LOCK: Locating Countermeasure-Capable Prefix Hijackers

Tongqing Qiu*, Lusheng Ji†, Dan Pei†, Jia Wang†, Jun (Jim) Xu * and Hitesh Ballani‡
*Georgia Institute of Technology
†AT&T Labs – Research
‡Cornell University

***Abstract*—Prefix hijacking is known as one of the security 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 an attacker is critical for deciding at the earliest time the proper mitigation mechanisms to invoke to limit the impact of the attack and successfully stopping the attack.**

In this paper, we propose a robust scheme named LOCK, LOcating Countermeasure-capable hijacKers, for locating the prefix hijacker ASes based on distributed data-plane Internet measurements. LOCK locates each attacker AS by actively monitoring paths to the victim prefix from a small number of carefully selected monitors distributed on the Internet. More importantly, 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 the data-plane path before a packet reaches the hijacker, and that the data-plane 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 of LOCK against three sets of hijacking attacks: synthetic attacks, reconstructed previously known attacks, and controlled attacks on the Internet. Our evaluation results show that LOCK is able to pinpoint the prefix hijacker AS with an accuracy of over 90%.

I. Introduction

Prefix hijacking refers to a scenario in which a misconfigured or malicious BGP router originates a route to an IP prefix it does not own, is becoming an increasingly serious security problem in the Internet.

The Internet consists of tens of thousands of Autonomous Systems (ASes), each of which is an independently administrated domain. The routing information within an AS is maintained by Interior Gateway Protocols (IGPs) while the Border Gateway Protocol (BGP) is employed to maintain and exchange inter-domain routing information. However, BGP is designed based on the assumption that there is implicit trust among the participants in the system. The lack of adequate authentication schemes implies that the inter-domain routing infrastructure is incapable of preventing misbehaving routers from arbitrarily advertising routes for prefixes and/or fabricate 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 or even service outage, as well as serious security breach. A canonical example happened in 1997 when AS7007 announced prefixes of a large portion of the Internet and interrupted the reachability to these prefixes for hours [1]. Recently, the prefix of YouTube was hijacked by an AS in Pakistan for more than 2 hours [2].

There are two main approaches to solving the prefix hijacking problem. The first approach focuses on prefix hijacking prevention. Many solutions fall into the category [3]–[14]. These solutions, however, all require some changes to current routing infrastructures such as router software, router configurations, network operations, or even the introduction of public key infrastructures. Therefore generally speaking they are difficult to deploy.

The second approach aims at prefix hijack detection and service restoration. Solutions that fall into this group either attempt to monitor the state of Internet and detect hijacking events when they happen [15]–[20], or recover from prefix hijacking and restore services for the victim prefixes [21]. An apparent advantage of these methods is that they are compatible with existing routing infrastructures and therefore can be deployed incrementally on today's Internet.

Unfortunately, in practice, several of the known prefix hijack events were detected and recovered, and service to the victim prefixes was restored through human interaction and manual intervention rather than any of the aforementioned solutions. This is because a critical step in putting together these solutions into an automated prefix hijack detection and recovery system is still missing and that is how to locate hijackers after hijacking events are detected. Without this step recovery and restoration proposals can not help because they typically assume that the identities of the hijackers are known. However, as we will elaborate, locating hijackers is far from a trivial problem, especially considering the various countermeasures that hijackers may invoke.

In this paper, we present a scheme called LOCK, LOcating Countermeasure-capable hijacKers. It is a light-weight and deployable scheme for locating hijacker ASes automatically. The LOCK monitors are deployed at diverse locations in the Internet. For any specific target prefix, the LOCK selects an optimum set of monitors from all available monitors at LOCK's disposal after balancing factors such as how many monitors to choose and how well the hijacker may be located. These monitors collect and report *traceroute* information for

the monitor-to-target prefix paths either periodically or on-demand.

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. Second, our locating scheme relies only on real-time measurements taken from the data plane. Not requiring privileged access to live BGP advertisement data makes our approach easy to deploy and appealing to prospective prefix hijack monitoring service providers. Third, we propose a monitor selection algorithm to make the monitors work more efficiently. 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 more than 90% accuracy.

The rest of the paper is organized as follows. Section II provides background information on prefix hijacking. Section III provides an overview of the framework of our LOCK scheme. Then we describe detailed monitoring and locating methodologies in Section IV and Section V respectively. Section VI evaluates the performance of the LOCK scheme. Section VII briefly surveys related works before Section VIII concludes the paper.

II. 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 to 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 may drop all traffic addressed to the victim prefix that it receives and effectively perform a denial-of-service attack against the prefix owner [22]. It can also redirect traffic to an incorrect destination and use this for a phishing attack [22]. More over, it can also use this technique to spread spams [23]. Thus, 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".

Based on how the attacker deals with the hijacked traffic, we classify prefix hijacks[1] 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 [24]. In addition, the characteristics of different hijack types are different, which often affect how different types of attacks are detected.

There have been a number of existing approaches proposed for detecting prefix hijacks. They utilize either information in BGP updates collected from control plane [15]–[17], [19], path or end-to-end probing information collected from data plane [20], or both [3], [18], [24]. 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 III-A. To locate a hijacker, the LOCK scheme only needs to know which prefix is hijacked. Therefore it does not depend on any particular detection method as long as the method is able to provide victim identity.

However, because LOCK follows a data-plane approach, it is beneficial to briefly introduce the methods used in [20]. [20] proposed a light-weight distributed scheme for detecting IP prefix hijacks in data-plane. The design of the detection scheme is based on two key observations made on the Internet, hop count stability and AS path similarity. This detection system continuously tests these two assertions in a distributed and light-weighted manner, and uses any departure from this stability and similarity as the trigger for target prefix hijack alarms. It can detect the hijacking events with low false negative and false positive ratios. We should also point out that because both LOCK and [20] require a distributed multi-vantage point monitoring system, and because both LOCK and the path disagreement test of [20] use *traceroute* to gather monitor-to-target prefix path information, a joint detection-location system based on LOCK and [20] has the potential of being very efficient.

III. Framework

In this section, we present an overview of key ideas of the hijacker locating algorithm in LOCK.

A. Challenges

Currently, the most commonly used hijacker-locating approach is to look at the origin ASes of the victim prefix, which

[1]We use the term "hijack" to refer to all three kinds of prefix hijack attacks unless otherwise specified.

can typically be identified by inspecting BGP announcements. When a victim prefix is found to be hijacked, any newly appearing origin AS would be considered as the hijacker AS. However, this simple approach will fail if the hijacker is capable of engaging countermeasures. For instance such a hijacker can fabricate an AS-path and include this fake AS-path in its announcements when it first announces the victim prefix to conceal the identity of the true origin AS, – itself. Thus, it is not trivial to locate a countermeasure-capable hijacker.

To locate hijacker in control plane, a fair number of tapping points must be installed at diversely distributed locations to collect BGP announcements in real time.AS paths from different vantage points to the same prefix are then computed and the intersection points of these AS paths are inspected to identify the hijacker. Currently ISPs many have limited resources to deploy such tapping points diversely enough. This entry barrier limits the opportunity to only very resourceful parties. There are some public available control plane data like Route Views. But it cannot cover all targets, and the updates are not real-time enough.

Our previous work [20] proposed a data-plane prefix hijacking monitoring framework to overcome the limitations of control-plane solutions. Following the same rationale, this paper takes that solution one step further by trying to locate the hijackers using only data-plane mechanisms as well.

It is actually more difficult to locate hijackers in the data plane than in the control plane. The reason is that in the control plane at least the hijacker AS will appear in the hijacked paths regardless what kind of countermeasures the hijacker may poll. However, in the data plane the hijacker may be completely invisible from path probes. 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 may do many things to interfere the path probing. 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. Blackholing and imposture hijacks are the typical examples of such interferences. With this kind of false router level path, how real any derived AS level knowledge can be is also in question.

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 need to design a more effective algorithm for locating these hijackers.

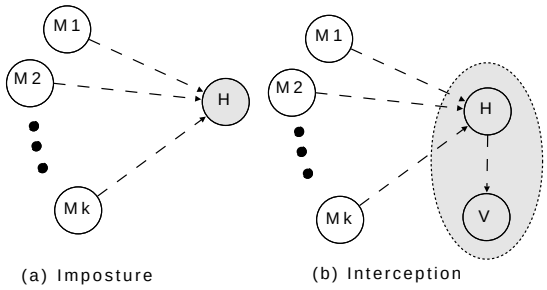


Fig. 1. "Honest" hijackers

B. Locating Countermeasure-capable Hijackers

The basic idea of LOCK is based on two key observations. The first observation is that the data-plane paths from multiple vantage points to a hijacked victim prefix, if the paths are hijacked, "converge" to the hijacker AS, independent of hijacker scenarios and whether the hijacker is countermeasure-capable. This is easy to understand because the goal of prefix hijacking is to redirect traffics destined to the victim prefix to the hijacker. The second observation is that the hijacker can not manipulate data-plane measurements that do not go through the hijacker's AS. More specifically, the *traceroute* responses from the routers between a vantage point and the router before the first router under hijacker's control can be trusted since they are not subject to hijacker's manipulation. Of course, LOCK does not know beforehand which routers are controlled by the hijacker. Instead, based on above two key observations, LOCK tries to combine the data-plane path information collected from multiple monitors and narrow down to a small number of candidate hijackers.

We use Figure 1 to illustrate the first observation. For the ease of presentation, for now we suppose hijacker H is "honest", i.e., it does not modify the *traceroute* information, and worry about the hijacker's countermeasures later. Suppose the routes of monitors M_1 M_2, ... M_k are polluted by the hijacker H, which means that the traffic sent by these monitors to the victim prefix will go through the hijacker H. In such a case, the hijacker AS H appears on all of these polluted paths. In other words, H should be among the "convergence points" of all these paths. By convergence points, we mean the ASes that show up in all the polluted paths. In Figure 1, the convergence points of the polluted paths are within the shaded area surrounded by the dashed eclipse. Moreover, it is intuitive that if the set of monitors are topologically diverse enough, the hijacker should be the first convergence point or among the first few convergence points.

Unfortunately, this basic "convergence points" approach does not directly work for blackholing or imposture attacks, or other hijacker countermeasures. In these cases, the hijacker

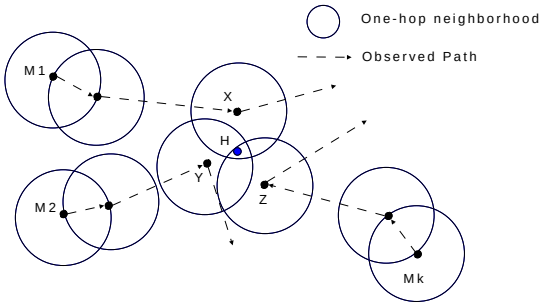


Fig. 2. Malicious hijackers, more general case

AS may not show up at all, leave alone being one of the "convergence points". More importantly, the hijacker may even generates invalid "tails" to trick the monitors. Figure 2 illustrates the situation.

To handle the case where hijacker AS does not show up on the AS-level traceroute path, LOCK seeks a solution from the second key observations we mentioned earlier: the *traceroute* results should be reliable for the portion of a path before the path enters the hijacker AS. In other words, the paths from all the polluted monitors to the victim prefix must include the upstream AS neighbors of the hijacker AS, with respect to the monitors. So if we can identify these upstream neighbors of the hijacker AS, the hijacker must be within the intersection of neighbor sets of the hijacker's neighbors. And chances are the size of the intersection set is very small if the monitors have diversified locations. For example, in Figure 2, ideally suppose we know ASes X, Y, and Z (which are on the polluted paths from monitors M_1, M_2, and M_k, respectively) are the upstream neighbors of the hijacker. We can then infer that the hijacker H must be within the intersection of the neighbor sets of X, Y, and Z. Of course in reality LOCK does not know beforehand which ASes are the 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 must be included in this neighborhood set.

For reasons that we will explain in the Section V, instead of using the neighborhood set of an arbitrary path the 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 V.

C. LOCK Scheme Overview

LOCK consists of three key steps: (i) For each target prefix, we select a number of monitors from a set of candidate monitors. (ii) Each monitor runs *traceroute* to each target prefix, either periodically or on-demand when a hijack detection system detects a hijack to the target prefix. (iii) The *traceroute* results from multiple monitors are analyzed to locate the hijackers. Details of the monitor selection algorithm and locating algorithm will be presented in Section IV and Section V, respectively.

LOCK's hijacker-locating scheme has a number of advantages. First of all, LOCK only collects live network information from data plane by running *traceroute* from a set of carefully chosen monitors to each target prefix. No live BGP feed is needed for the locating scheme, making it possible for those who only have data-plane access to deploy LOCK.

Another advantage of LOCK is that it is independent of hijacker detection system. It can work with any control-plane, data-plane, or hybrid detection system. The data-plane probing in LOCK can be run periodically for each target prefix, in parallel with a detection system. After the detection system detects a hijack of the victim prefix, LOCK is then consulted for hijacker location. Alternatively, LOCK's *traceroute* can be triggered on-demand by a detection system once a hijack is detected. In either case, as long as each monitor for the victim prefix has finished a *traceroute* to it after the hijack polluted the routes, the hijacker location result becomes available. This highlights another advantage of LOCK: it locates hijacker in a short period time, usually within a few minutes, which is much more quickly than any locating system which includes human factor.

Moreover, LOCK has one unified hijacker-locating algorithm, which works for all different hijacking scenarios (blackholing, imposture, or interception), with either "honest" or "dishonest" hijacker. This simplicity is important to the users of the system since they may not know what kind of hijacking scenarios they are facing. Solutions that have different algorithms for differen hijack types may have to test against all possible hijack types, while with LOCK the same approach works for all situations..

LOCK is light-weight in terms of monitoring overhead. This advantage comes from three factors. First, 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. Second, our locating scheme is naturally distributed. The probing load to a number of monitors are distributed diversely across the Internet. Third, the advanced monitor selection scheme can help us to reduce the number of monitors, so that the probing overhead can be further reduced.

Finally, LOCK is robust against errors for two reasons. First, the diversified locations of multiple monitors can improve the chance of catching hijacking events and locating the hijacker. Second, the way we produce the ranked suspect list is a majority rule type of approach, robust against individual errors.

D. Running LOCK with data-plane hijack detection system

While our LOCK system can work with any hijack detection system, running it together with a data-plane approach does offer some advantages. This is because the existing data-plane monitors and measurement data can also be (re)used by LOCK to locate the hijacker. For example, if we use the scheme in [20] as the detection system and LOCK as the locating system, LOCK can simply reuse the *traceroute* results from the detection system, and no extra measurements are needed. This results in less monitoring overhead and quicker hijacker-locating.

IV. 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 the improved accuracy gained by each additional monitor can gradually diminish as the number of monitors increases. 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 as monitors. 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 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.

A. 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 monitor selection problem as a hierarchical clustering problem. Such problems have well-known algorithms, such as [25], 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.

B. 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 [24].

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. 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 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 [24] for detailed analysis). Our ranking algorithm is shown in Algorithm 1.

V. Hijacker-Locating Algorithm

A. Pre-Processing

When a prefix is hijacked, a portion of the Internet will experience the hijack. Traffic originated from this portion

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

Algorithm 1: Ranking monitors in each cluster

```
foreach monitor i in the cluster
    if provider-route R[i] = 300;
    elseif peer-route R[i] = 200;
    else R[i] = 100;
    R[i] += D(i, t); /* add the AS-level distance */
```

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 [20] help separate real hijack induced path changes from changes caused by other non-hijack reasons.

The most common tool for acquiring IP forwarding path 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 [26] 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 [24], [27]. 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 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.

B. 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, resulted from *traceroute* and pre-processing. 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.

In this paper 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 $d(m_i, a_k)$.

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.

Algorithm 2: The pseudo-code of locating algorithm

```
Initializing
  set H, C, D empty;
Updating
  foreach m_i in the monitor set M
    foreach a_k ∈ N(P_i)
      if a_k ∈ H
        D(a_k) += d(m_i, a_k);
        C(a_k) += 1;
      else
        insert a_k in H ;
        C(a_k) = 0;
        D(a_k) = d(m_i, a_k);
Ranking
  sort a_k ∈ H by C(a_k);
  for a_k with the same value of C(a_k);
    sort a_k by D(a_k);
```

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.

C. 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.

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 [28], 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. 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.

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 [20]. 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.

VI. Evaluation

We implemented and deployed LOCK on PlanetLab [29].It is worth pointing out that LOCK can be deployed on any network nodes, not limited to the planetlab nodes. Even the end user can deploy such system using P2P fashion. We use Planetlab because it is relatively easy to control. We then 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 controlled real attacks.

A. Measurement Setup

1) Candidate Monitors: In our experiments, we choose a number of PlanetLab [29] nodes as candidate network location monitors. These candidate monitors are selected to ensure geographical diversity. We manually select 73 PlanetLab nodes in 36 distinct ASes at different geographical regions. More specifically, relying on the DNS name, we select half of US nodes, which covers east, west coasts and the middle area; and half from other different countries, which covers different continents. For each target prefix, a set of monitors will be selected among these candidate monitors using the algorithm presented in Section IV.

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 Route-Views [30] and RIPE [31] to identify the initial candidates of MOAS and SOAS prefixes. Then for each candidate prefix, we try 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 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. Using this method, we have 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 top 100 popular Web sites based on the Alex [32] ranking and obtain their IP addresses and corresponding prefixes. We also obtained IP addresses and prefixes of YouTube and 50 popular online social networks[3]. 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.

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 [20]. 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 [26].

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 [33]. We also used the AS relationship information captured customer-to-provider and peer links over 6 month (from June 2007 to December 2007) using the inferring technology described in [34].

B. 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) controlled prefix hijacking events on the Internet.

1) Simulating Synthetic Prefix Hijacking Events: We consider commonly used interdomain routing policies: "prefer customer routes" and "valley-free routing". In particular, an AS will prefer routes announced from its customer ASes than those announced from its peer ASes than 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.

In our evaluation, we simulated three hijacking scenarios based on the analysis in Section III-B: two honest scenarios (imposture and interception) and a malicious scenario. In each attack scenario, we selected one PlanetLab node as the monitor s, another PlanetLab node as the hijacker h, which attempts to hijack a target prefix t.

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, s, and t, except for cases where t's AS is on the AS path from s to

[3]The candidate list of online social networks is from http://en.wikipedia.org/wiki/List_of_social_networking_websites.

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 order to play an adversary of the locating algorithm, we simply randomly generate one tail and use it for all monitors. In this way, *covered counts* for these random AS should be high, making it harder to locate the real hijacker.

2) Reconstructing Previously-Known Prefix Hijacking Events: We obtained the list of previously-known prefix hijacking events from the Internet Alert Registry [35]. IAR provides the network operator community with the up to date BGP (Border Gateway Protocol) routing security information. Its discussion forum [4] posts suspicious hijacking events. We choose 7 events that have been verified, including some famous victims such as YouTube and eBay, during time period from 2006 to 2008.

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

3) Launching Controlled Prefix Hijacking Events: We conducted controlled prefix hijacking events on the Internet using hosts at four different sites, namely Cornell, Berkeley, Seattle, and Pittsburgh. Each host runs the Quagga software router and establishes 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 [24]. In our experiment, we focused on the imposture scenario. There are 12 hijacking cases by switching role of each site. These attacks are launched according to a pre-configured schedule during period from May 2, 2008 to May 4, 2008.

[4]http://iar.cs.unm.edu/phpBB2/viewforum.php?f=2&sid=e2e2dfedbbd77a560d0710c14a0cf5ff

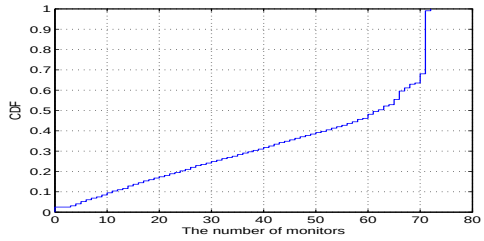


Fig. 3. The CDF of impacted monitors

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 [36]. We will then study the correlation between LOCK's locating accuracy of a given hijacker and the impact of its attack.

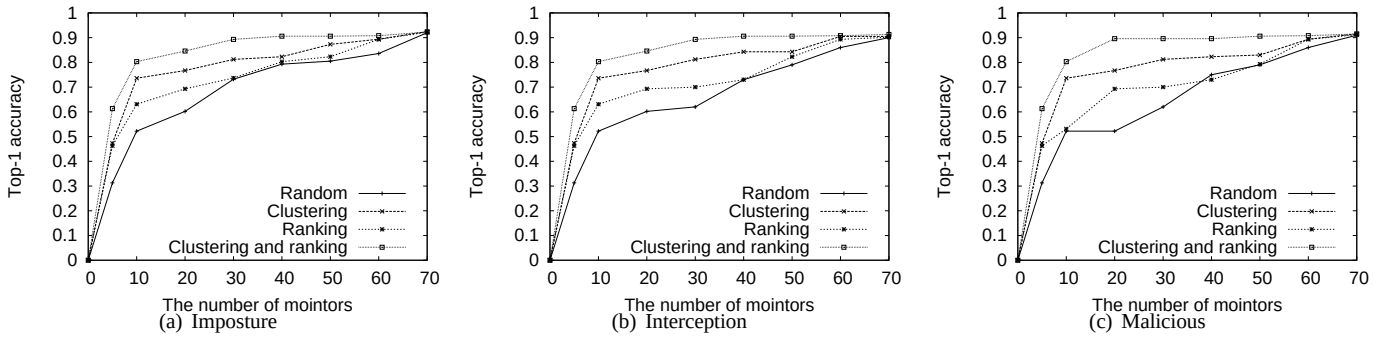
C. Monitor Selection

Not only the number of monitors affects the locating quality, the locations of the monitors and how they are distributed on the Internet also matter. In this sub-section, we investigate the effectiveness of various monitor selection algorithms.

1) Visibility of Candidate Monitors: Before evaluating the monitor selection algorithm, we first investigate exactly how many monitors can detect each hijack event. That is, for each hijack, how many monitor-to-prefix paths would be altered to go through the hijacker? This statistics offers important insight into what kind of hijacker-locating quality can be expected from LOCK because how well the locating algorithm performs is related to how many monitors can observe the hijack.

From Figure 3 we can see that more than 90% of the hijack events are observed by more than 10 monitors. This is very encouraging because generally the more monitors can observe a particular hijack event, the more likely the algorithm can locate the hijacker.

2) Performance of Monitor Selection Algorithms: We compare the performance of the monitor selection algorithm (referred as *clustering and ranking*) proposed in Section IV with



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

Fig. 4. Performance of monitor selection algorithms

the following three monitor selection algorithms: (i) *random:* randomly selecting m monitors from all M candidates; (ii) *clustering:* dividing M monitors into clusters based on the clustering algorithm proposed in Section IV-A, then randomly selecting one monitor from each cluster; and (iii) *ranking:* ranking M monitors based on the ranking algorithm proposed in Section IV-B, then selecting the first m candidates.

Figure 4 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; and (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. Figure 5 shows the work load on different monitors after 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 IV. Individual monitor's work load is computed as the number of target prefixes assigned to it divided by the total number of target prefixes. 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 and plotted as the horizontal baseline in Figure 5. As we can see, although some monitors may get more workload than others, only four monitors out of 73 have load above 0.5, which means that they monitor more than half of prefix targets, and the load the busiest monitor get is still below 0.55.

D. Synthetic Prefix hijacking events

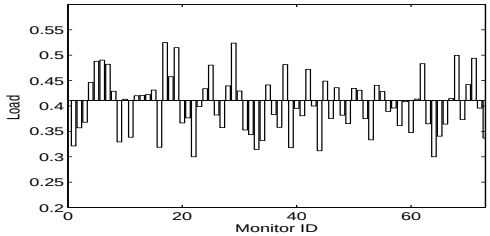


Fig. 5. Load distribution of monitors

1) 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 6 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:

- 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 V-B, 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 7 shows the ranking (x-axis) vs the median impact of all hijackers with the same ranking (Y-axis). All three plots in Figure 7 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.

2) Effectiveness of Improvements: Finally, we evaluate the quality of two improvements (I1 and I2) proposed in Section V-C. 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 I 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%.
- 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%

E. Effectiveness on different AS-levels

We study the locating accuracy when the hijacke located in different level in the AS hierarchy. We classify AS nodes into three tiers: Tier-1 nodes, transit nodes, and stub nodes like in [36]. [5] 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 II. The hijackers on the higher level could be loacted more easily. The hijackers on the edge is relatively hard to locate. We can still achieve more than 90% accuracy.

[5]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.

TABLE II
THE EFFECTIVENESS ON DIFFERENT AS-LEVELS

Categority	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 III
THE EFFECTIVENESS ON PREVENTION AFTER LOCATING

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

TABLE IV
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

F. Effectiveness of filtering after locating the hijacker

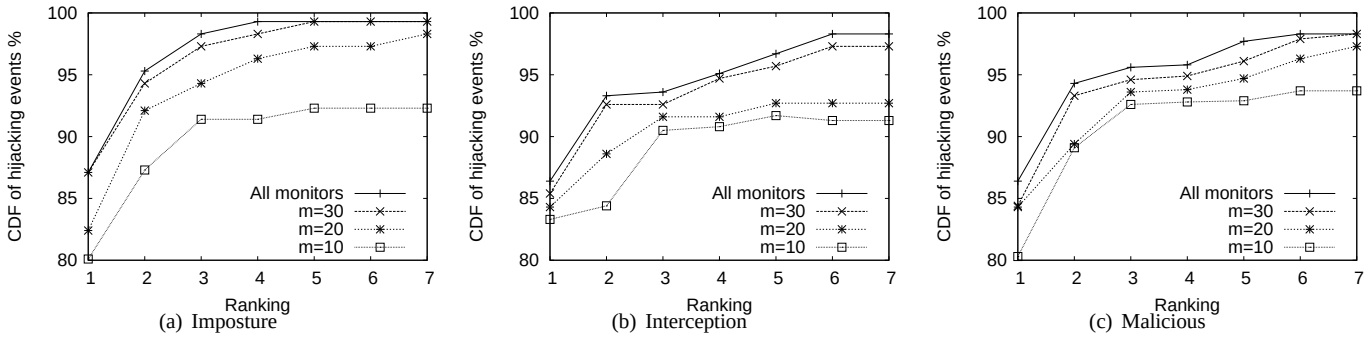
After locating the AS, the next step is to filter the fake AS annoucement from it. We compare the average percentage of impacted (polluted) AS, before and after the locating and filtering either stop on the origin or on the Tire1 AS. As a comparsion, we also select the last hop of AS of the observed paths as a hijacker (current approach) then do the same filtering. They are under malicious case. Table III shows that Lock is more helpful than manual method to prevent hijacks.

G. Evaluation on Previous-Known Attacks

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

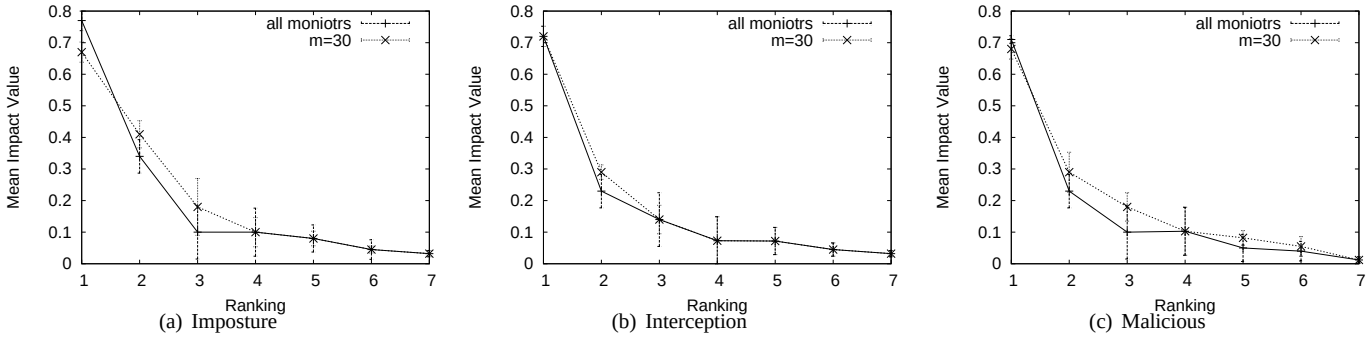
H. Evaluation on Controlled Real Attacks

In our experiments, we launched real imposture attacks using four controlled sites one by one. The schedule is shown in Table V. Each LOCK monitor probes the target prefix 204.9.168.0/22 once every 5 minutes. For the purpose of this experiment, we use the detection scheme proposed in [20], which was able to detect all the attacks launched from the controlled sites. The hijackers in these experiments are honest,



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

Fig. 6. The CDF of the rank of hijackers in synthetic attacks



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

Fig. 7. Correlating the impact with the ranking value

TABLE I
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%

i.e., no countermeasure is done by the hijackers. Thus we observe that LOCK locates the hijackers as top-1 suspects in all the real imposture attacks.

In this real Internet experiment, we are 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 locate 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 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). 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 V 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.

VII. 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 [3]–[9], require BGP routers

TABLE V
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

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 [10]–[14] require changing router softwares so that inter-AS queries are supported [10], [14], stable paths are more preferred [11], [13], or additional attributes are added into BGP updates to facilitate detection [12]. 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 [3], [15]–[20], [24] because such solutions use passive monitoring and thus are highly deployable. For example, [20] monitors the data plane, [15]–[17], [19] monitor the control plane [15]–[17], [19], and [3], [18], [24] monitor both control and data planes. The main limitation of the methods relying on control plane monitoring is that they either require the provision of live BGP feeds or cannot achieve real-time detection using historical BGP data. Similar to [20], LOCK utilizes information collected mostly from the data plane though LOCK aims at locating hijackers while [20] aims at detecting hijacking events.

Measurement-based solutions often require careful selection of monitors. In particular, LOCK selects monitors based on their likelihood of observing hijacking events, while [20] proposed an initial monitor selection algorithm to detect hijacks without further evaluation, and [36] tries to understand the impact of hijackers in different locations. In addition, there have been a number of studies [37]–[39] on the limitations of existing BGP monitoring systems (e.g. RouteView) and the impacts of monitor placement algorithms [40] for collecting BGP data for a boarder range of applications such as topology discovery, dynamic routing behavior discovery, etc [41].

Finally, existing works [21], [42], [43] proposed to mitigating prefix hijacking by using an alternative routing path [42], [43], or by modifying AS_SET [21]. Though LOCK does not directly handle the mitigation of prefix hijacking events, LOCK can provide the hijacker location information required by these mitigation schemes.

VIII. CONCLUSION

In this paper, we propose a robust scheme named LOCK, Locating Countermeasure-capable hijacKers, for locating the prefix hijacker ASes based on distributed data-plane measurements.

The LOCK scheme is motivated by two cornerstone observations. Firstly regardless of what type of prefix hijacking, hijacked data-plane paths "converge" around the hijacker AS. Secondly even the hijacker may employ countermeasures to interfere data-plane path probing processes; it can not affect probing results until the probing messages reach the hijacker. In other words, the hijackers cannot manipulate the portion of any measurement derived data-plane path that is before the hijacker.

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 and probing traffic distributed to different regions of the Internet; 3) LOCK is a robust scheme because multiple monitors help improving locating accuracy and discounting individual errors; and 4) LOCK is a data-plane scheme that does not require any modification to existing protocols and network infrastructure.

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.

It is interesting to consider the combination of data plane and control plane information when locating the hijacker.

REFERENCES

[1] "Wow, AS7007!" http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html.

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

[3] 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.

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

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

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

[16] "RIPE myASn System," http://www.ris.ripe.net/myasn.html.

[17] 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.

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

[19] 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.

[20] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A Light-Weight Distributed Scheme for Detecting IP Prefix Hijacks in Real-Time," in *Proc. ACM SIGCOMM*, Aug. 2007.

[21] Z. Zhang, Y. Yang, Y. C. Hu, and Z. M. Mao, "Practical Defenses Against BGP Prefix Hijackin," in *Proc. of CoNext*, Dec. 2007.

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

[23] A. Ramachandran and N. Feamster, "Understanding the Network-Level Behavior of Spammers," in *Proceedings of ACM SIGCOMM*, 2006.

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

[25] S. Johnson, "Hierarchical Clustering Schemes," in *Psychometrika*, 1967.

[26] "iPlane," http://iplane.cs.washington.edu/.

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

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

[29] "PlanetLab," http://www.planet-lab.org.

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

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

[32] "Alexa," http://www.alexa.com/.

[33] "Internet Topology Collection," http://irl.cs.ucla.edu/topology.

[34] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang, "On AS-Level Path Inference," in *Proc. ACM SIGMETRICS*, 2005.

[35] "IAR," http://iar.cs.unm.edu/.

[36] 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.

[37] P. Barford, A. Bestavros, J. Byers, and M. Crovella, "On the marginal utility of network topology measurements," in *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. New York, NY, USA: ACM, 2001, pp. 5–17.

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

[39] B. Zhang, R. A. Liu, D. Massey, and L. Zhang, "Collecting the Internet AS-level Topology," *Computer Communication Review*, vol. 35, no. 1, pp. 53–61, 2004.

[40] R. Oliveira, M. Lad, B. Zhang, D. Pei, D. Massey, and L. Zhang, "Placing BGP Monitors in the Internet," UW Technical Report, 2006.

[41] Y. Zhang, Z. Zhang, Z. M. Mao, Y. C. Hu, , and B. Maggs, "On the Impact of Route Monitor Selection," in *Proceedings of ACM Internet Measurement Conference (IMC)*, 2007.

[42] W. Xu and J. Rexford., "MIRO: multi-path interdomain routing," in *Proc. ACM SIGCOMM*, 2006.

[43] ——, "Don't Secure Routing Protocols, Secure Data Delivery," in *Proc. Workshop on Hot Topics in Networks*, 2006.