Locating Prefix Hijackers using LOCK

Tongqing Qiu
Georgia Tech
tongqqiu@cc.gatech.edu

Lusheng Ji
AT&T Labs – Research
lji@research.att.com

Dan Pei
AT&T Labs – Research
peidan@research.att.com

Jia Wang
AT&T Labs – Research
jiawang@research.att.com

Jun (Jim) Xu
Georgia Tech
jx@cc.gatech.edu

Hitesh Ballani
Cornell University
hitesh@cs.cornell.edu

Abstract

Prefix hijacking is one of the top known threats on today's Internet. A number of measurement based solutions have been proposed to detect prefix hijacking events. In this paper we take these solutions one step further by addressing the problem of locating the attacker in each of the detected hijacking event. Being able to locate the attacker is critical for conducting necessary mitigation mechanisms at the earliest possible time to limit the impact of the attack, successfully stopping the attack and restoring the service.

We propose a robust scheme named LOCK, for LOCating the prefix hijacKer ASes based on distributed Internet measurements. LOCK locates each attacker AS by actively monitoring paths (either in the control-plane or in the data-plane) to the victim prefix from a small number of carefully selected monitors distributed on the Internet. Moreover, LOCK is robust against various countermeasures that the hijackers may employ. This is achieved by taking advantage of two observations: that the hijacker cannot manipulate AS path before the path reaches the hijacker, and that the paths to victim prefix "converge" around the hijacker AS. We have deployed LOCK on a number of PlanetLab nodes and conducted several large scale measurements and experiments to evaluate the performance. Our results show that LOCK is able to pinpoint the prefix hijacker AS with an accuracy up to 94.3%.

1 Introduction

The Internet consists of tens of thousands of Autonomous Systems (ASes), each of which is an independently administrated domain. Inter-AS routing information is maintained and exchanged by the Border Gateway Protocol (BGP). The lack of adequate authentication schemes in BGP leaves an opportunity for misbehaving routers to advertise and spread fabricated AS paths for targeted prefixes. Originating such a false AS path announcement is referred to as "prefix hijacking". Once a BGP router accepts such a false route and replaces a legitimate route with it, the traffic destined for the target prefix can be redirected as the hijacker wishes. The victim prefix network of a successful hijacking will experience performance degradation, service outage, and security breach. The incident of the prefix of YouTube being hijacked by an AS in Pakistan for more than 2 hours [1] is just a recent and better known reminder of the possibility of real prefix hijacking attacks.

Recently proposed solutions for combating prefix hijacking either monitor the state of Internet and detect ongoing hijacking events [12, 21, 22, 26, 34, 45], or attempt to restore service for victim prefix networks [42]. Both approaches are compatible with existing routing infrastructures and generally considered more deployable than another family of proposals (e.g., [4, 8, 11, 13, 18, 19, 27, 32, 35–37, 44]) which aim at prefix hijacking prevention, because the latter usually require changes to current routing infrastructures (e.g., router software, network operations), and some also require public key infrastructures.

However, the aforementioned detection and service restoration solutions only solve parts of the problem and a critical step is still missing towards a complete and automated detection-recovery system. That is how to locate the hijackers. More importantly, the location of hijackers is one of the key information that enables existing mitigation methods against prefix hijacking (e.g., [42]). One may consider this step trivial. Indeed in current practice this step is actually accomplished by human interactions and manual inspections of router logs. However, we would argue that the success of the current practice is due to the fact that discovered attacks so far are still primitive. Many of them are simply not attacks but rather the results of router mis-configurations. As we will elaborate, locating sophisticated hijackers is far from a trivial problem and the current practice will have great difficulties in locating them.

In this paper, we present a scheme called LOCK to LOCate prefix hijacKers. It is a light-weight and incrementally deployable scheme for locating hijacker ASes. The main idea behind LOCK are based the following two observations: that the hijacker cannot manipulate AS path before the path reaches the hijacker, and that the paths to victim prefix "converge" around the hijacker AS. Our contributions are four-fold. First, to the best of our knowledge, it is the first work studying the attacker locating problem for prefix hijacking, even when countermeasures are engaged by the hijacker. Second, our locating scheme can use either data-plane or control-plane information, making the deployment more flexible in practice. Third, we propose an algorithm for selecting locations where data-plane or control-plane data are collected such that the hijackers can be more efficiently located. Finally, we have deployed LOCK on a number of PlanetLab nodes and conducted several large scale measurements and experiments to evaluate the performance of LOCK against three groups of hijacking scenarios: synthetic attacks simulated using real path and topology information collected on the Internet, reconstructed previously known attacks, and controlled attack experiments conducted on the Internet. We show that the proposed approach can effectively locate the attacker AS with up to 94.3% accuracy.

The rest of the paper is organized as follows. Section 2 provides background information on prefix hijacking. Section 3 provides an overview of the framework of our LOCK scheme. Then we describe detailed monitoring and locating methodologies in Section 4 and Section 5 respectively. Section 6 evaluates the performance of the LOCK scheme. Section 7 briefly surveys related works before Section 8 concludes the paper.

2 Background

As mentioned before, IP prefix hijacking occurs when a mis-configured or malicious BGP router either originates or announces an AS path for an IP prefix not owned by the router's AS. In these BGP updates the misbehaving router's AS appears very attractive as a next hop for forwarding traffic towards that IP prefix. ASes that receive such ill-formed BGP updates may accept and further propagate the false route. As a result the route entry for the IP prefix in these ASes may be *polluted* and traffic from certain part of the Internet destined for the victim prefix is redirected to the attacker AS.

Such weakness of the inter-domain routing infrastructure has great danger of being exploited for malicious purposes. For instance the aforementioned misbehaving AS can either drop all traffic addressed to the victim prefix that it receives and effectively perform a denial-of-service attack against the prefix owner, or redirect traffic to an incorrect destination and use this for a phishing attack [28]. It can also use this technique to spread spams [33].

We refer to this kind of route manipulation as *IP prefix hijack attacks* and the party conducting the attack *hijacker* or *attacker*. Correspondingly the misbehaving router's AS becomes the *hijacker AS*, and the part of the Internet whose traffic towards the victim prefix is redirected to the hijacker AS is *hijacked*. So do we call the data forwarding paths that are now altered to go through the hijacker AS *hijacked*. We also refer to the victim prefix as the *target prefix*.

Following the convention in [45], we classify prefix hijacks into the following three categories:

- **Blackholing**: the attacker simply drops the hijacked 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.

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 [6]. In addition, the characteristics of different hijack types are different, which often affect how different types of attacks are detected. In this paper, we use the term *hijack* to refer to all three kinds of prefix hijack attacks unless otherwise specified.

There have been a number of approaches proposed for detecting prefix hijacks. They utilize either information in BGP updates collected from control plane [21, 22, 26, 34], or end-to-end probing information collected from data plane [43, 45], or both [6, 12, 36]. We will not get into the details of most of these approaches here because LOCK is a *hijacker-locating scheme*, not a hijack detection scheme. The difference between these two will be explained later in section 3.1. To locate a hijacker, the LOCK scheme only needs to know whether a given prefix is hijacked. Therefore LOCK can be used together with any detection method to further locate the hijacker AS. Moreover, LOCK can locate the hijacker using either data-plane or control-plane information.

3 Framework

In this section, we present an overview of key ideas of the hijacker locating algorithm in LOCK. Similar to detecting prefix hijacking, locating hijacker AS can be done in either control-plane or data-plane. Either way, the goal is, to use the AS path information to the hijacked prefix observed at multiple and diverse vantage points (or monitors) to figure out who the hijacker is. In control-plane approach, the AS path information is obtained from BGP routing tables or update messages. In data-plane approach, the AS path is obtained via AS-level traceroute (mapping the IP addresses in traceroute to AS numbers).

Both methods have pros and cons. Realtime data-plane information from multiple diverse vantage points is easier to be obtained than realtime BGP information(e.g. the BGP updates from [3] are typically delayed for a few hours). On the other hand, it is relatively easier for the attacker to manipulate the data-plane AS path to countermeasure the locating algorithm than the control-plane AS path. LOCK can use either data-plane or control-plane AS paths to locate the hijackers.

3.1 Challenges

Currently, the most commonly used hijacker-locating approach (called *simple locating approach*) is to look at the origin ASes of the target prefix. For example, Figure 1 (a) shows the control-plane AS path information to target prefix p at vantage points $M1$, $M2$, and $M3$, respectively, before hijacker H launches the hijack. All three vantage points observe the origin AS is T. In Figure 1 (b), Hijacker AS H announces a path H to target prefix p, which ASes A, B, $M1$, and $M2$ accept since the paths via H are better than their previous ones via CDT. In this case, the simple locating approach can easily identify the newly-appearing origin AS H as the hijacker.

However, this simple locating approach can fail even without any countermeasures by the hijackers. For example, in Figure 1(c), hijacker H pretends there is a link between H and the target AS T, and announces an AS path HT, again accepted by A,B,$M1$, and $M2$. The simple locating approach does not work here since the origin AS in all the AS paths are still T.

One might try to look beyond just origin AS and check other ASes in the path, but the hijacker AS might counter this such that the hijacker AS might not even appear in any of the AS paths. For example, in Figure 1(d) H simply announces an AS path T without prepending its own AS number H [1].

Above challenges in control-plane locating approaches also exist in data-plane approaches. Almost all data-plane path probing mechanisms are derived from the well known *traceroute* program. In *traceroute*, triggering messages with different initial TTL values are sent towards the same destination. As these messages are forwarded along the path to this destination, as soon as a message's TTL reduces to zero after reaching a router, the router needs to send back an ICMP Timeout message to notify the probing source. If the triggering messages go through the hijacker, this happens when the triggering messages' initial TTL values are greater than the hop distance from the probing source to the hijacker, the hijacker can do many things to interfere the path probing as a countermeasure to the locating algorithm.

In Figure 2(a), the hijacker AS's border router responds to traceroute honestly in blackholing (in which for example the border router responds with a *no route* ICMP message with its own IP address) and imposture (in which for example a router in H responds "destination reached" message with its own IP address). In either case, the router address belongs to H and maps to AS H, and the simple locating approach can identify H as the newly appearing origin AS hence the hijacker AS.

However, in the interception attack shown in Figure 2(b), the hijacker further propagates the traceroute probe packets to the target via $XYZT$, thus the origin AS is still H. Hence the simple locating approach fails in this case.

Furthermore, the hijacker can use various countermeasures. For instance, the hijacker may simply drop the triggering messages without replying to interrupt *traceroute* probing from proceeding further. Or it may send back ICMP Timeout messages with arbitrary source IP addresses to trick the probing source into thinking routers of those addresses are en route to the destination. The hijacker may even respond with ICMP Timeout messages before the triggering messages' TTL values reach zero. In Figure 2 (c), hijacker H manipulates the traceroute response such that after the IP-to-AS mapping, the AS path appears to $M1$ to be $ACDT$, and appears to $M2$ to be BDT, neither of which contains hijacker AS H in it, making the hijacker locating difficult. We refer to above manipulation of traceroute response as *countermeasure* for data-plane locating approach, and call such hijackers *countermeasure-capable* or *malicious*.

In summary, sophisticated hijackers that are capable of engaging countermeasures can inject false path information into measurements collected in both control plane and data plane, easily evading simple hijacker-locating mechanisms. We therefore design a more effective algorithm for locating these hijackers in the next section.

3.2 Locating Hijackers

The basic idea of LOCK is based on two key observations, which apply to both data-plane and control-plane approaches, different types of hijacks (blackholing, imposture, and interception), and with or without countermeasures by the attackers.

The first observation is that *the hijacker cannot manipulate the portion of the AS path from a polluted vantage point to the upstream (i.e., closer to the vantage point) neighbor AS of the hijacker AS*. For example, in

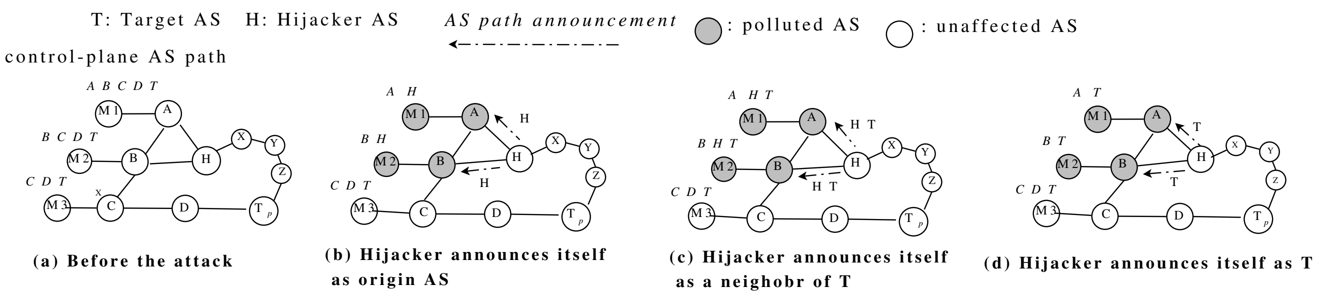

T: Target AS H: Hijacker AS
AS path announcement
: polluted AS
: unaffected AS
control-plane AS path
A B C D T
B C D T
C D T
(a) Before the attack
A H
B H
(b) Hijacker announces itself as origin AS
A H T
B H T
(c) Hijacker announces itself as a neighobr of T
A T
B T
(d) Hijacker announces itself as T

Figure 1: Control plane examples

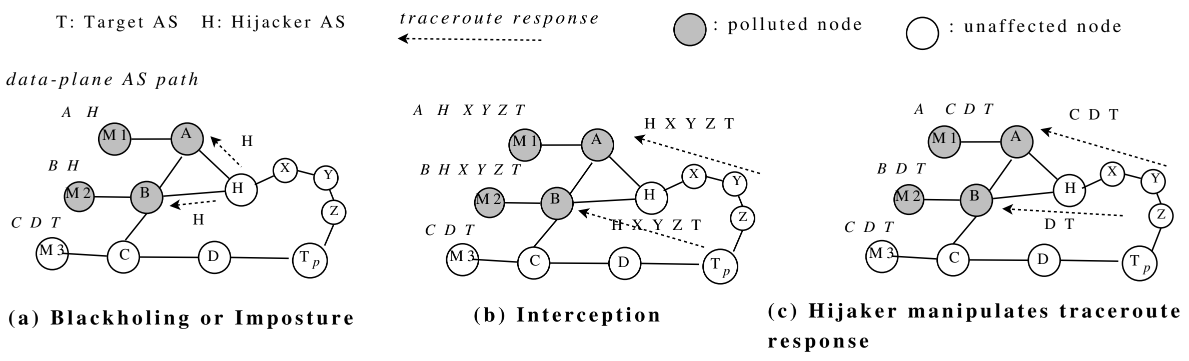


Figure 2: Data plane examples

Figures 1(c) and (d) and Figures 2 (b) and (c), for the polluted vantage points $M1$ and $M2$, the upstream ASes for hijacker AS H are A and B, and the portion of AS path $M1A$ and $M2B$ are trustworthy. This is easy to understand since the routers from the vantage points to hijacker upstream ASes are all well-behaving ones thus conform to BGP protocol in control-plane and ICMP protocol used in traceroute in data-plane.

The second observation is that *the trustworthy portion of polluted AS paths from multiple vantage points to a hijacked victim prefix "converge" "around" the hijacker AS.* This is also intuitive since, if the set of monitors are topologically diverse enough, the trustworthy portion of AS paths from all the polluted monitors to the target prefix must include the upstream AS neighbors of the hijacker AS (e.g. in Figure 1(d), and Figure 2 (c)) thus converge "around" the hijacker AS, or directly converge at hijacker AS (e.g. in Figure 1(b) and (c) and Figure 2(a) and (b)).

Since we do not know beforehand the hijack scenarios and whether there is any countermeasure, we focus on identifying these upstream neighbors of the hijacker AS, and then intuitively hijacker should be within the intersection of the 1-hop neighbor sets of the hijacker's neighbors. And chances are that the size of the intersection set is very small if the monitors have diversified locations. The neighbor sets of a given AS can be obtained from a daily snapshot of the state-of-arts AS level topology repository such as [16].

For example, in both Figures 1 and 2, ideally suppose we know that ASes A, B (which are on the polluted paths from vantage points $M1$ and $M2$, respectively) are the upstream neighbors of the hijacker. We can then infer that the hijacker AS should be within the intersection of $neighorset(A) = \{M1, B, H\}$ and $neighborse(B) = \{M2, A, H\}$, which is H. Of course in reality LOCK does not know beforehand which ASes are the upstream neighbors of the hijackers, thus each AS in a polluted path can potentially be such a neighbor of the hijacker AS. And hence the hijacker could be a neighbor of any of these nodes. We therefore put all the neighbors of *each* AS on a polluted path together with the path nodes themselves to form a *neighborhood set* of the polluted path. The hijacker should be included in this neighborhood set.

For reasons that we will explain in the Section 5, instead of using the neighborhood set of an arbitrary path, LOCK conservatively starts from the union of all the neighborhood sets of all polluted paths, $\mathcal{H}$. Then given that all polluted paths go through a neighbor AS of the hijacker, an AS which appears in more neighborhood sets is more likely to be the hijacker. We thus "rank" the ASes within $\mathcal{H}$ based on how many neighborhood sets an AS is in to narrow down to the handful of top ranked ASes.

Also when there are multiple convergence points, the earliest convergence point is more likely to be the hijacker than the later ones. More detailed ranking algorithm will be presented in Section 5.

As shown in this section, LOCK can utilize either control-plane or data-plane information. However, for the ease of presentation and due to space limitation, in the rest of paper we focus on data-plane approach unless otherwise specified.

4 Monitor Selection

LOCK operates in a distributed fashion from a number of monitors on the Internet. Both the number of monitors and locations of these monitors affect the accuracy in locating prefix hijackers. In general, the more monitors used by LOCK, the higher accuracy LOCK can achieve in locating prefix hijackers, and the more measurement overhead are incurred by LOCK. More importantly, the measurement overhead increase linearly as the number of monitors increases, while at the same time the improved accuracy gained by each additional monitor can gradually diminish. Therefore, it is hopeful to achieve very good accuracy with a limited number of carefully selected monitors.

In this section, we present a novel algorithm for selecting a number of monitors from a candidate set. In particular, we model the monitor selection problem as follows. Initially, we have M candidate monitors around the world. For each target prefix, we select a subset m monitors among the M candidates. In order to achieve the highest possible hijacker-locating accuracy with a limited number of monitors, the selection of monitors should be guided by two objectives: (i) maximize the likelihood of observing hijacking events on the target prefix; and (ii) maximize the diversity of paths from monitors to the target prefix so that a hijacking event can be observed from multiple distinct vantage points.

Our monitor selection algorithm consists of three steps:

1. **Clustering**: The M candidate monitors are grouped into m clusters. Monitors in the same cluster have more similar paths to the target prefix than those in different clusters.

2. **Ranking**: Candidate monitors in each cluster are ranked based on probability of their paths to the target prefix being polluted when the prefix is hijacked. The monitors with higher ranks are more likely to observe the hijacking event.

3. **Selecting**: The monitor which ranks the highest in each cluster is chosen to monitor the target prefix. Thus, a total of m monitors are selected for each target prefix.

4.1 Clustering

For a given target prefix, the candidate monitors are clustered based on similarity of their AS-level paths to the prefix. We measure the *similarity* between a pair of paths as the number of common ASes between these two paths over the length of the shorter path. If there is no common AS, the similarity score is 0. On the other hand, if the two paths are identical or one path is a sub-path of the other, the similarity score is 1. We also define the *similarity* between two clusters of paths as the maximum similarity between any two paths, one from each cluster.

We model the clustering part as a hierarchical clustering problem. Such problems have well-known algorithms, such as [17], that are polynomial-time complex. In this paper, we adopt the following simple clustering algorithm[2]. First, we start from M clusters, with one candidate site in each cluster, and compute similarity score for each pair of clusters. Second, we identify the pair of clusters with the largest similarity score among all pairs of clusters, and merge these two clusters into a single cluster. Third, we recompute the similarity score between this newly-formed cluster with each of the other clusters. We repeat steps two and three until only m clusters remain.

4.2 Ranking

We rank candidate monitors in each cluster based on their likelihood of observing hijacking events on the target prefix t (i.e., the path from monitor to target prefix is polluted by hijacking). For a given candidate site s, whether or not the route from s to t is polluted by hijacker h depends on the original best route (before the hijacking happens) from s to t and the fake route announced by h. This has been demonstrated by previous analysis in [6].

We assume that "prefer customer route" and "valley-free routing" are commonly adopted interdomain routing policies on today's Internet. We denote the original best route from s to t as a "customer-route", a "peer-route", or a "provider-route" if the next-hop AS on the route from s to t is a customer, a peer, or a provider of the AS to which s belongs, respectively. According to the interdomain routing policies, a customer-route would be the most preferable and a provider-route would be the least preferable by each router; similarly, when policy preferences are equal, the route with shorter AS path is more preferable [10]. Therefore, when hijacker h announces a fake path, the monitor whose original best route is provider-route is more likely to be polluted than a original route of peer-route, which in turn is more likely to be polluted

Algorithm 1: Ranking monitors in each cluster

```
1 foreach monitor $i$ in the cluster
2     if provider-route $R[i] = 300$; /* Assign the
      ranking. The larger the number is, the higher the
      rank is. */
3     elseif peer-route $R[i] = 200$;
4     else $R[i] = 100$;
5     $R[i] += D(i, t)$; /* Add the AS-level distance */
```

than a original route of customer-route; when the policy preferences are equal, the monitor whose original best route has a longer AS path to t is more likely to be polluted than the one whose original best route has a shorter AS path (Please refer to Table 1 of [6] for detailed analysis). Our ranking algorithm is shown in Algorithm 1. Note that establishing AS topology itself is a challenging problem. We use most advanced techniques [30] to infer the AS relationship. Admittedly, inferred results could be incomplete. However, the evaluation part will show that the ranking algorithm based on such data can still achieve high location accuracy.

5 Hijacker-Locating Algorithm

LOCK locates hijacker AS based on AS paths from a set of monitors to the victim prefix. The AS path from a monitor to the victim prefix can be either obtained from the control plane (e.g., BGP AS path) or from the data plane (e.g., traceroute path). In the latter case, LOCK will need to pre-process the path and compute the corresponding AS path (described in Section 5.1).

5.1 Pre-Processing

When a prefix is hijacked, a portion of the Internet will experience the hijack. Traffic originated from this portion of the Internet and destined for the hijacked prefix will be altered to go through the hijacker. Monitors deployed in this affected portion of the Internet can observe that their monitor-to-prefix paths being altered. These monitor-to-prefix paths are the foundation of our hijacker-locating algorithm. Only paths changed by the hijack event should be supplied to the hijacker-locating algorithm. Methods such as the one outlined in [45] help separate real hijack induced path changes from changes caused by other non-hijack reasons.

If the monitor-to-prefix path is obtained from the data plane, then LOCK pre-processes the path in the following way. The most common tool for acquiring IP forwarding path in the data plane is the well known *traceroute* program. This program sends out a series of triggering packets with different initial TTL values to trigger the routers en route to the destination to return ICMP Timeout messages as soon as they observe a triggering message's TTL value reaching 0, hence revealing these routers' identities. These *traceroute* results are router-level paths and they need to be converted to AS-level paths. During this conversion, NULL entries in *traceroute* results are simply discarded. This simplification rarely has any effect on the resulted AS path because as *traceroute* proceeds within a particular AS, only if all routers in this AS failed to show up in *traceroute* results our results may be affected, which we have found this to be very rare. These resulting AS paths are known as the "reported paths" by the monitors in the rest of the section.

We use publicly available IP to AS mapping data provided by the iPlane services [15] to convert router IP addresses to their corresponding AS numbers. It is known that accurately mapping IP addresses to AS numbers is difficult due to problems such as Internet Exchange Points (IXPs) and sibling ASes [6, 25]. We argue that the impact of these mapping errors on the results of our hijacker-locating algorithm is minor. Firstly the distribution of the nodes, either routers or ASes, that may cause any mapping error in their corresponding Internet topologies, either router level or AS level, is sparse. If our paths do not contain these problematic nodes, our results are not affected by mapping errors. Secondly, it will become apparent, as more of the details of the hijacker-locating algorithm are described, that our algorithm is rather robust against such mapping errors. As long as these errors do not occur when mapping nodes near the hijacker, they will not affect the result of our algorithm. It is also worthwhile noting that the IP to AS mapping data do not need to be obtained from realtime control plane data. That is, the IP to AS mapping can be pre-computed and stored since it usually does not change over short period of time.

It is also helpful to perform sanity checks on the AS paths before we begin the hijacker-locating algorithm. The hijacker may forge *traceroute* results if a *traceroute* triggering message actually passes through the hijacker. Since the prefix has been hijacked, triggering messages with large enough initial TTL values, at least larger than the hop distance between the probing monitor and the hijacker, will inevitably pass through the hijacker. For a sophisticated hijacker, this is a good opportunity to fabricate responses to these triggering messages to conceal its own identity. As a result, the AS paths mapped from such a fake *traceroute* results may contain erroneous ASes as well. It is easy to see that these "noises" only appear in the later portion of a path because the portion that is before the hijacker cannot be altered by the hijacker, – the ICMP triggering messages do not reach the hijacker. Hence if a node in a path is determined to be a fake node, we really do not need to consider any nodes beyond that point because this point must be already beyond the hacker's position in the path.

In the pre-processing part, we consider the duplicated appearances of AS nodes. If a node appears more than once in a path, any appearance beyond the first is considered fake. This is because real *traceroute* results should not contain loops.

5.2 Basic Algorithm

We denote the set of monitors that have detected the hijacking and reported their altered monitor-to-prefix paths by $\mathcal{M}$. For each monitor m_i within $\mathcal{M}$, there is an AS level monitor-to-prefix path P_i, either computed by pre-processing *traceroute* path or obtained directly from BGP routes. We define the neighborhood set of a specific path P_i, denoted as $\mathcal{N}(P_i)$, as the union of all path nodes and their one-hop neighbors. The target prefix' AS should be removed from all $\mathcal{N}(P_i)$. The reason is simple, – it is not the hijacker AS. Note that LOCK computes the neighborhood set based on AS topology inferred from RouteView [3] before the hijacking is detected, rather than real-time BGP data when the hijacking is ongoing. Though the hijacker can try to pollute the AS topology information before launching real hijacking attack on the victim prefix, the impact of such evasion is minimal on the neighborhood set computation because it is difficult for hijacker to "remove" an observed true link from the AS topology by announcing fake routes.

We are interested in the neighborhood sets of the AS paths instead of just the AS paths themselves because the hijacker may actually not show up in any of the AS paths if it manipulates *traceroute* results. However, even under this condition the ASes which are immediately before the hijacker along the paths are real. Thus, the union of all neighborhood sets of all reported AS paths, $\mathcal{H} = \bigcup_i \mathcal{N}(P_i)$, form our search space for the hijacker. We denote each node in this search space as a_k. The hijacker-locating algorithm is essentially a ranking algorithm which assigns each node in $\mathcal{H}$ a rank based on their suspicious level of being the hijacker.

The LOCK algorithm ranks each AS node $a_k \in \mathcal{H}$ based on two values, *covered count* $\mathcal{C}(a_k)$ and *total distance to monitors* $\mathcal{D}(a_k)$. The *covered count* is simply computed by counting a_k appearing in how many path neighborhood sets. For each neighborhood set $\mathcal{N}(P_i)$ that a_k is a member, we compute the distance between a_k and the monitor of the path m_i, $d(m_i, a_k)$. This distance equals to the AS-level hop count from m_i to a_k along the path P_i if a_k is on the path P_i. Otherwise, $d(m_i, a_k)$ equals to the distance from m_i to a_k's neighbor, who is both on P_i and the closest to m_i, plus 1. If a_k is not a member of a path neighborhood set $\mathcal{N}(P_i)$, the distance $d(m_i, a_k)$ is set to 0. The *total distance to monitors* equals to the summation of all $d(m_i, a_k)$.

Algorithm 2: The pseudo-code of locating algorithm

```
1  Initializing
2    set $\mathcal{H}$, $\mathcal{C}$, $\mathcal{D}$ empty;
3  Updating
4    foreach $m_i$ in the monitor set $\mathcal{M}$
5      foreach $a_k \in \mathcal{N}(P_i)$
6        if $a_k \in \mathcal{H}$
7          $\mathcal{D}(a_k)$ += $d(m_i, a_k)$;
8          $\mathcal{C}(a_k)$ += 1;
9        else
10         insert $a_k$ in $\mathcal{H}$ ;
11         $\mathcal{C}(a_k) = 0$;
12         $\mathcal{D}(a_k)$ = $d(m_i, a_k)$;
13 Ranking
14   sort $a_k \in \mathcal{H}$ by $\mathcal{C}(a_k)$;
15   for $a_k$ with the same value of $C(a_k)$;
16     sort $a_k$ by $\mathcal{D}(a_k)$;
```

After for each a_k in $\mathcal{H}$ both *covered count* $\mathcal{C}(a_k)$ and *total distance to monitors* $\mathcal{D}(a_k)$ are computed, we rank all nodes in $\mathcal{H}$ firstly based on their *covered count*. The greater the *covered count* a node a_k has, the higher it is ranked. Then for nodes having the same *covered count*, ties are broken by ranking them based on their *total distance to monitors*, –the lower the total distance, the higher the rank. If there are still ties, node ranks are determined randomly.

Hence, the final result of the locating algorithm is a list of nodes a_k, ordered based on how suspicious each node is being the hijacker. The most suspicious AS appears on the top of the list. The pseudo-code of the locating algorithm is shown in Algorithm 2.

The ranking algorithm described here may seem overly complicated for finding where the reported paths converge. However it is designed specifically to be robust against various measurement errors and possible hijacker countermeasures. One particular reason for this design is to reduce the effect of individual false paths. If a monitor-to-prefix path is changed due to reasons other than being hijacked and the monitor falsely assesses the situation as hijack, the path reported by this monitor may cause confusion on where the paths converge. Since it is difficult to distinguish this kind of paths beforehand, our algorithm has adopted the approach as described above to discredit the effect of these individual erroneous paths. For similar reasons, our ranking algorithm is robust against the IP-to-AS mapping errors if any.

Another reason for outputting an ordered list is that there are cases that hijacked paths converge before these paths reach the hijacker (*early converge*). This is more likely to happen when the hijacker is located far away from the Internet core where the connectivity is rich. In this case the hijacked paths may converge at an upstream provider of the hijacker in stead of the hijacker itself. Although as we will show later these hijacking scenarios typically have small impacts, in other words the portion of the Internet that is affected by such hijacks is small; still we wish to locate the hijacker. A list of suspects ranked by level of suspicion is well suited for addressing this issue.

5.3 Improvements

After the suspect list is computed, we can apply additional post-processing methods to further improve our results. The basic algorithm is very conservative in the way that $\mathcal{H}$ includes all possible candidates. Now we look into ways that $\mathcal{H}$ may be reduced. The hope is that with a trimmed suspect set to begin with, the locating algorithm can get more focused on the hijacker by increasing the rate that the most suspicious node on the list is the hijacker. Both improvements are designed to alleviate the early converge problem we mentioned before. Note that the improvements may exclude the real hijacker from the suspect set, but the evaluation (in Section 6.3.5) shows that chance is very small.

5.3.1 Improvement One: AS Relationship

In the basic algorithm, we have only taken AS topology into account. In other words, all topological neighbors of nodes on a reported AS path are added to the path's neighborhood set. In reality, not all physical connections between ASes are actively used for carrying traffic. In particular, some connections may be used only for traffic of one direction but not the other. This is largely due to profit-driven routing policies between different ISPs. Internet paths have been found to follow the "valley-free" property [10], i.e. after traversing a provider-to-customer edge or a peer edge, a path will not traverse another customer-to-provider path or another peer edge. If we constrain our suspect set using this AS relationship based property by removing the neighbors that do not follow the "Valley-free" property from the neighborhood set of each reported path, we are able to reduce the size of the neighborhood set and further on the suspect set $\mathcal{H}$.

One matter needs to be pointed out is that not all links on the reported paths are necessarily real due to the hijacker's countermeasures. Since we do not know what links are fabricated we should not trim the neighborhood sets too aggressively. We only perform this improvement on path links that we are reasonably certain that they are real. In particular, as we know that an attacker cannot forge path links that are before itself, thus we can reasonably consider that on each reported path the links that are before the node immediately before the most suspicious node are real, and the trimming is only done on neighbors of these links.

This AS relationship based improvement is incorporated into the basic algorithm in an iterative fashion. We first pre-compute AS relationship information using method proposed in [10]. Note that this is done offline and does not require any real time access to the control plane information because AS relationship rarely change over time. After each execution of the basic algorithm produces a ranked suspect list, we can assume that on each path from the path's reporting monitor to the node immediately before the most suspicious node, all AS paths are valid. Based on these valid links, we can further infer the valid link in each neighborhood set. When there is any change of neighborhood set, we run the locating algorithm again to update the suspicious list. The iteration will stop if there is no change of suspicious list.

5.3.2 Improvement Two: Excluding Innocent ASes

The second improvement focuses on removing nodes from the suspect set $\mathcal{H}$ of whose innocence we are reasonably certain. One group of these nodes are the ones that are on the reported paths that actually pass through the most suspicious node and before the most suspicious node. The reason for this exclusion is again that the attacker cannot forge the identity of these nodes.

The second group of the innocent nodes are selected based on the path disagreement test described in [45]. In path disagreement test, a reference point that is outside of the target prefix but topologically very close to the prefix is selected and the path from a monitor to this reference point and the path from same monitor to the target prefix are compared. If they differ significantly it is highly likely that the prefix has been hijacked. The high accuracy of this test leads us to believe that nodes on monitor-to-reference point paths are not likely to be the hijacker. They can be excluded from the suspect set.

The second improvement is again incorporated into the basic algorithm in an iterative fashion. After each execution of the basic algorithm, the suspect set is reduced by removing nodes of the two aforementioned innocent groups. Then basic algorithm is executed again using the reduced suspect set. The iteration is repeated until the basic suspect set is stable.

6 Evaluation

We implemented and deployed LOCK on PlanetLab [31]. This is a necessary step to show that LOCK is deployable in real world system. Also using the PlanetLab testbed, we evaluated the performance of LOCK based on measurements of the deployed LOCK system. In this section, we first present our measurement setup and evaluation methodology. Then we evaluate the performance of the monitor selection algorithm in LOCK, and the effectiveness of LOCK against against synthetic hijacks, reconstructed previously-known hijacking events, and real hijacking attacks launched by us.

6.1 Measurement Setup

6.1.1 Candidate Monitors

In our experiments, we first chose a number of geographically diversified PlanetLab [31] nodes as candidate network location monitors. We manually selected 73 PlanetLab nodes in 36 distinct ASes in different geographical regions. More specifically, relying on their DNS names, half of the nodes are in the U.S., covering both coasts and the middle. The other half were selected from other countries across multiple continents. Among these candidate monitors, a set of monitors were selected using the algorithm presented in Section 4 to monitor each target prefix.

6.1.2 Target Prefixes

We selected target prefixes from four different sources: (i) Multiple Origin ASes (MOAS) prefixes, (ii) Single Origin AS (SOAS) prefixes with large traffic volume, (iii) prefixes of popular Web sites, and (vi) prefixes of popular online social networks. To get prefixes from sources (i) and (ii), we first use BGP tables obtained from RouteViews [3] and RIPE [2] to identify the initial candidates of MOAS and SOAS prefixes. Then for each candidate prefix, we tried to identify a small number (up to 4) of live (*i.e.* responsive to *ping*) IP addresses. To avoid scanning the entire candidate prefixes for live IP addresses, we mainly used the prefixes' local DNS server IP addresses to represent the prefix. If we failed to verify any live IP address for a particular prefix, we discarded this prefix from our experiments. Using this method, we selected 253 MOAS prefixes. We also ranked all SOAS prefixes based on "popularity" (*i.e.* traffic volume observed at a Tier-1 ISP based on Netflow) of the prefix and selected top 200 prefixes with live local DNS server IP addresses.

We also selected prefixes that correspond to popular applications on the Internet: Web and online social networks. In particular, we selected the top 100 popular Web sites based on the Alex [5] ranking and obtain their IP addresses and corresponding prefixes. We also obtained IP addresses and prefixes of YouTube and 50 popular online social networks. Each of the selected online social networks has at least 1 million registered users in multiple countries. Combining prefixes from all above four sources, we have a total of 451 target prefixes.

6.1.3 Measurement Data Gathering

In our experiments, each monitor measures its paths to all selected IP addresses in all target prefixes via *traceroute*. We also measured paths from each monitor to reference points of target prefixes [45]. In addition, each monitor also measures its paths to other monitors. We obtain AS-level paths of above measured paths by mapping IP addresses to their ASes based on the IP-to-AS mapping published at iPlane [15].

The results presented here are based on monitoring data collected from March 20*th*, 2008 to April 20*th*, 2008. In particular, we measured each path (from a monitor to a target prefix) every 5 minutes.

In addition, we obtained the AS topology data during the same time period from [16]. We also used the AS relationship information captured for customer-to-provider and peer links over 6 month (from June 2007 to December 2007) using the inferring technique described in [24].

6.2 Evaluation Methodology

We evaluated LOCK based on three sets of prefix hijacking experiments: (i) synthetic prefix hijacking events based on Internet measurement data; (ii) reconstructed previously-known prefix hijacking events based on Internet measurement data; and (iii) prefix hijacking events launched by us on the Internet.

6.2.1 Simulating Synthetic Prefix Hijacking Events

We consider commonly used interdomain routing policies: "prefer customer routes" and "valley-free routing". In particular, an AS prefers routes announced from its customer ASes over those announced from its peer ASes, further over those announced from its provider ASes. These policies are driven by financial profit of ASes. If two routes have the same profit-based preference, then the shorter route (i.e., fewer AS hop count) is preferred. When the hijacker announces a fake prefix, we assume that it does this to all its neighbors (i.e. providers, peers, and customers) to maximize hijacking impact.

For each attack scenario, we simulated all three types of hijacking scenarios, namely imposture, interception, malicious, as shown in Figure 2 in Section 3. Each attack scenario is simulated as follows. In each attack scenario, we selected one PlanetLab node as the hijacker h and another PlanetLab node as the target prefix t. The hijacking is then observed from the monitors.

In the imposture scenario, the path from s to t will become the path from s to h if s is polluted by h's attack. Otherwise, the path from s to t remains the same as before the attack. This was repeated for all possible selections of h, t, and s, except for 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.

The setup for simulating interceptions and malicious scenarios is similar to that of the imposture scenario. In the interception scenario, the path from s to t will be the concatenation of paths from s to h and from h to t if s is polluted by h's attack. However, we exclude the cases that there is one or more common ASes between these two paths. This is because the hijacker h cannot successfully redirect the traffic back to the target prefix t, i.e., the interception hijack fails.

In the malicious scenario, the hijacker h has countermeasure against LOCK. The path from s to t will be the path from s to h (the AS of h will not show up) with a few random AS hops appended after h. The generation of these random AS hops is kind of tricky. If h generates different noisy tails for different monitors, these tails may not converge at all. In this case, it is easier for our locating algorithm to locate the hijacker. In our simulations, in anticipating that the hijacker may fill its replies to traceroute probes with fake identities, we replaced the node identities with random entries for all nodes that are farther than the hijacker (inclusive) in the paths resulted from running traceroute from different monitors.

6.2.2 Reconstructing Previously-Known Prefix Hijacking Events

We obtained the list of previously-known prefix hijacking events from the Internet Alert Registry [14]. IAR provides the network operator community with the up to date BGP (Border Gateway Protocol) routing security information. Its discussion forum [3] posts suspicious hijacking events. We chose 7 that had been verified and confirmed to be prefix hijacking events, including some famous victims such as YouTube and eBay, during a time period from 2006 to 2008.

We reconstructed these 7 hijacking events using the following method. First, we selected a traceroutable IP in each victim AS as the probing target t, and a traceroutable IP in each hijacker AS as the hijacker h. Then we collected the traceroute information from each monitoring site s to these targets t and hijackers h. The routing policy is based again on the profit driven model. Since we don't know what kind of behavior each hijacker took (imposture, interception or malicious), We conservatively assume that the hijacker will try to evade our measurement. So it follows the malicious scenario we mentioned before.

6.2.3 Launching Controlled Prefix Hijacking Events

We conducted controlled prefix hijacking experiments on the Internet using hosts under our control at four different sites, namely Cornell, Berkeley, Seattle, and Pittsburgh. Each host ran the Quagga software router and established eBGP sessions with different ISPs. Effectively, this allowed us to advertise our dedicated prefix (204.9.168.0/22) into the Internet through the BGP sessions. The idea behind the experiments was to use our prefix as the target prefix with one of the sites serving as the owner of the prefix and the other three sites (separately) serving as the geographically distributed attackers trying to hijack the prefix. More implementation details can be found in [6]. In our experiment, we focused on the imposture scenario. There were 12 hijacking cases by switching the role of each site. These attacks were launched according to a pre-configured schedule during period from May 2, 2008 to May 4, 2008.

6.2.4 Performance Metrics

LOCK identifies suspicious hijackers and ranks them based on their likelihood of being the true hijacker. The hijacker ranked at top one is most suspicious. We thus define the *top-n accuracy* of LOCK as the percentage of hijacking events that the true hijacker ranks as top n on the suspect list, where n is a parameter. We use this parameterized definition because different operators might have different preference. Some might prefer knowing just the most suspicious hijacker, in which top-1 accuracy is most important. Others might not mind learning a longer suspect list to increase the likelihood that the hijacker is included in the suspect list. We will later show that the top-2 accuracy is already very high.

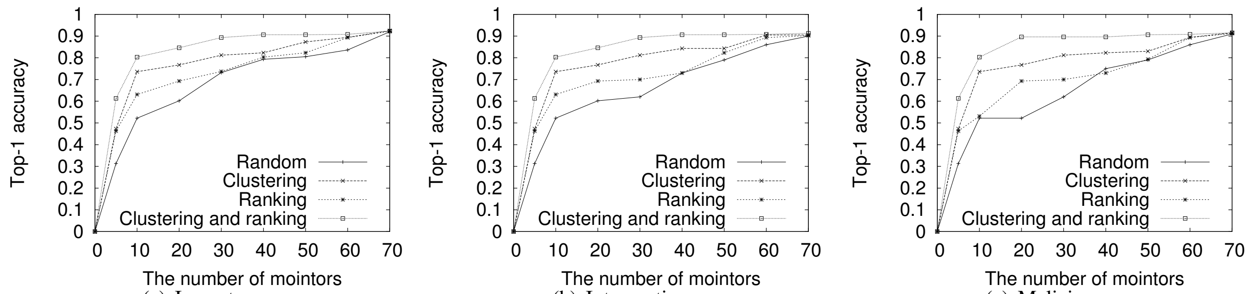
In addition, we define *impact* of a hijacker h as the fraction of the ASes from which the traffic to the target prefix t is hijacked to h, similar to what is done in [23]. We will then study the correlation between LOCK's locating accuracy of a given hijacker and the impact of its attack.

6.3 Evaluation on Synthetic Prefix Hijacking Events

In this section, we use the results of LOCK based on the data plane measurement to illustrate our findings.

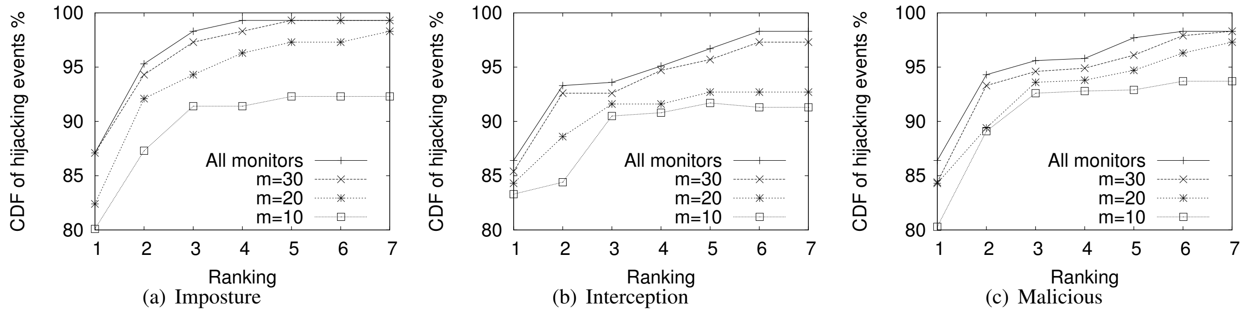
6.3.1 Monitor Selection

We compare the performance of the monitor selection algorithm (referred as *clustering and ranking*) proposed in Section 4 with the following three monitor selection algorithms: (i) *random:* randomly selecting m monitors



(a) Imposture (b) Interception (c) Malicious

Figure 3: Performance of monitor selection algorithms



(a) Imposture (b) Interception (c) Malicious

Figure 4: The CDF of the rank of hijackers in synthetic attacks

from all M candidates. (ii) *clustering:* dividing M monitors into clusters based on the clustering algorithm proposed in Section 4.1, then randomly selecting one monitor from each cluster; and (iii) *ranking:* ranking M monitors based on the ranking algorithm proposed in Section 4.2, then selecting the first m candidates.

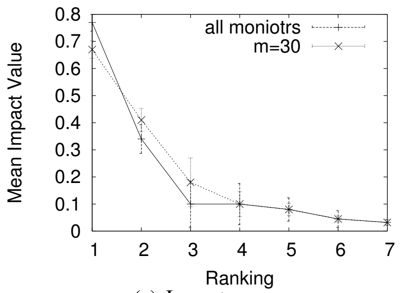
Figure 3 shows the top-1 accuracy of different monitor selection algorithms when varying the subsets of monitors. We focused on synthetic attacks since the dataset is much larger than previously-known hijacks and controlled real hijacks. We find that: (i) There is always a trade-off between the number of monitors selected and hijacker-locating accuracy. Note that even using all 73 monitors, the accuracy is less than 92%. It is not surprising because it is hard to detect the hijacking events which have small impact [23]. (ii) The *clustering and ranking* algorithm outperforms the rest. For example, for imposture attacks, selecting 10 monitors based on the ranking and clustering algorithm is enough for achieving 80% top-1 accuracy. This is only 1/3 of number of monitors needed to reach the same top-1 accuracy with either *ranking* or the *clustering* algorithm, or 1/6 if monitors are selected randomly. Hence in our experiments in the rest of the section, whenever we need to select monitors, we use the *clustering and ranking* algorithm, unless otherwise specified.

Moreover, we want to make sure that the monitor selection algorithm does not overload any monitors by assigning too many target prefixes to it for monitoring. For each target prefix we select $m = 30$ monitors from the total pool of $M = 73$ candidate monitors using the monitor selection algorithm described in Section 4. Individual monitor's work load is computed as the number of target prefixes assigned to it divided by the total number of target prefixes. Ideally, the average work load, which is the load each monitor gets if the monitoring tasks are evenly across all monitors equally instead of assigning prefixes to monitors that can monitor most effectively, is $m/M \approx 0.4$. As as comparison, we observe the real workload ranges from 0.3 to 0.55. In addition, only 4 monitors out of 73 have load above 0.5, which means that they monitor more than half of prefix targets.

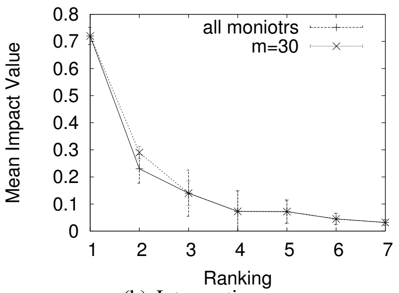
6.3.2 Effectiveness of Basic Algorithm

The evaluations of two different aspects of the effectiveness of the hijacker-locating algorithm are presented in this section. We show how well the ranked list captures the hijacker identity, as well as how well the ranked list reflects the impact of the hijack events.

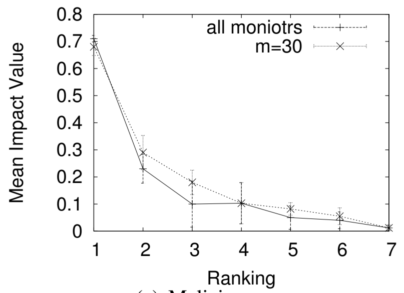
Figure 4 illustrates where the hijacker is ranked in the suspect list produced by the basic algorithm, for different number of monitors selected. Obviously, the higher the hijacker is ranked, the better the basic algorithm is. From this figure, we can see that:



(a) Imposture



(b) Interception



(c) Malicious

Figure 5: Correlating the impact with the ranking value

- More than 80% of the time, our basic algorithm pinpoints the hijacker by ranking it as *top 1* on the suspect list, regardless what kind of attack and with how many monitors, as long as more than the minimum number of 10 monitors.

- Because of the early convergence problem described in Section 5.2, the hijacker may not be ranked the first. Therefore as we look into not only the highest ranked node but even more nodes, the chance that the hijacker is included in this selective set increases. For example with 10 monitors, the chance that an imposture hijacker is found among the top three nodes is more than 94%, a 14% increase from only looking at the highest ranked suspect.

- The hijacker-locating algorithm performs best in imposture scenarios. The reason is that imposture paths are more likely to be straight without detouring.

- Obviously the more monitors we employ, the better the algorithm works. What is interesting is that seemingly by having $m = 30$ we have reached the point of diminishing return: having more than 30 monitors no longer improves the performance much.

Next, we study the relationship between the impact of a hijack event and where the hijacker is ranked in the suspect list. This shows another aspect of the quality of our hijacker-locating algorithm. That is, not only we want to locate hijackers, we especially want to locate the hijackers causing great damages. Figure 5 shows the ranking (x-axis) vs the median impact of all hijackers with the same ranking (Y-axis). All three plots in Figure 5 show that there is a positive relationship between the hijacker's rank and the impact of its hijack attack. In other words, the larger the impact caused by a hijacker, the more likely our locating algorithm will rank the hijacker high in the suspect list. This is mostly due to the fact that the early converge problems occur mostly at where hijacks have small impacts, near Internet edge.

6.3.3 Effectiveness of Improvements

Finally, we evaluate the quality of two improvements (I1 and I2) proposed in Section 5.3. In particular, we are not only interested in the increase in top-1 accuracy these improvements may bring, but also the false negative rate (FNR), which is the ratio that the improvements mistakenly exclude a hijacker from the suspect list.

Table 1 shows both sets of numbers for different kinds of attacks and different number of monitors. Different combinations of the basic algorithm and the improvements are shown in different rows of the table.

- I2 helps more. The reason is that for I1 we can only trust the path before converges. But for I2, we have more information provided by the reference point traceroute.

- When combining I1 and I2, the accuracy can be further improved. This is because the nodes that I1 and I2 remove from the suspect list are typically not the same.

- In general, LOCK (i.e., B+I1+I2) is able to pinpoint the prefix hijacker AS with an accuracy of over 91%, up to 94.3%.

- The false negative ratio introduced by improvements is relatively low. For example, when using all monitors we can improve the accuracy by more than 5% by applying both I1 and I2, while the false negative ratio resulted from applying the improvements is only 0.09%

6.3.4 Effectiveness on different AS-levels

We study the locating accuracy when the hijacker located in different level in the AS hierarchy. We classify AS nodes into three tiers: Tier-1 nodes, transit nodes, and

Table 1: The effectiveness of improvement

Algorithms	All monitors						m=30					
	Imposture		Interception		Malicious		Imposture		Interception		Malicious	
	Accuracy	FNR	Accuracy	FNR	Accuracy	FNR	Accuracy	FNR	Accuracy	FNR	Accuracy	FNR
B	88.7%	0.00%	86.3%	0.00%	85.4%	0.00%	86.2%	0.00%	84.7%	0.00%	83.5%	0.00%
B+I1	89.8%	0.03%	90.3%	0.17%	88.6%	0.14%	86.4%	0.05%	85.3%	0.14%	84.6%	0.11%
B+I2	91.3%	0.09%	93.1%	0.16%	90.4%	0.10%	90.7%	0.14%	90.6%	0.18%	88.3%	0.20%
B+I1+I2	94.2%	0.09%	94.3%	0.24%	93.1%	0.18%	92.4%	0.20%	91.4%	0.17%	91.8%	0.26%

Table 2: The effectiveness on different AS-levels

Category	Imposture		Interception		Malicious	
	Accuracy	FNR	Accuracy	FNR	Accuracy	FNR
All	92.4%	0.20%	91.4%	0.17%	91.8%	0.26%
Transit	97.6%	0.04%	96.3%	0.07%	94.8%	0.14%
Stub	90.2%	0.18%	90.1%	0.21%	90.4%	0.35%

Table 3: The effectiveness on prevention after locating

Methods	Initial	Stop the origin	Stop in Tier1
LOCK	23.43%	0.10%	2.31%
Simple Locating	23.43%	13.13%	21.90%

stub nodes like in [23]. [4] Our hijackers in planetlab belongs to transit nodes, or stub nodes. When using two improvements and 30 monitors, we compare the accuracy and false negative ration for these two classes, in Table 2. The hijackers on the higher level could be located more easily. The hijackers on the edge is relatively hard to locate. We can still achieve more than 90% accuracy.

6.3.5 Effectiveness of filtering after locating the hijacker

After locating the AS, the next step is to filter the fake AS announcement from it. We compare the average percentage of impacted (polluted) AS, before and after the locating and filtering either stop on the origin or on the Tier1 AS. As a comparison, we also select the last hop of AS of the observed paths as a hijacker (simple locating approach) then do the same filtering. They are under malicious case. Table 3 shows that Lock is more helpful than simple locating method to prevent hijacks.

6.3.6 Remarks

We have shown that LOCK performs well using monitor-to-prefix paths measured in the data plane. Similar observation would hold if control plane paths are used in LOCK. In the non-malicious cases, the monitor-to-prefix paths that observed in the control plane are the same as those observed in the data plane. However, in the malicious case, we have shown that the hijacker can employ more sophisticated evasion technique in the data plane than in the control plane. Therefore, our results shown in this section provide a lower bound of LOCK performance against malicious hijackers.

Table 4: Previously-Known prefix hijacking events

Victim AS	Hijacker AS	Date	#monitors
3691	6461	March 15, 2008	16
36561 (YouTube)	17557	February 24, 2008	9
11643 (eBay)	10139	November 30, 2007	7
4678	17606	January 15, 2007	8
7018	31604	January 13, 2007	13
1299	9930	September 7, 2006	5
701, 1239	23520	June 7, 2006	12

6.4 Evaluation on Previous-Known Attacks

We reconstructed 7 previously known prefix hijacking events. Table 4 shows the dates and ASes of the hijacker and the target prefix (i.e., the victim) of these events. By using all 73 monitors deployed on PlanetLab, LOCK is able to accurately locate the hijacker ASes as the top-1 suspects for all these hijacking events, i.e., the true hijackers are ranked first on the suspect lists. Using the monitor selection algorithm (clustering and ranking) presented in Section 4, we also identified the minimum set of monitors that were required by LOCK to accurately locate the hijacker in each of these previously-known events. The last column of Table 4 shows that all hijackers could be correctly located as top-1 suspects by using 16 or fewer monitors. A detailed investigation shows that these hijacks polluted majority of the monitors, resulting in LOCK's high locating accuracy.

6.5 Evaluation on Controlled Real Attacks

In this set of experiments, we launched real imposture attacks using four sites under our control. The schedule is shown in Table 5. During the experiments each LOCK

Table 5: Locating hijackers in real Internet attacks

Victim Site	Hijacker Site	Launch Time (EST)	Response Time (minutes)	Required monitors
Cornell	Berkeley	May 2 12:01:31	13	12
	Seattle	May 2 16:12:47	7	10
	Pittsburgh	May 2 17:34:39	9	9
Pittsburgh	Cornell	May 2 19:32:09	13	14
	Berkeley	May 2 22:50:25	11	15
	Seattle	May 3 02:26:26	12	15
Seattle	Cornell	May 3 11:20:42	9	8
	Pittsburgh	May 3 13:03:10	12	12
	Berkeley	May 3 19:16:16	8	18
Berkeley	Seattle	May 3 22:35:07	13	14
	Pittsburgh	May 4 00:01:01	12	16
	Cornell	May 4 11:19:20	11	10

monitor probed the target prefix 204.9.168.0/22 once every 5 minutes. For the purpose of this experiment, we used the detection scheme proposed in [45], which was able to detect all the attacks launched from the controlled sites. The hijackers in these experiments were "honest", i.e., no countermeasure was done by the hijackers. Thus we observed that LOCK locates the hijackers as top-1 suspects in all the real imposture attacks.

In this real Internet experiment, we were able to evaluate the response time of LOCK in addition to its accuracy. The response time is defined as the latency from the time the the attack is launched by the hijacker to the time that LOCK locates the hijacker. The response time highly depends on two major factors: the speed of propagation of invalid route advertisement and the probing rate employed by LOCK monitors. It usually takes up to a few minutes for a route advertisement to spread across the Internet. This is the latency that an attack takes before making full impact on the Internet. After a LOCK monitor is impacted by an attack, it may also take a few minutes for the monitor to detect and locate the hijacker because the monitor probes target prefixes periodically. There are also few minor factors that may affect the response time. For example, there can be a few seconds latency for LOCK monitors to get replies for each probe. However, they are neglected in our evaluation because they are orders of magnitude smaller than the above two major factors.

We record the timestamp each attack is launched from a control site and the timestamp LOCK locates the hijacker (i.e., that controlled site). Both of which are synchronized with a common reference time server. The response time is computed by taking the difference between the above two timestamps. If alternative detection scheme is used, the observed response time serves as a conservative upper bound of the latency that LOCK takes to locate the hijacker.

Table 5 shows the response time and minimum number of required monitors for locating these real prefix hijacking events. We observe that LOCK is able to locate the hijacker within $7 \sim 13$ minutes. Given that the probe frequency of LOCK monitors is 5 minutes, the results implies that it takes LOCK at most $2 \sim 3$ rounds of probes to detect and locate the hijacker. Moreover, all hijackers are correctly located as top-1 suspects by using 18 or fewer monitors.

7 Related Work

A number of solutions have been proposed to proactively defend against prefix hijacking. They can be categorized into two broad categories: crypto based and non-crypto based. Crypto based solutions, such as [4,8,13,19,27,35, 36], require BGP routers to sign and verify the origin AS and/or the AS path to detect and reject false routing messages. However, such solutions often require signature generation and verification which have significant impact on router performance. Non-crypto based proposals such as [11,18,32,37,44] require changing router softwares so that inter-AS queries are supported [11, 32], stable paths are more preferred [18, 37], or additional attributes are added into BGP updates to facilitate detection [44]. All the above proposals are not easily deployable because they all require changes in router software, router configuration, or network operations, and some also require public key infrastructures.

Recently, there has been increasing interest in solutions for reactive detection of prefix hijacking [6, 12, 21, 22, 26, 34, 36, 45] because such solutions use passive monitoring and thus are highly deployable. For example, [43, 45] monitor the data plane, [21, 22, 26, 34] monitor the control plane, and [6, 12, 36] monitor both control and data planes. LOCK is different from all these approaches because LOCK locates the hijacker AS for each prefix hijacking event, while the above approaches only focus on detecting a hijacking event without further revealing the location of the hijacker. In fact, LOCK can be used together with any of the above hijacking detection algorithm for identifying hijacker AS because the flexibility of LOCK on using either control plane or data plane information in locating hijacker.

Measurement-based solutions often require careful selection of monitors. In particular, LOCK selects monitors based on their likelihood of observing hijacking events, while [45] proposed an initial monitor selection algorithm to detect hijacks without further evaluation, and [23] tries to understand the impact of hijackers in different locations. In addition, there have been a number of studies [7,9,40] on the limitations of existing BGP monitoring systems (e.g. RouteView) and the impacts of monitor placement algorithms [29] for collecting BGP data for a boarder range of applications such as topology discovery, dynamic routing behavior discovery and network black hole discovery [20,41].

Finally, existing works [38, 39, 42] proposed to mitigating prefix hijacking by using an alternative routing path [38, 39], or by modifying AS_SET [42]. Though LOCK does not directly handle the mitigation of prefix hijacking events, LOCK can provide the hijacker location information required by these mitigation schemes.

8 Conclusion

In this paper, we propose a robust scheme named LOCK for locating the prefix hijacker ASes based on distributed AS path measurements. LOCK has several advantages: 1) LOCK is an unified scheme that locates hijackers in the same fashion across different types of prefix hijacking attacks; 2) LOCK is a distributed scheme with workload distributed among multiple monitors; 3) LOCK is a robust scheme because multiple monitors help improving locating accuracy and discounting individual errors; and 4) LOCK is a flexible scheme because it can use AS path measurement data obtained either from data-plane or from control-plane to locate the hijacker AS.

The performance of the LOCK scheme has been evaluated extensively through experiments in three kinds of settings: test topology constructed based on real Internet measurements, reconstructed known prefix hijack attacks, and controlled prefix hijack attacks conducted on the Internet. We have shown that the LOCK scheme is very accurate, highly effective, and rapid reacting.

Acknowledgement

Tongqing Qiu and Jun Xu are supported in part by NSF grants CNS-0519745, CNS-0626979, CNS-0716423, and CAREER Award ANI-023831.

References

[1] http://www.ripe.net/news/study-youtube-hijacking.html.

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

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

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

[5] Alexa. http://www.alexa.com/.

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

[7] Barford, P., Bestavros, A., Byers, J., and Crovella, M. On the marginal utility of network topology measurements. In *IMW '01* (New York, NY, USA, 2001), ACM, pp. 5–17.

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

[9] Cohen, R., and Raz, D. The Internet Dark Matter - on the Missing Links in the AS Connectivity Map. In *INFOCOM* (2006).

[10] Gao, L. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking* (2001).

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

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

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

[14] IAR. http://iar.cs.unm.edu/.

[15] iPlane. http://iplane.cs.washington.edu/.

[16] Internet topology collection. http://irl.cs.ucla.edu/topology/.

[17] Johnson, S. Hierarchical Clustering Schemes. In *Psychometrika* (1967).

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

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

[20] Kompella, R. R., Yates, J., Greenberg, A., and Snoeren, A. C. Detection and Localization of Network Black Holes. In *Proc. IEEE INFOCOM* (2007).

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

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

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

[24] Mao, Z. M., Qiu, L., Wang, J., and Zhang, Y. On AS-Level Path Inference. In *Proc. ACM SIGMETRICS* (2005).

[25] MAO, Z. M., REXFORD, J., WANG, J., AND KATZ, R. Towards an Accurate AS-level Traceroute Tool. In *Proc. ACM SIGCOMM* (2003).

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

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

[28] NORDSTROM, O., AND DOVROLIS, C. Beware of BGP Attacks. *ACM SIGCOMM Computer Communications Review (CCR)* (Apr. 2004).

[29] OLIVEIRA, R., LAD, M., ZHANG, B., PEI, D., MASSEY, D., AND ZHANG, L. Placing BGP Monitors in the Internet. UW Technical Report, 2006.

[30] OLIVEIRA, R., PEI, D., WILLINGER, W., ZHANG, B., AND ZHANG, L. In Search of the elusive Ground Truth: The Internet's AS-level Connectivity Structure. In *Proc. ACM SIGMETRICS* (2008).

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

[32] QIU, S. Y., MONROSE, F., TERZIS, A., AND MCDANIEL, P. D. Efficient Techniques for Detecting False Origin Advertisements in Inter-domain Routing. In *Proc. IEEE NPsec* (Nov. 2006).

[33] RAMACHANDRAN, A., AND FEAMSTER, N. Understanding the Network-Level Behavior of Spammers. In *Proceedings of ACM SIGCOMM* (2006).

[34] SIGANOS, G., AND FALOUTSOS, M. Neighborhood Watch for Internet Routing: Can We Improve the Robustness of Internet Routing Today? In *Proc. IEEE INFOCOM* (May 2007).

[35] SMITH, B. R., AND GARCIA-LUNA-ACEVES, J. J. Securing the Border Gateway Routing Protocol. In *Proc. Global Internet* (Nov. 1996).

[36] SUBRAMANIAN, L., ROTH, V., STOICA, I., SHENKER, S., AND KATZ, R. H. Listen and Whisper: Security Mechanisms for BGP. In *Proc. USENIX NSDI* (Mar. 2004).

[37] WANG, L., ZHAO, X., PEI, D., BUSH, R., MASSEY, D., MANKIN, A., WU, S., AND ZHANG, L. Protecting BGP Routes to Top Level DNS Servers. In *Proc. IEEE ICDCS* (2003).

[38] XU, W., AND REXFORD., J. Don't Secure Routing Protocols, Secure Data Delivery. In *Proc. ACM HotNets* (2006).

[39] XU, W., AND REXFORD., J. MIRO: multi-path interdomain routing. In *Proc. ACM SIGCOMM* (2006).

[40] ZHANG, B., LIU, R. A., MASSEY, D., AND ZHANG, L. Collecting the Internet AS-level Topology. *Computer Communication Review 35*, 1 (2004), 53–61.

[41] ZHANG, Y., ZHANG, Z., MAO, Z. M., HU, Y. C., , AND MAGGS, B. On the Impact of Route Monitor Selection. In *Proceedings of ACM IMC* (2007).

[42] ZHANG, Z., YANG, Y., HU, Y. C., AND MAO, Z. M. Practical Defenses Against BGP Prefix Hijacking. In *Proc. of CoNext* (Dec. 2007).

[43] ZHANG, Z., ZHANG, Y., HU, Y., MAO, Z., AND BUSH, R. iSPY: Detecting IP Prefix Hijacking on My Own. In *Proc. ACM SIGCOMM* (Aug. 2008).

[44] ZHAO, X., PEI, D., WANG, L., MASSEY, D., MANKIN, A., WU, S., AND ZHANG, L. Dection of Invalid Routing Announcement in the Internet. In *Proc. IEEE/IFIP DSN* (June 2002).

[45] ZHENG, C., JI, L., PEI, D., WANG, J., AND FRANCIS, P. A Light-Weight Distributed Scheme for Detecting IP Prefix Hijacks in Real-Time. In *Proc. ACM SIGCOMM* (Aug. 2007).

Notes

[1]Note that some vendor implementation does not check whether the neighbor has appended its own AS in the announcement , while some vendor implementation does check (in which this hijack does not succeed).

[2]The complexity is not a concern here because the number of clusters is relatively small comparing to traditional clustering problem.

[3]Disscussion form: `http://iar.cs.unm.edu/phpBB2/viewforum.php?f=2`

[4]To choose the set of Tier-1 nodes, we started with a well known list, and added a few high degree nodes that form a clique with the existing set. Nodes other than Tier-1s but provide transit service to other AS nodes, are classified as transit nodes, and the remainder of nodes are classified as stub nodes.