Accurate Real-time Identification of IP Hijacking

Xin Hu Z. Morley Mao
University of Michigan
huxin@umich.edu zmao@umich.edu

Abstract

In this paper, we present novel and practical techniques to accurately detect IP prefix hijacking attacks in real time to facilitate timely mitigation responses. There are strong evidences that IP hijacking is common on today's Internet. Attackers may hijack victim's IP address space to perpetrate malicious activities such as spamming and launching DoS attacks without worrying about disclosing their identity through source IP addresses. More seriously, they can disrupt network services or regular communication by temporarily stealing actively used addresses. Unintentional network misconfigurations can also have similar effects, possibly leading to severe impact on reachability. We propose novel ways to much more accurately detect IP hijacking by combining analysis of passively collected BGP routing updates and data plane fingerprints of suspicious prefixes. The key insight is to use data plane information in the form of edge network fingerprinting to disambiguate potentially numerous suspect IP hijacking incidences based on routing anomaly detection.

Previous work on identifying IP hijacking solely relies on control plane information in the form of anomalous routing updates or external data such as stale address registries. Such an approach is inaccurate, suffering from too many false positives to be useful in practice. In our proposed scheme, real-time fingerprinting provides confirming evidence for hijacking, while incurring little overhead. More importantly, we provide mechanisms to perform online mitigation rather than post-mortem analysis. Utilizing real-time BGP data from multiple feeds as well as RouteViews, we demonstrate the ability of our system to distinguish between legitimate routing changes and hijacking attacks.

I. Introduction

Analogous to identity theft, IP hijacking also known as fraudulent origin attacks is to steal IP addresses belonging to other networks. It is an attack on the routing infrastructure or the control plane of the Internet. To accomplish this, attackers announce the hijacked address prefixes to traverse networks they control, so that they can use the stolen addresses to send and receive traffic. On the current Internet, IP address allocation to different organizations is managed by ICANN (the Internet Corporation for Assigned Names and Numbers) which delegates local allocations to several Regional Internet Registries (RIR) such as ARIN, RIPE, and APNIC. Those registries in turn allocate IP addresses to different organizations and national registries. IP addresses provide identifying information; therefore, they are important resources associated with owners' identities and should be deemed as properties of organizations who must receive them from legitimate authorities.

Attackers may hijack IP address space for two purposes: (1) Use the stolen addresses to conduct malicious activities such as spamming and DoS attacks without worrying about disclosing their identity. Note that although source IPs can be easily spoofed due to lack of ubiquitous deployment of ingress filtering, establishing a TCP connection still requires using a routable IP address to receive traffic. (2) Intentionally disrupt the communication of legitimate hosts numbered with the stolen addresses, disrupting their reachability – effectively a more stealthy type of DoS attack. Both types of hijacking use can significantly interrupt the stability and security of the Internet. Moreover, stolen IPs were also found to be sold or leased to networks in need of IP address spaces [28]. In such cases, attackers often trick the address registries to insert erroneous address ownership information. Note that the symptom of IP hijacking from victim's perspective is similar to other outages, making it nontrivial to diagnose.

Besides malicious intent, IP address hijacking can also result from unintentional network misconfigurations. The most notable example is the incident involving AS7007 [13] which accidentally advertised a short path to a large number of network prefixes (belonging to other networks) to its upstream provider. The provider did not filter out the bogus routing announcements leading to a large blackhole for many destinations on the Internet.

IP hijacking sometimes also refers to BGP (Border Gateway Protocol) hijacking, because to receive traffic to hijacked IP addresses, the attacker has to make those IP addresses known to other parts of the Internet by announcing them through BGP [42], [30], [23], which is the interdomain routing protocol on the Internet today. The Internet consists of more than 22,000 Autonomous Systems (ASes) [4], each with its own independent routing policies. The basic function of BGP is to enable ASes to exchange reachability information and allow BGP speakers to build an internal model of AS connectivity. Neighboring ASes interact to exchange routing information. A BGP route consists of a particular prefix and the AS path used to reach that prefix. IP hijacking occurs if an AS advertises a prefix that it is not authorized to use either on purpose or by accident. Because the current BGP protocol implements little authentication and often assumes a significant level of trust between peering ASes, IP hijacking can easily succeed. Furthermore, because a BGP router cannot know routing policies of its neighbors and cannot accurately evaluate the validity of a routing announcement in general given only local information, this leads to significant difficulties in preventing malicious or misconfigured routing information from propagating through the entire Internet.

An obvious way to *prevent* IP hijacking is to ensure proper configurations of route filters at the links between network providers and their customers to preclude customers from announcing routes for prefixes they do not own. However, this is both difficult and insufficient due to several reasons: (1) Providers do not always know which address blocks their customers are assigned to due to the prevalence of multi-homing. As a result, customers often obtain address prefixes from multiple providers. (2) Similar to ingress filtering, as long as there is one provider that does not properly enforce route filtering, IP hijacking becomes possible. (3) Compromised routers in the core Internet can bypass such filters, as route filtering is impossible along peering edges due to lack of information on addresses allocated to customers belonging to one's peer, oftentimes one's competitor.[1]

Given the above difficulties with route filters and the possibility of rogue routers, it is highly necessary to detect and thwart potential IP hijacking attempts. Some of the existing work on detecting unauthorized prefix advertisement uses public route registry information such as whois database. Due to stale and inaccurate registry information, such an approach is ineffective. Other methods focus on detecting anomalous control plane information – relying on conflicts in origin ASes[2] in the announcements [52] and short-lived nature of routing updates [14]. These suffer from too many false positives as well as false negatives, making them impractical for real operational use. False positives result from legitimate reasons why seemingly anomalous routing updates occur. False negatives stem from the fundamental observation that the control plane path or the BGP AS-level path may not match the forwarding or data plane path. [36], [29]. Moreover, using timing behavior as an anomaly

[1] There are two dominant AS relationships: customer-provider and peer-peer. Customers pay their providers to obtain Internet connectivity. Peers exchange traffic on behalf of their customers for free.

[2] *Origin AS* is the AS originating the route announcement for a given IP prefix. It is also the last AS in the AS path, as each AS prepends its AS number when propagating the route.

indication further undermines online mitigation as the detection may need to wait for the hijacking attempt to disappear.

Our approach to defeating IP hijacking is to first detect in real time routing updates that indicate unauthorized announcement of address prefixes. *Our key insight is that a successful hijacking will result in* ***conflicting data plane fingerprints*** *describing the edge networks numbered with the announced address prefix.* Thus, we exploit this fundamental property by using light-weight active or passive fingerprinting that characterizes end-hosts or edge networks to accurately and efficiently ascertain IP hijacking attempts as soon as they occur. Such fingerprints can range from fine-grained host-based information like the host uptime or the number and types of open ports (collected through *nmap*) to coarse-grained network information such as firewall policies. Essentially these fingerprints are identifying signature information for the network using the IP address prefix in question. Typically a hijacking attempt cannot succeed in affecting the entire Internet, especially from the perspective of hosts topologically close to the actual network owning the prefix. A real hijacking routing update thus results in conflicting fingerprints obtained from different network vantage points.

Our work focuses on real-time detection of ongoing IP hijacking events as soon as they occur rather than post-mortem analysis. Online detection enables timely mitigation responses, for example in the form of requesting help through external channels. Our main contributions include the following aspects. We present a comprehensive framework for the attack model of IP hijacking, including attack types previously overlooked and cannot be addressed using anomaly detection on the control plane alone. We propose detection techniques for each IP hijacking attack type based on several novel techniques such as AS edge popularity checking, selectively examined closely using active probing to collect data plane fingerprints confirming the attacks. Unlike previous work, our approach successively reduces the amount of false positives using a variety of anomaly detection and constraint checking techniques on routing data. Only very few remaining incidents need to be finally confirmed using edge-network fingerprinting. Overall, we present an efficient, accurate, and general IP hijacking detection framework, readily deployed in today's Internet requiring no ISP nor end-host cooperation, and validated using empirical data.

The rest of the paper is organized as follows. We first summarize related work in Section II, followed by a description of a comprehensive classification of IP hijacking in Section III. Section IV proposes our detection techniques for each attack type. To demonstrate the real-time detection capability, we present experimental results in Section V. Validation using empirical data are shown in Section VI. Finally, Section VII concludes the paper.

II. Related Work

IP hijacking is an attack on the Internet's routing protocol, specifically on BGP. IETF's rpsec (Routing Protocol Security Requirements) Working Group provides general threat information for routing protocols [8] and in particular BGP security requirements [15]. Prefix origin authentication is one such requirement. Related to this is path authentication. As explained later, malicious AS inserted in the AS path can achieve similar damage as fraudulent origin ASes (at the end of the AS path). A recent survey written by Butler *et al.* gives a comprehensive overview on BGP security issues, currently proposed solutions, and operational practices to improve routing robustness.

According to recommendations in RFC1930 [25], a prefix is usually to be originated by a single AS. MOAS conflicts result if multiple origin ASes announce the same prefix. Zhao *et al.* first coined the term MOAS, providing several legitimate explanations for them aside from misconfiguration and hijacking attacks: prefixes of exchange points, multi-homing without BGP or with private AS numbers [52]. Their subsequent work [53] suggested the use of BGP community attribute storing a list of originating ASes to detect potential violations. However, such a list is unauthenticated and optional, thus cannot ensure accurate detection of IP hijacking. To protect routes to specific services such as DNS, another ensuing work by Wang *et al.* [50] proposes preferring a set of known stable routes over transient routes. However, this approach does not scale to arbitrary routes.

The well-known BGP security architecture S-BGP [46] relies on digitally signed routing updates to ensure integrity and authenticity, assuming the presence of PKIs. Its high overhead in terms of memory, CPU, and additional management overhead prevents its rapid deployment. Follow-up work such as psBGP [49] and [51] improve the efficiency of S-BGP. The subsequently proposed SoBGP [38] provides flexibility to trade off security and protocol overhead using protocol parameters, combining proactive security measures with anomaly detection. Both S-BGP and SoBGP can defend against IP hijacking attacks besides other security issues. Other work in this area relying on cryptography include [44], [26]. The Interdomain Routing Validation (IRV) project [22] uses an out-of-band mechanism to validate received routing information by querying the IRV server in the relevant AS. However, it does not prevent an AS from originating a prefix it does not own.

The Listen and Whisper scheme [48] proposed by Subramanian *et al.* also helps identify inconsistent routing advertisement, but does not deterministically detect IP hijacking attacks. Similar to our approach, it takes advantage of data plane information. However, we take a more proactive approach by collecting the relevant fingerprints to maximize the possibility of identifying potential conflicts as a result of IP hijacking. Complimentary to our approach, the recent work by Aiello, Ioannidis and McDaniel [7] investigates the semantics, design and application of origin authentication services by formalizing address delegation semantics and exploring the use of various cryptographic structures for asserting block ownership and delegation.

Compared to these related work, our approach focuses on practical, readily deployable mechanisms using information from the data plane to validate occurrences of IP hijacking in real time. Many operational requirements for secured BGP have not been addressed [12], hindering the deployment of solutions such as S-BGP. In contrast, our solution can be incrementally, easily deployed by end hosts today, requiring no additional infrastructure, modifications to BGP nor routers, nor ISP cooperation. Our work uses routing anomaly detection techniques, such as those by Kruegel *et al.* [34]. We improve these techniques and use them for narrowing down more suspicious incidents for further investigation based on edge network fingerprinting. *Essentially we combine anomaly detection of control plane information* i.e., *routing updates with more conclusive conflicting data-plane fingerprints identification associated with the network in question.*

In the area of anomaly detection of routing updates and complementary to our work is the recent paper by Lad *et al.* [35] which notifies the prefix owners in real time occurrences of new origin ASes. This method however can be evaded as changes in origin AS is not necessary for attacks to occur. Our approach is more general and identifies all possible hijacking attack types described in Section III. A recent presentation at the NANOG meeting by Boothe *et al.* [14] presents the idea of detecting IP hijacking based on heuristics of short-lived MOAS conflicts, similar to [27]. We do not use timing-based approaches, as they may produce significant false positives and false negatives due to evasion. Furthermore, our work achieves online detection without waiting for the hijacking event to disappear which is necessary to collect the timing behavior. Ramachandran and Feamster recently [41] confirmed a common suspicion that IP hijacking is correlated with malicious activities such as spamming. Many current best common BGP practices such as route filtering and TTL security hack [21] can make attacks more difficult.

Finally, our work benefits significantly from various fingerprinting approaches to characterize end hosts and networks: *e.g.*, OS-based fingerprinting using tools such as nmap [18] and xprobe2 [5], physical device fingerprinting by identifying clock skews [33], timestamp-based information using TCP and ICMP timestamp probing, as well as IP ID probing used for counting hosts behind NAT [11].

III. A Comprehensive Attack Model of IP Hijacking

We first provide a classification of IP hijacking scenarios. The comprehensive attack taxonomy provides the foundation for our discussion on detection, the explanation for attacker's motivations, and possible evasion attempts. A similar taxonomy is given by Lad *et al.* [35], but their work addressed only a subset of the attacks.

1) **Hijack a prefix:** The attacker announces the ownership of IP prefixes that belong to some victim ASes. This will lead to Multiple Origin AS (MOAS) conflicts in routing tables, because the same prefix appears to have originated from both the original owner's AS and the hijacker's AS.
2) **Hijack a prefix and its AS:** The attacker announces a route to a prefix with an AS path that traverses its own AS to reach the victim AS. No MOAS conflict will result, since only a new route to the legitimate origin AS is added to the routing table, with the origin AS of the hijacked prefix unchanged. This route is invalid, since all the traffic to the prefix goes through the attacker's AS, allowing the attacker to easily intercept, modify, and insert traffic, while pretending to own the prefix of the victim AS.
3) **Hijack a subnet of a prefix:** This is similar to the first case, except the attacker only announces a subnet of an existing prefix. For example, the attacker only hijacks a /24 subnet of an announced /23 prefix, which has not been further deaggregated into smaller prefixes. In this case, there is no directly observable MOAS for such a prefix in routing tables without examining its supernet prefixes. We call this type of MOAS involving a subnet of a prefix **subMOAS**.
4) **Hijack a subnet of a prefix and its AS:** The attacker announces a path to reach the victim AS and a subnet of this AS's prefix. The attacker may prefer this method since it introduces neither MOAS nor subMOAS into routing tables and is the most difficult to detect.
5) **Hijacking along a legitimate path:** Instead of forwarding the traffic to the expected next-hop network, the attacker intercepts traffic and originates traffic using the address block of the downstream network.

In the first four attack types, attackers attempt to announce an attractive route, so that routers in different networks on the Internet, even given alternative routes, will still select the hijacking route as the best route and subsequently install it in their forwarding tables. One of the steps in route selection process is preferring routes with the shortest AS path [42]. Retaining the origin AS (type 2) increases the path length and may cause the hijacking route not chosen by some routers. Note that given the shortest AS path preference, networks topologically close to the victim AS are less likely impacted as they tend to choose the correct routes which are usually shorter than the hijacking routes that may traverse several ASes before reaching such networks. Along the same reasoning, routing tables of networks close to the attacker's AS announcing the hijacking route are more likely polluted, choosing the hijacking route over the possibly longer but correct routes. Routes announced by top tier providers usually have shorter AS paths as observed from most networks and are less likely impacted by IP hijacking.

For the fifth attack type, the attacker does not need to announce a new route but merely violate the rule of forwarding traffic based on its advertised route. We do not focus on this attack type, but our techniques can also identify it by simply performing traceroute to show that traffic stops within the malicious network.

Based on the above taxonomy, we highlight two important attack strategies used by attackers to improve hijacking success and avoid detection. Such understanding helps devise detection techniques to combat these attacks. The first strategy is announcing a subnet P' of an existing prefix P, resulting in two advantages. First, if there are no other subnets of P' announced and the hijacking route is not filtered,[3] based on the longest prefix matching rule [42], each router receiving such a hijacking route is guaranteed to select it as the best path regardless of its AS path length. Second, simple MOAS-based routing anomaly detection can overlook this type of attack. Note that attackers do not have the incentive to announce a supernet or *covering prefix* (using the terminology from [35]) either with the correct origin AS or attacker's origin AS, as it makes the hijacked route less attractive. Such announcement is only useful if there exists address blocks within the supernet not covered by existing route announcements. Essentially, this hijacking involves allocated but unannounced routes, and can be identified in a similar fashion as unallocated routes through a bogon-like list. We leave this as future work.

Existing work on detecting IP hijacking usually focuses on anomalous routing updates such as MOAS conflicts. It is easy to extend such an approach to subMOAS cases to address type-3 attacks. However, attackers can avoid such conflicts altogether by retaining the correct origin AS with an AS path containing the attacker's AS but reaching the correct origin AS, *i.e.*, type 2 and 4. This is attacker's second strategy with the disadvantage that the announced AS path is longer and may not be selected as the best path. However, announcing a subnet combined with this strategy, as illustrated in type-4 attack, will overcome this disadvantage, creating the most devious attack. We next discuss the above four hijack models in detail to provide the background for proposed detection techniques in Section IV.

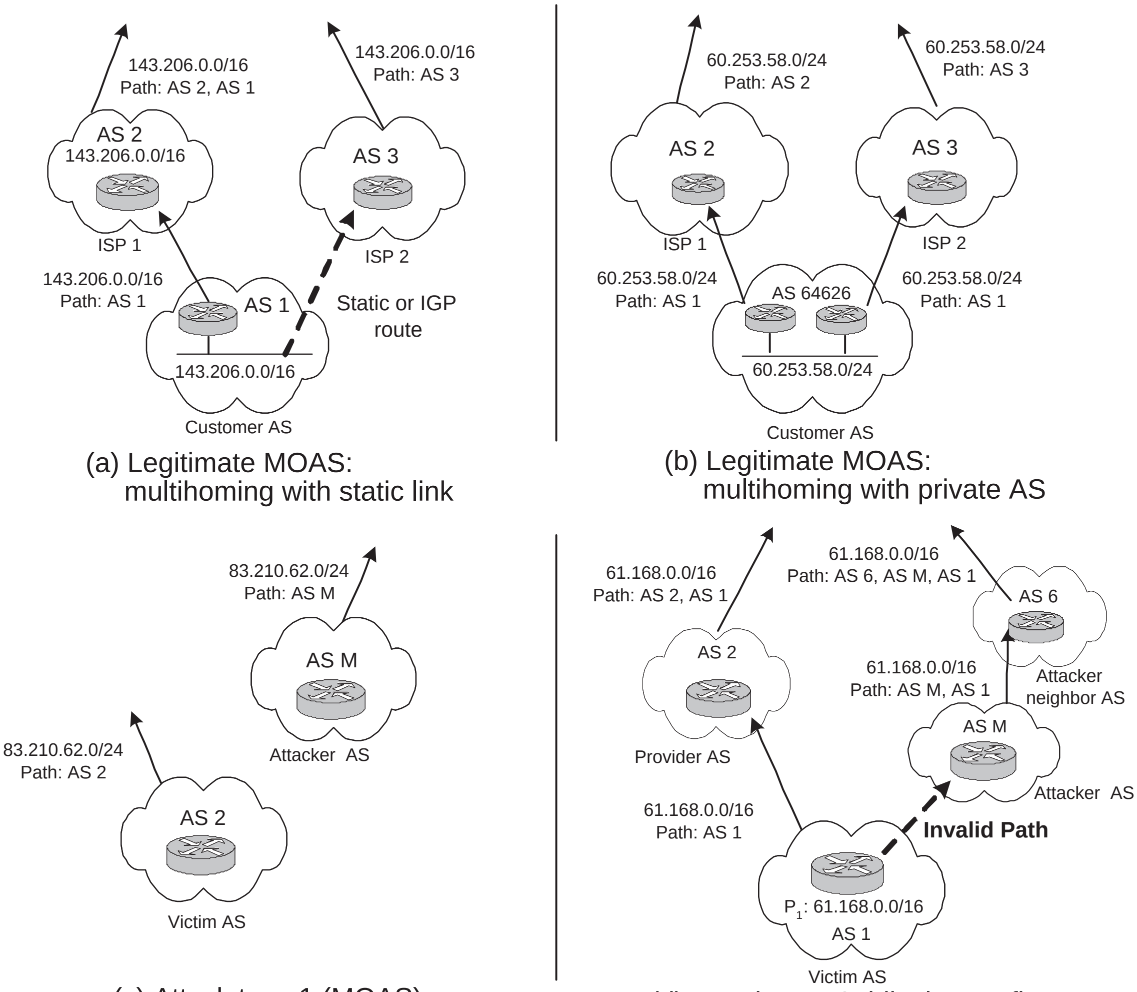
A. Hijack a prefix

The most direct way to hijack a prefix is to announce a BGP route to the prefix with the originating AS at least partially controlled by the attacker, who needs to be able to inject this hijacking route into a BGP session, possibly using a compromised router. The BGP neighbors subsequently propagate the route, if it is selected as the best path. Combining routing feeds from multiple vantage points will reveal a Multiple Origin AS or MOAS conflict [52], *i.e.*, a prefix with conflicting origin ASes. As an example, there are two AS paths to reach prefix P_1, namely $\{AS_1, AS_2, \cdots AS_n\}$ and $\{AS'_1, AS'_2, \cdots, AS_m\}$. An MOAS conflict occurs if $AS_n \neq AS_m$.

MOAS is only one possible indication of IP hijacking. There are nevertheless also valid reasons for MOAS. Therefore detecting MOAS alone serves only as one starting point, and we focus on distinguishing IP hijacking from legitimate MOAS cases. We describe two most common legitimate reasons as illustrated in Figure 1 (a),(b).

- **Multi-homing with static links:** In this case, an AS X uses statically configured route to connect to one of its provider ASes, AS Y. AS X establishes a BGP session to another provider AS. If the same prefix is announced to both providers, it will appear to have two origin ASes: X and Y in different AS paths.
- **Multi-homing with private AS numbers:** A customer may not be able obtain a registered AS number from the Internet Routing Registry (IRR) due to shortage of AS numbers. It can still use BGP to connect to its providers with a private AS number. Upon receiving the advertised routes, the provider will eliminate the private AS in the AS paths before announcing them externally. If

[3] In general, prefixes smaller than /24 are likely filtered to limit the size of routing tables [10].



(a) Legitimate MOAS: multihoming with static link

(b) Legitimate MOAS: multihoming with private AS

(c) Attack type 1 (MOAS): hijack a prefix.

(d) Attack type 2: hijack a prefix and its AS

Fig. 1. Common legitimate MOAS cases and type 1, type 2 IP hijacking attack using incorrect BGP paths.

a prefix is announced to both providers, it will appear to originate directly from the providers, resulting in an MOAS conflict.

Note that the above two cases can be combined, *e.g.,* in the form of statically linked to one provider, and using a BGP session with a private AS to connect to another provider. Other less common valid reasons for MOAS include Internet Exchange Point (IXP) Addresses, address aggregation, and IP anycast. IP hijacking and router misconfigurations can also lead to MOAS conflicts. The fundamental difficulty arises from the lack of authoritative information on the address ownership. Therefore by observing MOAS cases alone, we cannot identify IP hijacking. In Section IV, we develop an accurate algorithm to distinguish IP hijacking using data plane information.

B. Hijack a prefix and its AS

Despite several valid reasons for MOAS conflicts, they can still be considered possible abnormal BGP behavior requiring further investigation. Stealthy attackers can avoid MOAS by advertising a route to the stolen prefix retaining its origin AS. Whenever a BGP router advertises a route to its neighbor, it prepends its own AS number in the AS path. For locally owned prefixes, the AS path will just be the local AS. It is conceivable that the attacker uses a compromised router to pretend to be the victim AS X by advertising the route with AS path $\{X\}$. However, by default many BGP routers can reject routes with AS paths not starting with the AS number of their neighbor router in the BGP session. To ensure reachability, attackers in AS Y can instead advertise a route traversing its own AS reaching the victim AS X, *i.e.*, with AS path $\{Y, X\}$ for prefixes owned by AS X. It is difficult to filter such routes unless a BGP router has accurate knowledge of all the BGP neighbors of its neighbor. By creating fake AS edges, attackers can avoid MOAS conflicts, while still achieving the goal of using stolen prefixes to send and receive traffic. Interestingly, some of the DNS root servers use *IP anycast* which matches this attack profile.

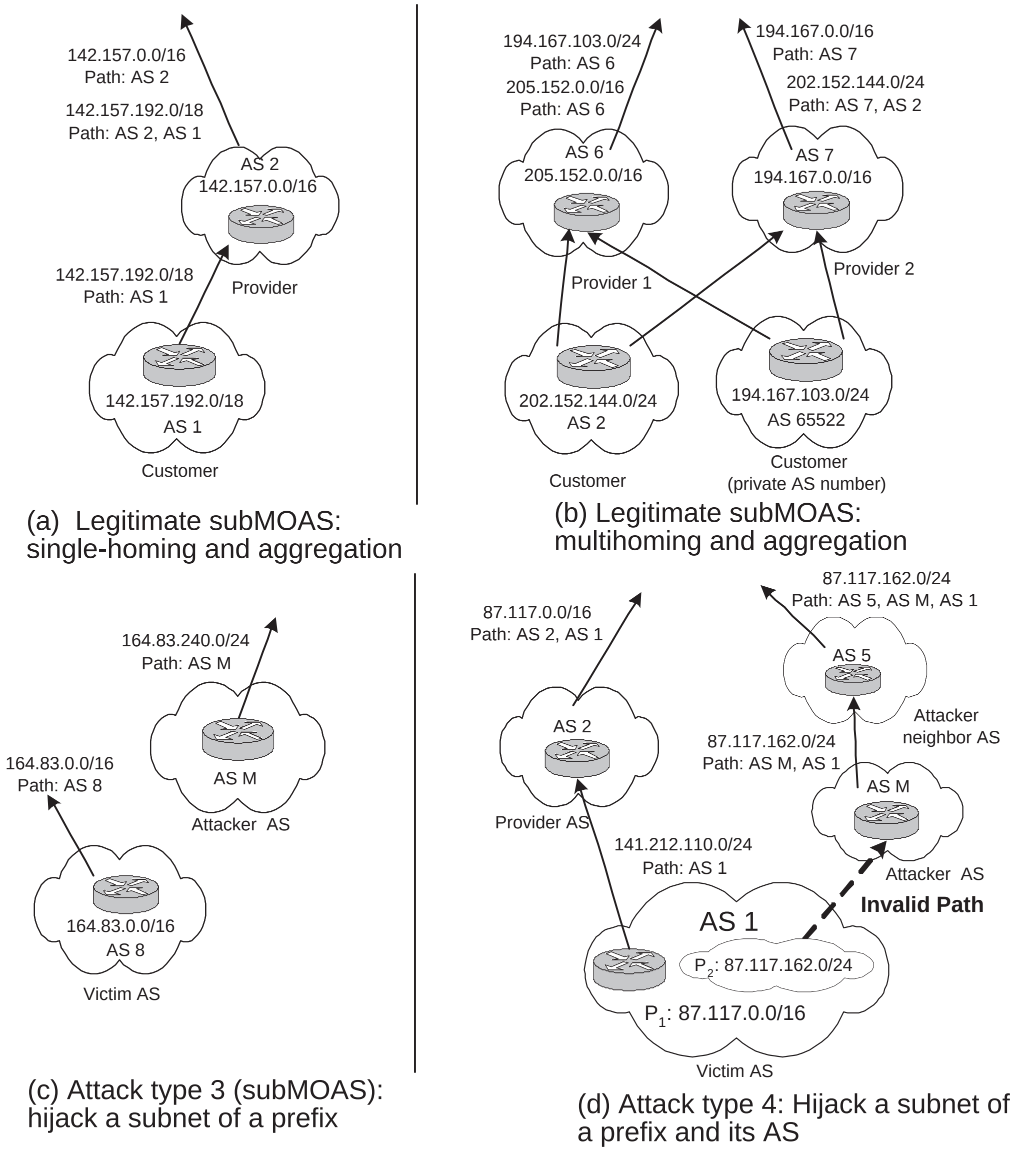
C. Hijack a subnet of a prefix

Another way to avoid MOAS conflicts is to announce a subnet of an existing prefix that has no other advertised subnets. For example, an attacker may hijack 129.222.32.0/19 given the existence of 129.222.0.0/16 in the routing table. As long as there are no other advertisements for such a prefix and no filtering for this route, the route is likely to be globally propagated or used due to longest prefix based forwarding. For attackers, this approach eliminates the challenging task of making the hijacked route attractive so that it is selected as the best path by other networks. To capture this routing anomaly, the definition of MOAS can be broadened to include such origin conflicts involving subnets of prefixes, as subMOAS conflicts. Similar to MOAS, there are several valid reasons for subMOAS enumerated here (shown in Figure 2 (a),(b),(c)).

- **Multi-homing with static links:** Similar to the MOAS case, except that the static routing between two ASes is configured to reach a subnet prefix, or the other session announces the subnet. This results in a subMOAS conflict as the origin AS of a prefix and its subnet disagrees.
- **multi-homing with private AS numbers:** For load balancing and redundancy reasons, a customer may multi-home to several providers and announce overlapping prefixes to its providers, *i.e.*, a bigger prefix to provider A and its subnet to provider B. If private AS number is used for these BGP sessions, the prefix and its subnet will appear to have the provider's AS as the origin AS, resulting in subMOAS conflicts.
- **Aggregation with single-homing or multi-homing:** A customer C obtains a prefix P from its provider A, who may aggregate P into a larger prefix and advertise only the less-specific aggregate to reduce routing table size with origin AS A. If the customer advertises P to its other provider B. B usually cannot aggregate P as its address block is likely to be discontinuous from that of A. A subMOAS conflict results: the bigger prefix with origin AS A and its subnet P with origin AS C. In the case of single-homing, the provider A announces both the aggregate prefix with origin AS A and P with origin AS C, resulting in an subMOAS conflict.

D. Hijack a subnet of a prefix and its AS

This is the most stealthy hijacking attack, combining the advantages of both the second and third attack types to avoid MOAS/subMOAS conflicts. Because of longest prefix matching, attackers can exclusively receive traffic destined to the hijacked prefix. For example, an attacker hijacks a subnet P' of prefix P owned by AS_1. Assume attacker's AS is AS_2. He announces the AS path $\{AS_2, AS_1\}$ for prefix P'. If attacker's neighbors cannot validate whether AS_2 really has a connection to AS_1, they will propagate this route. Since P' is more specific than P, most routers adopt it. Then the attacker is able to freely use P' to receive and send traffic from most external networks.



(a) Legitimate subMOAS: single-homing and aggregation

(b) Legitimate subMOAS: multihoming and aggregation

(c) Attack type 3 (subMOAS): hijack a subnet of a prefix

(d) Attack type 4: Hijack a subnet of a prefix and its AS

Fig. 2. Common legitimate subMOAS cases and type3, type 4 IP hijacking attack using incorrect BGP paths.

IV. Real-time Detection Techniques of IP Hijacking

We classified above different IP hijacking attacks and explained valid cases exhibiting similar behavior as malicious attacks. The focus of our detection algorithm is on distinguishing the unique characteristics of IP hijacking attacks based on data-plane properties of the network using the suspected prefix. It is critical operationally to have a highly accurate detection scheme with low false positives and negatives.

The fundamental and key difference between IP hijacking and valid routing updates lies in the ownership of the IP prefix and its connectivity on the Internet. For valid MOAS and subMOAS updates, despite multiple paths and disagreeing origin ASs, there is only one owner for the prefix, corresponding to a single network numbered with the prefix. Traffic sent from anywhere on the Internet destined to the prefix will arrive at the same network location. In the case of IP hijacking, the attacker illegally takes control over the prefix. Traffic sent from different network locations, depending on routing policies, may arrive at either the true network owner or the hijacked owner. Such a conflict must exist on the Internet, as traffic sent from the networks topologically close to the true owner or from that owner network must almost always arrive at the correct network. This holds even in the case for subMOAS, as IGP routing within the true owner network is unaffected. If hijacking is successful, as evidenced in the suspicious routing updates, networks advertising such updates will choose the hijacked route and reach the attacker network instead. To summarize, the consistency of the destination is the major criteria underlying our detection algorithm, which uses this inherent difference to detect IP hijacking attacks.

A. Fingerprinting-based consistency checks

When IP hijacking occurs, a given IP address in the hijacked prefix may be used by different end hosts. Similarly, two distinct networks can use the same IP prefix. Therefore we can check the consistency of destination hosts by verifying whether their properties match. Note that we do not require end-host cooperation, which can readily provide strong cryptographic authentication information. Instead, we propose a general approach using fingerprints to characterize properties of networks and hosts of the IP prefix. In general, we can focus on two types of fingerprints: host-based and network-based. End host properties such as the Operating System (OS), the actual physical device, host configurations (*e.g.*, firewall rules), host software, host services, *etc.* can all constitute host fingerprints serving as signatures or identifying information to help detect inconsistency. Network characteristics including network configurations like firewall policies, resource properties like bandwidth information, characteristics of routers connecting the network, *etc.* can provide distinguishing network fingerprints.

There are several considerations in choosing among these properties for detecting potential inconsistency implying real IP hijacking. One is the cost in terms of network overhead and probing duration, as some fingerprints, although can be obtained passively, require active probing given the lack of network cooperations. Thus, fingerprints collected through light-weight probing are preferred. Another consideration is accuracy. There are inherent errors in measurement due to limited precision and external influences. Combining multiple fingerprints (and assigning a weight to each based on its confidence) help reduce both false positives and negatives. Aside from measurement errors, false positives can also be due to intentional changes in such fingerprints. For example, load balancing redirects an incoming request destined to an IP address to a lightly loaded server, possibly resulting in conflicting fingerprints. Responses specific to the source IP address, such as those generated from firewalls, can also result in different externally observed host or network properties. Some properties are nondeterministic intentionally by design and should not be used for fingerprints. False negatives may result from distinct networks or hosts with identical fingerprints. Using multiple fingerprints and choosing discriminating properties such as host uptime[4] certainly reduce its likelihood. One type of uniquely identifying fingerprint is associated with the physical device, *e.g.*, [33] measuring the clock skews in target machines from TCP timestamp information.

From attacker's perspective, *evading fingerprints* by faking similar network or host properties of the original network is challenging given the use of a diverse set of properties, especially if properties are associated with available resources. It is not easy to fake more resources than what are available.

[4] Host uptime is how long the system has been running.

As initial examples, below we discuss the use of host OS, IP ID, TCP and ICMP timestamp based characterization as fingerprints.

Host OS properties: Attackers are likely to use a dissimilar OS or configure the OS differently in terms of open ports compared to legitimate users of the network. Even if the host is configured the same, the IP addresses used within the prefix may be different. The fact that certain IPs are reachable at certain ports from a given location, but not from another location is an indication for conflicts, barring intermediate network reachability issues and source-specific firewall rules. Different OSes (types or versions) typically implement the TCP/IP stack with slight variations, providing OS signatures. Popular remote OS probing tools including Nmap [19], [18] and xprobe2 [5] can be used to obtain such information. Nmap has a large fingerprint database including a wide range of OSes and devices such as firewalls, routers, switches, and even printers. In addition to OS information, open ports and services running on the host also provide identifying information.

IP Identifier probing: IP header includes a 16 bit identifier (IP-ID) field, designed to be unique for each IP datagram with the same source-destination to facilitate IP fragment reassembly. A common implementation is ”global” IP ID, *i.e.*, incrementing the IP ID by one for every packet sent, regardless of the destination IP. Similar to previous work on using IP IDs to uniquely identify hosts [11], we propose to use them to verify whether two machines are the same. We continuously send probe packets simultaneously to the same destination IP address but coming from different ASes associated the MOAS announcing the prefix. In the case of no hijacking, packets reach the same machine. Because of the global incremental properties of most implementations of IP ID, the target replies with IP ID values incremented by one or a fixed value for each probe packet, therefore the IP ID reply packets from two different ASes should exhibit roughly alternating incrementing pattern. In contrast, if there is IP hijacking, probe packets actually reach distinct machines, IP ID in reply packets from the two ASes appears unrelated.

There are several difficulties with this approach. Some implementations randomly set the IP ID field or reset it to be 0. As long as the DF (Don’t Fragment) bit is set, IP ID is no longer of any critical use. Some systems set IP ID field to be unique across every connection or peer rather than using a global counter.

TCP timestamp probing: The TCP timestamp option specified by RFC 1323 [31] is used for measuring round-trip times. It can also be used to estimate the time when the machine was last rebooted. According to [33], TCP timestamp is set based on the internal clock of machine’s TCP network stack which is reset upon system reboot. This clock runs at a certain frequency ranging from 1Hz to 1000Hz. Thus, the resolution of this virtual clock is between 1ms and 1second. Knowing the frequency and the TCP timestamp, we can infer the uptime of the target machine. If the inferred uptime based on TCP timestamp obtained from different locations is sufficiently diverse, even taking into account the measurement differences, it is very likely that a hijacking attack succeeded. Therefore the TCP timestamp is another good metric for judging the uniqueness of machines.

ICMP timestamp probing: Sending ICMP timestamp requests to the target machine will solicit the ICMP timestamp replies containing the system time of the target machine reported in millisecond. Because not all the machines connected to the Internet are synchronized with NTP, we can expect two different machines likely to have noticeable differences in their clock time and thus in their ICMP timestamp reply messages.

Though none of the above four methods guarantees to completely distinguish two different machines, the combination of them can reduce the false negative rate and improve the accuracy. In what follows, we discuss the techniques of detecting IP hijacking attacks for each of first four attack types summarized in Table I.

Attack type	Routing updates monitored	Detection techniques
1 (hijack prefix)	MOAS updates	fingerprinting-based consistency check (FP check)
2 (hijack prefix, AS)	all updates	edge, geographic, and relationship (EGR) constraints, FP check
3 (hijack subnet prefix)	subMOAS updates	customer-provider (C-P) check, reflect-scan
4 (hijack subnet prefix, AS)	new, nonsubMOAS prefixes	edge, geographic, and relationship constraints, reflect-scan
5 (hijack a legitimate path)	not triggered by updates	fingerprinting-based consistency check

TABLE I

SUMMARY OF DETECTION TECHNIQUES

B. *Type 1: Detection of prefix hijacking*

This type of IP hijacking has the characteristic of MOAS conflicts as shown in Figure 1(c), described in Section III-A. The essence of our attack detection algorithm outlined here is to check whether the prefix originated by multiple ASes has consistent data-plane signatures. To verify this, we send probing packets to the same IP in the prefix traversing different origin ASes and use the previously discussed fingerprinting-based consistency checks.

1) For each prefix involved in MOAS conflicts, find all AS paths reaching the prefix.
2) Build an AS path tree, rooted at the prefix.
3) Find a live host if possible in the prefix serving as the probing target.
4) Select probe locations so that packets traverse different AS paths and reach conflicting origin ASes.
5) Perform probing using techniques described in Section IV-A.
6) Analyze obtained fingerprints to check for mismatches implying potential IP hijack attacks.

One challenge is to select probe locations such that probe traffic goes into different origin ASes. We use the current best AS paths from publicly available BGP data to guide the selection. For example, assume prefix P_1 announced by both AS_1 and AS_2 has two AS paths reaching it: $\{AS_5, AS_3, AS_1\}$ and $\{AS_6, AS_4, AS_2\}$. Probe locations are chosen to be as close to the origin AS as possible – AS_1 is preferred over AS_3. Traffic may not conform the expected AS paths, because of possible inconsistency between data and control plane and disagreeing AS paths within the same AS (due to issues such as tie-breaking). Other difficulties include incomplete routing data to predict AS-level paths and limited probe locations. After selecting the probe location based on AS paths, we verify that traffic with high probability arrives at the intended AS. This is nontrivial, as translating a router IP from traceroute to AS numbers may result in multiple ASes [36]. Furthermore, traceroute may not reach the destination. We use any of the following two criteria to ensure that packets with high probability reach the origin AS AS_1.

- The traceroute IP-level path contains a router whose IP address is originated by AS_1 only.
- The traceroute IP-level path contains a router whose IP belongs to prefix P_2 using longest prefix matching, and P_2 is originated by the second last AS before reaching AS_1, *i.e.*, AS_3. In addition, AS_3 does not appear within the AS path originated by other conflicting origin ASes for the prefix.

To assist efficient probe location selection, we construct an AS path tree. A path from a leaf to the root denotes an AS path involved in the MOAS. Probe locations closer to the root are preferred. Locations directly within the preferred AS are selected. Given the limited probe locations, we improve our chance of finding one by identifying the “Largest Common AS Sets” (LCAS) which is the largest set of common ASes traversed by paths to any destination from a given probe location. Usually LCAS includes the upstream ISPs.

C. Type 2: Detection of prefix and AS hijacking

We now address the second attack type shown in Figure 1(d) and described in Section III-B. In this case, attackers avoid MOAS and subMOAS conflicts by retaining the correct origin AS. This is achieved either by creating a fake AS edge or violating routing policies. In the former case, attackers can simply append the correct origin AS after its own AS in the AS path, creating a fake AS edge between its network and the victim network. In the latter case, even if the AS path consists of physically connected networks, traffic cannot flow along the path due to routing policy violations. In both cases, the AS path is inconsistent with the data plane: the data packets do not flow along the advertised BGP AS path. One way verify the consistency between the control and data path is to use a tool such as AS-level traceroute [37]; however, not all routers respond with ICMP TTL exceeded messages. Our approach still relies on data-plane fingerprinting, but we enhance it by first using the following simple checks to reduce the false positive rate, especially given any update may be a possible attack in this category. Our goal is to minimize false negatives while significantly reducing false positives. Unlike the previous approach [35], our techniques are applicable independent of the position of the fake edge within the AS path.

- **Edge popularity constraint:** To retain the origin AS, an attacker may fake an AS edge between its AS and the victim AS. We identify such anomalies for computing the *popularity* of an AS edge. If the AS edge has never been previously observed in other route announcements or there are few prefixes using routes traversing this edge, it is highly suspicious.
- **Geographic constraint:** Similar to the above constraint, an fake AS edge can connect two geographically distant networks. BGP peering sessions between two ASes almost always occur between routers physically colocated. Thus, an AS edge corresponding to two distant networks signals an alarm.
- **Relationship constraint:** Extending the path constraint in previous work [34], we identify obvious violations of routing policies within the AS paths using inferred AS relationships [20], [47], [9].

We elaborate on the geographic constraint checking. We improve on previous work in two ways. First, rather than using data from registries such as whois, which provides only a single location for a registered AS, we exploit more fine-grained address prefix information. The previous study done by Michael *et al.* [16] showed that roughly 97% of all prefixes announced by stub ASes were announced from the same location. Thus, detecting fake edges involve stub ASes or ASes near the edge of the Internet is generally easier. Second, we build up a location set for each AS consisting of all distinct location information of its originated prefixes. The distance between ASes is the minimum distance between every pair of locations in their location set. Using location set of prefix level information eliminates as much as possible the influence of geographic diversity of each AS.

D. Type 3: Detection of prefix subnet hijacking

This attack shown in Figure 2(c), elaborated in Section III-C, occurs when the attacker hijacks a subnet of victim's currently advertised prefix by announcing it as originating from its *own* AS, resulting in a subMOAS conflict. This approach is more stealthy, as it does not create obvious MOAS conflicts and is less likely noticed if the subnet is unused. It is also preferred by attackers as more networks will adopt the hijacked route due to longest prefix matching. Our detection scheme relies on first identifying subMOAS conflicts, subsequently excluding the cases directly involving ASes with customer provider relationships using the *customer-provider check* explained below. Finally, we use fingerprinting checks to analyze the remaining cases.

The customer-provider check operates based on the assumption that providers will not intentionally hijack customer's routes due to lack of economic incentives and the ease of discovering such attacks through traceroute-like probing. Similarly, customers are incapable of hijacking provider's routes because traffic needs to first traverse the provider, and providers can easily detect such routing announcements. Given this justification, we now introduce a very simple yet robust and accurate technique for inferring customer provider relationships, improving on existing approaches of AS relationship inference [20], [47], [9]. Unlike peer to peer relationships, customer provider relationships can be viewed as transitive: if AS_1 is AS_2's customer and AS_2 is in turn AS_3's customer, AS_1 is also considered an "indirect" customer of AS_3. It is well-known that legitimate AS paths are valley-free [20] ("up" denotes customer to provider; "down" refers to provider to customer): no AS path can traverse a customer-provider edge after a provider-customer or peer-peer edge; no path can go through more than one peer-peer edge.

To infer customer provider relationships, we devise a simple rule justified by the valley-free routing policy: edges appearing before the tier-1 AS in the AS path are all customer-provider edges; edges appearing after the tier-1 AS must be all provider-customer edges. This holds, because tier-1 ASes are the highest point in the AS paths dictating preceding edges to be "up" edges and succeeding edges to be "down" edges to satisfy the valley free rule. Note that it is quite easy to infer tier-1 ISPs which do not have any providers. Given the prevalence of AS paths containing tier-1 ISPs, this check reduces false positives of subMOAS cases with very low false negatives. Furthermore, this simple filtering has low overhead and is suitable for real-time monitoring. However, it does not deal with conflicts involve two provider ASes who do not have a customer-provider relationship, *e.g.,* Figure 2(b). Thus, we still need to resort to fingerprinting for the remaining cases, but the challenge is that the more specific hijacked prefix can cause all traffic be influenced regardless of the probe location (as long as it does not filter the route). We can reach the correct owner network under the following two situations: (1) We can find probing locations inside the victim AS, so that the fingerprinting packets will be routed using IGP rather than BGP. (2) We can find probing locations inside the customer or provider of the victim AS that use static link to connect to victim AS and are thus unaffected by hijacking.

Given limited probe locations, neither condition is easily satisfied. We devise a new probing technique called *reflect-scan* for fingerprinting the victim network given the possibility of subnet hijacking. Our method is inspired by the TCP Idlescan technique [17] implemented in Nmap [19]. The basic idea is to make use of predictable IP ID increment in IP packet and IGP routing within victim AS which is unaffected by polluted BGP routes. We use IP spoofing to solicit traffic inside the victim AS. Let us assume a hijacking scenario where AS_1 has a large prefix P_1, *e.g.,* 195.6.0.0/16. AS_2 is malicious and hijacks subnet P_2 of P_1, *e.g.,* 195.6.203.0/24. Our probing technique works as follows (depicted in Figure 3 and Figure 4):

1) Find a live host ($H_2 or H_2'$: 195.6.203.3) in the hijacked prefix P_2 with predictable IP ID values (increment by 1) and is relatively idle (little outgoing traffic).
2) Find a live host (H_1: 195.6.216.26) whose IP address is in P_1 but not in P_2. More generally H_1 can be any live host in any prefix except P_2 originated by AS_1.
3) Assume due to hijacking, there exists a host H_2' in attacker's network AS_2 and a host H_2 with the same IP in victim's network AS_1. Because H_1 and H_2 are in the same same AS, packets sent from H_1 to 195.6.203.3 is routed using IGP *e.g.,* OSPF and reach H_2, the correct host. In contrast, if probing packets are sent from outside AS_1, they are routed using the polluted BGP route and reach H_2' instead, because P_2 is more specific than P_1.

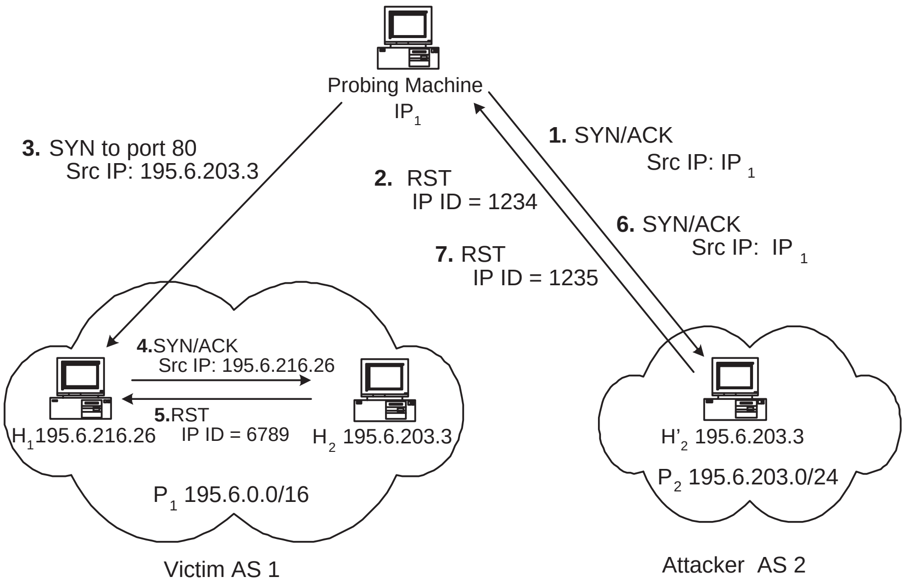


Fig. 3. Reflect-scan when hijacking occurs.

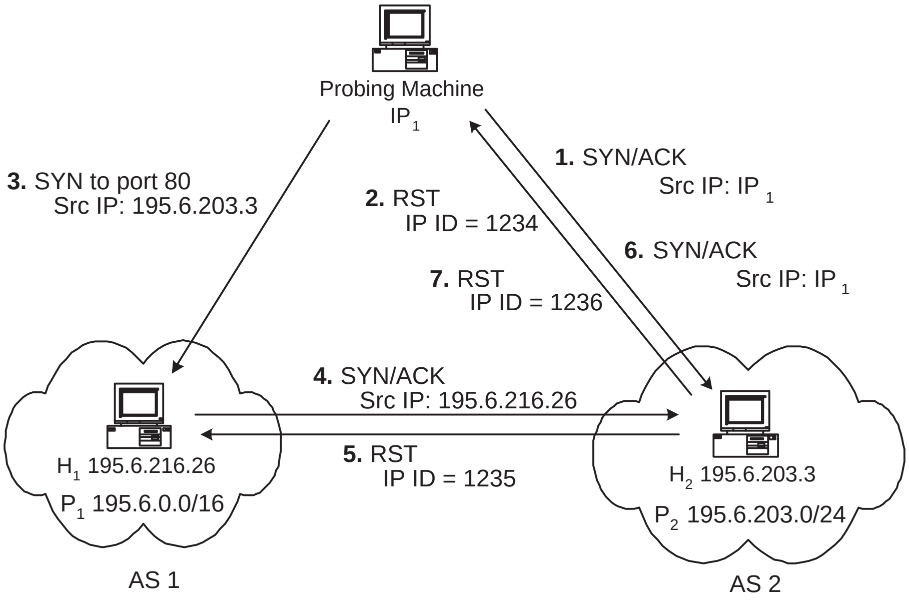


Fig. 4. Reflect-scan without hijacking.

4) Step 1-2: Send probe packets to 195.6.203.3 and record its current IP ID value. Remember because our probing comes from outside AS_1, in the case of hijacking, the traffic will be routed to the potentially hijacked prefix and the IP ID value is that of attacker's machine, *i.e.*, H_2'.
5) Step 3-5: Send a SYN packet to an open port of H_1 (195.6.216.26) with a spoofed source IP of H_2 (195.6.203.3). H_1 should reply with SYN/ACK to 195.6.203.3. Because IP address of H_1, 195.6.216.26 and 195.6.203.3 are inside the same routing domain, the response packet should reach H_2 in AS_1. After receiving this unsolicited SYN/ACK, H_2 will send a RST and increase

its IP ID value by one.

6) Step 6-7: Reprobe 195.6.203.3 and obtain the current IP ID value of H_2 or H_2'. If the IP ID value in the reply is only increased by 1, then it has not sent any outgoing packets. Very likely it did not receive H_1's SYN/ACK packet. If the increase in IP ID is 2 or more, it is highly likely that P_2 is not hijacked.

As demonstrated by the figure above, the target host with IP 195.6.203.3 will respond differently depending on whether the subMOAS is caused by hijacking. If there is no hijacking, H_2 will receive reply SYN/ACK packets from H_1, causing its IP ID number to be incremented by the number of spoofed packets received. Otherwise, the IP ID value of the target host would not increase. We now relax the restriction that H_2 needs to be idle to improve the robustness of the reflect-scan test. During our probing, H_2 may also send out other packets not triggered by our probing. To reduce the false negative rate (missing some hijacking attacks), we first measure the average increasing rate of H_2's IP ID value per unit time, for example n packets per second. Here we assume we can always find a host with not very large n *e.g.,* 20 packets/second. This is reasonable, since not all hosts are as busy as routers. We also send multiple (*e.g.,* $2 * n$) spoofed packets to H_1 in quick succession. After that, we probe H_2 again, if there is a significant increase in IP ID increase rate (much larger than n per second), we expect H_2 to have received the response packets from H_1 and therefore there is no hijacking. We can also repeat this test to further improve the accuracy.

Similar to Idlescan, our method relies on the following common properties (which may not hold due to firewalls). We send probe packets to verify these conditions hold and there are no ingress filtering for spoofed packets. (1) A live host will reply with a SYN/ACK packet upon receiving a SYN packet to an open port. (2) A host will reply with a RST packet when receiving an unsolicited SYN/ACK packet. (3) Every IP packet has an IP ID value. Many operating systems predictably increment it by some fixed value (usually one) for outgoing IP packet.

E. Type 4: Detection of prefix subnet and AS hijacking

This is the most devious attack type as illustrated in Figure 2(d), discussed in Section III-D, where the attacker hijacks a subnet *and* retains the correct origin AS. Similar to type-2 attack, there is no MOAS or subMOAS conflicts. To detect this attack type, we continuously monitor new prefixes that are subnets of existing prefixes in the routing tables. If they do not cause a subMOAS conflict, they may fall into this category. We can apply similar checks for type-2 attacks: edge popularity constraints, geographic constraints, and relationship constraints. After performing these checks, we apply reflect-scan probing to deal with the remaining cases that violate any of the checks. Note that we can still achieve real-time monitoring given that the space of suspicious cases for this attack type only include the new prefixes not present in the current routing tables.

V. Implementation of Real-Time Monitoring

One of the most important properties of our system is real-time monitoring. As hijacking sometimes lasts only for a short time period to avoid detection, a real-time detection system is essential to defend against malicious attacks in a timely manner, reduce the damage, and identify the culprit. We demonstrate next how we achieve the real-time capability in our prototype system.

A. System architecture

We developed a prototype system aimed at online detection of anomalous BGP routing updates and selective light-weight active probing to gather data-plane fingerprints for identifying real hijacking

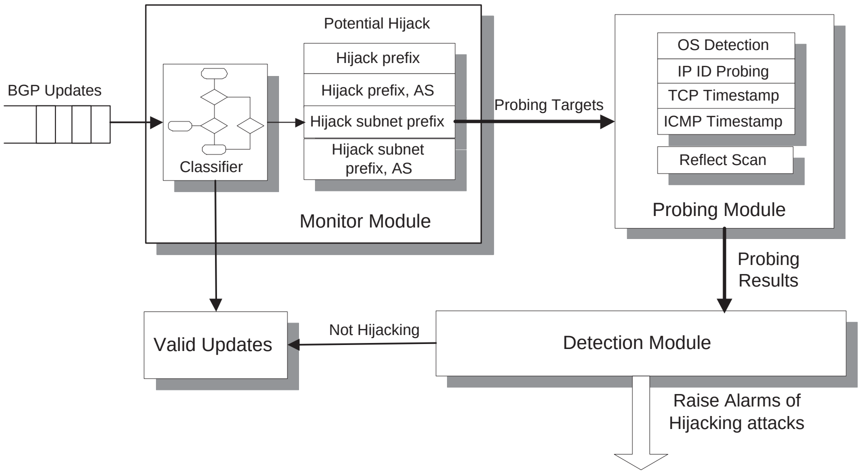


Fig. 5. Architecture of real time detection system for hijacking attacks.

attacks. Figure 5 illustrates the architecture of the prototype. It consists of three modules that closely interact with each other.

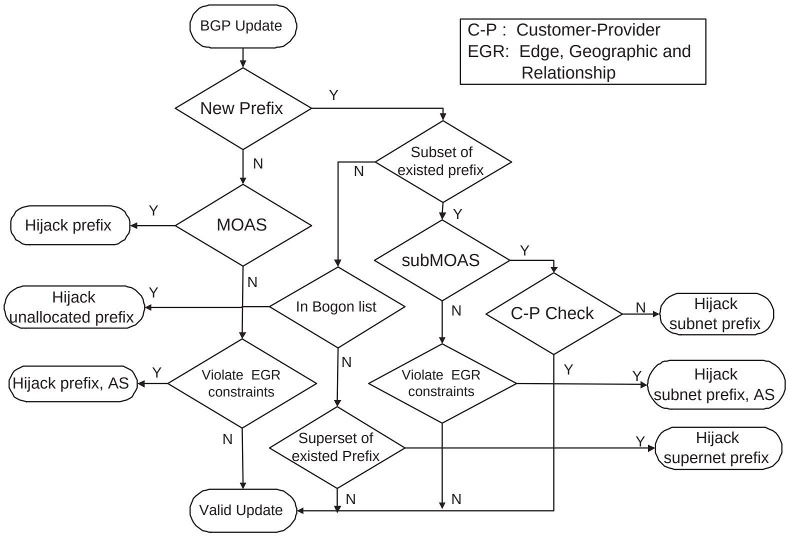


Fig. 6. (b) Classification for hijacking types.

1) **Monitor Module**: processes BGP updates in real time to identify potential IP hijacking. The classifier in this module depicted in detail by Figure 6 classifies each update into two types: valid and suspicious. For the latter case, it groups them into four hijacking types described in Section III. Then both the type and the update information (*i.e.,* prefix and AS path) are fed into the Probing Module for further analysis.
2) **Probing Module**: takes input from the Monitor Module and selects corresponding probing techniques. It chooses the appropriate probing locations and launches probing (*e.g.,* OS detection, IP ID reflect-scan) to the target prefix. Probe results are collected and sent to the Detection Module.

3) **Detection Module**: analyzes and compares the probe results to distinguish real hijacking from valid updates.

B. Experimental methodology

BGP data set: We use BGP update data primarily from two sources: University of Oregon RouteViews Server [3] which peers with 57 BGP routers in 46 different ASes and our own route monitor peering with 7 BGP routers in 7 distinct ASes including both academic and commercial networks. RouteViews data has better coverage, however, it has usually two hour lag in the availability of its 15-minute update files on its Web server. We can obtain real-time BGP updates from our own monitor. Because of the larger number of feeds in RouteViews data, we use it to evaluate our system's scalability and efficiency in processing large number of BGP updates. For update-triggered response, we use data from our own monitor to study timely responses to anomalous updates.

Probe location selection: We use the Planetlab testbed [2] as the candidate probing places for both type-1 and type-2 attacks. Note that reflect-scans can be conducted anywhere as long as IP spoofing is permitted. The PlanetLab testbed currently consists of 642 machines in 179 different ASes including 3 tier-1 ISPs. To increase the likelihood of finding a probe location reaching the desired origin AS, we construct LCAS for each Planetlab host. As a result, we can find probe locations for 89% MOAS cases and 75% type-2 attack cases.

Live IP addresses: Live IP addresses for probing are collected by combining locally collected DNS and Web server logs. We also use reverse DNS to look up authoritative DNS servers and mail servers of various domains. We also adopt light-weight ping sweeps for a very limited address range if we cannot find a live host from the list. Currently our list contains 1,165,845 unique IP addresses allowing us to find target hosts for 70.3% of all prefixes in MOAS conflicts, 55.2% for type-2 attacks, 71.0% for subMOAS conflicts, and 90.1% for type-4 attacks.

Geographic information of prefixes: There are several public databases for translating IP addresses or prefixes to corresponding locations. In our current implementation, we use the NetGeo [1] database, developed by CAIDA to map IP addresses and AS numbers to geographic locations. We queried locations for 198,146 prefixes, and NetGeo returned detailed longitude and latitude values for 98.4% of them. We plan to explore other techniques [39].

C. Real-time detection

To achieve real-time detection of ongoing IP hijacking, efficient BGP update processing and selective probing is of critical importance. To understand our system performance, we measure BGP update rate, detected anomaly rate, the probing time of different attack types, and the memory usage of the prototype. We use RouteViews data for its better coverage and simulate update processing by feeding RouteViews Data into the Monitor Module.

Update rate: The update rate determines the workload of our system. We take one week's updates (from 04/01/2006 to 04/07/2006) from RouteViews and calculate the average update rate for each BGP feed over a period of the seven days. The maximum update rate is 12 updates/second, the minimum rate is less than 1 update/second, and the average rate is about 2.45 updates/second. Because the classification process does not involve active probing, even a desktop machine can easily handle such a rate for many BGP feeds.

Anomaly rate: The anomaly rate is the number of suspicious updates per unit time after processing using the classifier. This determines the rate of active probing to detect hijacking attacks. Suspicious updates is divided into four types described in Section III. We show the rate using one day data from RouteViews in Table II:

Attack Type	Suspicious updates	Max rate (15 minutes)	Average rate (15 minutes)
1	Hijacking a prefix (MOAS conflicts)	0.42	0.08
2	Hijacking a prefix and its AS	28.17	1.60
3	Hijacking a prefix subnet (subMOAS)	2.92	0.16
	After Customer-provider check	0.86	0.09
4	Hijacking a prefix subnet and its AS	3.74	0.33
	After EGR constraint check	0.15	0.01

TABLE II
ANOMALY RATE OF SUSPICIOUS UPDATES/BGP FEED (1 DAY ROUTEVIEWS DATA)

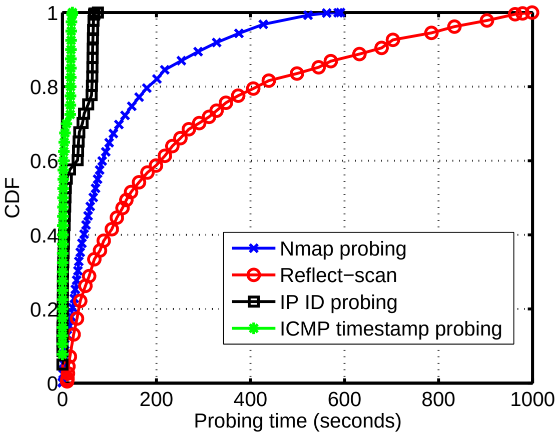


Fig. 7. The probing time distribution.

As illustrated in Table II, the average anomaly rate for all attack types is quite small. Therefore the overhead is relatively low. Furthermore, since all the probing can be done in parallel, our system can easily scale to monitoring a large number of BGP feeds.

Probing time: For each suspicious BGP update, the system performs active probing to identify IP hijacking. In the current implementation, we adopt four probing techniques: Nmap scan, IP ID probing, ICMP timestamp probing, and reflect scan. Based on one week's experiments, probing duration distribution is shown in Figure 7. In general, the probing takes less than 10 minutes, with the average time of less than 3 minutes for Nmap, and less than 4 minutes for reflect-scan (due to the overhead to find idle hosts and open ports). Considering the relatively low anomaly rate shown in Table II, our system is scalable.

Memory usage: We evaluate the memory usage of our system. The prototype is implemented using both Perl and C and runs on a desktop computer with P4 3.2GHz CPU and 1.5GB memory. For RouteViews data, it consumes around 66% of total memory. When monitoring the real-time BGP data collected from our own router, it uses less than 7% of total memory, demonstrating relative low overhead and high scalability of our prototype system.

VI. Evaluation of fingerprinting-based IP hijacking detection

In this section, we describe results in data probing using our prototype system and evaluate the effectiveness of the detection system by illustrating some interesting results collected during more than one week's time period.

A. Feasibility of selected probing techniques

We experiment with several probing techniques to collect fingerprints. Not all hosts respond to probes. For example, in response to SYN packets, systems with Fedora Core 3, 2.6.10 Linux kernel reply with zeroed IP-ID packets. Next we evaluate the feasibility of IP ID, ICMP and TCP timestamp probing on common OSes.

IP ID probing: As mentioned before, Fedora Core 3 with 2.6.10 Linux kernel replies a SYN/ACK packet with zero IP ID to an incoming SYN packet destined to an open port. However, its IP ID value in response to ACK packets are globally sequential. The IP-ID-based probing effectiveness relies heavily on the actual OS property. We did experiments on several popular operating systems and summarize results in Table III. The table indicates that for each OS we can always select appropriate probing technique to ensure the IP ID reply is globally sequential.

OS type	IP ID value pattern						
	SYN Open	SYN Close	ACK Open	ACK close	ICMP	Global	
						TCP	ICMP
Windows XP SP2 (firewall disabled)	S	S	S	S	S	G	G
Linux 2.4.0 - 2.5.20	0	0	0	0	S	N/A	G
Linux 2.6.12	0	S	S	S	S	G	L
Sun Solaris 8, 9 or 10	S	S	S	S	S	G	G
FreeBSD 6.0	S	S	S	S	S	G	G
Cisco router running IOS 12.3	S	S	S	S	R	G	R

TABLE III

IP ID for different OS (S: Sequential increment; 0: return value is all 0; R: Random; G: Global; L: local).

TCP/ICMP timestamp probing: Similar to IP ID probing, we test the feasibility of using timestamp probing on several popular operating systems in Table III and found that both ICMP and TCP timestamp are supported by all of them with the exception that Windows XP and the Cisco router do not support TCP timestamp. Some routers also disable ICMP timestamp replies due to policies.

B. Effectiveness of customer-provider checking

Our system uses several simple route anomaly detection techniques such as edge popularity checking and relationship checking to reduce false positives and fingerprinting-based consistency checking that requires active probing. One of these techniques is customer-provider check whose effectiveness hinges on the number of AS paths containing tier-1 ASes to help eliminate valid subMOAS cases. Using a tier-1 ISP list obtained based on [47], we found on average 84.4% of all AS paths in RouteViews data contains at least one tier-1 AS, and this increases to than 96% for our locally collected BGP data. Therefore the customer-provider heuristic introduced in Section IV-D is fairly effective at eliminating valid subMOAS conflicts, also demonstrated in Table II.

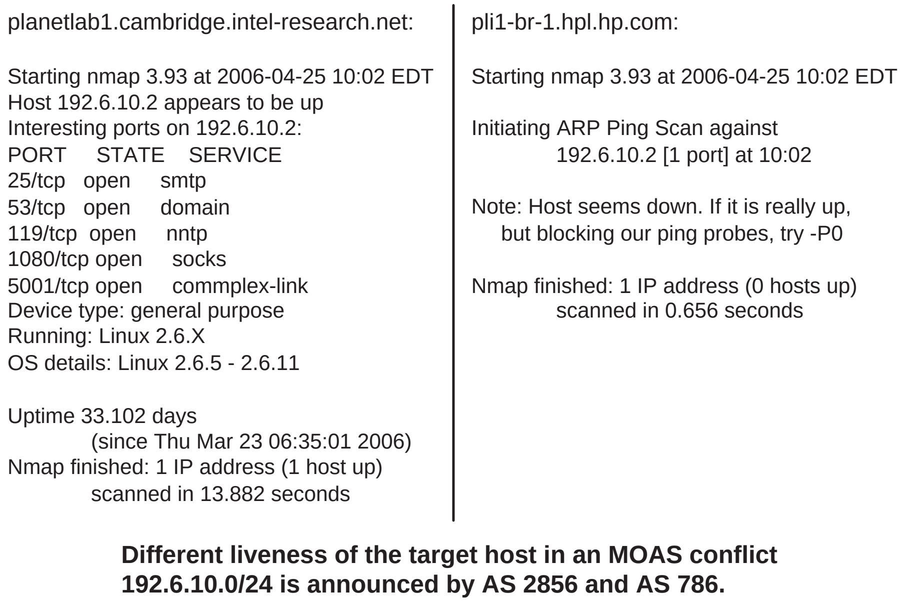
C. Monitoring results

We now present some interesting results obtained from over 111 hours of real-time monitoring across 8 days.[5] The type and number of anomalies are summarized in Table IV. The rate is the averaged over all 7 feeds monitored. We implement probing for IP-ID and ICMP timstamp using Scriptroute [45] and reflect-scan using hping [43]. The probing to the same IP across different paths are conducted at roughly the same time.

Suspicious update type	Total number	Average rate (per 15min)	Suspicious cases (after FP check)
Hijack a prefix (MOAS conflicts)	1485	0.48	55
Hijack a prefix and its AS	10418	3.35	137
Hijack a subset of a prefix (subMOAS conflicts)	1469	0.47	71
Hijack a subset of a prefix and its AS	473	0.15	35

TABLE IV
SUSPICIOUS UPDATES DETECTED DURING 4 DAYS' MONITORING AFTER APPLYING VARIOUS CONSTRAINT CHECKING.

Suspicious MOAS conflicts and type-2 attacks: Since we use similar probing techniques to identify suspicious MOAS conflicts (type-1 attacks) and type-2 attacks, we present them together here. We group the observed suspicious fingerprinting results into following categories.



```
planetlab1.cambridge.intel-research.net:

Starting nmap 3.93 at 2006-04-25 10:02 EDT
Host 192.6.10.2 appears to be up
Interesting ports on 192.6.10.2:
PORT     STATE    SERVICE
25/tcp   open     smtp
53/tcp   open     domain
119/tcp  open     nntp
1080/tcp open     socks
5001/tcp open     commplex-link
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.5 - 2.6.11

Uptime 33.102 days
          (since Thu Mar 23 06:35:01 2006)
Nmap finished: 1 IP address (1 host up)
          scanned in 13.882 seconds
```

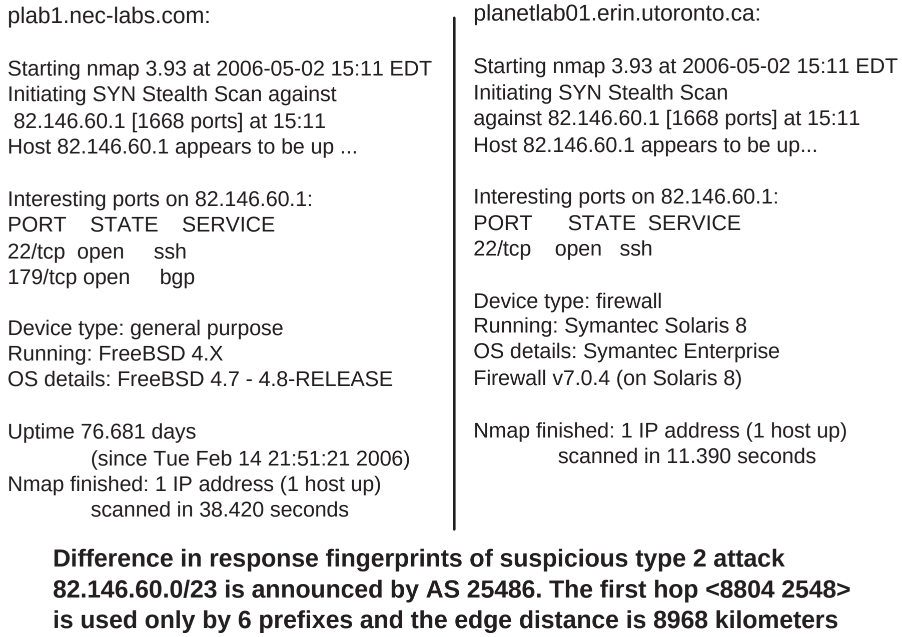
```
pli1-br-1.hpl.hp.com:

Starting nmap 3.93 at 2006-04-25 10:02 EDT

Initiating ARP Ping Scan against
          192.6.10.2 [1 port] at 10:02

Note: Host seems down. If it is really up,
    but blocking our ping probes, try -P0

Nmap finished: 1 IP address (0 hosts up)
          scanned in 0.656 seconds
```

Different liveness of the target host in an MOAS conflict
192.6.10.0/24 is announced by AS 2856 and AS 786.

Fig. 8. Conflicting fingerprints of Nmap probing of type-1 attacks The first line indicates the probe location.

- **Different liveness:** Using Nmap if the host appears alive from one location, but unresponsive from another location, it may be a real hijacking attack barring intermediate network problems and special firewall policies. An example is shown in Figure 8.
- **Different Operating Systems:** Figure 9 is an suspicious type-2 attack with different Nmap-inferred OS.

[5] We do not have results for the remaining hours due to network problems with the BGP monitor.



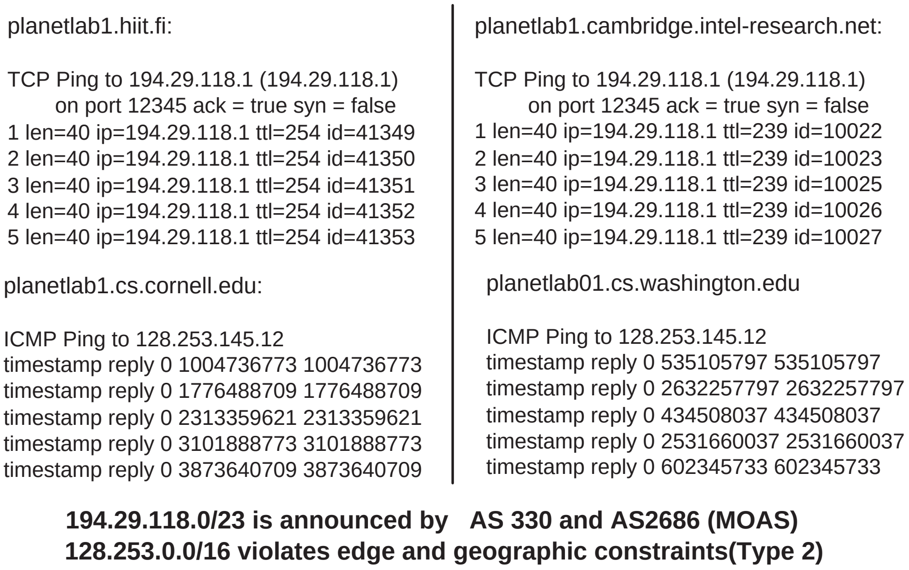
```
plab1.nec-labs.com:

Starting nmap 3.93 at 2006-05-02 15:11 EDT
Initiating SYN Stealth Scan against
 82.146.60.1 [1668 ports] at 15:11
Host 82.146.60.1 appears to be up ...

Interesting ports on 82.146.60.1:
PORT    STATE    SERVICE
22/tcp  open     ssh
179/tcp open     bgp

Device type: general purpose
Running: FreeBSD 4.X
OS details: FreeBSD 4.7 - 4.8-RELEASE

Uptime 76.681 days
          (since Tue Feb 14 21:51:21 2006)
Nmap finished: 1 IP address (1 host up)
          scanned in 38.420 seconds
```

```
planetlab01.erin.utoronto.ca:

Starting nmap 3.93 at 2006-05-02 15:11 EDT
Initiating SYN Stealth Scan
against 82.146.60.1 [1668 ports] at 15:11
Host 82.146.60.1 appears to be up...

Interesting ports on 82.146.60.1:
PORT      STATE  SERVICE
22/tcp    open   ssh

Device type: firewall
Running: Symantec Solaris 8
OS details: Symantec Enterprise
Firewall v7.0.4 (on Solaris 8)

Nmap finished: 1 IP address (1 host up)
          scanned in 11.390 seconds
```

Difference in response fingerprints of suspicious type 2 attack
82.146.60.0/23 is announced by AS 25486. The first hop <8804 2548>
is used only by 6 prefixes and the edge distance is 8968 kilometers

Fig. 9. Conflicting fingerprints of Nmap probing of type-2 attacks.

- **Different open ports:** Figures 9 exhibits inconsistency in open services: BGP (port 179).
- **Different TCP timestamps (uptime):** The host probed from one location may support TCP timestamp, but not from another location, *e.g.*, Figure 9. We also observed significantly different uptime values (Figure 12).



```
planetlab1.hiit.fi:

TCP Ping to 194.29.118.1 (194.29.118.1)
       on port 12345 ack = true syn = false
1 len=40 ip=194.29.118.1 ttl=254 id=41349
2 len=40 ip=194.29.118.1 ttl=254 id=41350
3 len=40 ip=194.29.118.1 ttl=254 id=41351
4 len=40 ip=194.29.118.1 ttl=254 id=41352
5 len=40 ip=194.29.118.1 ttl=254 id=41353
```

```
planetlab1.cambridge.intel-research.net:

TCP Ping to 194.29.118.1 (194.29.118.1)
        on port 12345 ack = true syn = false
1 len=40 ip=194.29.118.1 ttl=239 id=10022
2 len=40 ip=194.29.118.1 ttl=239 id=10023
3 len=40 ip=194.29.118.1 ttl=239 id=10025
4 len=40 ip=194.29.118.1 ttl=239 id=10026
5 len=40 ip=194.29.118.1 ttl=239 id=10027
```

```
planetlab1.cs.cornell.edu:

ICMP Ping to 128.253.145.12
timestamp reply 0 1004736773 1004736773
timestamp reply 0 1776488709 1776488709
timestamp reply 0 2313359621 2313359621
timestamp reply 0 3101888773 3101888773
timestamp reply 0 3873640709 3873640709
```

```
planetlab01.cs.washington.edu

ICMP Ping to 128.253.145.12
timestamp reply 0 535105797 535105797
timestamp reply 0 2632257797 2632257797
timestamp reply 0 434508037 434508037
timestamp reply 0 2531660037 2531660037
timestamp reply 0 602345733 602345733
```

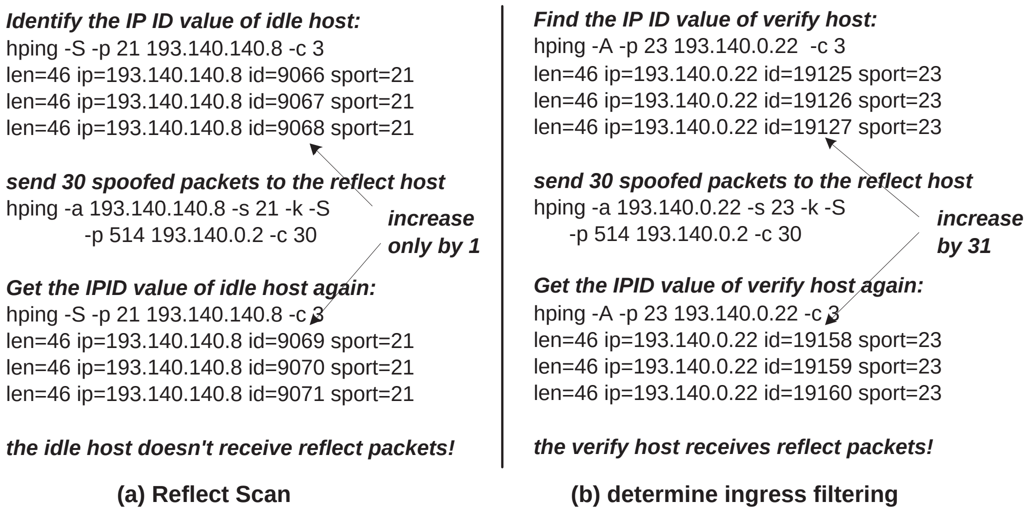
194.29.118.0/23 is announced by   AS 330 and AS2686 (MOAS)
128.253.0.0/16 violates edge and geographic constraints(Type 2)

Fig. 10. Different IP ID values and ICMP timestamp values (potential type-2 attacks).

- **Different ICMP timestamps (local time):** Figure 10 indicates significantly different ICMP timestamp values.

- **Different IP IDs:** For systems expected to have globally incrementing IP-ID patterns, there is a significant difference in IP ID return values or patterns,*e.g.*, Figure 10.

Suspicious subMOAS conflicts and type-4 attacks: For suspected subMOAS (type-3) and type-4 attacks, we use reflect-scan to identify hijacking incidents. The following is a found example of a suspicious subMOAS conflict with the probing results using reflect-scan shown in Figure 11. Prefix 193.140.140.0/24 is announced by AS15390 at 21:27 April 25th, 2006, which has a subMOAS conflict with prefix 193.140.0.0/16 owned by AS8517.



Identify the IP ID value of idle host:

```
hping -S -p 21 193.140.140.8 -c 3
len=46 ip=193.140.140.8 id=9066 sport=21
len=46 ip=193.140.140.8 id=9067 sport=21
len=46 ip=193.140.140.8 id=9068 sport=21
```

send 30 spoofed packets to the reflect host

```
hping -a 193.140.140.8 -s 21 -k -S
          -p 514 193.140.0.2 -c 30
```

Get the IPID value of idle host again:

```
hping -S -p 21 193.140.140.8 -c 3
len=46 ip=193.140.140.8 id=9069 sport=21
len=46 ip=193.140.140.8 id=9070 sport=21
len=46 ip=193.140.140.8 id=9071 sport=21
```

the idle host doesn't receive reflect packets!

(a) Reflect Scan

Find the IP ID value of verify host:

```
hping -A -p 23 193.140.0.22  -c 3
len=46 ip=193.140.0.22 id=19125 sport=23
len=46 ip=193.140.0.22 id=19126 sport=23
len=46 ip=193.140.0.22 id=19127 sport=23
```

send 30 spoofed packets to the reflect host

```
hping -a 193.140.0.22 -s 23 -k -S
     -p 514 193.140.0.2 -c 30
```

Get the IPID value of verify host again:

```
hping -A -p 23 193.140.0.22 -c 3
len=46 ip=193.140.0.22 id=19158 sport=23
len=46 ip=193.140.0.22 id=19159 sport=23
len=46 ip=193.140.0.22 id=19160 sport=23
```

the verify host receives reflect packets!

(b) determine ingress filtering

Fig. 11. A reflect-scan example (type-3).

1) 193.140.140.8 (H_2) in the subnet 193.140.140.0/24 is selected as the idle host, because its IP ID increases regularly by one and has the open port 21.
2) We send SYN/ACK packets to port 21 of H_2 to verify that H_2 responds with RST.
3) Live host 193.140.0.2 (H_1) in the larger prefix 193.140.0.0/16 but not in the subnet is chosen as the *reflect host* with an open port 514.
4) Compare the idle host H_2's IP ID values before and after sending spoofed packets to reflect host H_1 with source IP of H_2. We found the idle host did not receive 30 reflected packets, which may be dropped or delivered somewhere in AS8517 (Figure 11(a)).
5) To verify that the test did not fail due to ingress filtering[6] which may cause the idle host not to receive spoofed packets, we select another idle host 193.140.0.22 similar to H_1 to be the *verify host*.
6) We do the similar test to check for ingress filtering. By comparing the IP ID value of the verify host before and after sending spoofed packets using verify host as the source IP to the reflect host, we find that it receives all reflected packets indicating the lack of ingress filtering in AS8517 (Figure 11(b)).

Since we are confident that reflected packets are sent to the idle host (step 6) and the idle host responds to SYN/ACK packets (step 2), the idle host's IP ID value should be increased, if it received

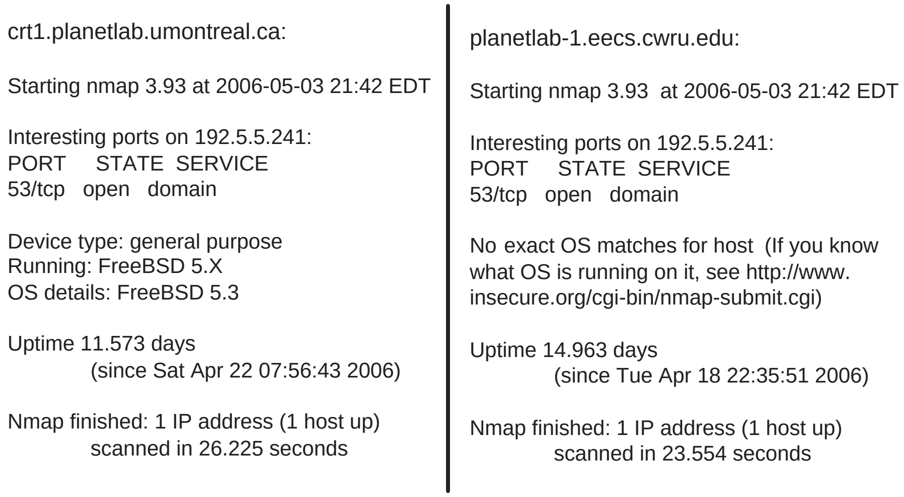
[6] If AS8517 has ingress filtering that filters out incoming traffic with source IP from inside the AS, the spoofed packet cannot reach the reflect host, and no reflect packets will be generated.

them. Thus, we can conclude that this case fails reflect-scan and is highly suspicious as a real hijacking attack.

D. Validation using IP anycast of root DNS servers

For load balancing and robustness consideration, a number of root name-servers are deployed using IP anycast [24]. IP anycast, defined in RFC 1546[40], is an internetwork service where multiple severs support the same service under the same IP address. Currently, 5 out of all 13 DNS root servers (C, F, I, J and K) are using IP anycast, each with multiple servers in different locations [32], [6]. IP anycast for root DNS is achieved by announcing the same prefix and AS number from multiple locations on the Internet, identical to hijacking both the prefix and its AS (type-2 attack). However, this is a valid case; thus, we use it to validate our techniques. The applicable techniques include geographic constraints to identify routing anomalies.

Across 8 days' monitoring, our system successfully captured suspicious updates from four root servers (F, I, J and K), with the exception of the C-root server (c.root-server.net in prefix 192.33.4.0/24 with origin AS2149). After investigating the updates for the C-root server, we find that it only have one upstream provider AS174 which is a large tier-1 ISP. Since AS174 also has a location near to AS2149, the updates for C-root server do not violate the geographical constraint and therefore cannot be captured using that constraint alone. The following is an example of the F-root server (f.root-servers.net) detected by our system. The IP address of the F-root server is 192.5.5.241 in prefix 192.5.5.0/24 announced by AS3557. Figure 12 and Figure 13 clearly show that probing from two different planetlab nodes actually reaches two distinct machines, validating our fingerprinting approach.



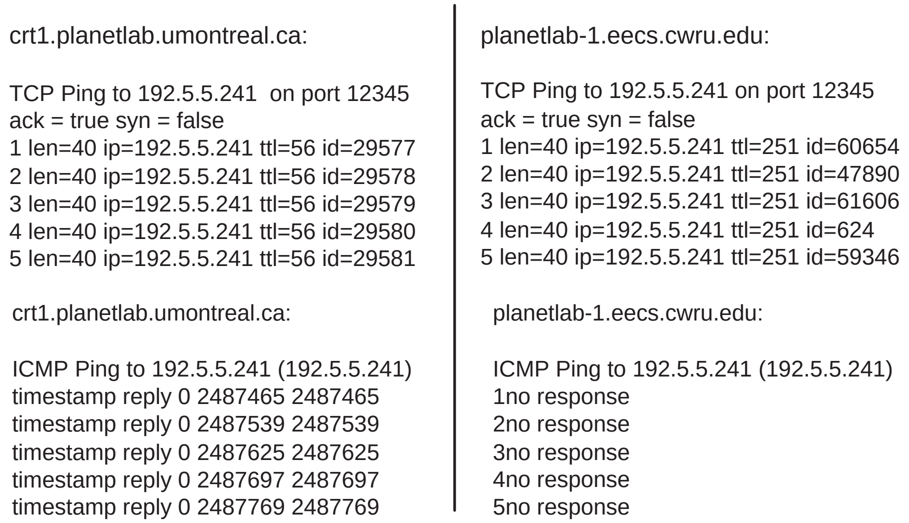
```
crt1.planetlab.umontreal.ca:

Starting nmap 3.93 at 2006-05-03 21:42 EDT

Interesting ports on 192.5.5.241:
PORT     STATE  SERVICE
53/tcp   open   domain

Device type: general purpose
Running: FreeBSD 5.X
OS details: FreeBSD 5.3

Uptime 11.573 days
          (since Sat Apr 22 07:56:43 2006)

Nmap finished: 1 IP address (1 host up)
          scanned in 26.225 seconds
```

```
planetlab-1.eecs.cwru.edu:

Starting nmap 3.93  at 2006-05-03 21:42 EDT

Interesting ports on 192.5.5.241:
PORT     STATE  SERVICE
53/tcp   open   domain

No exact OS matches for host  (If you know
what OS is running on it, see http://www.
insecure.org/cgi-bin/nmap-submit.cgi)

Uptime 14.963 days
          (since Tue Apr 18 22:35:51 2006)

Nmap finished: 1 IP address (1 host up)
          scanned in 23.554 seconds
```

Fig. 12. Different Nmap probing signatures for the F-DNS root server (legitimate type-2 case).

VII. Discussions and conclusions

We discuss several limitations with our work and plans for future improvement. First, our system is triggered based on anomalous routing updates. Although RouteViews provides a comprehensive set of routing updates, it is still conceivable that the system misses some routing anomalies. Another disadvantage is that hijacking may not be visible on the control plane, as the data plane is not



```
crt1.planetlab.umontreal.ca:

TCP Ping to 192.5.5.241  on port 12345
ack = true syn = false
1 len=40 ip=192.5.5.241 ttl=56 id=29577
2 len=40 ip=192.5.5.241 ttl=56 id=29578
3 len=40 ip=192.5.5.241 ttl=56 id=29579
4 len=40 ip=192.5.5.241 ttl=56 id=29580
5 len=40 ip=192.5.5.241 ttl=56 id=29581

crt1.planetlab.umontreal.ca:

ICMP Ping to 192.5.5.241 (192.5.5.241)
timestamp reply 0 2487465 2487465
timestamp reply 0 2487539 2487539
timestamp reply 0 2487625 2487625
timestamp reply 0 2487697 2487697
timestamp reply 0 2487769 2487769
```

```
planetlab-1.eecs.cwru.edu:

TCP Ping to 192.5.5.241 on port 12345
ack = true syn = false
1 len=40 ip=192.5.5.241 ttl=251 id=60654
2 len=40 ip=192.5.5.241 ttl=251 id=47890
3 len=40 ip=192.5.5.241 ttl=251 id=61606
4 len=40 ip=192.5.5.241 ttl=251 id=624
5 len=40 ip=192.5.5.241 ttl=251 id=59346

planetlab-1.eecs.cwru.edu:

ICMP Ping to 192.5.5.241 (192.5.5.241)
1no response
2no response
3no response
4no response
5no response
```

Fig. 13. Different IP ID and ICMP timestamp probing signatures for the F-DNS root server (legitimate type-2 case).

guaranteed to be consistent with advertised routes. We plan to explore continuous monitoring and performance-triggered probing to augment the current approach. We also plan to analyze in more detail the accuracy of fingerprinting techniques. A second more serious limitation is that probing will be limited by limited vantage points and the difficulties of collecting identifying fingerprints due to increasing deployment of firewalls. We plan to explore the coverage based on the probing location and network-based fingerprints. Note that our system can be deployed either by individual networks or by a centralized system. In the latter case, we have demonstrated the scalability of the system, but we did not address the issue of reliably notifying the victims. This is challenging as the victim may not be easily reached due to the impact of IP hijacking. Work by Lad *et al.* [35] suggests the use of diverse paths, without providing absolute guarantee.

In summary, we present a framework for accurate, real-time IP hijacking detection. Our work is based on the novel insight that a real hijacking attack will result in conflicting data-plane fingerprints describing the hijacked network. Using this key difference, we can significantly reduce false positives and more confidently identify IP hijacking without sacrificing efficiency. This is the first work exploiting the consistency between data-plane and control-plane information to identify IP hijacking attacks. Our system can be incrementally deployed without modifying any infrastructure nor requiring support from networks. We have demonstrated the effectiveness and efficiency of a prototype system using real data.

References

[1] NetGeo - The Internet Geographic Database. http://www.caida.org/tools/utilities/netgeo/index.xml.

[2] PlanetLab. http://www.planet-lab.org/.

[3] University of Oregon Route Views Archive Project. www.routeviews.org.

[4] BGP Table Data. http://bgp.potaroo.net/index-bgp.html, 2006.

[5] Fyodor Yarochkin and Meder Kydyraliev and Ofir Arkin. Xprobe2. http://www.sys-security.com/index.php?page=xprobe, 2006.

[6] J. Abley. Hierarchical Anycast for Global Service Distribution. ISC's Technical Note, 2003.

[7] William Aiello, John Ioannidis, and Patrick McDaniel. Origin Authentication in Interdomain Routing. In *Proc. of CCS*, 2003.

[8] A. Barbir, S. Murphy, and Y. Yang. Generic Threats to Routing Protocols. IETF Draft: draft-ietf-rpsec-routing-threats-07, April 2004.

[9] G. Battista, M. Patrignani, and M. Pizzonia. Computing the Types of the Relationships Between Autonomous Systems. In *Proc. IEEE INFOCOM*, March 2003.

[10] Steve Bellovin, Randy Bush, Timothy G. Griffin, and Jennifer Rexford. Slowing routing table growth by filtering based on address allocation policies. http://www.cs.princeton.edu/~jrex/papers/filter.pdf, 2001.

[11] Steven M. Bellovin. A Technique for Counting NATted Hosts. In *Proc. Second Internet Measurement Workshop*, November 2002.

[12] Steven M. Bellovin, John Ioannidis, and Randy Bush. Position Paper: Operational Requirements for Secured BGP. March 2005.

[13] V. J. Bono. 7007 Explanation and Apology. http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html.

[14] Peter Boothe, James Hiebert, and Randy Bush. How Prevalent is Prefix Hijacking on the Internet. NANOG36 Talk, February 2006.

[15] B. Christian and T. Tauber. BGP Security Requirements. IETF Draft: draft-ietf-rpsec-bgpsecrec-04, March 2006.

[16] Michael Freedman, Mythili Vutukuru, Nick Feamster, and Hari Balakrishnan. Geographic Locality of IP Prefixes. In *Internet Measurement Conference (IMC) 2005*, Berkeley, CA, October 2005.

[17] Fyodor. Idle Scanning and related IPID games. http://www.insecure.org/nmap/idlescan.html.

[18] Fyodor. Remote OS detection via TCP/IP Stack Fingerprinting. http://www.insecure.org/nmap/nmap-fingerprinting-article.html, June 2002.

[19] Fyodor. Nmap free security scanner. http://www.insecure.org/nmap/, 2006.

[20] Lixin Gao. On Inferring Autonomous System Relationships in the Internet. In *Proc. IEEE Global Internet Symposium*, 2000.

[21] Vijay Gill, John Heasley, and David Meyer. The Generalized TTL Security Mechanism (GTSM). RFC 3682, February 2004.

[22] Geoff Goodell, William Aiello, Tim Griffin, John Ioannidis, Patrick McDaniel, and Avi Rubin. Working Around BGP: An Incremental Approach to Improving Security and Accuracy of Interdomain Routing. In *Proc. of NDSS*, February 2003.

[23] S. Halabi and D. McPherson. *Internet Routing Architectures*. Cisco Press, Indianapolis, Indiana, second edition, 2000.

[24] T. Hardy. RFC 3258 - Distributing Authoritative Name Servers via Shared Unicast Addresses. RFC 3258, April 2002.

[25] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registation of an Autonomous System(AS). RFC 1930, 1996.

[26] Yih-Chun Hu and Marvin Sirbu. SPV: Secure Path Vector Routing for Securing BGP. In *Proc. ACM SIGCOMM*, 2004.

[27] Geoff Huston. Auto-Detecting Address Hijacking? Presentation at RIPE-50, May.

[28] Carl Hutzler and Ron da Silva. The Relationship Between Network Security and Spam. NANOG 29 Meeting: `http://www.nanog.org/mtg-0310/spam.html`, October 2003.

[29] Young Hyum, Andre Broido, and k claffy. Traceroute and BGP AS Path incongruities. 2003. `http://www.caida.org/outreach/papers/2003/ASP/`.

[30] John W. Stewart III. *BGP4 Inter-Domain Routing in the Internet*. Addison-Wesley, 1999.

[31] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323, May 1992.

[32] Daniel Karrenberg. Distributing K-Root Service by Anycast Routing of 193.0.14.129. RIPE 268, 2003.

[33] Tadayoshi Kohno, Andre Broido, and K. C. Claffy. Remote physical device fingerprinting. In *In Proc. of the 2005 IEEE Symposium on Security and Privacy*, pages 211–225, Washington, DC, USA, 2005. IEEE Computer Society.

[34] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. Topology-Based Detection of Anomalous BGP Messages. In *Proc. of Recent Advances in Intrusion Detection: 6th International Symposium, RAID*, 2003.

[35] Mohit Lad, Daniel Massey, Dan Pei, Yiguo Wu, Beichuan Zhang, and Lixia Zhang. PHAS: a Prefix Hijack Alerting System. In *To appear in Proc. of USENIX Security*, August 2006.

[36] Z. M. Mao, D. Johnson, J. Rexford, J. Wang, and R. Katz. Scalable and Accurate Identification of AS-Level Forwarding Paths. In *Proc. IEEE INFOCOM*, March 2004.

[37] Z. M. Mao, J. Rexford, J. Wang, and R. Katz. Towards an Accurate AS-Level Traceroute Tool. In *Proc. ACM SIGCOMM*, August 2003.

[38] James Ng. Extensions to BGP to Support Secure Origin BGP (soBGP). IETF Draft: draft-ng-sobgp-bgp-extensions-01.txt, November 2002.

[39] Venkata N.Padmanabhan and Lakshminarayanan Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *Proc. ACM SIGCOMM*, 2001.

[40] C. Partridge, T. Mendez, and W. Milliken. Host Anycasting Service. RFC 1546, 1993.

[41] Anirudh Ramachandran and Nick Feamster. Understanding the Network-Level Behavior of Spammers. Technical Report GT-CSS-2006-001, Georgia Tech, January 2006.

[42] Y. Rekhter and T. Li. A Border Gateway Protocol. RFC 1771, March 1995.

[43] Salvatore Sanfilippo. Hping. `http://www.hping.org/`, 2006.

[44] Bradley R. Smith and J.J. Garcia-Luna-Aceves. Securing the Border Gateway Routing Protocol. In *Proc. of Global Internet*, 1996.

[45] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public internet measurement facility, 2002.

[46] Stephen Kent and Charles Lynn and Karen Seo. Secure Border Gateway Protocol (Secure-BGP). *IEEE J. Selected Areas in Communications*, 2000.

[47] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proc. IEEE INFOCOM*, 2002.

[48] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy H. Katz. Listen and Whisper: Security Mechanisms for BGP. In *Proc. first Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[49] Tao Wan, Evangelos Kranakis, and P.C. van Oorschot. Pretty Secure BGP (psBGP). In *Proc. NDSS*, 2005.

[50] Lan Wang, Xiaoliang Zhao, Dan Pei, Randy Bush, Daniel Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. Protecting BGP Routesto Top Level DNS Servers. In *Proc. of IEEE International Conference on DistributedComputing Systems (ICDCS)*, 2003.

[51] Meiyuan Zhao, Sean Smith, and David Nicol. Aggregated Path Authentication for Efficient BGP Security. In *Proc. of CCS*, 2005.

[52] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. An Analysis of BGP Multiple Origin AS (MOAS) Conflicts. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[53] Xiaoliang Zhao, Dan Pei, Lan Wang, Daniel Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. Detection of Invalid Routing Announcement in the Internet. In *Proc. of DSN*, 2002.