of the whole probing round. We further calculate the amount of total probing traffic per round. A traceroute TTL-limited probing packet accounts for 26 bytes, an ICMP ping sends two packets, 64 bytes each, and a TCP ping sends two packets, 26 bytes each. The total probing traffic per round is 1.1 ~ 2.1 MB.

Second, the average execution time of a single traceroute is between 9.6 ~ 19.5 seconds. Keeping 50 concurrent traceroutes, one round of probing finishes in 15 ~ 29 minutes. This short turn-around time of probing enables iSPY to obtain the up-to-date vPath. Furthermore, as we will describe in Section 5, iSPY determines there is an ongoing hijacking if it detects the number of cuts has exceeded a threshold, and hence a hijacking can be detected sooner than the finish of one complete probing round, achieving even shorter detection latency.

Finally, the bandwidth consumed by the probing traffic from the prefix-owner network is only 2 ~ 3 KB/s, confirming iSPY is lightweight. iSPY’s probing module can run on a low-end PC and incurs very low bandwidth consumption to the access bandwidth, and the probing is non-intrusive to the Internet.

**Coverage.** We next evaluate the coverage of our probing module. The purpose of probing is to test the reachability of other ASes and discover the AS-level paths to them. Accordingly, we measure what fraction of the ASes 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 ASes.

First, we focus on one node at Princeton. Table 6 lists the coverage in terms of ASes by traceroute probing, complementary ICMP and TCP ping probing, and the overall probing. Traceroute probing successfully reaches 91.4% of the 3470 transit ASes, 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%) ASes 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 ASes. The incompletely resolved paths are mainly due to unmapped \*’s in traceroute. This imperfection of path discovery poses a challenge for locating cuts in the vPath by iSPY. We will show in Section 5 how to tackle this problem.

Second, we show the coverage on five sampled nodes as well as across all nodes in Table 7. Across all nodes, the reachable AS coverage is 95.6% ~ 100% with a median of 99.9%, and the portion of ASes having complete paths ranges between 69.7% ~ 85.9%. This latter property depends on how frequently the probing traverses traceroute-filtering networks, which depends on the

**Table 6: Coverage of probing and breakdown of different methods.**

	Transit ASes	
	Num	Perc
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%

location of the probing source network.

## 5. iSPY: PREFIX-OWNER-CENTRIC HI-JACK 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 ASes 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. 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 4.1 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 6 via Internet experiments that the gap between the two bounds is small.

**Continuous Decision Making.** Although iSPY performs probing round by round, each round probing a total of  $N$  transit ASes in some fixed order, since probing rounds are issued continuously, at any point in the middle of a round, taking out the past  $N$  consecutive probes would also make up a complete round of probing.

iSPY exploits the above observation to perform continuous real-time decision making. It continuously streams new vPath data produced by the probing module into the decision making module. On

**Table 7: Coverage of probing on 108 PlanetLab nodes.**

	Five sample sources (by location)					Overall (108 sources)		
	UK	Pitts,US	LA,US	Norway	Japan	min	max	median
ASes reached by probing (%)	99.9	100.0	99.9	100.0	100.0	95.6	100.0	99.9
ASes having complete path (%)	79.7	74.5	80.7	82.4	82.4	69.7	85.9	81.0

receiving each updated AS-level path  $P(i)$ , the detection module identifies if there is a cut in it. It then updates the set  $\Omega$  with the number of unique cuts in the past  $N$  probes. Whenever the lower bound of  $|\Omega|$  exceeds a predetermined threshold  $C$ , the decision making module reports the occurrence of a hijacking. Because the cuts are calculated on the fly and the total number is updated continuously, the system can detect hijacking well before all cuts in a complete round of probing are witnessed.

As with any threshold-based decision system (e.g., [37]), 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 3.4 shows that using a threshold value of 10 leads to a low false negative ratio of 0.45%. In Section 6.1, we measure the false positive ratios from Internet experiments and analyze the sensitivity of and the tradeoffs between the two ratios with varying threshold values.

## 6. 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 when there is a hijacking.

### 6.1 PlanetLab Experiments

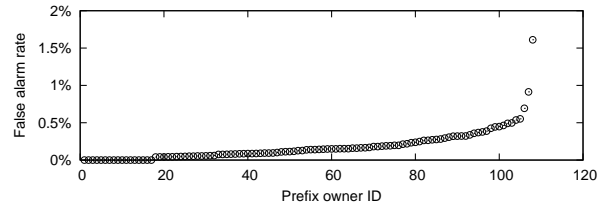
ISPY has been continuously running on over 100 PlanetLab sites since Nov 2007. Since hijacking is a rare event in the Internet, in this section, we use this deployment to evaluate the detection false positive ratio of ISPY. For this evaluation, we report on ISPY's deployment on 108 PlanetLab hosts (distinct prefixes) located in 88 ASes 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.

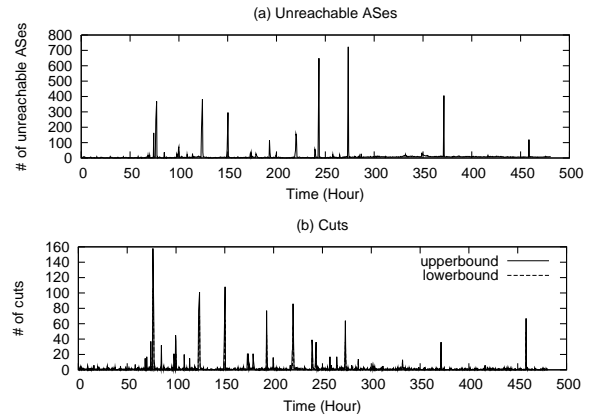
We believe that the alarms reported were all false positive, i.e., all cases with the number of cuts larger than 5 were not due to hijacking. This is based on two evidences. First, these cuts never lasted more than one round, whereas hijacking would generally last longer. Second, we did not find any MOAS announcement in RouteViews for these hosts' prefixes during the 25-day period.

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

To understand the cause for the high false positive ratio cases, we closely examined the three hosts that triggered the most false alarms. Figure 6 shows the number of unreachable transit ASes and the number of cuts witnessed by ISPY at `stella.planetlab.ntua.gr` which triggered the most alarms, a total



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



**Figure 6: Unreachable ASes and cuts witnessed by ISPY during its 500-hour running on Planetlab host `stella.planetlab.ntua.gr`.**

of 22 alarms out of 1363 probing rounds, as indicated by the spikes in the figure. The total probing time was 500 hours 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 ASes, 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 RTTs returned by successful traceroute, ping and TCP ping probes are higher than normal. Moreover, such abnormal behavior never lasted 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 above false positive ratios from the above deployment study and the false negative ratios from Section 3.4 in Figure 7, varying the detection threshold. The figure shows that the threshold of 10 cuts 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%. A threshold value of 10 yields a better false negative ratio than 20, and a much better false positive ratio than 5 with almost no degradation in the false negative ratio. This is because as

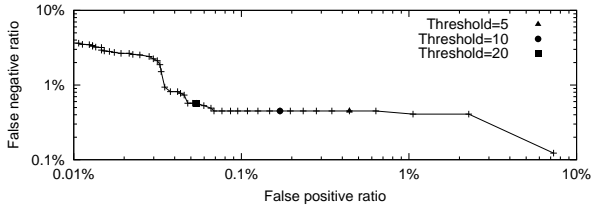


Figure 7: False positive ratio vs. false negative ratio.

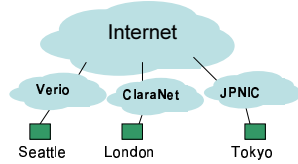


Figure 8: The setup of our hijacking testbed.

observed in Section 3.4, if the number of cuts created by a hijacking is less than 10 cuts, this number is very likely to be less than 5.

## 6.2 Hijacking Experiment

While our analysis in Section 3 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 testbed which launched a total of 15 hijacking attacks of our own prefix, and we deployed iSPY in this testbed and observed its live action during the hijacking events.

**Experiment Setup.** Our prefix hijacking testbed consists of three hosts located at three sites, Seattle, London and Tokyo, 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 Figure 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 alone injected the target prefix. Two hours later, the attacker also injected the prefix, and hence started the hijacking event. Another two hours later, the attacker withdrew the prefix. We performed a total of 15 hijackings during two one-week periods in January and June 2008. The time of hijacking events and the corresponding attackers and victims are listed in Table 8.<sup>6</sup> 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.

<sup>6</sup>We could not enumerate all six hijacking scenarios due to machine unavailability.

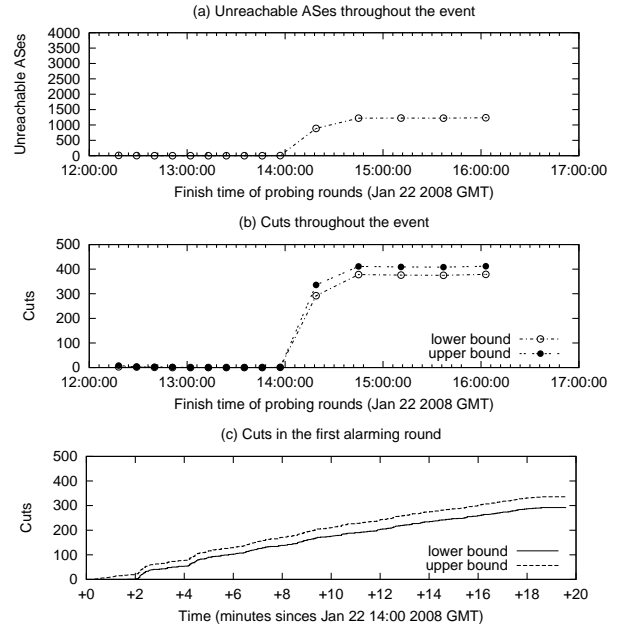


Figure 9: The number of unreachable ASes and the number of cuts as the probing progresses in event 1.

**Details of Detecting One Hijack.** We first discuss one hijacking event in detail, and in doing so introduce all the entries shown in Table 8. This hijack is event 1 in Table 8. It started on January 22 at 14:00 GMT by the London site, and upon convergence rendered 35.6% of the transit ASes unreachable from the Seattle site which resulted in cuts in the vPath in the range of [376, 409]. Figures 9(a)-(b) show the number of unreachable ASes and the number of cuts that were computed continuously in a rolling fashion during the 2 hours before and the 2 hours after the hijack injection moment of 14:00 GMT. The dots mark the boundary of probing rounds. Note 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 ASes and hence proceeded slower than before hijacking. We observe that before the hijack happened, the numbers of unreachable ASes and of cuts were both insignificant. At the hijack injection moment, the current probing round had been running for 2.1 minutes. As the probing progressed, iSPY updated the observed cuts continuously as shown in Figure 9(c). At 2.1 minutes 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 minutes, well before the round finished 19 minutes later.

**Overall Detection Performance.** Table 8 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 3 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 8. 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 rolling fashion cut computation, these cuts are accumulated to the round immediately after, and the alarm was triggered quickly in that round.

In summary, Table 8 shows that despite that the start time of the

**Table 8: 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

first alarming round varies from 10.9 minutes before to 0.2 minutes after the hijack injection, the detection latency remains consistently low, in the range of 1.4 ~ 3.1 minutes. 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 8 shows iSPY only witnessed 0.3 ~ 1.0% of ASes 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 ASes 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.

## 7. DISCUSSION AND FUTURE WORK

**Counter Measures.** One potential counter measure by attackers to data-plane based techniques is to modify the information contained in the probing replies (e.g. [37]). Doing so under a prefix-owner-centric scheme such as iSPY where the victim AS probes a large number of ASes requires the attacker AS to manipulate replies to all traceroute probes that are drawn to itself from polluted ASes. 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 [4]. We plan to investigate performance-based approaches in our victim-centric detection framework to detect interception attacks in our future work.

Another potential counter measure is pollution shaping, *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 adopted by those ASes. However, such a counter measure is unlikely to pose a threat for the following reasons. First, shaping a small-cut pollution is difficult. Evading pollution at a few specified ASes may not affect the number of cuts due to pollution, as the large number of peering and multi-homing links of other polluted ASes will still lead to many cuts. On the other hand, if the bogus route is padded with too many ASes, its competitiveness and hence the impact of the hijacking become limited. Second, to calculate the exact set of ASes 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 counter measures that attackers may

launch against iSPY as part of our 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 [34].

When a network owns multiple prefixes, it can potentially improve iSPY’s detection accuracy of hijacking events by comparing vPaths generated from probing from different prefixes. Dissimilar cuts, *e.g.*, with some above the threshold and other below, in these vPaths potentially provide further evidence of hijacking of some of its prefixes. However, a detail 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 [18], it remains interesting and we are currently exploring ways to extend iSPY to also identify the attacker in real time.

**Difference from Reachability Analysis.** A recent study by the Hubble system [16] reported that the extent of reachability problems is much greater than previously expected. Based on the actually reported numbers (31692 events with a median duration of about 2.75 hours from observing 110,000 prefixes for 3 weeks), we estimate the number of prefixes unreachable if sampling from 3000 prefixes within a 15-minute duration (iSPY’s average probing round) to be about 5. In our 25-day Internet deployment of iSPY in over 100 network locations, we observed the average number of transit ASes unreachable in a probing round to be 0.9. We attribute the difference to the fact that iSPY only monitors the reachability to the transit ASes, which are generally much more stable than stub ASes as considered in the Hubble study.

## 8. RELATED WORK

iSPY is closely related to previous work on IP prefix hijack detection [18, 12, 37, 25]. As summarized in the introduction, none of 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 In-

ternet 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 [26]. 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 [9, 30].

Upon detecting prefix hijacking events, the natural next step of action is to mitigate their impact. Many existing mitigation schemes include manual response to install filters, ACR [31], MIRO [33], route purge-promotion [36], and overlay routing. Detection followed by reactive mitigation falls into the category of passive counter measures against prefix hijacking. A number of proactive prevention solutions have also been proposed [17, 22, 13, 29, 15]. Another area related to prefix hijacking is on impact analysis [19, 35].

Finally, the probing module design of iSPY builds on previous work of light-weight end-host-based monitoring systems such as Rocketfuel [28], PlanetSeer [34], iPlane [20], and Hubble [16]. Our work also relates to previous work on using data-plane information to troubleshoot routing problems such as missing routes [7] and bogus prefixes [6].

## 9. 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 ASes in the Internet, a prefix hijacking almost always pollutes a significant percentage of the ASes. 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 ASes. 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 light-weight probing and scans for the hijacking signature as the trigger for hijacking alarms.

The prefix-owner-centric design makes iSPY 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 minutes in our hijacking experiments; (3) it is light-weight with the average probing traffic rate between 2 ~ 3 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 existing infrastructure-based designs of hijacking attack detection and demonstrates the effectiveness of end-host-based probing and analysis.

## 10. ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their helpful comments. This work was supported in part by NSF grants CAREER-0238379, CyberTrust-0430204, and CAREER-0643612.

## 11. REFERENCES

- [1] RIPE RIS. <http://www.ripe.net/ris/>.
- [2] University of Oregon Route Views Archive Project. <http://www.routeviews.org>.

- [3] 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.
- [4] H. Ballani, P. Francis, and X. Zhang. A Study of Prefix Hijacking and Interception in the Internet. In *Proc. ACM SIGCOMM*, August 2007.
- [5] P. Boothe, J. Hiebert, and R. Bush. How Prevalent is Prefix Hijacking on the Internet. NANOG36 Talk, February 2006.
- [6] R. Bush, J. Hiebert, O. Maennel, M. Roughtan, and S. Uhlig. Testing the reachability of new address space. In *Proc. ACM SIGCOMM INM*, 2007.
- [7] D.-F. Chang, R. Govindan, and J. Heidemann. Exploring the Ability of Locating BGP Missing Routes from Multiple Looking Glasses. In *Proc. ACM Workshop on Netw. Troubleshooting*, September 2004.
- [8] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Towards capturing representative AS-level Internet topologies. *Comp. Netw.*, 2004.
- [9] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs. Locating Internet Routing Instabilities. In *Proc. ACM SIGCOMM*, 2004.
- [10] L. Gao. On Inferring Autonomous System Relationships in the Internet. In *Proc. IEEE Global Internet Symposium*, 2000.
- [11] Y. He, G. Siganos, M. Faloutsos, and S. V. Krishnamurthy. A systematic framework for unearthing the missing links: Measurements and Impact. In *Proc. NSDI*, 2007.
- [12] X. Hu and Z. M. Mao. Accurate Real-time Identification of IP Prefix Hijacking. In *Proc. IEEE Security and Privacy*, 2007.
- [13] Y.-C. Hu, A. Perrig, and M. Sirbu. SPV: A Secure Path Vector Scheme for Securing BGP. In *Proc. ACM SIGCOMM*, 2004.
- [14] B. Huffaker. Caida as ranking project. July, 2006, [http://www.caida.org/analysis/topology/rank\\_as/](http://www.caida.org/analysis/topology/rank_as/).
- [15] J. Karlin, J. Karlin, S. Forrest, and J. Rexford. Pretty Good BGP: Improving BGP by Cautiously Adopting Routes. In *Proc. ICNP*, 2006.
- [16] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying Blackholes in the Internet with Hubble. In *Proc. NSDI*, 2008.
- [17] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (Secure-BGP). *IEEE J. Selected Areas in Communications*, 2000.
- [18] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. PHAS: A Prefix Hijack Alert System. In *Proc. USENIX Security Symposium*, 2006.
- [19] M. Lad, R. Oliveira, B. Zhang, and L. Zhang. Understanding resiliency of Internet topology against prefix hijack attacks. In *Proc. DSN*, 2007.
- [20] H. V. Madhyastha, T. Anderson, A. Krishnamurthy, N. Spring, and A. Venkataramani. iPlane: An information plane for distributed services. In *Proc. OSDI*, Nov. 2006.
- [21] Z. M. Mao, J. Rexford, J. Wang, and R. Katz. Towards an accurate AS-level traceroute tool. In *Proc. ACM SIGCOMM*, 2003.
- [22] J. Ng. Extensions to BGP to Support Secure Origin BGP (soBGP). IETF Draft: draft-ng-sobgp-bgp-extensions-01.txt, November 2002.
- [23] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. In search of the elusive ground truth: The Internet's AS-level connectivity structure. In *Proc. ACM SIGMETRICS*, 2008.
- [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.
- [25] J. Qiu, L. Gao, S. Ranjan, and A. Nucci. Detecting Bogus BGP Route Information: Going Beyond Prefix Hijacking. In *Proc. SECURECOMM*, 2007.
- [26] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proc. of SIGCOMM*, 2006.
- [27] C. A. Shue, A. Kalafut, and M. Gupta. The web is smaller than it seems. In *Proc. ACM SIGCOMM IMC*, 2007.
- [28] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1):2-16, 2004.
- [29] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and Whisper: Security Mechanisms for BGP. In *Proc. NSDI*, 2004.
- [30] R. Teixeira and J. Rexford. A measurement framework for pin-pointing routing changes. In *Proc. ACM Workshop on Netw. Troubleshooting*, 2004.
- [31] D. Wendlandt, I. Avramopoulos, D. Andersen, and J. Rexford. Don't Secure Routing Protocols, Secure Data Delivery. In *Proc. ACM HotNets*, 2006.
- [32] J. Wu, Y. Zhang, Z. M. Mao, and K. Shin. Internet Routing Resilience to Failures: Analysis and Implications. In *Proc. ACM CoNEXT*, 2007.
- [33] W. Xu and J. Rexford. MIRO: multi-path interdomain routing. In *Proc. ACM SIGCOMM*, 2006.
- [34] 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.
- [35] 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.
- [36] Z. Zhang, Y. Zhang, Y. C. Hu, and Z. M. Mao. Practical Defenses Against BGP Prefix Hijacking. In *Proc. ACM CoNEXT*, 2007.
- [37] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis. A Light-Weight Distributed Scheme for Detecting IP Prefix Hijacks in Realtime. In *Proc. SIGCOMM*, 2007.