A Light-Weight Distributed Scheme for Detecting IP Prefix Hijacks in Real-Time

Changxi Zheng[‡], Lusheng Ji[§], Dan Pei[§], Jia Wang[§], and Paul Francis[‡]
[§]AT&T Labs - Research, Florham Park, NJ
[‡]Dept. of Computer Science, Cornell University, Ithaca, NY
[§]{lji, peidan, jiawang}@research.att.com, [‡]{cxzheng, francis}@cs.cornell.edu

ABSTRACT

As more and more Internet IP prefix hijacking incidents are being reported, the value of hijacking detection services has become evident. Most of the current hijacking detection approaches monitor IP prefixes on the control plane and detect inconsistencies in route advertisements and route qualities. We propose a different approach that utilizes information collected mostly from the data plane. Our method is motivated by two key observations: when a prefix is not hijacked, 1) the hop count of the path from a source to this prefix is generally stable; and 2) the path from a source to this prefix is almost always a super-path of the path from the same source to a reference point along the previous path, as long as the reference point is topologically close to the prefix. By carefully selecting multiple vantage points and monitoring from these vantage points for any departure from these two observations, our method is able to detect prefix hijacking with high accuracy in a light-weight, distributed, and real-time fashion. Through simulations constructed based on real Internet measurement traces, we demonstrate that our scheme is accurate with both false positive and false negative ratios below 0.5%.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing Protocols*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Monitoring*

General Terms

Measurement, Security

Keywords

Routing, BGP, Hijacking, Interception, Detection

1. INTRODUCTION

Hijacking IP address prefix is a known threat that disrupts the Internet routing infrastructure. The Border Gateway Protocol (BGP), which is the de facto inter-domain routing protocol used on today's Internet, has no mechanism for authenticating routing announcements. Thus, misbehaved routers can arbitrarily advertise routes for prefixes and/or fabricate Autonomous System (AS) paths associated with the prefixes. Such false announcements can quickly spread to a large number of BGP routers across multiple ASes and pollute their routing tables. As a result, the victim prefix network will experience performance degradation and a serious security breach. For instance, packets addressed to a hijacked prefix can be dropped by intermediate routers; blackholed by the attacker; or forced to take a longer detour to reach the true destination. An attacker may also impersonate the victim prefix to communicate with other parties; send spam emails or launch DDoS attacks from the hijacked prefix; intercept communications; or conduct Man-in-the-middle attacks.

Most existing proposals [32, 3, 11, 25, 16, 6, 31, 9, 34, 36, 15, 28] require changes to router software, router configurations, network operations, and some require public key infrastructures. These solutions are therefore not easily deployable. When partially deployed, these approaches usually offer limited security [7]. Other proposals [18, 23, 19, 30] do only passive monitoring and thus are deployable. But they can suffer from high false positive ratio because hijackings can be indistinguishable from legitimate routing changes, or because the routing registry or allocation data used in some of these approaches can be outdated and inaccurate. Recently, utilizing data plane information together with control plane information in hijacking detection is gaining attention [32, 10, 4]. Despite the differences among existing approaches, in order to provide timely hijack detection, they all require privileged access to live BGP feeds.

In this paper, we present a light-weight and deployable distributed scheme for detecting prefix hijacking in real-time. Besides its effectiveness, what truly separates our approach from others is that our approach utilizes real-time data collected only from the data plane. BGP live feed dependent schemes require the availability of such feeds. As [24] shows, however among hundreds of Planetlab [27] nodes, only a very small number of nodes have live BGP update feeds. Not requiring privileged access to live BGP advertisement data makes our approach easy to deploy and appealing to prospective prefix hijack monitoring service providers. Independence from the BGP control plane also emancipates the hijacking detection mechanism from the updating cycles of BGP data collection points, and results in a potentially more timely detection alarm. The distributed nature of the scheme enables the monitor nodes to work collaboratively, improving system robustness and spreading out monitoring traffic overhead over the Internet.

There are three key steps in our approach: (i) for each target prefix we identify a number of vantage points (sites) from a list of

SIGCOMM'07, August 27–31, 2007, Kyoto, Japan.

candidates that are most suitable for conducting monitoring operations and use them as "monitors" for hijack detection; (ii) we continuously monitor the network location of a prefix from multiple vantage points by measuring the network distance from each vantage point to the target prefix and detect significant changes in the network distance from any vantage point to the target prefix (which indicates changes in the target prefix's network location); (iii) if network location change is indeed detected, we further verify that the location change is not caused by legitimate route changes on the Internet. In this paper, we use *hop count* as the measure of network distance between two hosts on the Internet. Alternative choices are discussed in Section 6.

Our contributions are three-fold. First, we propose a new lightweight distributed framework for detecting prefix hijack attacks. To the best of our knowledge, this is the first distributed hijack detection approach that relies only on real-time measurements taken from the data plane[1]. Second, we present a novel three-step scheme for detecting prefix hijacking. Third, we conduct analysis, experiments, and large scale simulations that are derived from real Internet measurements to evaluate our approach. We show that the proposed approach can effectively detect IP prefix hijacks in real-time with very low false negative and false positive ratios.

This paper is organized as follows. Section 2 provides BGP background information and defines the prefix hijacking problem. Section 3 overviews our framework for detecting IP prefix hijacks based on network distance measurements. We describe detailed monitoring and detection methodologies in Section 4. Section 5 evaluates our approach using analysis and large scale simulations derived directly from Internet measurements. We discuss remaining issues and related work in Sections 6 and 7. Finally, Section 8 concludes the paper.

2. PREFIX HIJACKING

The Internet is composed of tens of thousands of Autonomous Systems (ASes) that are under separate administrative domains. The Border Gateway Protocol (BGP) is the standard inter-domain routing protocol. BGP is a path vector protocol in that a BGP update includes a list of ASes which describes the path to a destination address prefix. A destination prefix is usually announced either by the prefix owner itself if it runs BGP and has an AS number; or by its upstream provider AS(es).

Because there is no authentication mechanism used in BGP, a mis-behaving router can announce routes to any destination prefix on the Internet and even manipulate route attributes in the routing updates it sends to neighboring routers. Taking advantage of this weakness has become the fundamental mechanism for constructing prefix hijack attacks.

Prefix hijacking happens in various forms in the control plane. The attacker can announce himself as the origin AS for the target prefix, it can announce a more specific IP prefix with a longer prefix length than the target prefix, or it can announce a very attractive AS path that may not exist in reality. Upon receiving these fabricated advertisements, other BGP routers may be fooled into thinking that a better or more specific route has become available towards the target prefix and start forwarding future traffic along the false path. As a result of the prefix hijacking, part (if not all) of the traffic addressed to the target prefix will be forwarded to the attacker instead of the target prefix.

[1]Hop count has been used before to detect spoofed address in a DDoS case [13], but not as part of a distributed system that can detect other cases of route manipulation as we discuss later.

Based on how the attacker deals with the hijacked traffic, we classify prefix hijacks into the following three categories:

- **Blackholing:** the attacker simply drops the attracted packets.
- **Imposture:** the attacker responds to senders of the hijacked traffic, mimicking the true destination's (the target prefix's) behavior.
- **Interception:** the attacker forwards the hijacked traffic to the target prefix after eavesdropping/recording the information in the packets.

In the rest of the paper, we use the term "hijack" to refer to all of these behaviors. Furthermore, we use the terms *attacker* and *hijacker* interchangeably.

While the conventional view of the damage of prefix hijacking has been focused on blackholing, the other two types of hijacking are equally important, if not more damaging. The most serious consequence of blackholing is the loss of reachability *only*, and is typically not accompanied by other dangers such as breach of confidentiality. Also the detection of blackholing is trivial. Communication peers of a target prefix can perceive blackholing attacks if they do not receive any response from the target prefix for a period of time, especially when such loss of communication does not occur to all the peers of the target prefix.

On the other hand, imposture and interception are more challenging to detect than blackholing. From any peer's point of view, the target prefix is still "reachable". Even from the BGP control plane it is often difficult to identify these two kinds of hijacks. For instance, as shown in [35] and [21], one MOAS (Multiple Origin ASes) prefix can be legitimately announced by multiple origin ASes. It is not easy to distinguish a hijack from a legitimate routing change of an MOAS prefix because they can both appear as a change of origin AS. On the other hand, although [4] shows that interception is easy to accomplish, there have been no reported imposture or interception attacks on the Internet yet. However, this does not mean that these attacks never happened nor that there are no ongoing attacks. It could be due to the difficulty to detect such attacks.

Because they are hard to detect, the interception and imposture hijacks can last a long period of time before being detected and reported to authorities and the target prefix owner. Furthermore, the hijacker can potentially cause more damage by conducting further attacks such as those similar to *online phishing* (with correct address as opposed to the normal phishing), spam emails [29] or DDoS attack. Or, the hijacker can intercept the traffic to retrieve important information for malicious purposes [17, 26, 4]. Given the threats of interception and imposture hijacks and the much greater challenges of detecting them than detecting blackholing, in this paper we focus on presenting a solution for the detection of both imposture and interception hijacks.

It should be mentioned that the detection scheme proposed in this paper is not limited to detecting imposture and interception hijacks, nor is it restricted in detecting *IP prefix* hijacking. It inherently can be extended to detect some other types of hijacking or misconfigurations, such as hijacking by faking the DNS response.

3. FRAMEWORK

In this section, we present an overview of our prefix hijack detection scheme.

3.1 Monitoring Network Location

One of the key observations behind our scheme is that the network location of a prefix generally remains unchanged over time.

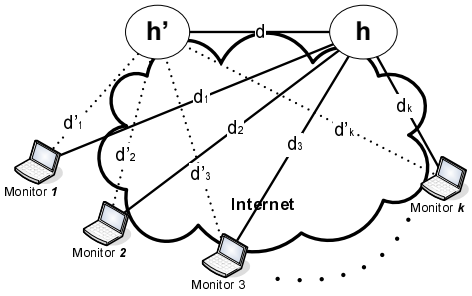


Figure 1: Monitoring Network Location

This is mainly due to the fact that IP prefix assignment on the Internet is usually on a very long term basis.[2] Once an IP prefix is assigned, it can be announced either by the prefix owner if it runs BGP, or by its immediate upstream service provider AS(es). In either case, the network location of the prefix viewed from external vantage points should belong to the same topological region. Note that depending on the context, we use the term "prefix" referring either to a set of IP addresses or network devices which are named by these IP addresses.

Due to Internet's vast size, network topological dynamics such as link status changes generally affect only a fraction of the overall Internet topology dramatically. Because routes are constructed based on the actual network topology of the Internet, network distance measurements obtained from the data plane, which are indeed network distances in the routing topology configured by BGP and other routing protocols, generally reflect the network distances in the actual network topology. Hence, the network distance measured from a given vantage point to a destination network is likely to remain the same over time. Previous work such as [33] have also confirmed this observation.

However, if a prefix is hijacked, the association between the routes to the prefix and the underlying network topology disappears. Thus, the network distances measured from certain vantage points to the target prefix would likely exhibit *significant* differences from what these distances were prior to the hijacking. In imposture scenarios, the network distances from certain vantage points to the target prefix may appear to be either shorter or longer, depending on the network locations of the vantage points and the network locations of the attackers. In interception scenarios, it is more likely that the network distances from certain vantage points to a target prefix appear to be longer because the paths towards the target prefix now take a detour going through the attacker's AS. Although such location change becomes small if the hijacker's location is very close to the victim's location, statistically the opposite case is more likely to happen due to the size of the Internet. Therefore, prefix hijacking can be effectively detected if significant changes in the network distances from certain vantage points to the target prefix are observed.

Figure 1 illustrates the idea behind our network location monitoring framework. There are k vantage points for monitoring a target prefix P (not shown in figure). Suppose that the prefix P is previously announced by h. The network distance between a vantage point i and h is denoted as d_i. Now h' also announces P. The distance between the vantage point i and h' is denoted as d'_i. Then the distance d between h and h' is bounded by $d \geq max_{i=1}^{k}|d_i - d'_i|$. If h and h' are co-located (e.g., h' is a provider or a customer of h, or h and h' are both providers of the owner of of P), d would be small. Therefore, $d_i \approx d'_i$ $(i = 1, \ldots, k)$.

However, in the scenarios of imposture where h' hijacks P, with high probability h and h' are not co-located, $\exists i \in 1, \ldots, k$, s.t. $|d_i - d'_i| \geq \delta$, where δ can be considered as the detection threshold. Therefore, the value of $D = max_{i=1}^{k}|d_i - d'_i|$ is an indication of the likelihood of prefix P being hijacked. The larger D is, the more likely P is hijacked. For the cases of interception, the distance from a vantage point i to P would be $d'_i + d$. Such attack scenarios can be detected with high probability as long as $\exists i \in 1, \ldots, k$, s.t. $|d_i - d'_i + d| \geq \delta$.

It is worth noting that typically a prefix hijacker can only hijack traffic from a portion of the Internet to the target prefix. This is because some ASes will prefer the true route from the target prefix to the one from hijacker due to shorter AS path or policy reasons [20, 4]. Consequently if our vantage point happens to be located in such regions that are not affected by the hijacking, it will not detect the hijacking either. Therefore, we must establish multiple topologically diverse vantage points for effectively monitoring a target prefix. Using multiple vantage points also increases the difficulty for an attacker to conduct any countermeasures because now it has to cheat all these vantage points. In addition, multiple vantage points may also help in reducing false positive ratios. From now on, we use the term *monitor* to refer to a vantage point that keeps probing the network location of a target prefix.

3.2 Detecting Path Disagreement

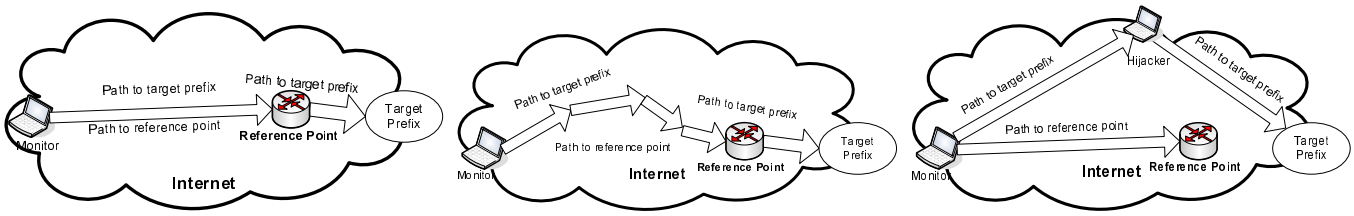
Our first detection mechanism focuses on significant network location changes. But the problem is that not all significant network location changes are the results of prefix hijacking. Internet topology changes regularly due to reasons such as link status changes and policy-based route changes. In contrast to prefix hijacking induced location changes, these changes will be referred to as *legitimate* in this paper. As we have mentioned before, most of these legitimate changes are not expected to result in dramatic widespread location changes and these "minor" legitimate changes are expected to be filtered out by the aforementioned location based hijack detection algorithm.

However, routes in the Internet are not always configured based on network topology due to special routing policies, in which case the inherent stability of the network topology of Internet does not translate to stability in routing topology. Also in rare occasions link status changes may actually alter the Internet topology dramatically. These kinds of routing topology changes can be significant and may be mistakenly identified as hijacking by the location based mechanism. Such false alarms require additional efforts to groom and filter. Correcting false positive detections is often a difficult task as it may require detailed configuration information that the network operators are unwilling to share.

All of these motivate us to develop the second detection technique, *path disagreement detection*. It is intended to be used in conjunction with the network location monitoring to produce highly accurate detection results. In this technique we focus on one particular difference between legitimate route changes and prefix hijacking attack induced route changes: the portion of the network being affected. A prefix hijacking attack usually only targets a specific network prefix while legitimate route changes usually affect larger number of prefixes.

For each monitor, we need to identify one *reference point* along the path from the monitor to the target prefix. This reference point needs to be topologically very close to the target prefix but still has an IP address outside of the target IP prefix. Because of the topological closeness, from the same monitor, the route to the ref-

[2]It is possible that a prefix is reassigned to name a different set of network devices which can appear as changes in the network location. However, this case is extremely rare and is out of scope of this paper.



(a) Original (b) Legitimate route change (c) Prefix hijacking

Figure 2: Path Disagreement

erence point of a target prefix is very likely (if not always) to be identical to, or more precisely a sub-path of, the route to the target prefix. Also for the same reason, chances are that legitimate route changes in the Internet would likely affect the target prefix and its reference point equally. We will provide measurement data to support this conjecture in Section 5. On the other hand, because the reference point has an IP address outside of the target prefix, any prefix hijacking attacks targeting the prefix will not affect the reference point. In other words, we will detect disagreement between the path from a monitor to a target prefix and the path from the same monitor to the corresponding reference point of the target prefix. Significant disagreement signals prefix hijacking attack at the target prefix.

Figure 2 illustrates the concept of path disagreement-based detection. Figure 2(a) shows a path from a monitor to a target prefix. A reference point that fits the criteria described previously is identified along this path. When the target prefix is not under attack, the path from the monitor to the reference point is a sub-path of the path from the same monitor to the target prefix. Figure 2(b) shows what may happen when legitimate route changes occur. No matter how the paths may twist and turn, as long as the reference point is topologically close to the target prefix, chances are that the path from the monitor to reference point is still a sub-path of the path to the target prefix. In contrast, Figure 2(c) shows that these two paths are now very different after the target prefix is hijacked, or more precisely in this example, intercepted. The path to the target prefix may take a detour through a hijacker-controlled site while the path to the reference point remains as before. In other words, how much "disagreement" there is between these two paths separates path changes caused by legitimate route changes from path changes caused by prefix hijacking attacks. It is also worth noting that not only path disagreement, but changes in how much paths disagree with each other can also be used for such detection.

We now discuss how a reference point can be identified. In a commonly seen configuration, the external interface (facing ISP AS) of the target prefix's network access router is assigned an IP address provided by the ISP, which is outside of the target prefix. In this case both this external interface or the customer-facing interface of the ISP's edge router can be ideal candidates for becoming a reference point. This reference point location is either on a router managed by the target prefix's administrator, or the next hop for the prefix's outgoing routes. The administrator obviously has this piece of configuration information and the administrator can provide such information at the time of signing up for the prefix hijacking monitoring service.

For configurations other than the one described in the previous paragraph, or if the identified candidate reference point is not willing to participate in hijack detection operations, *i.e.*, it does not respond to ICMP requests, we would need to discover a candidate reference point location by retreating along the route from the monitor to the target prefix backwards hop by hop to the first point that could actually assist in detection operations. A modified "traceroute" program can easily discover such a location for being used as a reference point. The discovery can be done either from the monitor side or from the target prefix network side.

If a reference point is not immediately connected to the target prefix, then a portion of the Internet actually lies between the target prefix and the reference point. If a hijacker is located within this portion of the Internet, chances are that the monitor will not see any path disagreement because the path to the reference point is still a sub-path of the path to the target prefix. This is clearly undesirable. This is why the reference point should be as close to the target prefix as possible.

Obviously reference points need to be established on a per-monitor basis. The reason is simple: the reference point selected for one monitor may not be on the path from a different monitor to the same prefix. On the other hand, multi-homed target prefixes also present additional challenges. In this case, a legitimate route change may actually cause the monitor's probe traffic to reach the target prefix via a difference access router. The new path to the target prefix may be quite different from the path to its current reference point. Instead of classifying this path change as a prefix hijacking attack induced as in single-homed customer case, additional steps are necessary. These additional steps require that reference points for a target prefix be established not only on a per-monitor basis but also on a per-access router basis. In other words, a monitor has to know all of its reference points for the target prefix, with each reference point corresponds to an access router of the target prefix. Once an instance of path disagreement is detect, the monitor needs to compare the path to the target prefix with paths to all of the monitor's reference points. If the path to the target prefix differs from *all* of these paths to reference points, the path change is likely caused by a successful prefix hijacking attack.

3.3 Hijack Detection Scheme Overview

With the two mechanisms already explained, we now outline our detection scheme. Our scheme consists of three key steps. First, for each target prefix, we select a number of monitors from a set of candidate monitors. Second, each monitor periodically measures the network distance to each target prefix and detects significant changes in hop count distance measurement. Third, if a significant change is detected, the monitor will measure the disagreement between the path to the target prefix and the path(s) to the reference point(s) of the target prefix. An alarm will be triggered if there is a significant disagreement between paths. In the next section, we present the three steps in detail.

Our scheme has a number of advantages. First of all, a significant difference between our approach and related work [32, 3, 11, 25, 16, 6, 31, 9, 34, 36, 15, 28, 18, 23, 19, 30, 10] is that our monitoring is conducted on the data plane. All live information required is collected from data plane. Hence, our approach does not require any alteration to Internet routing infrastructure such as setting up BGP update feeds. In related work, because the monitoring and detection mechanisms rely on access to live control plane information (*i.e.*, BGP updates), IP prefix hijacking detection has been a privilege that is only available to network operators and those with close ties to network operators. Our approach has changed this pattern and opened up the same opportunity to parties with only data plane access, which can be virtually anyone on the Internet. This new direction, combined with the distributed nature of our approach, dramatically changes the road map for how prefix hijacking monitoring and detection services can be built and deployed. In Section 6, we further discuss more details about how to deploy such a service in practice.

Another advantage of conducting detection on the data plane is that it can detect hijacking more quickly. Obviously, because prefix hijacking is a control plane mechanism, the results of hijacking also firstly emerge in control plane, *i.e.*, ill purposed BGP updates. It may be perceived that monitoring live BGP feeds in real-time provides the fastest diagnostics. However in reality due to the vast size of today's Internet and the volume of data traffic on the Internet, it is impossible for the routing infrastructure to spare enough resources to provide full BGP feeds to prefix hijacking monitors in real-time. Instead, most of the related work depend on periodically retrieving BGP information from a handful of BGP information collection points. As a result, the update period on those BGP collection points (e.g., every two hours at RouteView [2]) bounds the reaction time of these detection schemes. On the data plane, however our approach is far less restricted in terms of how often the monitors may probe; an important system design tradeoff.

Moreover, our scheme is robust. The advantage of having such a distributed architecture is that it provides good fault tolerance and a good channel to notify the target prefix regarding the detected attacks. Once a prefix is hijacked, it becomes nontrivial to send alarms to the target prefix if the path from the detection server to the target prefix is also affected by the hijack [20]. If the target prefix does not receive alarm traffic addressed to it, then it will not get the alarm notification either. However, in a distributed framework, it is likely that some subset of monitors are not affected by the attack and will still have valid paths to the target prefix. These monitors can alarm the target prefix.

Our scheme is light-weight in terms of monitoring overhead. This comes from two factors. First, the distributed nature amortizes the probing load to a number of monitors, which are distributed diversely across the Internet. Second, the probing packets on the data plane are quite small. Each monitor only needs several dozen bytes to get the probing information for a target prefix. Comparing with downloading large volume of BGP feeds on control plane, this approach can save bandwidth. At the same time, our scheme is very accurate with both low false positive and false negative ratios. As we will show in Section 5, both of the ratios are lower than 0.5%.

4. PREFIX HIJACKING DETECTION

In this section we describe our prefix hijacking detection algorithms in detail.

4.1 Monitor Selection

We model the monitor selection problem as follows. Initially we have M candidate sites around the world. Each of these sites is capable of executing the two detection techniques: network location change detection and path disagreement detection. Using all of these sites as monitors is possible, but usually not necessary. Also it may generate unnecessary monitoring traffic overhead at the target prefix network. Thus, for each target prefix, we select a subset m sites among the M candidates as monitors, and run the two detection procedures only on these m sites. The choices for monitors for a particular target prefix, however, should not be arbitrary because the locations of the monitors affect the quality of detection. In general, the monitors should be distributed in different geographical regions, and the less the paths from the monitors to the target prefix network share common links the better.

To better formulate the monitor selection problem, we first define the *correlation* between a pair of paths as the number of common links between the two paths over the length of the shorter path. If there is no shared link, the correlation is 0. On the other hand, if the two paths are identical or one path is a sub-path of the other, their correlation is 1. We also define the *correlation* between two sets of paths as the maximum path correlation between any two paths, one from each path set.

We construct the monitor selection problem as a hierarchical clustering problem. Such problems have well-known algorithms, such as [14], that are polynomial-time complex. First, we start from M clusters, with each candidate monitor being a single item cluster, and compute the correlations for all possible cluster pairs. Second, we identify the two clusters with the largest correlation among all cluster pairs, and merge these two clusters into a single cluster. Third, we recompute the correlations between all cluster pairs again. Then we repeat steps two and three till we have only m clusters. At the end we randomly select one monitor from each of the m clusters to identify the m desired monitors that will be used in monitoring service for the target prefix.

This algorithm works well in practice, and is easy to compute. The routes from all the potential monitors to the target prefix can be obtained from programs such as "traceroute". Monitor selection can be computed at a central location or even by the target prefix when it requests monitoring service.

4.2 Location Monitoring

The first detection procedure is about monitoring the "location" of the target prefix. Normally the network location of the target prefix is relatively stable. However, if hijacked, significant location change may occur. In practice, it is not necessary to pinpoint a target prefix's location on the Internet. Instead we describe a target prefix's location by its hop count distances to the set of monitors we selected using the algorithm described above. When multiple monitors detect that their hop count distances to the target prefix has changed, we conclude that the topological location of the target prefix has changed.

The detection algorithm falls into the general category of online change-point detection algorithms [5]. The problem can be stated as follows. Consider a sequence of random variable $x_t, t \in [0, T]$ with probability density $p_\theta(x)$ where θ is a parameter which may change over t. If $\theta = \theta_0$ for all $t \in [0, T]$, then there is no change. If $\theta = \theta_0$ for $t \in [0, \tau]$ but $\theta = \theta_1$ for $t \in (\tau, T]$, a change has happened at τ. The goal of this type of analysis is to identify such a change as soon as possible.

Compared to many other applications that require change-point detection, our problem is relatively simple. As shown in the next section, the hop count distance measurement is generally stable, which makes the probability density function clean and trivial. Also when change occurs, there is no build up phase. The hop count changes to a different value and remains at that value. As a

result, we decided to use the simplest classical time series change detection method: moving average with a fix-sized sliding window of S data points. Only data points obtained within this time window are taken into consideration. More specifically, the moving average is calculated as:

$$a = \frac{1}{S} \sum_{i=n-S+1}^{n} h_i,$$

where h_i is the i-th hop count, S is the sliding window size, and the n-th measurement is the newest measurement.

If a new hop count measurement departs dramatically from the previous moving average, we raise a flag indicating underlying pattern change. If multiple monitors discover significant hop count distance changes at the same time, this indicates that the topological location of the target prefix on the Internet has changed.

In practice, there are often transient problems with hop count measurements. We primarily use two techniques to smooth and filter out the "noisy" measurements.

The first is that in addition to the sliding window just mentioned, denoted as W_1, we use another sliding window, denoted as W_2 to "smooth" out current hop count measurement. The sizes of these two windows are S_1 and S_2 respectively. Because W_1 represents the past average hop count and W_2 is only used to smooth out measurement errors and noise, S_1 should be greater than S_2. The choice of S_2 represents a tradeoff between detection delay, what kind of transient problems are dominant, and how well they need to be handled. In our experiments, we used 12 and 10 respectively for S_1 and S_2.

When the network is stable, W_2's moving average hop count $a_2 = \frac{1}{S_2} \sum_{h\ in W_2} h$ is very close to W_1's moving average hop count $a_1 = \frac{1}{S_1} \sum_{h\ in W_1} h$. On the other hand, if the difference between these two averages is significant, the location of the target prefix has just changed. Therefore, if

$$\frac{\max\{a_1, a_2\}}{\min\{a_1, a_2\}} \geq T,$$

where T is a threshold, the system reports a potential hijacking.

In our experiments, we have observed that sometimes hop count measurement undergoes dramatic but very short-lived changes in the time scale of several seconds. To filter out these transient spikes, we discredit dramatic hop count changes at their first appearance. In order to do this, we define another threshold T'. Given two consecutive hop count measurements, h_1 and h_2. We remove h_2 from the sliding window if

$$\frac{\max\{h_1, h_2\}}{\min\{h_1, h_2\}} \geq T'.$$

Another problem is absent measurement. Due to packet loss or other reasons, it may happen that during a measurement interval there is no hop count data being collected at all. In our experiments, this is treated as a null data point, which effectively shortens the corresponding time window size by 1. In our experiment environment, we often have co-located monitors. They can help reduce the absent measurement problem. Measurements obtained from all co-located monitors can be combined and used as a single measurement. This way only when all co-located monitors fail to obtain a measurement, do we have an absent measurement.

A final point worth mentioning is that although we have been using hop count distance in our descriptions so far, because we are interested in location *changes*, we do not even need to obtain the absolute hop counts, which can be difficult to measure at times. In fact, only *change* in hop count is necessary for detecting location *change*. This greatly simplifies monitoring because hop count changes are indicated by the changes in residual TTL values in received IP packets, which is much easier to obtain. The change-point detection method can be easily adapted to using residual TTL instead of hop count distance.

4.3 Path Disagreement Detection

The concept of *path disagreement* described in Section 3 is fairly general. In this paper we actually study in particular the disagreement between AS paths instead of hop by hop router paths. Although router paths can be readily discovered using data plane probing tools such as traceroute, router paths are less stable than AS paths due to minor intra-AS path adjustments (e.g. for the purpose of load balancing). Also oftentimes traceroute results contain null entries for various reasons. They make hop by hop path comparison more difficult. On the other hand, AS paths have much less null entries.

Because we collect real-time measurements only from the data plane, the AS paths are not directly obtainable. We need to convert the IP addresses in traceroute results into AS numbers. This can be done with the help of some public web sites such as iPlane [12], which publishes IP-to-AS mapping data periodically.

We define the similarity between two AS paths starting from the same origin but ending at two different destinations as follows. Given two AS paths, P_1 and P_2, let's say the length $|P_1| \geq |P_2|$. We first identify a sub-path of P_1, P_1', which starts from the same origin as P_1 but has the length of $|P_2|$. Then we calculate the Hamming distance between P_1' and P_2 and denote it as d. Similarity, s, is then defined as:

$$s = 1 - \frac{d}{|P_2|}.$$

The "subtracted from 1" part of the definition makes s follow the convention of "similarity", that is, the larger the Hamming distance is, the less similar two paths are.

Once a potential hijack is reported from the location change detection procedure, we need to further check if there is still significant similarity between the AS path from the monitor to the target prefix and the AS path from the same monitor to the corresponding reference point of the target prefix. We denote the path similarities before and after the reported potential hijack as m_1 and m_2. If m_2 is less than m_1, which means the two AS paths become more inconsistent after the suspected hijacking, and m_1/m_2 is greater than some threshold T^*, which means the similarity between two AS paths decreases dramatically, the monitor raises an alarm for prefix hijack. If there are alarms from multiple monitors, the system is confident that it has just detected a prefix hijack attack.

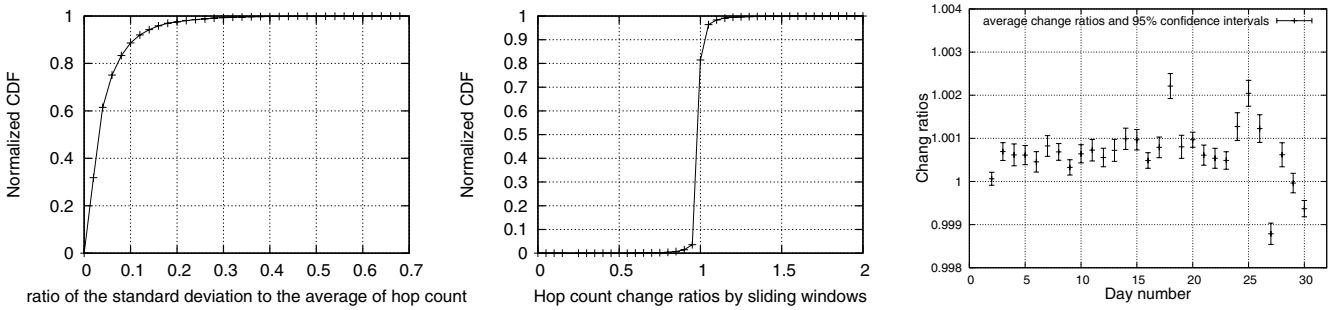
5. EXPERIMENTS AND EVALUATION

In this section, we first provide experimental results that justify the design of our hijack detection method. We then evaluate our detection scheme using analysis and simulation based on large scale Internet measurements.

5.1 Justification of Hijack Detection Design

5.1.1 Measurement Setup

In our experiments, we choose a number of nodes as network location monitors from the Planetlab [27] network. These monitors are selected to ensure geographical diversity. We manually select 43 Planetlab nodes in 25 distinct ASes at different geographical regions. In our experiments, we do not execute the monitor selection algorithm as described in Section 4. Instead, we use the co-located monitors conjunctively to avoid absent measurements.



(a) Hop count relative variance (b) Hop count change ratios (c) Hop count change ratios over time

Figure 3: Stability of hop counts

Generally speaking two kinds of prefixes can be found from a BGP table: those have Multiple Origin ASes (MOAS) and those have only a Single Origin AS (SOAS). Our experiments include prefixes of both kinds. We firstly use BGP tables obtained from RouteViews [2] and RIPE [1] to identify the initial candidates. Then for each candidate prefix, we try to identify a small number (up to 4) of live (*i.e.* responsive to "ping") IP addresses and use their network locations to approximate the network location of the prefix. To avoid scanning the entire candidate prefixes for live IP addresses, we mainly use the prefixes' local DNS server IP addresses. If we fail to verify any live IP address for a particular prefix, we discard this prefix from our experiments. MOAS prefixes are good candidates for evaluating false positives since their route changes are similar to those caused by hijacking on the control plane. So we retain all 242 MOAS prefixes that have live IP addresses. For SOAS prefixes, since there are too many, we rank them based on "popularity" (*i.e.* traffic volume) of the prefix and select the top 125 prefixes with live local DNS server IP addresses.

In our experiments, each monitor measures the hop count lengths of its paths to all selected IP addresses in all candidate prefixes. Each monitor also measures the hop count lengths of the path to other monitors. There are many ways to obtain hop count distance. Because asymmetrical routes are common on the Internet, it is important to obtain the "to" path hop count, not the "from" path hop count. The "traceroute" program is sufficient for this purpose. Normally traceroute only needs to execute a partial sequence starting from a TTL value fairly close to the known path hop count. It is only necessary to use the full sequence starting from $TTL = 1$ when the path hop count is unknown or partial sequence fails to discover the destination. If permitted, a special "ping" program that echoes the residual TTL value of the received ICMP_ECHO_REQUEST packets can also help finding the hop count length of the "to" path.

The results presented here are based on monitoring data collected from June 24*th*, 2006 to July 23*rd*, 2006. In particular, we measured the hop count length of each path (from a monitor to a target prefix) every 12 minutes. Thus, we have about 3600 hop count measurements for each path. We collected traceroute data for all paths from all monitors to all target prefixes as well as to all reference points of the target prefixes. We also collected traceroute data for paths between any two potential monitors.

Next, we justify two key assumptions of our hijack detection techniques: 1) the stability of hop count length of the path from a monitor to a target prefix, and 2) the similarity between the AS path from a monitor to a target prefix and the AS path from the same monitor to the target prefix' reference point.

5.1.2 Stability of Hop Counts

One of our prefix hijack detection techniques as described in Section 4.2 is built upon the assumption that when there is no prefix hijacking, the network location of a target prefix is relatively stable. We now verify this assumption.

From the large amount of path hop count length measurements we collected, we have observed that for a given path the hop count is relatively stable over time. The path may change from time to time, but not dramatically. For each path, we compute the standard deviation of the measured hop counts and normalize it by the average hop count of the path. Figure 3(a) shows the cumulative distribution function (CDF) of the normalized standard deviation of path hop counts from their averages. We observe that for 99% of the paths, the normalized standard deviation is less than 0.2, which indicates that if we observe a stable change of hop count with the normalized standard deviation much higher than 1.2, we should suspect that the network location of a target prefix has changed sufficiently, possibly as the result of a prefix hijacking.

Of course, the hop count measurements may contain noise due to many reasons. Instead of using individual hop count measurements directly, we aggregation them into bins of size 10 and calculate the average hop count of each bin. Then for every two consecutive bins, we define the *hop count change ratio* to be the ratio of the average hop count in the later bin to that of the earlier bin. Figure 3(b) shows the CDF of change ratios. We have found that about 98% of the change ratios are between 0.9 and 1.1. From the above two figures, we conclude that the hop count of a path from a monitor to a target prefix on the Internet is stable over the period of measurements (*i.e.* one month). Figure 3(c) shows the short term (intra-day) average change ratios on all the paths for each day, and its 95% confidence intervals. This figure shows that short term hop count stability is also very good: both the average change ratios and the 95% confidence intervals are between 0.998 and 1.003.

Note that, although we have shown that hop count is a good metric of network distance in detecting prefix hijacking, hop count may not the only choice in measuring network distance. Determining the best metric of measuring network distance is not the focus of this paper. We will defer the discussion of other potential metrics to Section 6.

5.1.3 Similarity Between AS Paths

As stated in Section 4.3, we compare the AS path from a monitor to a target prefix with the AS path from the same monitor to the

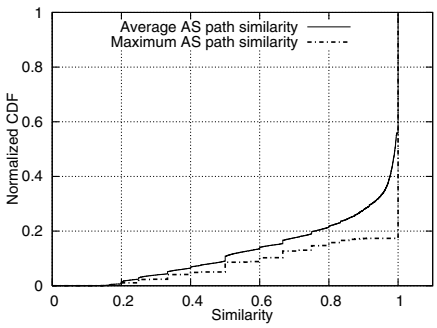


Figure 4: Average/Maximum AS path similarity

reference point of the target prefix to separate hop count changes caused by hijacking from those caused by legitimate route changes. Of course the implied assumption is that under normal conditions these two paths are very similar to each other. However, routing policies can cause the path to the target prefix to be different from that to the reference point of the target prefix. In this case, the path disagreement based detection algorithm may not produce accurate results. Thus, in this section, we study how similar the two AS paths are under normal network conditions.

In order to compare the similarity of the AS paths from a monitor to a target prefix and to its reference point, we map the paths obtained from traceroute measurements described in Section 5.1.1 to an AS path using methods proposed in [22]. Then we compute the similarity between two paths as defined in Section 4.3.

Figure 4 shows the CDF of the average and maximum AS path similarities. As we can see, 80% of the paths have similarities larger than 0.8. Since the average length of these AS paths is about 6, we can conclude that in most of the cases, there is at most 1 different AS hop on the AS paths. A low similarity between AS paths upon a sudden change in path hop counts may indicate a possible prefix hijacking.

5.2 Evaluation of Hijack Detection Scheme

In this section, we evaluate our method by running our detection scheme against simulated hijacking attack scenarios. On both January 20*th* and 21*st*, 2007, we ran our data collection program again to measure paths between the Planetlab nodes and live IP addresses that we identified in the justification stage. In particular we collected traceroute data for paths from the Planetlab nodes to the prefixes as well as paths between Planetlab nodes. In total there are 531 live IP addresses in the selected prefixes this time. Because we use real traceroute data to construct the simulation scenarios, the simulated network topology is in fact the portion of the Internet that appeared in our measurements.

5.2.1 Simulating Prefix Hijacking Attacks

In imposture scenarios, suppose the hijacker h attacks a target prefix t at time T. From a given monitor s to the target prefix t ($t \neq s$), the hop count from s to t before time T can be obtained directly from the traceroute results we have obtained. After time T, if s becomes closer to h than to t, then h has successfully hijacked traffic from s to t. The hop count from s to t is now actually the hop count from s to h, which can also be obtained directly from traceroute results. If s is still closer to t than to h after T, then h is not able to hijack traffic from s to t. The hop count from s to t after time T remains the same as that before time T.

To simulate an imposture attack, we selected one Planetlab node as the monitor, one different Planetlab node as the hijacker, which attempts to hijack the prefix of a live IP address. This was repeated

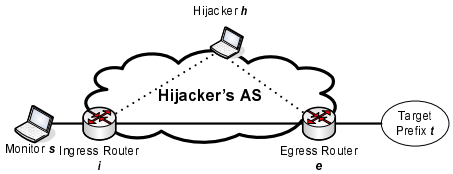


Figure 5: Simulating interception

for all possible selections of h, s, and t, except for all the cases where t's AS is on the AS path from s to h because the hijack will never succeed in these cases. In addition, since some paths were not traceroute-able, we had to discard combinations that require these paths. In total we simulated about 34000 imposture scenarios.

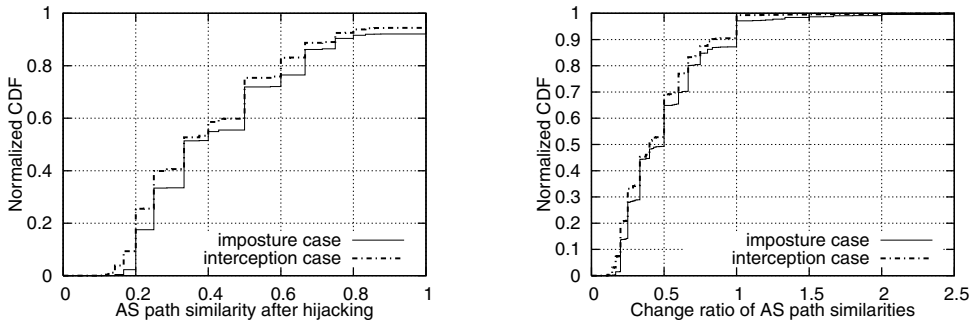
The setup for simulated interceptions is very similar to that of imposture. However, the hop count from s to t after time T is computed in a slightly different way. Given a hijacker, h, a source s, and a target prefix t, if the attack is successful (*i.e.*, s is closer to h than to t), h would forward the intercepted traffic to t. So the path that the traffic takes along is from s to h's AS then to t. However, the hop count from s to t after the interception can not be computed by simply summing up the hop count from s to h and that from h to t. As illustrated in Figure 5, h may not capture the hijacked traffic by itself, even though the hijacked traffic will be carried through h's AS. Here, we make a conservative approximation. We concatenate the route from s to h's ingress router i with the route from i to h's egress router e and with the route from e to t. If i and e are the same router, then there is no hop between them. Otherwise we assume that there is only 1 hop between them. This is usually true for small ASes. And the resulting hop count can be considered as a lower bound of the actual hop count after interception. Because for interception scenarios we must have valid traceroute data for both monitor to hijacker and hijacker to victim prefix paths, we ended up having fewer, about 25000, scenarios simulated.

5.2.2 Hop Count Changes Due to Hijacking

We use the ratio of the average hop count h_2 after hijacking to the average hop count h_1 before hijacking for the same monitor-target prefix pair to measure the hop count changes. Figure 6 plots the CDF of the ratios. We observe that while the hop count change ratios as the result of impostures is distributed almost uniformly from 0.5 to 1.5, there are more than 82% cases where the maximum change ratios among all potential monitors are larger than 1.2, and there are about 88% cases where the minimum change ratios are less than 0.8. For interception cases, the hop count is increased in most of the cases, since the hijacker forwards the intercepted traffic to the target prefix. About 98% of the cases, the maximum change ratio is lager than 1.2. The results indicate that, for a given hijacking instance, dramatic hop count changes will be observed at some (if not all) of the monitors if they are topologically diverse.

5.2.3 AS Path Disagreement Due to Hijacking

Using the simulation scenarios described in Section 5.2.1, we study how much AS path disagreement there is as a result of hijacking. This evaluates our path disagreement detection algorithm described in Section 4.3. Figure 7(a) shows the CDF of the AS path similarities resulting from hijack attacks. In 90% of the cases, the similarities after hijacking are less than 0.8. Comparing with Figure 4, where about 80% of the cases have the similarity higher than 0.8, it is expected that the change ratio on AS path similarity is significant enough that we can distinguish hijacking from legitimate routing changes. Figure 7(b) confirms our expectation. It shows the distribution of ratios of the AS path similarity before hijacking



(a) AS path similarity

(b) Change ratio of AS path similarities

Figure 7: The changes of AS path similarities due to hijacking (a) The AS path similarities resulting from hijacking. (b) The change ratio of the AS path similarity before hijacking to that after hijacking.

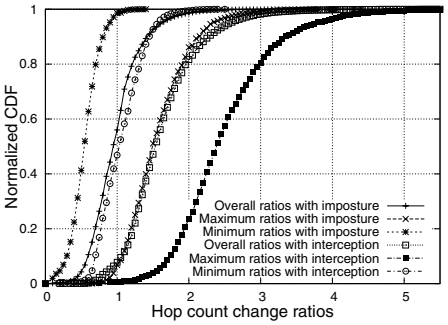


Figure 6: Hop count changes upon imposture/interception

to that of after hijacking. In about 90% cases the change ratio is less than 0.8 (*i.e.*, the AS path similarity is 20% lower than that before hijacking). Note that, even though 10% cases still show no significant difference in the AS path similarities before and after the hijacking (*i.e.*, it pass the AS path disagreement detection), if we choose the right set of monitors the results can be improved. We show more detailed results in the next section.

5.2.4 Performance of Hijack Detection Method

Now we evaluate the overall performance of our hijacking detection method as described in Section 4.

Detection Accuracy. Two important metrics for evaluating the accuracy of a detection algorithm are *false positive ratio* and *false negative ratio*. False positive ratio evaluates the percentage of falsely reported hijacking attacks, while false negative ratio evaluates the percentage of hijacking attacks that are not reported.

To evaluate the false positive ratio, we run our detection method on the series of hop counts collected in our real experiments on the Internet as described in Section 5.1.1 to see how many hijacks can be reported. While it is hard to tell whether a reported hijacking is a real hijacking, the result can serve as an upper bound for the false positive ratio of our detection algorithm. To evaluate the false negative ratio, we simulate imposture and interception scenarios based on the traceroute data using the method described in Section 5.2.2. We run our detection algorithm on all the hijacking scenarios as constructed in Section 5.2.1 to see how many cases are ignored by our algorithm.

Table 1 shows the evaluation results. The first column shows the thresholds used in the detection algorithm. There are two thresholds. Their usages are provided in Section 4.2 and 4.3 respectively. The first one is for how much hop count change is needed for the algorithm to consider the change to be sufficient to raise an alarm for target prefix network location change. The second threshold is for how much change in path similarity suffices that the two AS paths are now considered to disagree with each other. The subsequent columns present the false positive ratios and false negative ratios after hop count change detection and after both hop count change and AS path disagreement detection. We observe that as the thresholds increase the false negative ratio increases and the false positive ratio decreases. This is expected because the lower the thresholds are, the more sensitive the detection algorithm is.

It is also interesting to know the influence of the AS path disagreement detection as described in Section 4.3 on the detection result. On one hand, it filters out the legitimate route changes quite effectively and decreases the false positive ratio. On the other hand, it may also happen to filter out route changes caused by hijacks, which leads to an increase in the false negative ratio. However, as we can see from Table 1, the AS path disagreement detection can significantly reduce the false positive ratios, while only slightly increasing the false negative ratios.

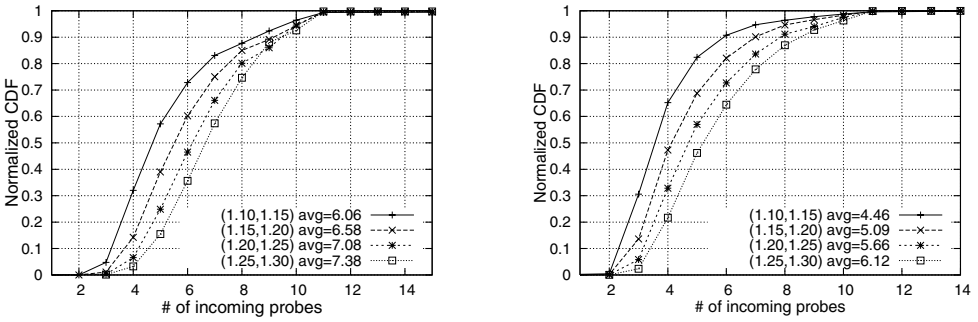
The choice for the two detection thresholds represents a tradeoff between the acceptable false positive and false negative ratios. From Table 1 we can see that reducing the false negative ratio can be achieved by using sensitive (low) thresholds and reducing false positive ratio can be achieved by using insensitive (high) thresholds. Generally speaking because the AS path disagreement test is very effective at filtering out falsely identified hijacks, the bias should be more towards having more sensitive thresholds.

It is also worthwhile to note that the false positive ratio reported in Table 1 is reasonable for operational use. For example, even if we use the most sensitive threshold choices of (1.10, 1.15) the false positive ratio is only around 0.22%. This translates to for every prefix hijacking incident, our system only generates 0.0022 false alarms. Let's assume that 1000 target prefixes are monitored and that conservatively there is at most one hijack for each target prefix per day, the number of false alarms triggered is only about $2 \sim 3$ per day, which is acceptable for manual inspection from an operational prospective.

Detection Latency. Next, we evaluate how fast our detection algorithm can detect a hijack. We measure the detection latency in terms of the number of hop count measurements needed in detecting a hijack. The reason that we do not specify latency in time is

Thresholds (Hop count, AS path)	False positive ratio		False negative ratio (imposture)		False negative ratio (interception)	
	Hop count	Hop count + AS path	Hop count	Hop count + AS path	Hop count	Hop count + AS path
(1.10, 1.15)	9.7573%	0.2248%	0.0519%	0.1413%	0.0142%	0.0149%
(1.15, 1.20)	6.5166%	0.1930%	0.0750%	0.2223%	0.0183%	0.0204%
(1.20, 1.25)	4.5034%	0.1802%	0.3316%	0.5852%	0.0376%	0.0960%
(1.25, 1.30)	3.1916%	0.1739%	0.6141%	1.0452%	0.2068%	0.3220%

Table 1: False positive and false negative ratios of hijack detection scheme



(a) Number of probes for imposture

(b) Number of probes for interception

Figure 8: Hop count measurements needed for detecting a hijacking: (a) imposture (b) interception

that it is highly dependent on how frequently the detection probing messages are sent and there is a trade off between how fast we can detect v.s. how much measurement traffic overhead we generate. This is best decided by the implementer.

Figure 8 shows the distributions of the detection latency for impostures and interceptions. We observe that it takes fewer than 9 hop count measurements to detect impostures. The average detection latency is $6.06 \sim 7.38$ measurements for different detection thresholds. Compared to imposture, the detection latency for interception is even shorter since interception usually leads to more significant hop count changes. As shown in Figure 8(b), the average detection latency is $4.46 \sim 6.12$ measurements.

Multiple Monitors. In the above experimental results, a hijacking is detected as soon as there is at least one monitor that reports a hijacking alarm. Intuitively, the more monitors reporting hijacking alarms for a target prefix at the same time, the higher confidence we have in believing that it is a real hijack. So it is interesting to explore, for imposture and interception, how many hijack reporting monitors are statistically required to conclude that a hijack indeed has succeeded. Figure 9 shows the CDF of the ratios of monitors which report the hijacking by varying detection thresholds. The ratios are presented for both imposture and interception cases after hop count change detection and after both hop count change and AS path disagreement detections. We observe that, the lower the thresholds are, the more likely that multiple monitors report hijacking both imposture and interception cases. This is consistent with the observation in Table 1. Although there can be a potential benefit of using multiple monitors in detection, simply setting a threshold on the number of monitors which report a hijacking alarm does not necessarily assure high confidence on detection results in every attack case. It is possible that a hijacking is not visible to some monitors (e.g., those monitors are "closer" to the target prefix than the hijacker, and thus not affected by the hijacks) [20, 4]. How to choose the optimal set of monitors is a complex issue and is among our future work.

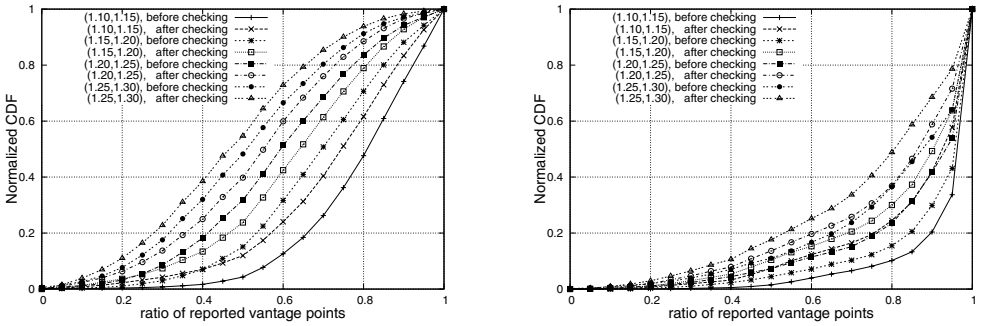
Furthermore, we have found that on average more monitors detect interceptions than impostures. This is because interceptions more likely lead to more increase in the hop counts. This observation is also consistent with the conclusion in Figure 6.

6. DISCUSSIONS AND FUTURE WORK

Granularity of Detection. The granularity of detecting possible hijacks by our scheme is at the prefix level. One possible form of hijacking is for attacker to announce a route to a subnet within a publicly announced prefix and use it for spam and phishing. This allows an attacker to gain routed address space in a fairly untraceable way. Our current scheme might not be able to detect this type of sub-prefix hijack unless that individual sub-prefix is monitored. Because ISPs currently prohibit any subnet of a /24 prefix to be announced publicly, /24 prefix is the smallest unit that can be hijacked. One possible solution to the sub-prefix hijacking problem is to sample more /24 prefixes within a target prefix to increase the likelihood of detecting such malicious behavior. On the other hand, hijacking a super-prefix (i.e., a supernet of one or more publicly announced prefixes) can only succeed on bogon prefixes (i.e., prefixes that are not allocated) because the Internet routing applies the longest prefix matching. This can be easily detected by detecting bogon prefixes.

Counter Measures. Two important metrics used in our scheme for hijacking detection are the hop count distance from a monitor to a target prefix and AS paths from a monitor to a target prefix and its reference point. A sophisticated attacker, since it has hijacked the traffic flow, may try to masquerade this information to hide ongoing attacks from being detected. We now discuss briefly how the attacker may counter these measurements and how that may affect our detection scheme.

To counter the hop count measurement, the most obvious method is for the attacker to modify the TTL value in the IP header. There are basically two kinds of modifications the attacker can perform. The first is to mimic the before-attack network location of the target prefix. This may be effective for blocking the view of a single detection monitor if the attacker knows the location of the monitor. However, in our scheme, multiple monitors are measuring the tar-



(a) Detection ratios for imposture

(b) Detection ratios for interception

Figure 9: Detection ratio among monitors: (a) imposture (b) interception

get prefix simultaneously. Unless the attacker knows the locations of all the monitors and the correct hop count distances from these monitors to the target prefix, it is unlikely for the attacker to mimic the TTLs so the network location change of the target prefix is not detected by any of the monitors. Of course the attacker may also randomly change the TTL value in IP header so that the detection scheme may observe a lot of noise. This kind of behavior change itself constitutes changes and should be taken as an equivalent to network location change as well. When this occurs, together with a positive result of AS path disagreement test, it can be enough for generating hijack alarm.

Fortunately for AS path disagreement tests, the attacker cannot fake the AS path from the monitor to the reference point of the target prefix because that traffic is not hijacked. For the AS path to the target prefix, the hijacker can not affect the correctness of portion of path that is before the hijacker either. This portion of the AS path is likely already enough to produce enough AS path disagreement. Thus, the AS path disagreement test result can still be trusted.

To sum up, in practice, it is not very easy for the hijacker to counter all of the hop count and AS path measurements of our scheme. This is an advantage of using a distributed multi-vantage point approach. As part of our future work, we will continue to investigate other counter measures that attackers may launch and how to suppress them.

Network Distance Metrics. Hop count is not the only choice in measuring network distance. Another potentially applicable metric is end-to-end latency. The basic idea is similar: monitors report hijacking alarms by detecting significant changes on the latency characteristics between themselves and the target prefix. Compared with hop count, end-to-end latency measurement is harder for hijacker to evade. However, the egregious noise in the end-to-end latency measurement caused by congestion and network usage makes hijacking detection more difficult, probably requiring more complicated signal processing technologies. We will explore the feasibility of using end-to-end latency measurement as part of our future work.

Deployment. There are various ways to deploy our detection scheme on the Internet. A content distribution company can easily deploy this scheme. Because our scheme is a data plane scheme, a more interesting deployment is an incremental and collaborative deployment very much like the popular peer-to-peer system. Different from the traditional P2P system, where the collaborations are among individual nodes, our peering relationships can be built based on prefixes. Each prefix may provide a monitoring service to other peers, while receiving the monitoring service from other prefixes by applying the monitor selection algorithm in Section 4. In this approach, some interesting problems emerge. For example, how does a prefix trust its monitors? How can a malicious monitor be prevented from collaborating with a hijacker or flooding the peer-to-peer monitoring network with false hijacking alarms? How can the monitoring load among different monitors be balanced? With a real deployment in place, it would be also useful to further quantify the extent of the prefix hijacking problem on the Internet. All of these leave us with an interesting future work.

7. RELATED WORK

Existing proposals to the IP prefix hijack problem can be categorized into two broad categories: crypto and non-crypto based. Crypto-based solutions, such as [32, 3, 11, 25, 16, 6, 31], require BGP routers to sign and verify the origin AS and/or the AS path to reject false routing messages as soon as they are detected, but the signature generation and verification have significant impact on router performance. Non-crypto proposals such as [9, 34, 36, 15, 28] require changing router softwares so that inter-AS queries are supported [9, 28], stable paths are more preferred [34, 15], or additional attributes are added into BGP updates to facilitate detection [36]. All the above proposals are not easily deployable because they all require changes to router software, router configuration, or network operations, and some require public key infrastructures.

There are some existing proposals [18, 23, 19, 30] that only do passive monitoring and thus are deployable, but they often suffer from high false positives. [23, 19] monitor the origin AS(es) of the target prefix from RouteViews or RIPE, and then notify the prefix owner about the changes via email [23, 19]. Because hijacking is often not distinguishable from legitimate routing changes, it is up to the prefix owner to determine which is the case. [18, 30] check the routing registry data to see whether geographic location of the target prefix changes [18] or whether the routing update conform to prefix origin information and the routing policy [30] but the routing registry data can be outdated and inaccurate.

Recently, utilizing data plane information together with control plane information in hijacking detection is gaining attention [32, 10, 4]. The *Listen* approach in [32] determines whether a target prefix is blackholed by checking whether the prefix has any complete TCP sessions. [4] conducted postmortem analysis of prefix interception in the Internet by comparing the AS-level traceroute [22] to the target prefix with the BGP path to the same prefix. [10] first detects control plane anomalies using live BGP feeds and

then reduces false positives by checking whether the target prefix shows any inconsistencies in its fingerprints such as Host OS, TCP timestamp, ICMP timestamp, *etc*. In contrast, our approach detects hijacking in real-time using data plane information, without relying on live BGP feeds.

In this paper we used residual TTL as the network location of a target prefix. We would like to acknowledge that the residual TTL or hop-count has been used in different contexts before. For example, [8] checks whether the residual TTL in BGP updates are in a legitimate range to make sure they are from expected neighbors in order to defend against DDOS attack against BGP. Another example is [13] which proposes an approach to defend spoofed DDOS attacks by checking the distribution of the hop counts from the packet sources.

8. CONCLUSION

This paper proposes a light-weight distributed scheme for detecting IP prefix hijacks. Different from most, if not all, other previous work on this topic, our proposal detects the hijacking by conducting measurements in the data plane.

The design of the detection scheme is based on two key observations we have made on the Internet, hop count stability and AS path similarity. Our scheme continuously tests these two assertions in a distributed and light-weight manner, and uses any departure from this stability and similarity as the trigger for the hijack alarms of the target prefix.

Our scheme has several advantages over the previous hijacking detection schemes: 1) It is light-weight, detecting with less probing overhead; 2) it is highly accurate in hijack detection with both very low false positive and false negative ratios; 3) it can detect prefix hijacking in real-time; 4) it does not require any modification of existing protocols and network infrastructure, making it suitable for incremental deployment; and 5) it is highly robust in terms of monitoring failure and attackers' evasion.

9. ACKNOWLEDGMENTS

We thank Tom Scholl, Aman Shaikh and anonymous reviewers for their valuable suggestions. We are also grateful to Michalis Faloutsos, our shepherd, for his help in revising the paper.

10. REFERENCES

[1] RIPE RIS Raw Data. `http://www.ripe.net/projects/ris/rawdata.html`.

[2] University of Oregon Route Views Archive Project. `http://www.routeview.org`.

[3] W. Aiello, J. Ioannidis, and P. McDaniel. Origin Authentication in Interdomain Routing. In *Proc. of ACM CCS*, Oct. 2003.

[4] H. Ballani, P. Francis, and X. Zhang. A Study of Prefix Hijacking and Interception in the Internet. In *Proc. ACM SIGCOMM*, Aug. 2007.

[5] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. 1993. Prentice-Hall, Inc.

[6] K. Butler, P. McDaniel, and W. Aiello. Optimizing BGP Security by Exploiting Path Stability. In *Proc. ACM CCS*, Nov. 2006.

[7] H. Chan, D. Dash, A. Perrig, and H. Zhang. Modeling Adoptability of Secure BGP Protocols. In *Proc. ACM SIGCOMM*, Sept. 2006.

[8] V. Gill, J. Heasley, and D. Meyer. The Generalized TTL Security Mechanism (GTSM). RFC 3682, Feb. 2004. http://www.ietf.org/rfc/rfc3682.txt.

[9] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working Around BGP: An Incremental Approach to Improving Security and Accuracy of Interdomain Routing. In *Proc. NDSS*, Feb. 2003.

[10] X. Hu and Z. M. Mao. Accurate Real-time Identification of IP Prefix Hijacking. In *Proc. IEEE Security and Privacy*, May 2007.

[11] Y.-C. Hu, A. Perrig, and M. Sirbu. SPV: Secure Path Vector Routing for Securing BGP. In *Proc. ACM SIGCOMM*, Aug. 2004.

[12] iPlane. `http://iplane.cs.washington.edu/`.

[13] C. Jin, H. Wang, and K. G. Shin. Hop-Count Filtering: An Effective Defense Against Spoofed DDos Traffic. In *Proc. ACM CCS*, Oct. 2003.

[14] S. Johnson. Hierarchical Clustering Schemes. In *Psychometrika*, 1967.

[15] J. Karlin, S. Forrest, and J. Rexford. Pretty Good BGP: Protecting BGP by Cautiously Selecting Routes. In *Proc. IEEE ICNP*, Nov. 2006.

[16] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (S-BGP). *IEEE JSAC Special Issue on Network Security*, Apr. 2000.

[17] J. Kim, S. Y. Ko, D. M. Nicol, X. A. Dimitropoulos, and G. F. Riley. A BGP Attack against Traffic Engineering. In *Proc. of the 36th conference on Winter simulation*, 2004.

[18] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Topology-based Detection of Anomalous BGP Messages. In *Proc. RAID*, Sept. 2003.

[19] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. PHAS: A Prefix Hijack Alert System. In *Proc. USENIX Security Symposium*, Aug. 2006.

[20] M. Lad, R. Oliveira, B. Zhang, and L. Zhang. Understanding Resiliency of Internet Topology Against Prefix Hijack Attacks. In *Proc. IEEE/IFIP DSN*, June 2007.

[21] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP Misconfiguration. In *Proc. of ACM SIGCOMM*, Aug. 2002.

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

[23] RIPE myASn System. http://www.ris.ripe.net/myasn.html.

[24] Aki Nakao. PLUTO BGP Sensor/ BGP Feed. http://www.planet-lab-jp.org/pluto/pluto-bgp-sensor.html.

[25] J. Ng. Extensions to BGP to Support Secure Origin BGP. ftp://ftp-eng.cisco.com/sobgp/drafts/draft-ng-sobgp-bgp-extensions-02.txt, April 2004.

[26] O. Nordstrom and C. Dovrolis. Beware of BGP Attacks. *ACM SIGCOMM Computer Communications Review (CCR)*, Apr. 2004.

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

[28] S. Y. Qiu, F. Monrose, A. Terzis, and P. D. McDaniel. Efficient Techniques for Detecting False Origin Advertisements in Inter-domain Routing. In *Proc. IEEE NPsec*, Nov. 2006.

[29] A. Ramachandran and N. Feamster. Understanding the Network-Level Behavior of Spammers. In *Proc. ACM SIGCOMM*, Sept. 2006.

[30] G. Siganos and M. Faloutsos. Neighborhood Watch for Internet Routing: Can We Improve the Robustness of Internet Routing Today? In *Proc. IEEE INFOCOM*, May 2007.

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

[32] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and Whisper: Security Mechanisms for BGP. In *Proc. USENIX NSDI*, Mar. 2004.

[33] R. Teixeira, S. Agarwal, and J. Rexford. BGP Routing Changes: Merging Views from Two ISPs. In *ACM SIGCOMM Computer Communication Review*, Oct. 2005.

[34] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. Wu, and L. Zhang. Protecting BGP Routes to Top Level DNS Servers. In *Proc. IEEE ICDCS*, 2003.

[35] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Wu, and L. Zhang. An Analysis BGP Multiple Origin AS(MOAS) Conflicts. In *Proc. ACM IMW*, Oct. 2001.

[36] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Wu, and L. Zhang. Dection of Invalid Routing Announcement in the Internet. In *Proc. IEEE/IFIP DSN*, June 2002.