Detecting Prefix Hijackings in the Internet with Argus

Xingang Shi[†§] shixg@cernet.edu.cn
Yang Xiang[‡§] sharangxy@gmail.com
Zhiliang Wang[†§] wzl@cernet.edu.cn
Xia Yin[‡§] yxia@tsinghua.edu.cn
Jianping Wu[†‡§] jianping@cernet.edu.cn

[†]Institute for Network Sciences and Cyberspace, Tsinghua University
[‡]Department of Computer Science & Technology, Tsinghua University
[§]Tsinghua National Laboratory for Information Science and Technology (TNList)

ABSTRACT

Border Gateway Protocol (BGP) plays a critical role in the Internet inter-domain routing reliability. Invalid routes generated by mis-configurations or forged by malicious attacks may hijack the traffic and devastate the Internet routing system, but it is unlikely that a secure BGP can be deployed in the near future to completely prevent them. Although many hijacking detection systems have been developed, they more or less have weaknesses such as long detection delay, high false alarm rate and deployment difficulty, and no systematic detection results have been studied. This paper proposes *Argus*, an agile system that can accurately detect prefix hijackings and deduce the underlying cause of route anomalies in a very fast way. *Argus* is based on correlating the control and data plane information closely and pervasively, and has been continuously monitoring the Internet for more than one year. During this period, around 40K routing anomalies were detected, from which 220 stable prefix hijackings were identified. Our analysis on these events shows that, hijackings that have only been theoretically studied before do exist in the Internet. Although the frequency of new hijackings is nearly stable, more specific prefixes are hijacked more frequently. Around 20% of the hijackings last less than ten minutes, and some can pollute 90% of the Internet in less than two minutes. These characteristics make *Argus* especially useful in practice. We further analyze some representative cases in detail to help increase the understanding of prefix hijackings in the Internet.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: [General Security and Protection]; C.2.3 [**Network Operations**]: [Network Monitoring]

Keywords

BGP, Security, Prefix Hijacking, Hijacking Detection

1. INTRODUCTION

The Internet is composed of tens of thousands of Autonomous Systems (ASes) which operate individual parts of the infrastructure. As Border Gateway Protocol (BGP) [27] controls the packet forwarding path between ASes, it plays a critical role in the efficiency and reliability of the Internet. However, because of the lack of security considerations, several BGP security problems have not been well resolved yet [25]. For example, the route information received from neighbors can not be validated, so invalid routes may cause packets being forwarded along wrong paths. This is known as the prefix hijacking problem.

Prefix hijacking is often generated by accidental misconfigurations, and may cause serious routing problems and economic losses [24]. For instance, in 2008, Pakistan Telecom hijacked YouTube for two hours [29], while in 2010, one AS of China Telecom hijacked more than 50,000 prefixes (15% of the Internet) of 170 different countries [28]. Malicious users also use prefix hijacking to intercept or drop certain traffic, or to launch spam or DDoS attacks.

To improve the security of BGP, several methods have been proposed, which broadly fall into three categories: cryptographic based prevention, anomaly mitigation, and anomaly detection. Cryptographic approaches [17, 21] usually use the Public Key Infrastructure (PKI) to ensure the authentication of routing announcements and prevent hijacking. They often consume significant router resources due to their computation and storage complexity, and are unlikely to be widely deployed very soon [10]. Anomaly mitigation approaches [30, 18] propose to ignore or demote suspicious routes once they are detected. These approaches are hard to deploy, and their misjudgments may disturb the routing system. At last, anomaly detection approaches [36, 16, 34, 12] aim to discover anomalous information or behavior in BGP announcements and raise alarm. As a viable secure scheme in the current stage, they offer valuable information for the understanding and diagnosis of the routing system, and help to push ISPs to deploy cryptographic solutions.

This paper focuses on the third category, anomaly detection. Although many workable detection systems have been developed, more or less, they have weaknesses such as long detection delay, high false alarm rate and great deployment difficulty. Our solution, *Argus*, exploits a key observation about IP prefix hijacking: polluted routers (i.e., routers whose traffic is hijacked) usually can not communicate with the hosts in the victim prefix, while normal routers can. This observation makes hijacking distinguishable from other anomalous route events, and let *Argus* achieve four superior capabilities over existing solutions: (1) Low false positive rate: *Argus* closely correlates the control-plane anomaly and the data-plane reachability information of a large number of vantage points, so that prefix hijacking can be accurately distinguished from other BGP events. (2) Low false negative rate: *Argus* monitors a much wider range of routing anomalies, including origin AS and AS-path anomalies, and considers the sub-prefix problem, so the chance of missing a hijacking event is much lower. (3) Realtime detection: *Argus* usually detects a hijacking within a few seconds, while existing methods often need orders of magnitudes longer. (4) Agile: *Argus* is very easy to deploy, and does not need to upgrade any existing network infrastructure or device.

Based on our earlier prototype [32] of *Argus*, [1] this paper makes four new contributions as follows:

- System improvement and service. We carefully evaluate the performance of *Argus*, and continuously improve it from several aspects. In particular, we tune the system parameters based on our monitoring practice, and add a scheme to collect and choose live IP addresses, resulting in better detection accuracy. We contribute it to the community by providing several public online services, and by raising alerts in realtime via different channels. We also discuss the limitations of *Argus* to facilitate further studies in this field.

- Hijacking monitoring and analysis. Since being deployed one year ago, *Argus* has already identified 220 stable prefix hijackings (i.e., hijackings that last longer than 10 seconds) from 40K anomalous BGP events. We systematically analyze these events, and get many interesting results. In particular, hijackings that have only been theoretically studied before do exist in the Internet. Although the frequency of new hijackings is nearly stable, more specific prefixes are hijacked more frequently. Around 20% of the hijackings last less than ten minutes, and some can pollute 90% of the Internet in less than two minutes. These characteristics make *Argus* especially useful in practice, and to the best of our knowledge, we are the first to present these statistics and trend of prefix hijackings in the Internet.

- Hijacking confirmation and diagnosis. Starting from March 2012, we directly contact the corresponding network operators if a stable prefix hijacking is detected. We provide detailed information collected by *Argus* to help their diagnosis work. More than 30% of those hijackings have been confirmed, and we did not received any objection to our judgments.

- Cause analysis of anomalous route events. We extend *Argus* to not only detect prefix hijackings, but also deduce the possible cause of anomalous route events and classify them into different categories. We further analyze several representative cases in detail, in order to help increase the understanding of hijackings in the Internet.

[1]In particular, most of Section 4.1 and Section 4.2, and part of Section 4.4 are replications of those in [32], with moderate revisions.

This paper is organized as follows. Section 2 introduces the necessary background of prefix hijacking and some related work. Section 3 illustrates our key observation for hijacking detection, and Section 4 describes the design details of *Argus*. Section 5 presents our Internet monitoring practice in the past year, while Section 6 further studies some representative hijacking cases and other anomalous route events. The pros and cons of our system are discussed in Section 7, and finally, Section 8 concludes.

2. BACKGROUND AND RELATED WORK

2.1 Prefix Hijacking

Each AS in the Internet manages a number of networks, which can be expressed as IP prefixes. BGP propagates routes (AS-paths) between neighboring ASes to determine how these networks can be reached, i.e., through which AS-path. An AS-path $p = \langle a_n, \cdots, a_0 \rangle$ is a sequence of ASes, where the last one, a_0, is called the origin AS of p. When the network connectivities change, BGP UPDATEs will be generated, and new AS-paths will be announced throughout the Internet.

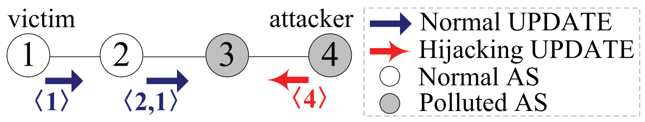


Figure 1: An example of BGP prefix hijacking.

As shown in Figure 1, AS1 announces a route for its prefix f with an AS-path $\langle 1 \rangle$, and AS2 and AS3 will get a route to f (i.e., $\langle 1 \rangle$ and $\langle 2, 1 \rangle$ respectively). However, AS4 can also announce a route for f with a forged but shorter AS-path $\langle 4 \rangle$. As AS3 considers the shorter path to be better [2] than the valid path it has previously received (i.e., $\langle 2, 1 \rangle$), it will forward the traffic destined to f to AS4. In this way, AS4 effectively hijacks the traffic by forging a route. We call AS3 *polluted* by the route announced by AS4, and indicate that by a gray circle in the figure.

A manipulator hijacks the traffic destined to a victim prefix either accidentally or intentionally. Based on how the hijacked traffic is finally dealt with, hijackings can be classified into the following two categories:

- Blackholing: the traffic is dropped, although the manipulator can eavesdrop on it at first.

- Imposture or Interception: the traffic is handled either by mimicking the victim's action, or by forwarding to the victim.

This paper only focuses on detecting blackholing due to the following reasons. First, mis-configurations will typically cause blackholing. Second, Although a malicious attacker may perform any of these actions, it is very difficult for him to mimic all behaviors of the victim, or to successfully forward the traffic to the victim, if not impossible. Third, detecting interception is very hard [9] since any AS can be viewed as a man-in-the-middle. Last, an end-to-end mechanism can be much more effective in protecting traffic from either imposture or interception.

[2]This is a simplified example, as the length of an AS-path is only one factor in BGP's route decision process.

Table 1: Pros and Cons of Hijacking Detection Methods

Methods	Detection delay	Attacker Information	Accuracy	Sub-prefix hijacking	Scalability	Deployment
Control-plane [8, 22, 12, 4]	Realtime	Yes	Low	Yes	Good	Easy
Data-plane [36, 34]	Minutes	No	Medium	No	Poor	Easy
Combination [16]	Minutes	Yes	High	Yes	Good	Hard
Correlation (*Argus*)	Realtime	Yes	High	Yes	Good	Easy

2.2 Related Work

Existing hijacking detection proposals can be classified into three categories based on the type of information they use, as compared in Table 1.

The control-plane approaches [8, 22, 12, 4] passively monitor the BGP routing information to collect anomalous events. They are easy to deploy, react in realtime, and can deduce information like who the attacker is or when the attack starts. However, they are fairly inaccurate and usually raise many unnecessary alarms, since they can not tell normal route events from hijackings very well [35].

The data-plane approaches [34, 36] continuously probe the Internet to detect whether any data path changes. The probing often involves a large number of networks, and the paths are collected by tools such as *traceroute*, hence these methods suffer from poor scalability and large latency. Due to a lack of control-plane information, they can not detect sub-prefix hijacking, and can provide little diagnosis such as who the attacker is.

Control and data plane information can be combined to complement each other. In [16], data-plane probing is launched only after an anomalous UPDATE is received, so the scalability is much better than a pure data-plane method. However, this system relies on some complicated probing methods like traceroute, nmap and IP/TCP timestamp, and also bears large latencies up to several minutes. More importantly, since customized software have to be installed on its vantage points, this solution is hard to deploy. Typically, a limited number of Planetlab nodes are used, which suffer from many side effects. In particular, the Planetlab nodes are unaware of BGP, so the benefits of combining control and data plane are limited. For example, the method can not distinguish prefix hijacking from BGP anycast, since both will result in different probing results. The nodes are mostly located in stub academic networks, so the data paths that can be probed are not diversified enough, and the capability of detecting hijackings is reduced.

We think the deep-rooted reason of their inefficiency is a lack of *close and pervasive correlation* between the control and the data plane. *Argus* makes a step forward, through correlating the control-plane route and data-plane reachability on a number of distributedly located diagnosis nodes in realtime.

There are also many measurement studies on BGP related problems, for example, analyzing route changes and their root causes [11, 14], measuring how the Internet end-to-end performance is impacted by BGP policy, timer or updates [31, 13], quantifying path exploration and slow convergence [26], and studying global black-holes in the Internet [20]. We monitor and identify anomalous route events and prefix hijackings, and systematically study the statistics and trend of these events that occurred over the past year, to help better understand the characteristics of Internet hijackings.

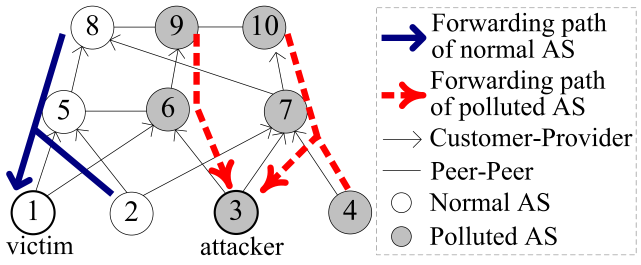


Figure 2: Different forwarding paths from normal ASes and polluted ASes during a hijacking.

3. KEY OBSERVATION

Prefix hijacking can be caused by changing either the origin AS, or some AS-path segments in a route announcement. Typically, only a portion of the ASes in the Internet can be polluted by a hijacking, due to the complexity and diversity of the Internet. As shown in Fig. 2, the attacker AS3 announces itself as the origin AS of the prefixes owned by the victim AS1,[3] and successfully pollutes AS4, AS6, AS7, AS9 and AS10. [4] These ASes will change their routes and forward the traffic which is originally destined to AS1 along the dashed line, while the other unpolluted ASes will forward such traffic along the solid line as usual. Unless the attacker forwards the hijacked traffic to AS1, a host in a polluted AS will not be able to communicate with a host in the victim, AS1, since the traffic goes to a black-hole.

Generally speaking, when the prefix hijacker acts as a black-hole, it is likely that a data-plane probe sent from an unpolluted AS to the victim prefix can successfully arrive at the destination and get a reply, while a probe from a polluted AS will most probably get no reply. This simple observation is the key basis of *Argus*, our prefix hijacking detection system. Next, we analyze this data-plane reachability problem in more detail.

As shown in Fig. 3, after an anomalous AS-path affects a portion of ASes in the Internet, the reachability to the affected prefix reflects different scenarios: [5]

1. When multi-origin (e.g., due to AS multi-homing, BGP anycast or static route) or traffic engineering is deployed, a new origin AS or new AS-path segments are introduced, and traffic may reach the destination prefix via different paths. In this case, both the normal ASes and the affected ASes can reach the destination prefix. (Fig. 3(a))

[3]Other cases, such as announcing false AS-path segments, will be discussed in Section 4.
[4]We assume AS6 prefers AS3 to AS1.
[5]By reachability, we mean one host can communicate with the other one in both directions, i.e., probe and reply.

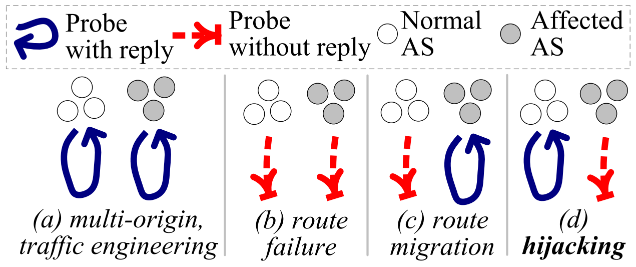


Figure 3: Data-plane reachabilities in different route events.

2. When node or link failure happens, there may not exist a valid route to the destination prefix, then no AS can reach it. Firewalls or inactive hosts may also lead to this result. (Fig. 3(b))

3. When route migrates (e.g., due to network merge or prefix acquisition, the ownership of a prefix changes, or new AS-path segments are used, either temporarily or permanently), ASes that have learned the new route are able to reach the destination prefix, while ASes have not learned the new route are not. (Fig. 3(c))

4. Under a hijacking as explained above, normal ASes can reach the destination prefix while polluted ASes can not. (Fig. 3(d))

The relation between the routes (control-plane) and the reachabilities (data-plane) in different ASes can be regarded as a fingerprint of a route event, and by this fingerprint, we can figure out whether a route change is actually caused by a hijacking. In *Argus*, we use *show ip bgp* to get the BGP routes, and use *ping* to test the reachability. These two commands are very simple, and are commonly enabled in a vast number of public route-servers and looking-glasses. These diagnosis nodes lie in different ASes nearly all over the Internet, and using them can greatly enhance our detection capabilities.

However, the real situation in the Internet is more complicated than the four scenarios above. Since there exists a convergence process after a route event happens, different ASes often experience route change at different time, and the control-plane does not always coincide with the data-plane. For example, as shown in Fig. 4, AS6 has already recovered from the pollution but its route UPDATE has not yet reached AS9. At this moment, probe from AS9 can get a response, even though AS9 is using a polluted AS-path.

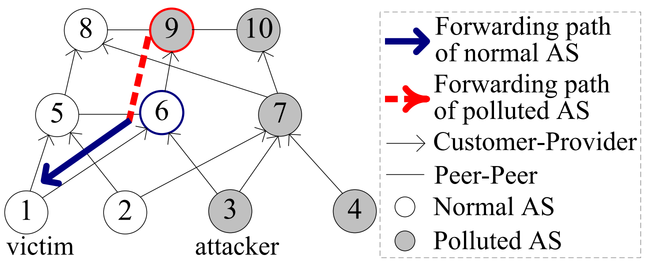


Figure 4: A polluted AS can reach the victim prefix during route convergence.

On the other hand, a normal AS not affected by a route change may also lose connectivity to certain destinations due to reasons not related to BGP [20]. Therefore, directly matching the control-plane status and data-plane probing results at each diagnosis node is error-prone. Instead, *Argus* correlates the control-plane and data-plane information of a large number of diagnosis nodes in a statistical manner. The details of this correlation process will be described in Section 4, and evaluated in Section 5.

4. PREFIX DETECTION SYSTEM - ARGUS

4.1 Overview

Argus is composed of three main modules: the Anomaly Monitoring Module (*AMM*), the Live-IP Retrieving Module (*LRM*), and the Hijacking Identification Module (*HIM*). The whole system is depicted in Figure 5.

The Anomaly Monitoring Module receives live BGP UPDATEs from BGPmon [3], a real-time BGP feed in the Route Views project [33] that collects UPDATEs from routers all over the world. When the *AMM* receives an UPDATE, it will check whether the embedded AS-path has an anomalous origin AS or an anomalous AS-path segment according to its local routing information database, as described in Section 4.2.

The Live-IP Retrieving Module maintains a pool of live IPs by periodically collecting from various sources. For an anomalous prefix f, it will carefully select an live IP as the probing target of the identification module. Details of the *LRM* will be discussed in Section 4.3.

The Hijacking Identification Module is responsible for collecting information from a number of vantage points, computing the fingerprint for the anomalous route event, and making final decisions about whether the suspicious prefix is really hijacked. It is the core of *Argus*, and Section 4.4 will present it in detail.

4.2 Monitor Anomalies

In *Argus*, we consider three types of route anomalies: origin anomalies (OA), adjacency anomalies (AA), and policy anomalies (PA), as illustrated in Fig. 6.

An origin anomaly (OA) happens when the origin AS in the AS-path of a prefix changes to a different AS, or a new prefix appears. This is the simplest case of prefix hijacking, and is the focus of all existing hijacking detection solutions. For example, in Figure 6(a), AS3 is not the origin AS of the prefix f, thus any AS-path for f ending with $\langle 3 \rangle$ is anomalous. *Argus* maintains a database which keeps track of the normal origin AS of each prefix to help detect origin anomalies.

Hijacking can also be caused by changing some AS-path segments, e.g., by using a command like *as-path prepend* incautiously, or by intentionally removing some ASes to shorten the path length. Such hijackings are key security issues considered by the IETF [5], but have not got enough attentions in measurement studies. To detect such anomalies, it is impractical to maintain all the AS-paths due to their enormous quantity. Instead, *Argus* only focuses on the neighboring ASes pairs and triples that occur in an AS-path.

In an AS-path of a prefix, if any pair of neighboring ASes has not appeared in any UPDATE before, *Argus* will report an adjacency anomaly (AA), since such a pair indicates that the adjacency between two ASes changes. Figure 6(b) illus-

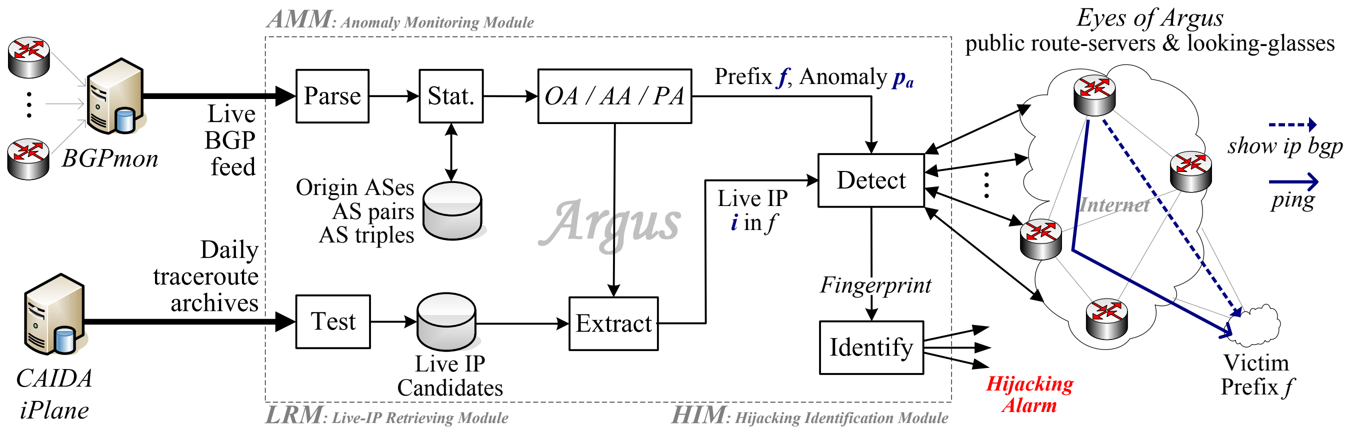


Figure 5: The architecture of *Argus*.

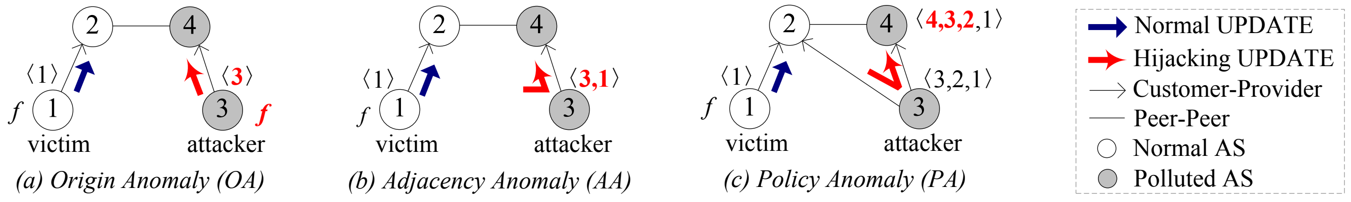


Figure 6: The three kinds of anomalies considered in *Argus*.

trates such an example, where AS3 does not directly connect to AS1, but erroneously announces an AS pair $\langle 3,1 \rangle$ in the AS-path.

On the other hand, since BGP is policy based, it often cares about whether routes learned from one neighbor should be propagated to another neighbor. Such a relationship between one AS with its two neighbors usually constitutes a BGP policy. For example, in Figure 6(c), although AS3 directly connects to AS2 and AS4, it should not announce routes learned from its provider AS2 to another provider AS4, so the triple $\langle 4,3,2 \rangle$ should not appear in any route. In case that a new triple of neighboring ASes appears, e.g., an AS-path containing $\langle 4,3,2 \rangle$ is received in this example, *Argus* will report a policy anomaly (*PA*).

In most cases, an AS uses the same policy for all the prefixes learned from the same neighbor, and does not care about the other non-adjacent ASes, so we think monitoring adjacency and policy anomalies can be an effective replacement for monitoring the whole AS-path. The latter needs to maintain all normal AS-paths while the former only needs to maintain all normal neighboring AS pairs and triples, as is done in *Argus*.

By considering the above three kinds of anomalies, *Argus* can detect a much wider range of hijackings, as will be demonstrated in Section 5.

Once the *AMM* detects an anomaly, it will notify the *HIM* to identify whether a hijacking truly happens. If not, it will add the new origin AS or the AS pair/triple into the local database since they are normal. The database will also be refreshed periodically by removing those origin ASs or AS pairs/triples inactive for a long period (i.e., more than two months).

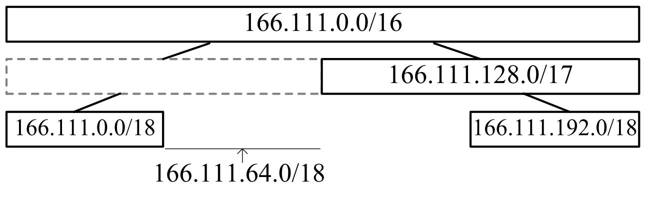


Figure 7: Finding an address block covered by 166.111.0.0/16 but not by any of its sub-prefixes.

4.3 Retrieve Live IPs

When anomaly is detected for a prefix f, the Live-IP Retrieving Module (*LRM*) tries to find a live IP address i in f. The reachability to this address is a key factor for determining whether f is hijacked. Due to the longest prefix matching mechanism, i should be covered by f, but not by any sub-prefix of f. Fig. 7 shows such an example, where the anomalous prefix 166.111.0.0/16 has three sub-prefixes announced, and our algorithm will try to find a live IP in the range of 166.111.64.0/18.

The *LRM* accomplishes this task by maintaining a local database of candidate live IP addresses in all announced prefixes in the Internet. These IPs are collected from CAIDA's Ark [2] and iPlane [23] daily traceroute results [6], and from many DNS records [6].

When a live IP address is asked for, the *LRM* will first retrieve all live IP addresses in the database according to the given prefix, and then launch parallel probings to select one that is currently reachable. This process is very quick, and

[6]These two projects perform traceroute from various vantage points to construct a router level atlas of the Internet.

usually completes within one second. When no reachable IP can be found, for example, when the server hosting the *LRM* is also polluted, the most active IP in the probing history will be selected. Inevitably, using an IP reachable in history may introduce false negatives if it is not alive at the moment, but our experiments show that this probability tends to be relatively low.

4.4 Identify Hijackings

The Hijacking Identification Module (*HIM*) employs a number of public route-servers and looking-glasses, called as the *eyes of Argus*, to finally determine whether an anomaly detected by the *AMM* is actually a prefix hijacking or not. At each *eye*, *Argus* activates two processes, one for gathering the control-plane route status of the victim prefix f, and the other for obtaining the data-plane reachability to the selected live IP i.

Specifically, *Argus* uses *show ip bgp* to extract the best BGP route $p_{t,j}(f)$ that its j-th *eye* chooses for the anomalous prefix f, in the t-th second. Then it checks whether $p_{t,j}(f)$ is affected by the anomalous origin AS (or AS pair/triple) reported by the *AMM*. Meanwhile, *Argus* uses *ping* to test whether the live IP i is reachable from each *eye*. Both of the two commands are very fast, so realtime results can be collected once every second.

After an anomaly is reported, we continuously do that for W seconds on N *eyes*. At the t-th second ($1 \le t \le W$), we compose the control-plane results of all *eyes* into a binary vector $C_t = \{C_{t,j} | 1 \le j \le N\}$, where

$$C_{t,j} = \begin{cases} 0, & \text{if } p_{t,j}(f) \text{ is affected by the anomaly} \\ 1, & \text{if } p_{t,j}(f) \text{ is not affected by the anomaly} \end{cases}$$

and compose the data-plane reachability results into a binary vector $D_t = \{D_{t,j} | 1 \le j \le N\}$, where

$$D_{t,j} = \begin{cases} 0, & \text{if the } j\text{-th } \textit{eye} \text{ gets no reply from the IP } i \\ 1, & \text{if the } j\text{-th } \textit{eye} \text{ gets a reply from the IP } i \end{cases}$$

Our key observation in Section 3 tells that, the relationship between C_t and D_t can expose the underlying cause of a route event, e.g., users in an unpolluted AS should be able to get a reply ($C_{t,j} = D_{t,j} = 1$) while users in a polluted AS should not ($C_{t,j} = D_{t,j} = 0$). Due to the possible inconsistency between the control and data plane at a node, we do not directly compare $C_{t,j}$ and $D_{t,j}$ of the same j-th *eye* one by one. Instead, we utilize the *correlation coefficient* of the two vectors C_t and D_t to measure this relationship, mathematically defined as

$$F_t = \frac{\sum_{j=1}^{N} \left[(C_{t,j} - \overline{C_t})(D_{t,j} - \overline{D_t}) \right]}{\sqrt{\sum_{j=1}^{N} (C_{t,j} - \overline{C_t})^2 \times \sum_{j=1}^{N} (D_{t,j} - \overline{D_t})^2}} \qquad (1)$$

where $\overline{C_t}$ and $\overline{D_t}$ are the average of $C_{t,j}$'s and $D_{t,j}$'s ($1 \le j \le N$) on all *eyes*, respectively. We also call F_t and $\overline{D_t}$ the *fingerprint* and *reachability* of the route event at time t, respectively.

Fig. 8 schematically illustrates the fingerprint and reachability distribution of different route events. When F_t is close to 1, D_t and C_t have a strong positive correlation (i.e., most polluted *eyes* cannot get reply from the victim prefix, while most normal *eyes* can), then very probably a prefix hijacking is going on. Since routing may be inconsistent and

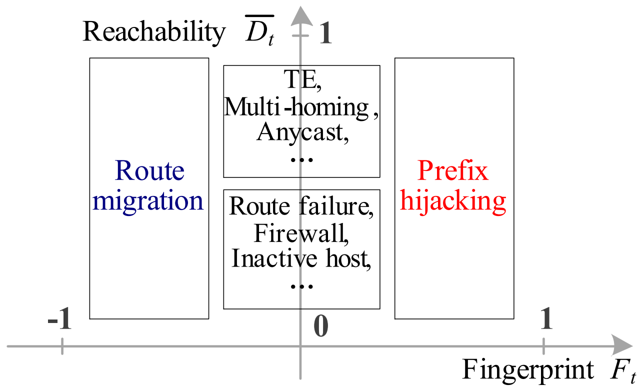

Reachability $\overline{D_t}$
1
Route migration
TE, Multi-homing, Anycast, ...
Route failure, Firewall, Inactive host, ...
Prefix hijacking
-1
0
1
Fingerprint F_t

Figure 8: Fingerprint and reachability of different route events.

instable, we raise a hijacking alarm only when F_t is greater than a threshold μ for at least a period of T seconds. The choice of μ and T will affect the detection accuracy, as will be discussed in Section 5. We compute F_t in each second for a total of 120 seconds.

When F_t is close to -1, a strong negative correlation exists between C_t and D_t (i.e., most polluted *eyes* can get reply while most normal *eyes* cannot), then the anomaly is very possibly just a route migration.

When D_t and C_t are not strongly correlated (i.e., F_t is close to 0), there are two possible situations: (1) if the average data-plane reachability $\overline{D_t}$ approaches 1, which means most *eyes* can communicate with hosts in the target prefix, then the anomalous route is often a normal backup route, used for multi-homing, BGP anycast, backup path, or traffic engineering; (2) if $\overline{D_t}$ approaches 0, which means most *eyes* can not reach the target prefix, then a route failure or a firewall may exist in the middle, or because the probing target we choose is not alive.

We can see that *Argus* has very little dependency on external nodes. It only receives live BGP feed, collects active IP addresses, and logins to route-servers or looking-glasses to execute two simple commands. All of these resources are publicly and widely available, thus it is very easy to deploy, and can monitor the Internet closely and pervasively.

We note that, when the denominator of equation (1) is zero, we set F_t to be zero. This happens when either C_t or D_t is a vector of all zero's or all one's. In particular, when C_t is a vector of all one's, all our *eyes* are polluted, and we will not raise hijacking alarms any more. However this rarely happens, as will be shown in the next section.

Based on *Argus*, we have built several online services. We raise realtime hijacking alarms via a public mailing list and twitter, which network operators can subscribe freely. We post the information of all anomalous routes and hijackings, together with their statistics, on our website, to facilitate further processing and analysis. We also provide web service APIs that can access our monitoring system in realtime so that other systems can integrate the hijacking detection capability. We make the services robust by deploying them in different ASes, so even when some of them are polluted under an attack, the victim AS can still be informed as soon as possible. All the detail of these services can be found at `argus.csnet1.cs.tsinghua.edu.cn`.

5. INTERNET MONITORING PRACTICE

Argus has been continuously monitoring the Internet for one year, starting from May 2, 2011. During this period, 40K anomalous route events were reported [7], and 220 stable hijackings were identified. Due to a lack of the ground truth, we use several methods to verify our identification results. First, we query all valid Route Origin Authorizations (ROAs) [17], which are digitally signed and can be used to verify whether a prefix is announced by its authorized owner AS. Those anomalies with ROA records can be used as a validation set to test our algorithm: if any such anomaly is identified as a hijacking, then it must be a false positive. However the ROA records are still rather incomplete, and only 266 anomaly alarms can be used for validation, so we also query the Internet Routing Registry (IRR)[7] to get more prefix-origin pairs, where matching records are found for 3988 anomalies. [8] Although the IRR information is not guaranteed to be correct, we use it for a rough evaluation. Second, we announce all the identified hijackings via several public channels like twitter and mailinglist, [9] and have recently started to directly contact the corresponding network operators. Till now, we have not got any objection to our identification results, and 10 out of 31 network operators being contacted have confirmed our results. Last, we also query the IRR to get other auxiliary information, such as import/export policies, and query other databases such as *whois* and Cyclops [12] to validate our results. Since that needs a lot of labor work, the analysis is still on going.

5.1 System Parameters and Performance

The higher the correlation between the control and the data plane, the more likely a hijacking is going on. When an anomaly is reported by the Anomaly Monitoring Module (AMM), the Hijacking Identification Module (HIM) starts to compute the fingerprint F_t for this anomaly, in every second of a continuous period of W seconds. Since the Internet often converges in less than two minutes except for some unusual cases, and prefix hijacking is one kind of T_{short} event, which moves from a longer or less preferred to a shorter or more preferred path, with convergence time typically less than 40 seconds [26], we set $W = 120$ seconds. During this period, if F_t is larger than a hijacking fingerprint threshold μ, we say the event is in a *suspicious hijacking state*, and if the duration of the suspicious state, denoted by d, is no smaller than a hijacking duration threshold of T seconds, we classify this anomaly as a stable hijacking. Using a large hijacking fingerprint threshold μ or duration threshold T may miss some real hijacking events, while using a small μ or T may cause unnecessary alarms.

To model the high correlation coefficient a hijacking should have, μ can not be too small. Fig. 9 shows how the false positive rate (FPR) changes with μ, where μ varies from 0.4 to 1.0. When we use the 266 route anomalies that have corresponding ROAs as the validation set [10], and the threshold T is set as 10 seconds, no anomaly is ever falsely identified as a hijacking. If the 3988 alarms with IRR *route origin* records are used as the validation set, only 0.3% alarms are falsely identified as hijackings when $\mu = 0.5$, and the error decreases to 0.2% when $\mu = 0.6$, as represented by the line with squares. Since IRR records are often outdated and not guaranteed to be correct, the real error should be even smaller. For comparison, the dash line represents the results using ROAs but without filtering the alarms by their duration (i.e., when $T = 1$). If $\mu = 0.5$, there are four cases misclassified as hijackings, and when $\mu = 0.6$, there is no false identification.

Fig. 10 plots the fingerprint distribution of all identified stable hijackings when we use the default threshold $\mu = 0.6$ and $T = 10$. It is clear that most stable hijackings have a fingerprint much larger than 0.6, as indicated by the peak near 1.0. This confirms our conjecture that the control-plane route status C_t and the data-plane reachability D_t has a very high positive correlation when hijacking happens.

Fig. 11 shows the cumulative distribution function (CDF) of the duration of all suspicious hijacking alarms (i.e., when the fingerprint $F_t > \mu$). Many alarms are transient with a short lifetime, and the distributions do not change much when μ varies from 0.6 to 0.9. In practice, we use $T = 10$ seconds to filter those non-stable route events that may cause μ temporarily high. Even If they are real hijackings,

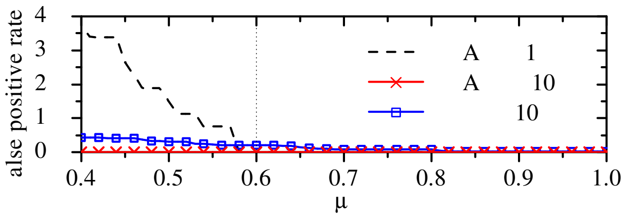


Figure 9: False Positive Detection Rate v.s. μ.

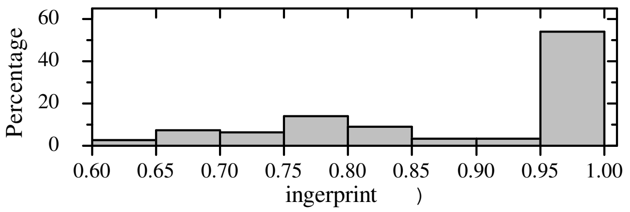


Figure 10: Fingerprint distribution of all identified stable hijackings ($\mu = 0.6$).

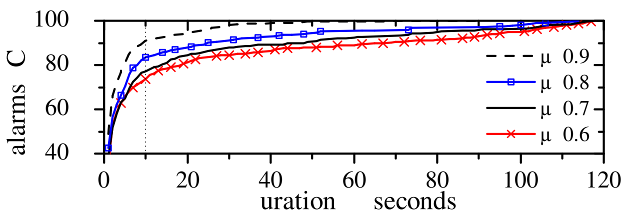


Figure 11: Duration of suspicious alarms (CDF).

[7] We have aggregated the raw BGP UPDATEs from different routers into anomalous events, using algorithms like in [26].

[8] The IRR currently contains about 36% of all the Internet prefixes, while the ratio that an origin anomaly can be matched in it is 20%.

[9] Full details about these channels can be found at http://argus.csnet1.cs.tsinghua.edu.cn/about/.

[10] These anomalies are all caused by normal route events, so FPR is the percentage of hijackings identified in this set.

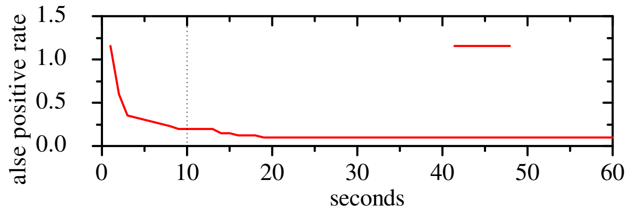


Figure 12: False Positive Detection Rate v.s. T.

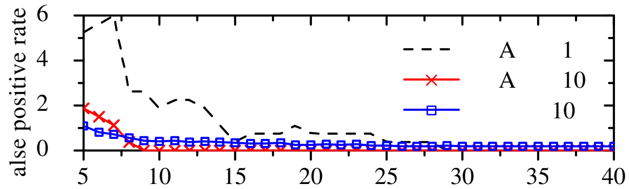


Figure 14: False positive detection rate v.s. #*eyes*.

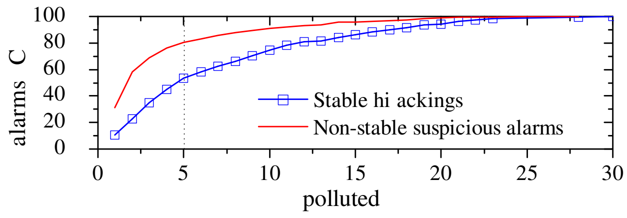


Figure 13: The numbers of polluted *eyes* in stable hijackings and non-stable suspicious alarms.

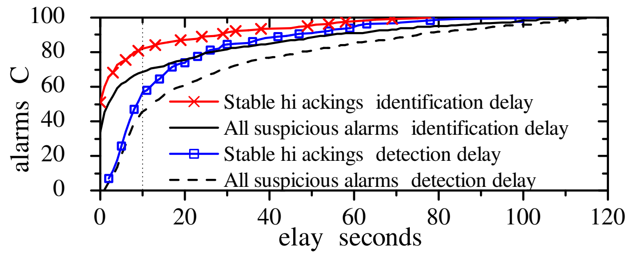


Figure 15: Hijacking detection delay (CDF).

they can do little harm to the Internet since the routes can hardly converge in such a short time.

Fig. 12 shows how the FPR decreases with T, under a default $\mu = 0.6$. Since there is no false identification when the ROA records are used, this curve is plotted using the IRR records as the validation set. When $T > 10$, the false rate is less than 0.2%, while if we do not use such a duration threshold, the false rate will increase to 1.1%.

To illustrate the effectiveness of T, we compare the numbers of *eyes* that are polluted by the 220 stable hijackings and the 570 non-stable suspicious alarms in Fig. 13. The distributions show that, non-stable hijackings usually affect a much fewer number of *eyes* than stable hijackings. For example, in 81% cases, less than five *eyes* are affected if the alarm duration $d < T$, while in more than 50% cases, more than five *eyes* are affected if $d > T$. We note that, it rarely happens that more than 40 *eyes* are affected, even for stable hijackings, and we will elaborate on this point later.

Fig. 14 presents how the false positive rate decreases when more *eyes* are used, where the 266 anomalies with ROAs and the 3988 anomalies with IRR records are used as the validation set, respectively. It is clear that using more than 40 *eyes* achieves a fair accurate result, while less than ten *eyes* may bring a few errors.

Fig. 15 reports the cumulative distribution of the detection delay (i.e., from when the first anomalous BGP UPDATE is received to when the first time that F_t exceeds μ), for the 790 suspicious hijacking alarms and 220 stable hijackings, respectively. About 50% suspicious alarms, and 60% stable hijacking have a detection delay of no more than 10 seconds. This is considered as much faster than existing data-plane based hijacking detection methods, which cost at least a few minutes.

We note that, the speed of *Argus* can be further improved, since this delay includes the time used for logging in to the *eyes* and the delay of route propagation, which are in average 6.4 and 4.6 seconds. They can be effectively reduced by putting *Argus* in a network of better quality, and by using more *eyes*. If these latencies are deducted from the detection delay, the real identification delay costs less than 10 seconds in 70% suspicious alarms and 80% stable hijackings, and costs less than one second in 35% suspicious alarms and 50% stable hijackings. In this manner, we claim that *Argus* achieves *realtime* detection.

5.2 Do we miss many real hijackings?

It is very hard to judge whether *Argus* can detect all or at least a large fraction of blackholing prefix hijackings in the Internet, since no one knows the ground truth. We make several efforts to improve its detection capability.

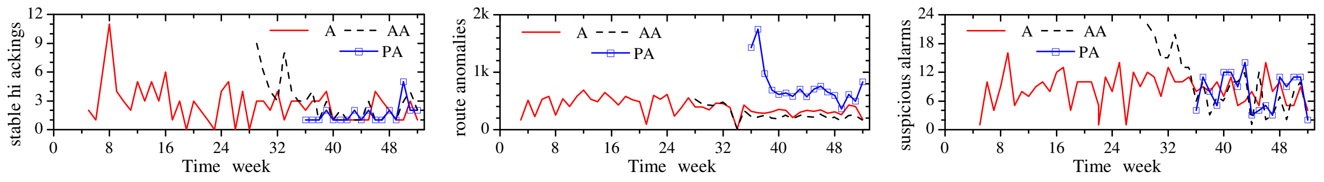
The BGP UPDATEs from BGPmon [3] is considered to cover a large portion of the Internet [33]. For example, the number of peers from which BGPmon collected live data ranged from 110 to 130 in the last year, and these peers were distributed in more than 70 ASes. To handle the relative large data volume [11], we used a server with a six-core Intel Xeon CPU and 16G memory, and experienced no data loss in the whole year. [12] In these UPDATEs, the anomaly monitoring module monitors three types of anomalies, including *OA*, *AA* and *PA*. To the best of our knowledge, we are the first to monitor hijackings which utilize forged adjacency or policy, although they have been theoretically studied before [15]. The successful detection of such hijackings demonstrates the superior capabilities of *Argus*.

Our identification method depends on the reachability to a live host in the target prefix. If no such a host can be found, we just guess one and probe it from our *eyes*. However, if no *eye* can reach it, a possible hijacking may be missed, and we say this is a blindfolded case. We improve the live-IP retrieving module by collecting live IPs from multiple sources, and the number of blindfolded cases decreases from 58% to 19% [13]. We also consider the sub-prefix problem

[11]The peak volume exceeded 10MB/s.

[12]On average, we handled around 10GB BGP data each day, and the peak volume exceeded 20Mbps.

[13]This is just an upper bound for missing hijackings, since



(a) Stable hijackings (b) Route anomalies (c) Suspicious hijacking alarms

Figure 16: The weekly numbers of stable hijackings, route anomalies and suspicious alarms.

as explained in Section 4.3 and illustrated in Fig. 7. In our results, around 10% stable hijackings will not be identified if sub-prefix is ignored, i.e., the live IP belongs to a sub-prefix of the prefix we are interested in.

In the identification module, we now keep a pool of 389 route-servers and looking glasses widely distributed in 41 transit ASes, [14] to sense both the control and the data plane status of the Internet in a pervasive manner. Transit ASes, including default-free ASes, can sense the diverse paths to different prefixes in the Internet. In contrast, other distributed monitoring platforms, such as the Planetlab, often concentrate in stub ASes, such as academic networks, and often use a default route for their traffic. To avoid overloading the eyes, for each anomalous event, we randomly choose $40 \sim 80$ eyes from our server pool, and use them to compute the corresponding fingerprint and reachability. Using more than one eye in the same transit AS can improve accuracy and provide better redundancy, since different eyes may experience different paths to the same prefix [15] and different inconsistencies between their control and data plane.

When computing the fingerprint F_t, a hijacking may be missed due to the division-by-zero problem. However, in our data, the chance that $\overline{C_t}$ equals 1 (i.e., no eye is polluted) is only 3%, and the chance that $\overline{D_t}$ equals 0 (i.e., all eyes are blindfolded) is less than 20%. [16] The case that $\overline{C_t} = 0$ (i.e., all eyes are polluted) or $\overline{D_t} = 1$ (i.e., all eyes can reach the live IP) never occurs. So the probability that a hijacking is missed due to the failure point of F_t is very likely to be small.

It is worth to note that, the number of ASes in which eyes are employed is very important. In the early stage of our system deployment, our eyes covered less than 20 transit ASes in the Internet, and almost half of the anomalous events could not be seen by any of our eyes. This is definitely a major contributor to the false negatives of our system at that time. However, we kept adding more usable eyes, especially in more ASes, and now less than 2% anomalous events may be missed by all the eyes used in identification.

Last, we do not use very large μ and T, so hijackings with unstable fingerprints can also be identified. As has been demonstrated, they do not affect the false positive rate very much.

In conclusion, although there is no direct proof for the false negative possibility of *Argus*, we believe it can detect a large fraction of blackholing hijackings in the Internet, and can liberate network operators from a vast number of alarms. In the rest of this paper, we will make a systematic analysis on our monitoring results.

many such cases are indeed not hijackings, but are caused by route failure, firewall, no active hosts, etc..

[14] A complete list of the eyes we employ can be found at http://argus.csnet1.cs.tsinghua.edu.cn/static/eyes.txt.

[15] The chance that two eyes in the same AS experience different paths is 39% in our experiments.

[16] When our eyes cover more than 20 transit ASes and the improved live IP selection algorithm is used.

5.3 Monitoring Results

The 40K anomalous route events reported by the *AMM* consist of 20K origin anomalies, 6.7K adjacency anomalies and 13.3K policy anomalies. From these events, the *HIM* identified 220 stable hijackings, in which 122 hijackings introduced origin anomalies, while 71 and 27 hijackings introduced adjacency and policy anomalies, respectively. However, since the monitoring of the three kinds of anomalies started at different time, these numbers are not directly comparable. Instead, in Fig. 16(a) and 16(b), we plot the weekly numbers of stable hijackings and route anomalies of each type, starting from May 2011. Notice there is no data for *AA* and *PA* anomalies until November 2011. For comparison, we also plot the weekly numbers of the 790 suspicious hijacking alarms in Fig. 16(c).

As shown in Fig 16(a), the frequencies of different types of stable hijackings are somewhat steady and comparable in the whole year, with an average of three new hijackings per week. The higher values in the starting points may just be caused by accumulation. One notable point is that, adjacency and policy based hijackings do exist in the Internet, and to the best of our knowledge, we are the first to detect and study them.

The trend of anomalies and suspicious hijackings is almost similar to that of stable hijackings. The only difference is that, the number of policy anomalies is far larger than the other types of anomalies, but the number of policy based hijackings is not that large. It is reasonable because the number of neighboring AS triples (i.e., policies) is much larger than the number of origins or neighboring AS pairs [17], while most anomalies are not hijackings.

Fig. 17 plots the CDF of the hijacking duration for all stable hijackings. The duration is computed off-line by examining when the corresponding route anomalies disappear from the routing UPDATEs, representing how long a hijacking will affect the Internet. It covers a very large range, from less than 10 minutes to longer than one week, both of which have a non-negligible fraction and are worth further studies. For example, more than 20% hijackings last less than 10 minutes, and these short (but not transient) hijackings ask for a realtime detector such as *Argus*. On the other hand, those long hijackings indicate that, this security problem is either ignored by or is still unknown to some ISPs, or it is

[17] In the past year, 925K neighboring AS triples, 606K prefix-origin pairs and 108K AS edges were monitored by *Argus*.

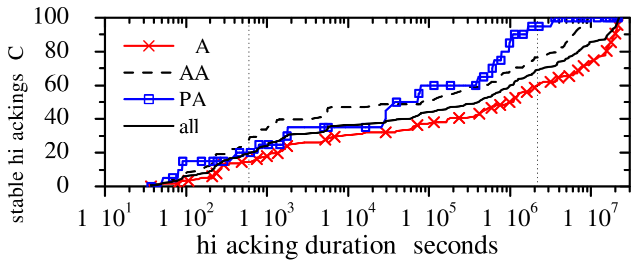


Figure 17: Duration of stable hijackings (CDF).

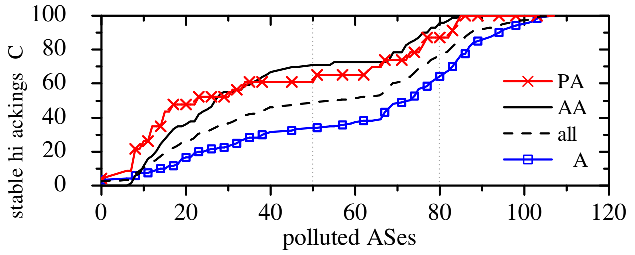


Figure 19: The number of polluted ASes (CDF).

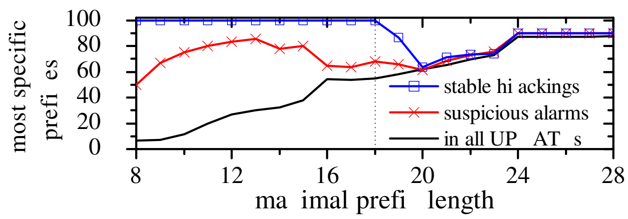


Figure 18: Ratio of most specific prefixes at different prefix length.

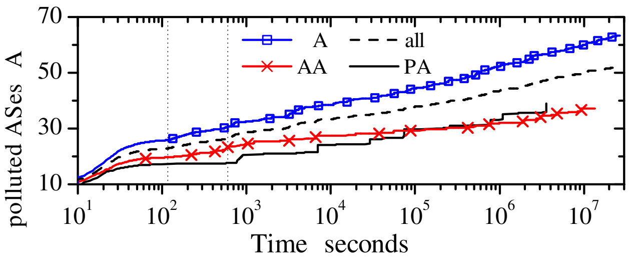


Figure 20: Hijacking pollution speed.

very hard to defend against. It should also be noted that, in order to reduce the workload of the eyes, we only use BGP UPDATEs to determine when a hijacking event ends, so it is possible that some long lasting hijackings are just route migrations which happen to exhibit hijacking characteristics in their early stage.

In total, we detected around 40K anomalous prefixes, of which 1132 appeared in suspicious alarms, while 263 appeared in stable hijackings. We call a prefix *most specific* if it has no sub-prefix announced to the Internet. Since routers prefer most specific prefixes, it is reasonable that hijackings also prefer them so that they can pollute the Internet more effectively, as in the Youtube event [29]. Fig. 18 shows the ratio of such most specific prefixes, in all announced prefixes, hijacked prefixes and suspicious prefixes, respectively, where hijackings show a clear favor for most specific ones. In stable hijackings, about 91% prefixes are most specific, while the ratio in all announced prefixes is 87%. This favor is especially evident for shorter prefixes whose impact is greater. In all the prefixes with a length less than 18, the average percentage of most specific prefixes is only 50%, while the ratio becomes 100% in stable hijackings (the ratio is 70% in all suspicious alarms). On the other hand, sub-prefix hijackings (using a *more specific* prefix than a normal one) contribute around 10% in stable hijackings.

To illustrate how these hijackings affect the Internet, Fig. 19 depicts how many ASes in the Internet a hijacking can successfully pollute (hijack), and Fig. 20 depicts how fast hijackings can pollute them. Since most ASes in the Internet are stub/customer ASes which use default routes to their providers for many prefixes, they cannot be seen in any of the UPDATEs for those prefixes. The numbers in these two figures only cover the transit ASes seen in the UPDATEs for the corresponding (hijacked) prefixes, while the actually affected area should also include their customers. In Fig. 19, more than 20% stable hijackings can pollute at least 80 ASes, while in Fig. 20, on average, more than 20 ASes are polluted within only two minutes. For each hijacking that affects more than 80 transit ASes, we also compute the ratio of the number of polluted ASes (in a certain period) to the number of ASes that can be seen in all UPDATEs for the corresponding prefix. This metric can be regarded as a rough estimator of *what percentage of the Internet is polluted*, and in some cases, more than 50% of the Internet are polluted within 20 seconds, and more than 90% of the Internet are polluted in less than two minutes.

On the other hand, in Fig. 19, origin based hijackings usually can pollute more ASes than adjacency or policy based hijackings. For example, about 70% in the former case can pollute at least 50 transit ASes, while only around 30% of the latter two cases have this capability. Similarly, origin based hijackings often pollute the Internet in a faster way, as shown in Fig. 20. For example, on average, an origin based hijacking can pollute about 26 transit ASes in two minutes, and can pollute more than 30 in ten minutes. Part of the reasons may be that, when forging prefix-origin pairs, most specific prefixes are often used, with short AS-paths.

It is also interesting that a hijacking can hardly pollute all ASes. Due to this reason, the chance that all our *eyes* are polluted is also low, which helps them to get a more accurate detection result.

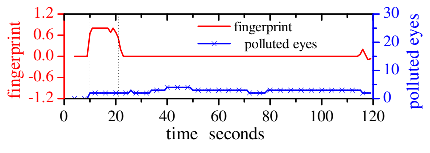
6. CASE STUDIES

In this section, we analyze the characteristics and root causes of some interesting hijackings, as well as some non-hijacking anomalies, in order to help increase the understanding of prefix hijackings in the Internet.

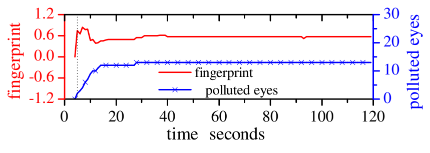
Table 2 summarizes the hijacking cases discussed in this section, including five cases with origin anomalies, three with adjacency anomalies and two with policy anomalies. Most of these hijackings have been confirmed by the corresponding network operators, while for other cases, we will present enough evidence.

Table 2: Case study: ten prefix hijackings detected in the past year.

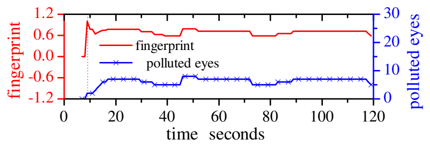
	Time	Prefix	Normal Origin	Anomalous Origin	Delay
OA_1	2011-12-27	166.111.32.0/24, ...	AS4538	AS23910	10
OA_2	2012-03-20	193.105.17.0/24	AS50407	AS15763	5
OA_3	2012-04-04	91.217.242.0/24	AS197279	AS48559	9
OA_4	2011-06-23	76.72.238.0/24, ...	AS701	AS27005	6
OA_5	2012-03-22	12.231.155.0/24	AS7018 (12.128.0.0/9)	AS13490	7
	Time	Prefix	AS-path	Anomalous AS Pair	Delay
AA_1	2012-04-23	210.1.38.0/24	3043 174 38082 38794 24465	38794 24465	12
AA_2	2012-03-31	184.164.255.0/24	4739 6939 2381 47065 19782 47065	47065 19782	4
AA_3	2011-12-19	205.153.112.0/22	3303 174 17368 26263	17368 26263	3
	Time	Prefix	AS-path	Anomalous AS Triple	Delay
PA_1	2012-04-19	77.223.240.0/22	4739 24709 25388 21021 12741 47728	21021 12741 47728	9
PA_2	2012-04-16	195.10.205.0/24	3043 174 20764 31484 3267 3216 35813	20764 31484 3267	5



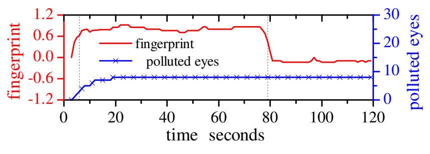
(a) OA_1



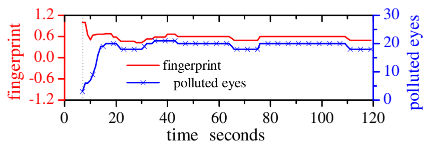
(b) OA_2



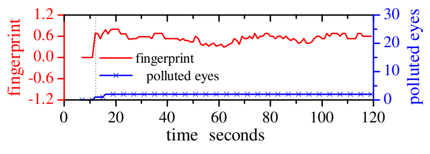
(c) OA_3



(d) OA_4



(e) OA_5



(f) AA_1

Figure 21: Case studies of hijackings: fingerprint and the number of polluted eyes.

In Fig. 21, for each hijacking with origin anomaly, we plot the corresponding fingerprint and the numbers of polluted *eyes* in each second, to demonstrate the timely and accurate reaction of *Argus*. We also indicate the time delay when *Argus* begins or stops to report hijacking alarms by a dotted line, if it exists. For other cases, since the curves are very similar, we just present one of them, and omit the others due to space limitations.

6.1 Hijackings with Origin Anomalies

Missing route filters: OA_1 was caused by a route filter failure. In this case, AS23910 learned two routes to 166.111.32.0/24 and 166.111.111.0/24, which were owned by AS4538. The Network operators confirmed that, since AS23910 accidentally removed its corresponding filters for a short period, these two routes were incorrectly redistributed into BGP, and then announced to the Internet, with AS23910 as the origin. This corresponds to the period when $F_t > 0.6$ in Fig. 21(a).

Network maintenance misplay: in OA_2, AS50407 was a customer of the victim AS15763. The network operators of AS50407 said they made a mistake when they were carrying out network maintenance, and hijacked the prefix 193.105.17.0/24 for 12 minutes. In Fig. 21(b), after the hijacking was detected, the fingerprint kept above 0.6 for the remaining window.

Premature migration attempt: OA_3 was caused by a premature route migration that was not well planned. In this case, AS197279 was to migrate 91.217.242.0/24 to another AS AS48559 "in the coming week or so", as said in their reply to our alarm. However, before the planned time, AS48559 announced itself as the owner of this prefix, which AS197279 was still keeping announcing at that time. Since live hosts have not moved into AS48559, polluted *eyes* could not reach them, and F_t increased above 0.6. The hijacking lasted about 17 minutes and *Argus* detected it within 9 seconds, as shown in Fig. 21(c). OA_4 is a similar case, where AS27005 announced the same prefix as AS701 did. There is a little difference from OA_3: live hosts in AS27005 became reachable after 70 seconds (still in the detection window), while live hosts in AS701 also kept reachable until several hours later, so F_t later decreased to 0 in Fig. 21(d).

Sub-prefix hijacking: OA_5 was confirmed by AS7018, which was the owner of a prefix 12.128.0.0/9. One of its sub-prefixes, 12.231.155.0/24, was incorrectly announced by AS13490 at March 22, 2012, and the hijacking lasted about 16 minutes. As shown in Fig. 21(e), this sub-prefix hijacking polluted around 20 *eyes* in 20 seconds, which is faster than the other cases. Interestingly, we also noticed that this sub-prefix was officially assigned to and announced by AS54357 nine days later, and *Argus* successfully detected that event and classified it as a route migration (i.e., the fingerprint was close to -1.0).

6.2 Hijackings with Adjacency Anomalies

Mis-configuration in traffic engineering: in the case

```
import: from AS12741 action pref=150; accept AS12741
export: to AS12741 announce AS21021
```

(a) PA_1: Routing policy of AS21021.

```
remarks: --- Uplinks ---
import:  from AS3267 action pref=85; accept ANY
export:  to AS3267 announce AS31484 AND AS196931
import:  from AS20764 action pref=85; accept ANY
export:  to AS20764 announce AS31484 AND AS196931
```

(b) PA_2: Routing policy of AS31484.

Figure 23: IRR records of PA_1 and PA_2.

of AA_1, AS24465 was connecting to two providers, and was using AS-Path prepending (ASPP) to achieve inbound Traffic Engineering. It normally announced its prefix 210.1.38.0/24 to one provider AS4750, but prepended itself for 4 times in the AS-path announced to the other provider AS7693. The hijacking happened when AS24465 connected to a new provider AS38794 and announced the prefix to it. However traffic could not go through the new one, and *Argus* immediately notified the operators of AS38794. When they replied 50 minutes later, we found the problem was fixed, and ASPP was also used in the new route.

AS-path poisoning experiment: AA_2 was caused by a rerouting experiment launched by Georgia Tech and University of Washington [19], using AS47065. They announced a looped path $\langle 47065, x, 47065\rangle$ for prefix 184.164.255.0/24, so that AS x would not accept this route later, and the corresponding traffic would not go through x. They repeated the experiments periodically for many times, and *Argus* continuously reported adjacency anomalies. If things went well, we should have identified no hijackings, since the destination should be reachable even when the path changed. [18] However, in an experiment when x was set to 19782, the target prefix became unreachable on our *eyes* that were using the new path, but was still reachable on the *eyes* that were using the normal path without loop, and *Argus* reported hijackings accordingly.

Short-lived neighbor: AA_3 was caused by AS17368 pretending to be a neighbor of AS26263, which was the owner of the victim prefix 205.153.112.0/22. The hijacking polluted 4 *eyes*, and lasted 14 minutes. Later, this adjacency disappeared and was never announced again.

6.3 Hijackings with Policy Anomalies

Import policy violation: PA_1 was caused by an import filter failure of AS21021. As described by the IRR records of AS21021 (Fig. 23(a)), it will import a route from AS12741 if and only if the corresponding AS-path is originated by AS12741. However, in this case, it accepted a route originated by AS47728, and the anomalous triple $\langle 21021, 12741, 47728\rangle$ was detected by *Argus*. When the traffic was hijacked by this route, *Argus* reported this case.

Export policy violation: PA_2 was caused by an export filter failure of AS31484. According to the IRR records of AS31484, it has two providers, AS20764 and AS3267. The route filter in the IRR, as shown in Fig. 23(b), means AS31484 will accept ANY route announced by the two providers, but will only announce routes originated from AS31484 or AS196931 (its customer) to them. In this case, AS31484 incorrectly announced routes learned from its provider AS3267 to the other provider AS20764, so *Argus* detected the anomalous triple $\langle 20764, 31484, 3267\rangle$. Later, it successfully identified the hijacking.

[18]Actually *Argus* reported route migrations, as expected.

6.4 Non-hijacking Anomalies

As we have mentioned before, *Argus* can also classify non-hijacking anomalies into different types, including route migration, traffic engineering, etc., as shown in Fig. 8. We analyze three cases to demonstrate this capability. Fig. 22 explains how they are different from hijackings, where for each case, we plot the corresponding fingerprint F_t, the reachability $\overline{D_t}$ to the anomalous prefix, and the number of polluted eyes.

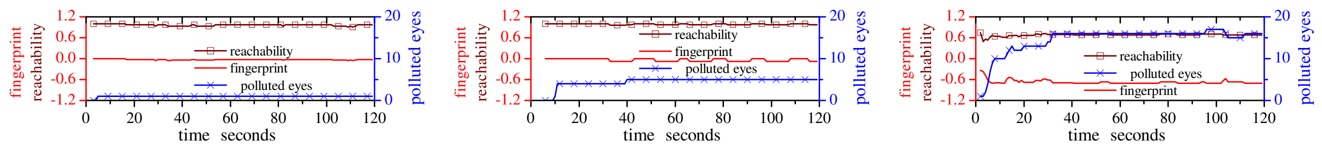
TE using BGP anycast: Root DNS servers usually use BGP anycast to balance and optimize DNS requests. In this case, the anomalous prefix 193.0.16.0/24 was used by the root DNS server k, and was operated by RIPE NCC. *Argus* detected that this prefix was suddenly announced by a new origin AS197000. Since all its *eyes*, either polluted or not, could reach the target prefix (i.e., $D_t = 1$), while the fingerprint F_t was close to 0, it was identified as traffic engineering. Fig. 22(a) shows the corresponding F_t and D_t.

TE with backup links: In this case, AS12476 announced an AS-path with ASPP to AS6453, which was its provider but had never appeared as the its neighbor before. The new path exposed this backup link accidentally. Similarly to the first case, F_t was always 0 while D_t was always 1, as shown in Fig. 22(b).

Route migration: AS12653 and AS7700 were operated by the same ISP (KB Impuls Hellas, Greece). During the route migration, AS12653 stopped announcing routes, while AS7700 started to take the place. In the route convergence process, almost all *eyes* that had learned the new routes could reach the target prefix, while *eyes* using the old routes could not. This led to a negative F_t (i.e., less than -0.6), as is in Fig. 22(c).

7. DISCUSSION

To show the advantages of *Argus*, we compare it with some representative hijacking detection systems proposed earlier. A control plane based system such as Cyclops [12] will report the 20K *OA* anomalies, while *Argus* can identify 122 stable hijackings out of them. This can lower the burden to handle hijackings by two orders of magnitude. Data plane methods [34, 36] have to continuously measure the Internet for each prefix they care, while *Argus* only probe the Internet for two minutes when an anomaly appears. The measurement overhead in one year can be roughly computed as (40K anomalies) $\times$ (2 minutes) $\times$ (80 eyes) for *Argus*, and as (400K prefixes) $\times$ (1 year) $\times$ (1 eye) for the system in [36], resulting in a difference of four orders of magnitude. The overhead of iSPY [34] is less since it only monitors the prefixes owned by the organization who deploys the system, but if the entire Internet is to be protected, the overhead will be comparable to that in [36]. Moreover, these data plane methods cannot detect sub-prefix hijackings (around 10% in our data), unless every possible sub-prefix is monitored. They also cannot distinguish BGP anycast and route migration from hijackings since they are based on data path changes. The method utilizing both data and control plane information [16] relies on host-specific fingerprints, such as



(a) BGP anycast (b) Backup link (c) Route migration

Figure 22: Case studies of non-hijacking anomalies: fingerprint, reachability and the number of polluted eyes.

OS property, IP ID, timestamp, etc., thus may also fail when BGP anycast or route migration happens. Besides, its detection delay is at least a few minutes while *Argus* needs only a few seconds.

Argus also has some limitations. It relies on ping to detect blackholing behavior, so malicious attackers can evade the system by actively responding to ping. However, we believe our earlier measurement results are still reasonable, and we will integrate other fingerprinting methods into *Argus*. A carefully crafted AS-path based on existing path segments may also evade our detection although building such a path is very difficult, and advanced detection and protection methods are still needed. Another weakness may lie in our metric F_t, which can be specifically attacked by carefully choosing the time/prefix to hijack, and how to devise more robust hijacking fingerprints is worth further study. Many components of *Argus* can be continuously improved, e.g., by adding eyes in different ASes, and refreshing live hosts [1]. Finally, much more can be done on the classification of route anomalies, and the characterization of realworld hijackings.

8. CONCLUSIONS

In this paper, we present *Argus*, a system for detecting IP prefix hijackings. Compared with existing methods, *Argus* has many advantages, and our one year practice in monitoring the Internet shows it can accurately distinguish prefix hijackings from a vast number of anomalies in a very short time, and can further deduce the possible root causes. It is very easy to deploy, and we are now providing it as a public service, which has already successfully helped many network operators to diagnose their network operations. In the future, we plan to incorporate more diagnosis information and make it more intelligent.

9. ACKNOWLEDGEMENT

We are grateful to the anonymous IMC reviewers and our shepherd Mark Crovella for their insightful comments. We also want to express our gratitude towards the network operators we have contacted for their generous cooperation and feedback. This work is supported by the National Key Technology R&D Program of China (Grant No. 2008BAH37B03), and the National Basic Research Program of China (973 Program, Grant No. 2009CB320502).

10. REFERENCES

[1] Ant censuses of the internet address space. `http://www.isi.edu/ant/traces/index.html`.

[2] Archipelago Measurement Infrastructure. `http://www.caida.org/projects/ark/`.

[3] The BGPmon project. `http://bgpmon.netsec.colostate.edu`.

[4] BGPmon.net. `http://bgpmon.net/`.

[5] Charter of the IETF Secure Inter-Domain Routing Working Group. `http://tools.ietf.org/wg/sidr/charters`.

[6] DNS records collected by Hurricane Electric. `http://bgp.he.net/net/166.111.0.0/16#_dns`.

[7] Irr - internet routing registry. `http://www.irr.net/`.

[8] Ripe myasn system. `http://www.ris.ripe.net/myasn.html`.

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

[10] S. M. Bellovin, R. Bush, and D. Ward. Security requirements for bgp path validation. `http://tools.ietf.org/html/draft-ymbk-bgpsec-reqs-02`, 2011.

[11] M. Caesar, L. Subramanian, and R. H. Katz. Towards root cause analysis of internet routing dynamics. In *Berkeley EECS Annual Research Symposium*, 2004.

[12] Y.-J. Chi, R. Oliveira, and L. Zhang. Cyclops: The AS-level connectivity observatory. *ACM SIGCOMM Computer Communication Review*, pages 7–16, 2008.

[13] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek. Measuring the effects of internet path faults on reactive routing. In *SIGMETRICS*, 2003.

[14] A. Feldmann, O. Maennel, Z. M. Mao, A. W. Berger, and B. M. Maggs. Locating internet routing instabilities. In *SIGCOMM*, 2004.

[15] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford. How secure are secure interdomain routing protocols? In *SIGCOMM*, 2010.

[16] X. Hu and Z. M. Mao. Accurate real-time identification of ip prefix hijacking. In *IEEE Symposium on Security and Privacy*, pages 3–17, 2007.

[17] G. Huston and G. Michaelson. RFC 6483: Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs). `http://tools.ietf.org/html/rfc6483`, 2012.

[18] J. Karlin, S. Forrest, and J. Rexford. Pretty good BGP: Improving BGP by cautiously adopting routes. In *ICNP*, pages 290–299, 2006.

[19] E. Katz-Bassett. Announcement of university of washington routing study. `http://mailman.nanog.org/pipermail/nanog/2011-August/039337.html`.

[20] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. E. Anderson. Studying black holes in the internet with hubble. In *NSDI*, 2008.

[21] S. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18:103–116, 2000.

[22] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. PHAS: A prefix hijack alert system. In *USENIX*, 2006.

[23] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, and A. Krishnamurthy. iPlane: An information plane for distributed services. In *OSDI*, pages 367–380, 2006.

[24] R. Mahajan, D. Wetherall, and T. Anderson. Understanding bgp misconfiguration. In *SIGCOMM*, pages 3–16, 2002.

[25] S. Murphy. Rfc 4272: Bgp security vulnerabilities analysis. `http://tools.ietf.org/html/rfc4272`, 2006.

[26] R. Oliveira, B. Zhang, D. Pei, R. Izhak-Ratzin, and L. Zhang. Quantifying path exploration in the Internet. In *Proc. of the 6th ACM SIGCOMM Internet Measurement Conference (IMC)*, Rio de Janeriro, Brazil, 2006.

[27] Y. Rekhter, T. Li, and S. Hares. RFC 4271: Border gateway protocol 4. `http://tools.ietf.org/html/rfc4271`, 2006.

[28] Renesys. China's 18-minute mystery. `http://www.renesys.com/blog/2010/11/chinas-18-minute-mystery.shtml`, 2010.

[29] RIPE. Youtube hijacking: A ripe ncc ris case study. `http://www.ripe.net/news/study-youtube-hijacking.html`, 2008.

[30] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and whisper: Security mechanisms for BGP. In *NSDI*, pages 127–140, 2004.

[31] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush. A measurement study on the impact of routing events on end-to-end internet path performance. In *SIGCOMM*, 2006.

[32] Y. Xiang, Z. Wang, X. Yin, and J. Wu. Argus: An accurate and agile system to detecting ip prefix hijacking. In *Workshop on Trust and Security in the Future Internet*, 2011.

[33] B. Zhang, R. Liu, D. Massey, and L. Zhang. Collecting the internet as-level topology. *SIGCOMM Comput. Commun. Rev.*, 35(1):53–61, 2005.

[34] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush. iSPY: Detecting ip prefix hijacking on my own. In *SIGCOMM*, pages 327–338, 2008.

[35] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. An analysis of BGP multiple origin as (MOAS) conflicts. In *1st ACM SIGCOMM Workshop on Internet Measurement*, 2001.

[36] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis. A light-weight distributed scheme for detecting ip prefix hijacks in real-time. In *SIGCOMM*, pages 324–334, 2007.