PHAS: A Prefix Hijack Alert System [*]

Mohit Lad [†]
mohit@cs.ucla.edu

Dan Massey [‡]
massey@cs.colostate.edu

Dan Pei [§]
peidan@research.att.com

Yiguo Wu [†]
yiguowu@cs.ucla.edu

Beichuan Zhang [¶]
bzhang@cs.arizona.edu

Lixia Zhang [†]
lixia@cs.ucla.edu

Abstract

In a BGP prefix hijacking event, a router originates a route to a prefix, but does not provide data delivery to the actual prefix. Prefix hijacking events have been widely reported and are a serious problem in the Internet. This paper presents a new Prefix Hijack Alert System (PHAS). PHAS is a real-time notification system that alerts prefix owners when their BGP origin changes. By providing reliable and timely notification of origin AS changes, PHAS allows prefix owners to quickly and easily detect prefix hijacking events and take prompt action to address the problem. We illustrate the effectiveness of PHAS and evaluate its overhead using BGP logs collected from RouteViews. PHAS is light-weight, easy to implement, and readily deployable. In addition to protecting against false BGP origins, the PHAS concept can be extended to detect prefix hijacking events that involve announcing more specific prefixes or modifying the last hop in the path.

1 Introduction

The Internet relies on the Border Gateway Protocol (BGP)[16] to convey routing information. However, if BGP provides incorrect routing information, packets may never reach the intended destination, and may even be misdirected to malicious destinations. The inability to ensure the integrity and correctness of routing information leads to many known vulnerabilities in BGP [12]. This paper considers a simple and common vulnerability called prefix hijacking. In a common prefix hijacking event, an Autonomous System (AS) originates a route for an address space, termed as prefix, but does not provide data delivery for that prefix. In other words, an AS reports "use me to reach prefix p", but does not actually provide data delivery for prefix p. For example, on December 24, 2004, AS 9121 incorrectly originated routes to 106,089 prefixes, almost 70% of all the prefixes. BGP routers throughout the Internet selected the route originating from AS 9121 as the best path to some or all of these prefixes. Traffic for these prefixes was then forwarded to AS 9121, who then essentially dropped the packets, affecting thousands of organizations [13]. When a prefix is hijacked, sensitive data from unsuspecting users could easily fall into the wrong hands, resulting in serious security and privacy breaches. A recent study has also found that spammers hijack BGP prefixes to send spam mail [14]. Thus, prefix hijacking is real operational concern in the Internet, and securing Internet routing against prefix hijacking is an important problem.

Simply detecting the occurrence of a prefix hijack event is an essential, but difficult task. Large-scale events where an AS mistakenly hijacks thousands of prefixes may be detected relatively quickly due to their size and impact. For example, in the AS 9121 event described above, thousands of prefixes from different origins, suddenly changed to origin AS 9121, a clear indication of prefix hijack. But smaller scale errors and intentional attacks can be much more difficult to detect. For example, suppose a malicious AS originates a false path to only one prefix, 131.179.0.0/16 (UCLA). Some BGP routers will accept the new false path while others may continue to use the correct path originated by UCLA. An origin change for a single prefix is a common occurrence and is unlikely to trigger alarm. As we will show later in the paper, there are quite a few origin AS changes during a typical day and most of these changes are valid. A prefix may change its origin AS at any time due to contrac-

[*]This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No N66001-04-1-8926 and by National Science Fundation(NSF) under Contract No ANI-0221453. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA or NSF.

[†]Computer Science Department, UCLA

[‡]Computer Science Department, Colorado State University

[§]AT&T Labs–Research

[¶]Computer Science Department, University of Arizona tual arrangement, multi-homing, traffic engineering, and a host of other factors. Only the origin itself (UCLA in our example) could easily and accurately distinguish between a legitimate origin change and a prefix hijack [4]. The legitimate origin is best able to identify this type of prefix hijack, but it has very little information about the BGP routes taken by others to its own prefix. In this case, UCLA may notice a drop in traffic and/or reports of connectivity problems, but there are numerous potential causes for this. Even if UCLA suspected a prefix hijacking attack, UCLA's local data can only confirm that it has correctly announced its own route. To determine if others are incorrectly announcing a route to UCLA, the UCLA administrators would need data from other remote sites.

One of the goals behind the existing BGP monitoring projects such as Oregon RouteViews and RIPE RRC is to provide network operators with a remote view of their own prefixes. Through establishing BGP peering with operational routers, the RouteViews and RIPE RRC projects collect routing data from a few hundred BGP routers around the globe placed in critical exchange points, tier-1 ISPs, and so forth.[1] These BGP data collectors obtain information in real time, which can be used to quickly detect prefix hijacking and identify the source of the problem. For example, a prefix hijack event occurred on January 22, 2006 and affected close to 80 prefixes including a financial organization. Within a few seconds of the event occurrence, RouteViews data collector received update messages from several of its BGP peers indicating a new origin to the prefix of the financial organization. If the prefix owner had received this data, it could have immediately detected the prefix hijacking and could have quickly taken corrective measures using operator channels. However, prefix owners do not have any way to easily access the data. The current BGP monitors collect vast amounts of data and dump the raw data, unsorted, onto the disk. It is impractical to assume that all the prefix owners would be able to download this the data and then extract the information about their own prefixes, let alone doing so in real-time.

In this paper, we build on the premise that the prefix owner is the only one who can accurately distinguish between legitimate changes and prefix hijacking events, and propose a scalable system for providing prefix owners with timely and reliable notifications of potential prefix hijacks. During a prefix hijack, the notification itself may reach the hijacker instead of the prefix owner, and thus the prefix owner would not be informed of the ongoing hijack. To increase the chances of notification delivery, we use a multi-path delivery mechanism using the existing email infrastructure to increase the chances of notification delivery. Our design is readily deployable and easy to use. Once our system has detected the problem, the owner can then take necessary actions, including soliciting help through operator channels like North American Network Operators Group (NANOG) mailing lists, and the NSP-Security mailing lists to either resolve the problem with the hijacker or its upstream ISPs.

The remainder of the paper is organized as follows. Section 2 presents some basics about routing and prefix hijacks. Section 3 presents our system design in detail. Notification generation, notification delivery, and local notification filtering are presented in Section 4, Section 5, and Section 6, respectively. Section 7 evaluates our design. Section 8 shows how the detection capability of our system can be extended to handle other forms of prefix hijacking. Section 9 reviews related work and Section 10 concludes the paper.

2 Background

The Internet consists of a large number of networks called Autonomous Systems (AS), and BGP (Border Gateway Protocol [15]) is the protocol used to exchange routing information between these ASes. Each AS is represented by a unique numeric ID and destinations are in the form of prefixes, where each prefix represents a network space. For example, a prefix 131.179.96.0/24 represents a network space of 2^8 addresses belonging to AS 52 (UCLA). Authorities such as ARIN and RIPE assign prefixes to organizations, who then become the owner of the prefixes.

As a path vector routing protocol, BGP lists the entire AS path to reach a destination prefix in its routing updates. In Figure 1(a), AS 52 (UCLA) is the owner of prefix 131.179.0.0/16 and announces this prefix to its upstream service providers. The AS announcing a prefix to the rest of the Internet, is called the origin AS of the prefix. In this example, AS 52 is the origin AS of prefix 131.179.0.0/16.[2] An AS receiving a path to reach a prefix may choose to propagate the path to some or all of its neighbors. An AS intending to propagate a received path, prepends its own AS ID to the path before sending the announcement to its neighbors. Therefore, in Figure 1(a), AS A has an AS path of (A, P, Q, R, 52) for prefix 131.179.0.0/16 and AS B has an AS path of (B, R, 52). When multiple prefixes cover the same address, the longest prefix match rule is used to forward the traffic. Thus, if a BGP routing table contains paths to reach prefix 131.179.0.0/24 as well as 131.179.0.0/16, then a

[1]Admittedly a few hundred routers represent only a small fraction of the overall Internet. A prefix hijack that affects only a small local region may not be observed by any of the current BGP monitors. In a separate project, we are studying the optimal BGP monitor placement problem, however those results are beyond the scope of this paper.

[2]Sometimes the owner of a prefix might not run BGP and its provider AS serves as the prefix origin.

packet destined to 131.179.0.128 would choose the path to reach 131.179.0.0/24.

2.1 Prefix Hijacking

In a prefix hijack event, the announced path to the prefix cannot actually be used to deliver data to the prefix. In some parts of the Internet, the false path replaces the authentic route to the prefix and traffic that follows the false path will eventually be dropped or delivered to someone who is pretending to be the legitimate destination. In other words, the traffic sent along the false path has been hijacked. We term the AS injecting false information as an attacker AS, and the AS that owns the route as a victim AS. For example, in Figure 1b, AS 110 announces 131.179.0.0/16, while the true origin for this prefix should be AS 52. It can be seen in this example, that AS 110 successfully effected a hijack, since AS A decided to pick the route to AS 110 instead of AS 52. In this case, AS 52 is the victim, AS 110 is the attacker and any traffic sent by AS A is delivered to AS 110 rather than the legitimate origin. Note that AS 52 may see a drop in its overall traffic volume, but variations in traffic load are the norm for most networks and AS 52 may be completely unaware that hijack event is occurring.

An attacker AS can hijack a prefix in various ways such as falsely announcing itself as the origin for a prefix (as discussed in the example above), falsely modifying some portion of the path other than origin, or falsely announcing a more specific prefix. Our presentation of PHAS is particularly concerned with the first case case where the origin AS is not valid. Section 8 discusses how the PHAS concept can be extended to handle path modifications adjacent to the origin AS and announcement of more specific prefixes.

3 System Design

Our basic approach is to examine BGP routing data collected at RouteViews (or RIPE, or any other BGP collectors), and provide real time notifications of any potential prefix hijacking to the prefix owner in a reliable way. In particular, we should immediately notify the prefix owner anytime a new origin AS is associated with their prefix. At a potentially slower rate, the prefix owner should be notified when an origin AS is no longer used to announce its prefix. The net result is that the prefix owner is able to track the set of AS numbers that originate its prefix. Presumably, the prefix owner also knows which AS numbers are allowed to serve as origins and can thus detect any false origins, as well as know when the false origins have stopped announcing its prefix.

More formally, we define an *origin set* for each prefix and track changes of this origin set. Existing BGP monitoring projects such as RouteViews and RIPE, peer with a few hundred BGP routers around the globe and collect BGP updates in real-time. Each monitored BGP router, or monitor in short, reports its best path to a prefix P and the last hop in this path is an origin AS for P. We define the origin set $O_{SET}(P, t)$ for a prefix P as the union of all the origins seen by all the monitors for that prefix at time t. For example, on January 22, 2006 before 8:31 hours GMT, all RouteViews monitors reported paths ending in AS 19758 for prefix 65.173.134.0/24, and thus for time $t <$ 8:31 on 01/22/2006, $O_{SET}(65.173.134.0/24, t) = \{19758\}$. When the prefix was hijacked at 8:31AM, some monitors switched to paths ending with AS 27506 and thus for the time t = 8:31 on 1/22/2006, $O_{SET}(65.173.134.0/24, t) = \{19758, 27506\}$. Our objective is to immediately notify the owner of 65.173.134.0/24 of this new origin set, and the owner could then work to resolve this issue with the offending AS 27506 or its upstream providers. Later, when the origin AS 27506 would not be seen as announcing this prefix anymore, we would like to send a notification to the prefix owner indicating that the origin set $O_{SET}(65.173.134.0/24, t) = \{19758\}$, so that the prefix owner also knows that the problem has been resolved.

Our design consists of the following four components.

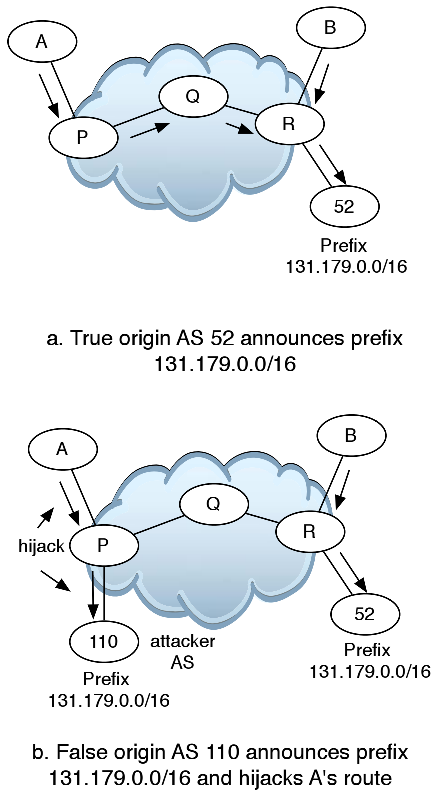
A
B
Q
P
R
52
Prefix 131.179.0.0/16
a. True origin AS 52 announces prefix 131.179.0.0/16
A
B
Q
hijack
P
R
52
110
attacker AS
Prefix 131.179.0.0/16
Prefix 131.179.0.0/16
b. False origin AS 110 announces prefix 131.179.0.0/16 and hijacks A's route

Figure 1: Example of prefix hijack

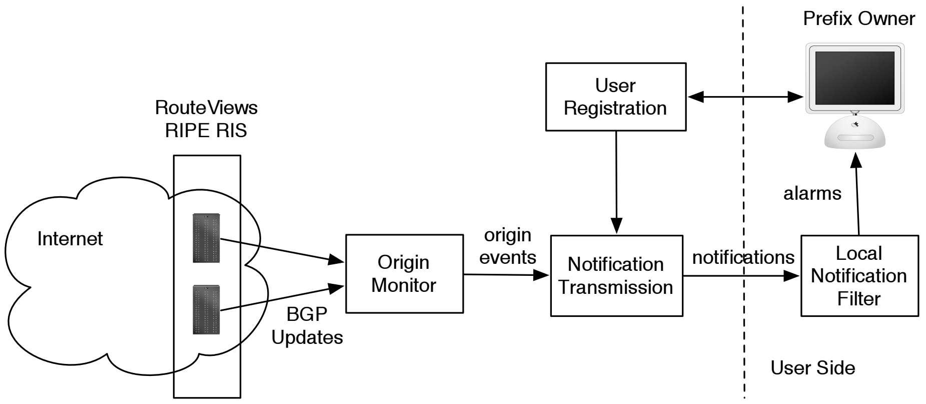

Figure 2: Components of PHAS

1. *User Registration:* All prefix owners who are interested in using our system need to register with the PHAS server and provide contact email addresses. PHAS aims to provide a web-based registration service, similar to the standard mailing list registration process. Each new user opens an account by his/her email address and a password via a secure HTTPS session. This action is sent to the email address for confirmation. Once confirmed, the registration is committed, and any later change to the account is done via HTTPS and requires the password. The registration specifies which prefixes are of interest and each registrant is strongly encouraged to submit multiple email addresses hosted by different email systems (such as a GMail address), to maximize the chance of email reception in face of prefix hijacking. Ideally, only the legitimate owner of a prefix should register, but verifying the correct contact address for each prefix is a challenging problem in its own right with no immediately deployable solution. In PHAS, an attacker may register and falsely claim to be the prefix owner. However, this action does not cancel the registration by the legitimate owner and all notifications are based on publicly available data so the attacker gains no new information by successfully registering.

2. *Origin Set Monitoring:* Using the BGP monitor data, PHAS maintains a current origin set for each registered prefix. If there is a change to this origin set, an origin event is generated.To control the number of origin events for prefixes with frequent origin changes, we use a time-window based mechanism to reduce the *repeated reporting* of the origin changes but still guarantee the immediate notification for any new origin announced for a prefix. We increase the duration of this window for prefixes that report lot of origin changes even after the default time window is used. The window duration is decreased if the number of origin event reduces. This adaptive window scheme is central to ensuring the system scales from the perspective of the origin set monitoring and also limits the number of false positives sent to the prefix owners. It is discussed further in Section 4.

3. *Notification Transmission:* Once an origin event is generated, PHAS decides whether the origin event translates to a notification message to be sent. For this, it checks the user registration to see if there are email addresses registered for the prefix involved. However, the seemingly simple task of sending a notification message, could be difficult in face of prefix hijack. For example, when the route to UCLA has been hijacked, email from PHAS to noc@ucla.edu may follow the hijacked route and never reach the intended receiver. To protect against this case, we strongly recommended two practices for prefix owners in order to set up "multiple diverse paths" for email delivery. First, in registering with our system, prefix owners should provide multiple email accounts on different email servers that are topologically diverse. Second, prefix owners should have Internet access via multiple prefixes. ISPs often have multiple prefixes of their own. For one that only owns a single prefix, a backup plan like a dial-up Internet access account is recommended. With the combination of multiple email addresses and multiple prefixes for Internet access, prefix owners can achieve a high success rate of notification delivery even in face of prefix hijack. All notification messages are also signed by PHAS server, whose public key is well-known. More details on the notification scheme will be discussed in Section 5.

4. *Local Notification Filter:* Although the notifications could be sent directly to network administrators, our

design assumes an automated processing of the received notifications. Tasks such as verifying the message is properly signed, checking whether periodic notifications has been received, and so forth are better handled by an automated receiver. In addition, many prefixes have multiple legitimate origins and thus not every change in the origin set is necessarily an attack that should be reported to the local network administrators. To make the system more user friendly, we provide a local filter program for processing the notification email. The local filter manages the external email addresses, checks any change in origin against a locally configured set of valid origins, and only reports an alarm to administrator when an unexpected origin change occurs. Local administrator can easily customize the filter program or even provide their own filter program. By incorporating a local filter, all the legitimate origin changes are simply screened out by the filter and only notices requiring human intervention are reported to the network operator. Local notification filter is discussed in more detail in Section 6.

Figure 2 shows the four components in our design and the interaction between them. Note how the origin events translate to notifications and finally to alarms.

4 Origin Change Detection

PHAS detects changes in BGP prefix origins and sends notification messages to registered prefix owners. For traffic engineering purposes, some networks may change their prefix origins frequently, which may trigger a large volume of notification messages if we want to keep track of every change. The main challenge in the system design is how to notify the owner in a timely manner while not being overwhelmed by the volume of messages.

4.1 Instantaneous Origin Changes

We first consider a simple scheme (Algorithm 1) that maintains an *origin set* for each prefix and sends a notification whenever the origin set changes. It takes input from a BGP monitoring project such as Route Views or RIPE. Let $\{M_1, M_2, ..., M_i, ..., M_N\}$ denote the set of N BGP routers providing data. By observing the BGP updates sent by router M_i, we can determine M_i's current route to prefix P. If M_i has a route to P at time t, $origin(M_i, P, t)$ denotes the origin AS of P in this route. If M_i has no route to P at time t, $origin(M_i, P, t)$ is empty. The *origin set* for prefix P at time t is defined as $O_{SET}(P, t) = \cup_{i=1}^{N} origin(M_i, P, t)$. In other words, the instantaneous origin set is simply the union of the origins currently used by any of the monitors to reach this prefix. As updates from M_i arrive, $origin(M_i, P, t)$ may change and thus the origin set may change as well. Whenever the origin set changes, we say an *origin event* is triggered.

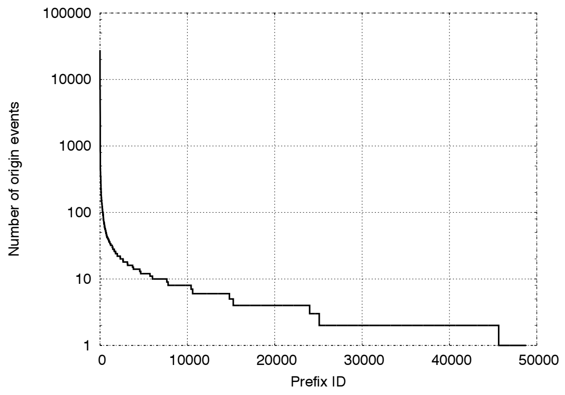

Figure 3: Origin events per prefix - December 2005

Algorithm 1: Instantaneous Origin Change

```
Initialize $origin(M_i, P, t_0)$ using the initial routing table of $M_i$ at time $t_0$;
$O_{SET}(P, t_0) = \cup_{i=1}^{N} origin(M_i, P, t_0)$;
if update for prefix $P$ at time $t$ from router $M_i$ is an announcement then
    $origin(M_i, P, t)$ = the last AS in the announced path;
else
    $origin(M_i, P, t) = \{\}$;
$O_{SET}(P, t) = \cup_{i=1}^{N} origin(M_i, P, t)$;
if $O_{SET}(P, t) \neq O_{SET}(P, t-1)$ then
    $origin\_gain = O_{SET}(P, t) - O_{SET}(P, t-1)$;
    $origin\_loss = O_{SET}(P, t-1) - O_{SET}(P, t)$;
    send $[O_{SET}(P, t), origin\_gain, origin\_loss]$ to prefix owner;
```

To study the algorithm behavior, we used data for the month of December 2005, from the RouteViews collector at the Oregon Internet Exchange. This BGP data collector peers with 42 operational routers from around the globe and thus the origin set is the union of the origin ASes seen by these 42 peers. The number of prefixes involved is close to 170,000. Algorithm 1 generated 511,513 origin events involving 48,768 prefixes during December 2005. Thus, close to 30% of the prefixes had one or more origin set changes. Figure 3 shows the distribution of the number of origin events per prefix (prefixes with no origin events are not plotted).

As the figure shows, some prefixes generated a large number of origin events. In Algorithm 1, even when the same origin leaves the set and comes back again

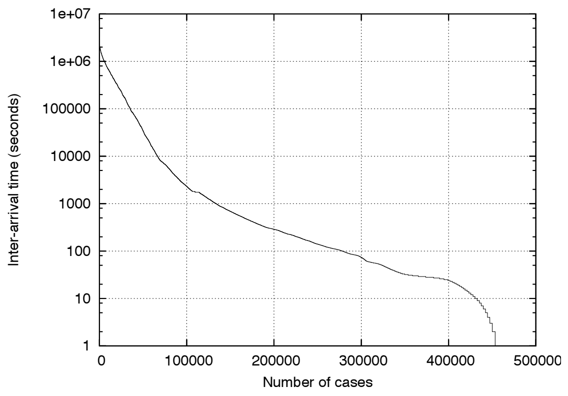

Figure 4: Inter-arrival time between origin events for a prefix for December 2005

on a repeated basis, each appearance and each disappearance triggers an origin event. For example, prefix 207.135.82.64/26 generated 5747 origin events during December 2005, simply due to the fact that its origin set switched frequently between $\{2828, 65000\}$, $\{2828\}$, and $\{65000\}$. Since some prefixes have unstable connectivity to the Internet, repeated withdrawal and announcement sequence causes the origin to frequently leave and join the set, resulting in repeated origin events. In order to detect prefix hijacking events, it is essential to immediately notify the owner when a new origin appears. However, reporting oscillations between already reported origins, as in this particular example, can be reduced.

4.2 Windowed Origin Changes

We now introduce the notion of windowed origin set. We can mask off repeated and frequent origin changes by reporting observed origin set over some time window, instead of reporting instantaneous origin set changes. Figure 4 plots the inter-arrival time between origin events. From the figure we can see that the inter-arrival time is less than 1000 seconds in close to 75% of the cases.

Let $O_{SET}(P,t,k)$ denote the set of all the origins for prefix P observed over the last k time units. In other words, this windowed origin set consists of all the origins for P that were observed by at least one router M_i during the time $[t-k,t]$. More formally, define $origin(M_i,P,t,k) = \cup_{i=t-k}^{t} origin(M_i,P,t)$ and $O_{SET}(P,k,t) = \cup_{i=1}^{N} origin(M_i,P,t,k)$. The definition includes the last k units at each time and thus provides a continuously moving window over which the origins of P are recorded. The algorithm to detect origin changes with a moving window is the same as Algorithm 1, except that we now have to include the time window k and only send origin events when $O_{SET}(P,t-1,k) \neq O_{SET}(P,t,k)$.

It is important to note that this revised algorithm only reduces the number of *repeated origin events*. The prefix owner will be immediately notified whenever a *new* (potentially false) origin appears for the first time during the last k time units. Suppose router M_i is the first to observe a new origin O for prefix P. If this new announcement first appears at time t, $Origin(M_i,P,t,k) = O$ and thus $O \in O_{SET}(P,t,k)$. Since M_i is the first to observe this origin, it must also be the case that $O \notin O_{SET}(P,t-1,k)$. Thus $O_{SET}(P,t-1,k) \neq O_{SET}(P,t,k)$ and an origin event is triggered at time t, i.e., as soon as the new origin appears. This feature guarantees timely detection and notification of potential prefix hijacking.

However, the addition of a time window does delay the notification of origin-loss events. Suppose origin O was in fact a prefix hijacking attempt. As discussed above, the prefix owner is immediately notified when O first appears. Assume as a result of this fast notification, the owner took actions and quickly resolved the attack. Let M_j denote the last monitored router to remove O from its routing table at time t_{end}. Although O has been removed from the routing tables, it will not be removed from M_j's origin set until time $t_{end}+k$. Thus O is also not removed from $Origin(M_i,P,t,k)$ until time $t_{end}+k$. The net result is that the prefix owner is not notified that O has been removed until k time units after O has vanished from the routing system.

4.3 Adaptive Window Size

Our objective is to reduce the number of repeated origin events for prefixes with frequent origin changes, but not penalize well-behaved prefixes by delaying reports that an origin has been removed. We start with a base time window of one hour. This masks transient changes for most prefixes, at a cost of delaying notification of origin loss events by one hour. However, some prefixes still generate a large number of notification messages even with the one hour window. Increasing the window size can further limit the number of repeated origin events for these prefixes but at the cost of further delaying origin-loss events for other prefixes. Rather than attempt to assign a uniform time window for all prefixes all the time, we introduce an adaptive window resizing scheme for each prefix. Essentially, prefixes that generate a large number of messages will be penalized by large window size, while other prefixes still use small window size.

Initially, each prefix starts with a penalty value of $penalty(P) = 0$ and a time window of one hour. Anytime a notification is generated for this prefix, $penalty(P)$ is increased by 0.5. The penalty value decays exponentially over time and the rate of decay is determined by a half-life parameter. We currently use a

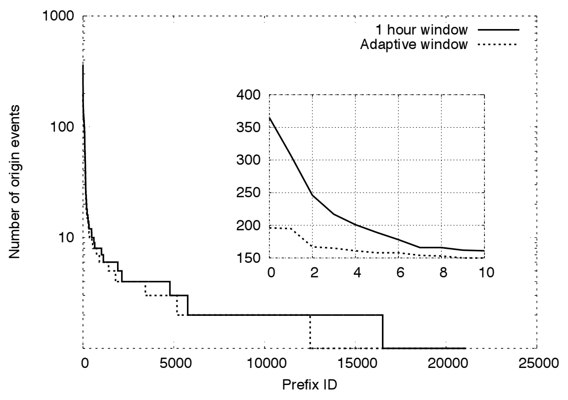

Figure 5: Distribution of origin events per prefix using adaptive window

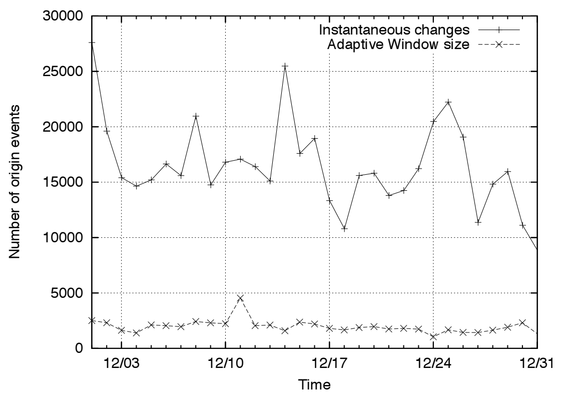

Figure 6: Comparison of origin events per day using instantaneous and adaptive window

half-life of 2 hours[3]. The size of the prefix's time window is set to $2^{\lfloor penalty(P) \rfloor}$ hours. In other words, a prefix with $penalty(P) < 1$ uses a time window of $2^0 = 1$ hour; a prefix with a $penalty(P)$ in the range $[1, 2)$ uses a time window of $2^1 = 2$ hours; a prefix with $penalty(P)$ in range $[2, 3)$ uses a time window of $2^2 = 4$ hours; and so forth.

Figure 5 shows the distribution of origin events generated using this adaptive window. For comparison, we also show the distribution using a fixed window size of 1 hour and show a zoomed in portion of the plot for the top 10 most active prefixes. Figure 6 shows the number of origin events generated per day using adaptive windows with a default as 1 hour along with the number of origin events using instantaneous origin changes for comparison. The introduction of the adaptive window reduces the number of origin events due to unstable prefixes, while still ensuring that any newly announced origin is immediately reported to the prefix owner. Prefixes that experience large number of origin changes would experience a longer delay before being notified of origin loss events, but would still receive immediate notification when a new origin appears.

5 Notification Delivery

Once a notification message is generated, it is delivered to the prefix owner's registered mailboxes through email. We choose email for delivery, since it is a ubiquitous delivery method on the Internet and uses TCP, which provides reliable data transfer. The email body is signed by the monitor to ensure its integrity. There are two types of messages: event-driven notifications and periodic refreshes. The event-driven notifications are triggered by origin set changes, and the email contains corresponding origin gains or losses. For example prefix 60.253.48.0/24, the notification messages look like the following:

<TYPE=gain, seqnum=1, GMT-TIME=20041221 12:52:33, PREFIX=60.253.48.0/24, NEW-SET={23918 31050 29257}, ORIGIN-GAINED=29257>

<TYPE=loss, seqnum=2, GMT-TIME=20041221 13:52:49, PREFIX=60.253.48.0/24, NEW-SET={29257 31050}, ORIGIN-LOST=23918>

The periodic notification is sent at fixed time interval (1 day by default), and the email contains the complete origin set at that moment. The periodic refresh message is a soft-state mechanism to provide additional system resilience against unforeseen errors. For instance, even if a notification is lost due to email server crash, the next refresh message will bring the owner's knowledge about the origin set up to date.

The major challenge in our system design is how to deliver notifications successfully even in the face of prefix hijacks. When a prefix is being hijacked, some data traffic on the Internet would go to the false origin instead of the true one. If the path from our server to the prefix owner is diverted to the false origin, then the owner would not receive the notification at the time when it is needed the most.

Due to the large scale of Internet routing, a prefix hijack is unlikely to affect all the paths towards the true origin. Thus in delivering the notification messages, our system uses multiple diversified paths to improve the chances of successful delivery. Ideally, we can send notifications from the monitors that still have path to the old origin. But this type of email forwarding service is not part of current BGP monitoring arrangement with commercial ISPs. Requiring email forwarding from monitors would undermine the deployability of our service. Thus we leave this as an option for future development,

[3]In other words, the penalty at time t is exactly one half of the penalty at time $(t - 2)$ hours, assuming no additional origin events were generated during that time

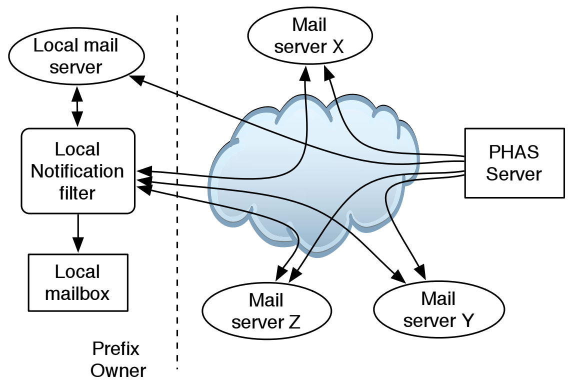

Figure 7: Notification setup

and instead ask prefix owners to take the responsibility of setting up multiple diversified delivery paths.

There are two practices recommended for prefix owners. First, when registering with our system, they should provide multiple email accounts on different servers that are also topologically diverse, for instance popular email services like GMail and Yahoo! mail. Secondly, they should have Internet access through multiple prefixes. ISPs often have several prefixes themselves, so this should not be a problem. For ones that only own a single prefix, a backup plan, such as a dial-up Internet access account, is recommended.

Figure 7 shows how the multiple diversified path delivery works. The owner of prefix P registers four email addresses, one within P, and three others, X, Y, and Z, in three different networks. Every notification message will be sent to all four mailboxes. The prefix owner's local filter will retrieve these four messages, and then process them. The email body will contain a sequence number, based on which the local filter decides whether it is a duplicate or is obsolete. Only emails with new contents pass through the filter and result in an alarm used for hijack detection. When a prefix P is hijacked, as long as the owner can access one of X, Y, Z, and our server, the notification will be delivered. Even if all four mailboxes are not accessible directly from the owner site, as long as the owner can access the Internet through another prefix, he/she can still retrieve the notification messages regarding the prefix P. The local filter also periodically polls the mailboxes. In the event that none of them is reachable, it is very likely that prefix owner's Internet access has problems, and the filter will generate an alarm to the operator. In summary, the combination of multiple topologically diversified mailboxes and multiple prefixes used for Internet access, ensures high delivery rate for notifications.

6 Local Notification Filter

PHAS does not associate a prefix with a true origin or false origin, and thus reports all origin set changes to the prefix owner. However, not all origin set changes may be of interest to the prefix owner, especially in the event that the origin set changes frequently. The local notification filter, is an important tuning block at the user side that enables the prefix owner to filter out unwanted alarms and alert the user for potential hijacks. In this section, we explain some basic building blocks for constructing filter rules and use examples to show how simple rules can control the notifications delivered to the user.

6.1 Constructing filtering rules

We define a rule to have the form "IF <condition> THEN <action>". There are two basic actions possible; ACCEPT results in the message being delivered and REJECT results in the message being dropped. The default action is ACCEPT, in case no rules are specified or no rules are fired. The local filter can contain various rules ordered by preference, and IF clauses can also be nested. While, multiple rules can be listed, for each notification message, an action of ACCEPT or REJECT can be performed only once. In other words, once an action is performed, no more rules are matched for that notification message. Hence, we encourage users to use rules that are simple and easy to understand and analyze.

To construct rules, we define the following constructs.

1. CONTAINS: defines what a particular key may contain.
2. DIFF: difference between sets.
3. LT, EQ, GT: correspond to the mathematical $<, =$ and $>$.
4. NOT: negates the construct it follows. E.g. one may use it with CONTAINS.
5. AND, OR: for combinations of conditions.
6. ANY and ALL: used to deal with sets in rules.

Examples

1. A rule specific to a prefix, and checking to see if the new origin is a known origin:

   IF <ORIGIN-GAINED EQ 29257 AND PREFIX EQ 60.253.48.0/24> THEN REJECT

2. A rule asking to drop all origin loss notifications:

   IF <TYPE EQ "loss"> THEN REJECT

Example of a bad Rule

1. A rule that checks for the existence of an AS in the ORIGIN-SET:

   IF <ORIGIN-SET CONTAINS 23918> THEN REJECT

In the event of a hijack that changes the origin set from $\{23918\}$ to $\{23918, X\}$, where X is the hijacker, the notification will not be delivered to the user, since the origin set still contains AS 23918.

6.2 Case Study

We now use a case study to show how simple rules can be used to deal with a real scenario. We choose prefix 60.253.48.0/24 as an example and look at the notifications from December 21, 2004 to December 28, 2004, when a known prefix hijack event happened. A sample of the notifications seen by the filter is shown below.

<TYPE=gain, GMT-TIME=20041221 04:44:45, PREFIX=60.253.48.0/24, NEW-SET={23918, 31050}, ORIGIN-GAINED=31050>

<TYPE=gain, GMT-TIME=20041221 12:52:33, PREFIX=60.253.48.0/24, NEW-SET={23918, 31050, 29257}, ORIGIN-GAINED=29257>

<TYPE=loss, GMT-TIME=20041221 13:52:49, PREFIX=60.253.48.0/24, NEW-SET={29257, 31050}, ORIGIN-LOST=23918>

<TYPE=loss, GMT-TIME=20041221 13:53:56, PREFIX=60.253.48.0/24, NEW-SET= {29257}, ORIGIN-LOST=31050>

For this prefix, we observed three origin ASes: AS 29257, AS 31050 and AS 23918. The origin set fluctuated between various combinations of these three ASes causing notifications to be sent to the owner. Without local filtering, all these legitimate changes would have resulted in alarms being sent to the prefix owner. However, the prefix owner, knowing all these three legitimate origin ASes, can set simple rules to filter out these changes:

IF <ORIGIN-GAINED EQ ANY {23918,31050,29257} > THEN REJECT

IF <ORIGIN-LOST EQ ANY {23918,31050,29257} > THEN REJECT

Note, each notification contains only one value for ORIGIN-GAINED or ORIGIN-LOST, and hence we can use EQ (equals) clause here. With this rule in place, the prefix owner would only receive an alarm when the origin changes passes both rules. Around 9:30 AM on Dec 24, 2004, such an alarm happened:

<TYPE=gain, GMT-TIME=20041224 09:30:29, PREFIX=60.253.48.0/24, NEW-SET={23918 9121}, ORIGIN-GAINED=9121>

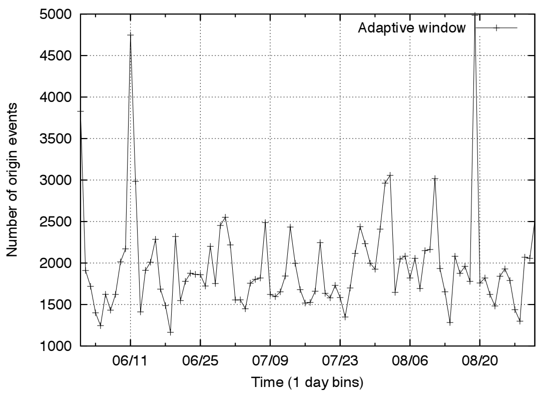

Figure 8: Origin events per day from June 1, 2005 to August 31, 2005

<TYPE=loss, GMT-TIME=20041224 11:35:02, PREFIX=60.253.48.0/24, NEW-SET= {23918}, ORIGIN-LOST=9121>

The first alarm indicates that AS 9121 is now hijacking the prefix 60.253.48.0/24. The owner knows that this is not a legitimate origin for this prefix, and can then take appropriate actions. An alarm is also generated to inform the owner that AS 9121 stopped announcing the prefix, indicating the matter has been resolved.

7 Evaluation

To evaluate the overhead of the system, we use BGP log data to calculate the number of origin events generated by the PHAS server, and the number of notifications received by each AS. We also apply our method to the data collected during known hijack events to show that PHAS can indeed catch those events. Finally, we use simulations to evaluate the success ratio of notification delivery using multi-path delivery scheme.

7.1 Notification Messages

Figures 8 and 9 plot the number of origin events per day over a 6 month period from June 2005 to November 2005. The origin events generated per day for month of December 2005 were shown in Figure 6 in Section 4. Throughout this period, we observed the number of events captured per day to be around 2000, with a few occasional spikes. From a system point of view, sending 2000 messages per day is manageable, even with multiple email delivery.

We now look from users' point of view to see how many notification messages they would receive if subscribed to receive PHAS alerts. We treat each origin event as one notification, assuming all prefixes are registered to receive alerts. For our evaluation, we use the

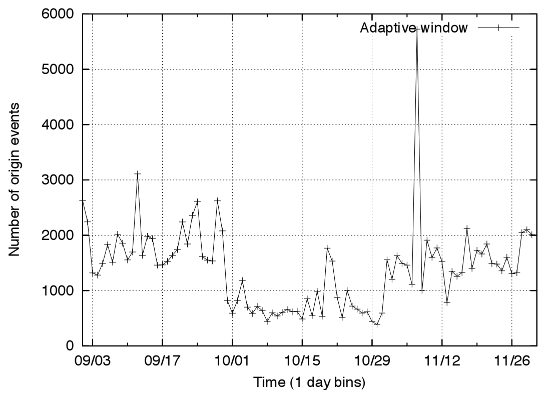

Figure 9: Origin events per day from September 1, 2005 to November 30, 2005

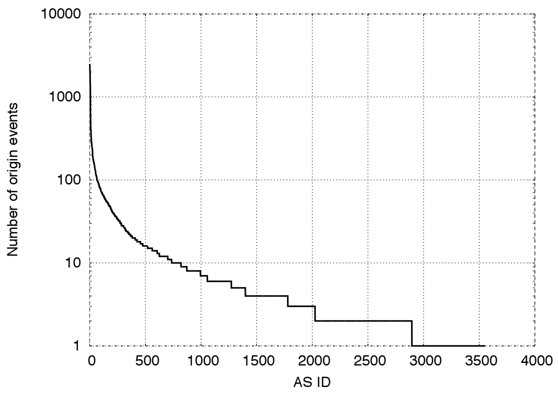

Figure 10: Distribution of events per AS for December 2005

events generated for the month of December 2005. We first evaluate the notifications received per prefix. From Figure 5 in Section 4, we see that only 20K out of more than 150K prefixes were involved in origin events. Of those 20K prefixes, almost all of them had less than 10 origin events per month. Only a handful of prefixes had more than 100 origin events per month. The worst case being 209.140.24.0/24 with 196 origin events. A closer look at the alarms revealed that the origin set alternated between $\{\}$ and $\{3043\}$, which indicates the prefix was unstable. From, these numbers for origin events, one can see that the number of notifications expected per prefix is quite small, except for some unstable prefixes. For cases of unstable prefixes, the owner's local filter will be able to handle such redundant notifications easily.

Since a prefix owner may register multiple prefixes, we also look at number of notifications expected per AS for the month of December 2005. For evaluation purpose, we estimated the prefixes registered by each AS by using the routing table to map every prefix to its origin AS. Figure 10 shows the number of origin events per AS for December 2005. Only about 3.5K ASes out of the total 18K ASes received notifications. Of those ASes that received notifications, 97% of them received less than 100 notifications in the entire month. The worst case was AS 29257, receiving 2501 notifications, with the $O_{SET}(P)$ fluctuating between combinations of 4 origins. These numbers for origin events per AS indicate that in most cases, an AS would receive a small number of notifications, and in extreme cases, local filters can once again deal with the common pattern of notifications. All of the above results show that the load of notification generation, transmission, and processing are easily manageable by a single machine, even when all the prefixes are registered with PHAS.

7.2 Detecting Known Events

We now check if our system would have caught some known prefix hijack events. One such prefix hijack occurred on May 7, 2005 when AS 174 hijacked one of Google's prefixes, 64.233.161.0/24, causing Google to be unreachable during this time. When run over this period of time, PHAS caught this origin set change and indicated AS 174 as the origin gained during this event.

A larger scale hijack event occurred on Dec 24, 2005. AS 9121 announced itself as origin to over 106K prefixes. PHAS detected 106082 unique prefixes with origin 9121 added to its origin set and a total of 217884 origin events. Most prefixes had 2 notifications, one reporting the addition of AS 9121, and the other reporting the removal of AS 9121.

Another case of hijack occurred on Jan 22, 2006, when AS 27506 announced itself as origin to some other's prefixes. For this day we detected 41 unique prefixes with AS 27506 as a new origin, and a total of 141 origin change events. For some prefixes, the AS 27506 was announced as origin, then withdrawn, and then reannounced and withdrawn again resulting in multiple origin events.

Overall, PHAS successfully caught every known prefix hijack due to false origin in a timely manner, and the timing matched reports from other sources.

7.3 Notification Delivery

To have multiple diverse paths for notification delivery, we recommend the prefix owners register multiple mailboxes and have multiple prefixes for Internet access. If they do have multiple prefixes, they can always receive the notification messages assuming only one is hijacked. In this subsection, we evaluate the effectiveness of using multiple mailboxes through simulations on Internet topology.

The approach is to take an Internet AS graph as the topology, tag each link with inferred relationship, assume the widely adopted "no-valley" routing policy on every node, then compute the shortest policy-compliant path between any two nodes. For each calculation, the input includes one true prefix origin, one false origin, and a set of mailboxes. Based on the computed shortest paths, we can find out the success ratio of notification delivery.

The AS Topology is collected from multiple sources, including BGP monitors, route servers, looking glasses, and routing registry [22]. The AS relationship is inferred using the method in [21]. Two set of mailboxes are used for comparison. The first set is RouteViews (AS 3582) only, which is called "direct delivery" without other mailbox. The second set is RouteViews plus GMail (AS 15169), Yahoo Mail (AS 10310), and Hotmail (AS 12076). We randomly picked 276 ASes to form the origin pairs. They are 15 tier-1 ASes, 21 tier-2 ASes, 20 tier-3 ASes, 20 tier-4 ASes, and 200 tier-5 ASes. We exhaust all the combinations of origin pairs, a total of 75900 cases.

Given an origin pair, some nodes will take the path to the true origin, while the others will take the path to the false origin. If a mailbox node takes the path to the true origin, the prefix owner will be able to access this mailbox and receive the notification. Otherwise the notification is lost. *Delivery ratio* is defined as the percentage of mailboxes that take the path to the true origin.

Note the simulation results will be symmetric. That is, suppose there is 20% delivery ratio for a given pair of true origin and a false origin, then it will be 80% when the role of these two origins switches. Since we exhaust all combinations of origin pairs, whenever there is a case of $a\%$ delivery ratio, there will be a corresponding case of $(1 - a\%)$ delivery ratio.

In our path computation, we use random tie-breaking when there're multiple shortest paths. For example, if a mailbox has two equal paths, one leads to the true origin, the other leads to the false origin, we count this as 0.5 notifications from this mailbox can be delivered.

Figure 11 compares the delivery ratio of with and without additional mailboxes. Without the three additional mailboxes, about 30% of notifications will be guaranteed delivered, about 30% of notifications will be lost for sure, and the rest may be delivered by certain probability. With the three additional mailboxes, the non-delivery ratio drops to about 10%.

Figure 12 shows the number (not the ratio) of notifications that can get delivered. In about two thirds ($x \geq 1$) of the cases, we have at least one messages are guaranteed to be delivered. It doubles compared with using only the direct delivery (30%). This suggests that three additional mailboxes can greatly improve the notification delivery, but we may still need more mailboxes for higher

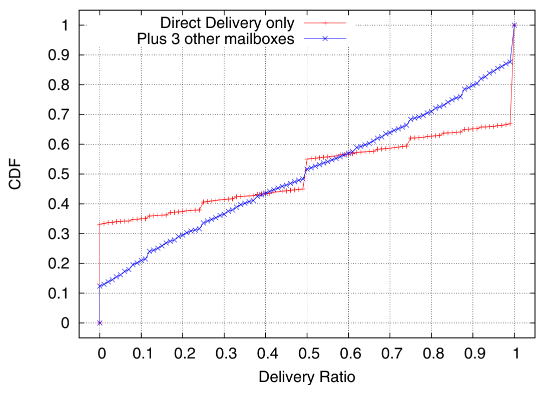

Figure 11: Delivery Ratio

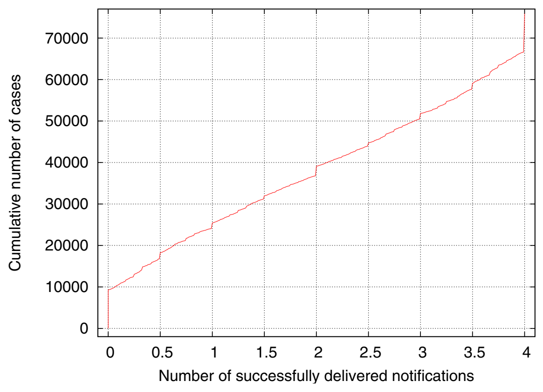

Figure 12: Delivery Number

success ratio.

8 Extensions to basic system

So far we have focused on detecting false origins. In this section, we discuss other ways of hijacking a prefix besides directly announcing a prefix and discuss extensions to the current system to deal with some of these cases.

8.1 Classification of Prefix hijack

At the highest level, the attacker AS could target a prefix that is already being announced by another AS, which we term as valid prefix. The attacker may pretend to be the owner of this prefix and originate the prefix resulting in a false origin hijack, that is the focus of this paper. Another way to hijack a prefix is by announcing a valid origin, but report invalid path to the origin. For false paths, we separate the case of *false last hop*, from false information on any other hop in the path, since the prefix

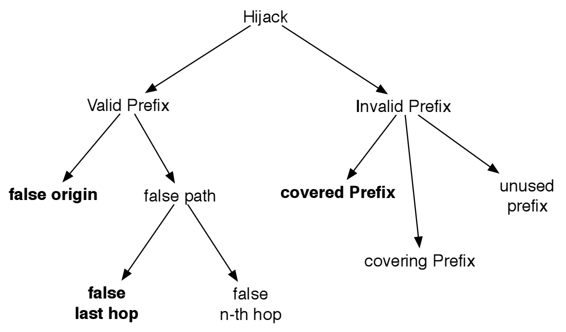

Figure 13: Types of prefix hijacks

owner's AS knows its immediate neighbors, and hence can identify whether the last hop is valid or not.

An attacker AS may also announce a prefix that is not being announced by another AS, termed as invalid prefix. If the attacker announces a sub-prefix of some valid prefix, termed as a *covered prefix hijack*, then routers in the Internet may contain routes to both the victim AS's prefix as well as the attacker's prefix. However, if the destination IP of a packet being routed, falls under the attacker's prefix space, then due to longest prefix match, the data would be forwarded to the attacker. An attacker AS may also announce a less specific prefix than a valid prefix, termed as a *covering prefix hijack* but will receive traffic, only when the route to the valid prefix is withdrawn. For example, if AS 110 announces 131.0.0.0/8, then AS A would route traffic destined to the valid prefix 131.179.0.0/16, to AS 110 only when the prefix 131.179.0.0/16 is withdrawn. Finally, an AS may announce an invalid prefix that does not conflict with any used prefix space. For example, spammers are known to use unused prefixes for spam purpose. Figure 13 shows the classification explained above.

Prefix hijacks could also include combinations of various types in Figure 13. E.g. AS 110 announcing 131.179.0.0/24 (invalid covered prefix) with the path $\{110, 52\}$ (invalid last hop). In Figure 13, the hijacks in bold (false origin, covered prefix, false last hop) are the ones where the prefix owner knows of what is legitimate and what may not be, and protection against these attacks is the focus of PHAS. We now discuss two other sets to deal with covered prefix hijack and false last hop hijack.

8.2 Sub-prefix Set

The idea of using a sub-prefix set is to provide the owner of an IP prefix with the information about whether anybody is announcing a more specific prefix under its assigned space. This would catch hijacking event where a prefix, say 131.179.96.0/26 is announced by a hijacker AS 100, but the prefix is part of the address space covered by 131.179.0.0/24, which is owned by AS 52.

For an IP prefix x, some or all of its assigned address space might get further divided into a number of longer prefixes. Each of these prefixes is a known as a covered prefix of x . The set of all covered prefixes of x observed from the BGP monitors, is denoted as $CP(x)$. For example, if UCLA announces 131.179.0.0/16 as well as 131.179.96.0/24 and 131.179.59.0/24, then $CP(131.179.0.0/16)$ is $\{131.179.96.0/24, 131.179.59.0/24\}$.

We define a sub-prefix set $SP_{SET}(x)$ to consist of all $y \in$ *CP(x)* such that there does not exist another prefix $z \in$ *CP(x)* with $y \in CP(z)$. In other words, the set $SP_{SET}(x)$ contains only the first level covered prefix for prefix x.

As an example of how this SP_{SET} could be useful, we present a case from Jan 22, 2006. The prefix 208.0.0.0/11, owned by Sprint, generated one origin event at 5:06 am UTC indicating that the sub-prefix set had changed from $\{\}$ to $\{208.28.1.0/24\}$ with origin $\{27506\}$. The prefix in question, 208.28.1.0/24 is not usually seen in the global routing tables, but in this case AS 27506 announced this prefix, which covers a portion of Sprint's 208.0.0.0/11 prefix space, thus resulting in a hijack.

8.3 Last Hop Set

The last hop set is maintained with the objective of detecting false last hops in BGP announcements. Once again, the owner of the prefix would know the legitimate next hops based on peering agreements and reports of such changes would allow the owner to detect false last hops in BGP paths.

We define an last hop set $LH_{SET}(A)$ as the set of last hops for all prefixes with AS A as the origin. For example, if M_1 observed a path (7018, 1239, 52) to prefix P_1, M_2 has a path (3356, 1239, 52) to P_2, and M_3's path to P_3 is (701, 852,52), then the last hop set of AS 52, or $LH_{SET}(52)$, is $\{1239, 852\}$. Note, that last hop is defined for an AS, and not for a prefix, since it is reflecting topological connectivity.

The main objective of using the sub-prefix set and the last hop set is to identify potential hijacks involving more specific address space and last hop changes. However, the sub-prefix set for large address blocks like 12.0.0.0/8 can be potentially huge, and may cause lots of dynamics. Similarly, the size of last hop sets for nodes with rich connectivity (e.g. tier 1 ISP) can also be significant, and may fluctuate a lot. For future work, we plan to understand the dynamics of these two sets, define how to use these sets, and include them as a part of the PHAS system.

9 Related Work

Various prefix hijack events have been reported to NANOG [10] mailing list from time to time. [23] and [8] studied the exact prefix hijack as part of the MOAS (Multiple Origin AS) problem, in which one prefix has multiple origin ASes in the routing table. These studies show that one prefix can be legitimately announced by multiple origin ASes, but can also be hijacked due to mis-configurations.

Existing proposals to address prefix hijack problem can be categorized into two types: cryptography based, and non-crypto based. Crypto-based solutions, such as [18], [1],[3], [11], [6], [17], require BGP routers to sign and verify the origin AS and the path, which have significant impact on router performance. Furthermore, these solutions are not easily deployable because they all need changes to router software, and some require public key infrastructures.

Non-crypto proposals include [2], [20], [24], and [5]. IRV approach in [2] lets each AS designate a server that answers queries regarding BGP security. [20] lets the router give preference to stable routes over transient ones which can be results of prefix hijacks. Similarly, in PG-BGP [5], a router detects prefix hijacks by monitoring the origin ASes in BGP announcements for each prefix over time. A transient origin AS of a prefix is considered as anomalous, and router avoids using the anomalous routes whenever possible. PG-BGP also detects covered prefix hijacks using similar approach. In [24], prefix owners attach additional information to the routing updates, so that remote routers could detect prefix hijacks. All the Above non-crypto proposals require changes to router softwares, router configurations, or the ways that operators run their networks.

Compared to all of the above proposals, the biggest advantage of our system is that it is fully deployable. PHAS can be up and running without requiring cooperation from multiple ISPs, registry authorities, router vendors, or even end users. While other approaches focus on detecting prefix hijack at remote ASes, we simply notify the prefix owner about the origin changes, thus allowing the prefix owners to detect prefix hijacks with a high accuracy.

Three other related works [19, 7, 9] are also fully deployable. [19] utilizes the data from RouteViews or RIPE and visualizes the origin AS changes of the prefixes for visual detection of the prefix hijacks. [7] proposes an alarming algorithm for prefix hijacks and path hijack, based on the the public BGP data, and the geographic information of the each AS from the $whois$ database. The key observation is that if two edge ASes are connected to each other or legitimately originate the same prefixes, they are geographically close. Violation of this observation will trigger alarms.

The RIPE MyASN project [9] is probably the most similar service to ours, but its design is based on a fundamentally different philosophy. In the MyASN project, a prefix owner registers the valid origin set for a prefix. MyASN then tracks roughly the equivalent of our instantaneous origin set for this prefix. An alarm is triggered when any invalid origin AS appears. Our approach reports the origin set changes to the prefix owner, and any filtering or checking is done at the user site. This is a subtle difference, but has important implications.

First, filtering at the user side provides the greatest degree of flexibility to the detection algorithm. Users can apply any filtering criteria or detection algorithm on the data. When the filtering is done at the service site like MyASN, it is limited to what the service interface could provide. Obviously for security reason, the service site cannot allow arbitrary filtering script to be uploaded. If prefix owners cannot achieve their filtering goal at the service site, they have to deploy local filter anyway.

Second, it is critical for the server-based filtering to have the most up-to-date information needed for prefix hijack detection. The valid origin set must be updated at MyASN server whenever the prefix has a different origin set. It's especially hard to do update in face of an on-going prefix hijack. When a new hijack happens, the prefix owner may want to change the filtering rule, but is unable to do so due to the attack. Our approach does not does not suffer from this problem.

10 Conclusion

In this paper we described the design of PHAS, a Prefix Hijack Alert System. Rather than attempting an accurate route hijacking detection algorithm, PHAS aims at providing timely notification of origin AS changes to the owners of individual prefixes in a reliable way. The prefix owners can then easily identify real hijack alerts and filter out normal origin changes. By avoiding running complex data processing at BGP data collectors, PHAS can be quickly implemented and run with little overhead at the data collectors. By automating the email processing at the user end, PHAS provides network operators with realtime alerts to potential prefix hijacks while adding virtually no overhead to the operation tasks.

PHAS leverages on the existing routing logs for data input and the universally available email system for notification delivery. PHAS is light on authentication of users because its information is derived from publicly available data, and is light on data filtering because it simply provides information to users for hijack detection. As a result PHAS is light weight and readily deployable. As next step we plan to implement and install PHAS at RouteViews for trial deployment. We expect to gain further insight on how to improve PHAS through experience.

References

[1] W. Aiello, J. Ioannidis, and P. McDaniel. Origin Authentication in Interdomain Routing. In *Proceedings of 10th ACM Conference on Computer and Communications Security*, pages 165–178. ACM, October 2003. Washington, DC.

[2] 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 *NDSS*, 2003.

[3] Y.-C. Hu, A. Perrig, and M. Sirbu. SPV: Secure path vector routing for securing bgp. In *Proceedings of ACM Sigcomm*, August 2004.

[4] Geoff Huston. Auto-detecting hijacked prefixes? a presentation at RIPE-50, May 2005 http://www.potaroo.net/presentations/index.html.

[5] J. Karlin, S. Forrest, and J. Rexford. Pretty good bgp: Protecting bgp by cautiously selecting routes. Technical Report TR-CS-2005-37, University of New Mexico, Octber 2005.

[6] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE JSAC Special Issue on Network Security*, 2000.

[7] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. Topology-based detection of anomalous bgp messages. In *6th Symposium on Recent Advances in Intrusion Detection (RAID)*, 2003.

[8] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *Proceedings of ACM Sigcomm*, August 2002.

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

[10] The NANOG Mailing List. http://www.merit.edu/mail.archives/nanog/.

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

[12] D. Pei, D. Massey, and L. Zhang. A Framework for Resilient Internet Routing Protocols. *IEEE Network Special Issue on Protection, Restoration, and Disaster Recovery*, 2004.

[13] Alin C. Popescu, Brian J. Premore, and Todd Underwood. Anatomy of a leak: As9121. http://www.nanog.org/mtg-0505/underwood.html.

[14] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. In *Proceedings of ACM SIGCOMMq*, 2006.

[15] Y. Rekhter and T. Li. Border Gateway Protocol 4. RFC 1771, SRI Network Information Center, July 1995.

[16] Y. Rekhter and T. Li. A Border Gateway Protocol (BGP-4). *Request for Comment (RFC): 1771*, 1995.

[17] B. R. Smith and J. J. Garcia-Luna-Aceves. Securing the border gateway routing protocol. In *Global Internet'96*, November 1996.

[18] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and whisper: Security mechanisms for bgp. In *Proceedings of ACM NDSI 2004*, March 2004.

[19] S. T. Teoh, K.-L. Ma, S. F.Wu, D. Massey, X. Zhao, D. Pei, L. Wang, L. Zhang, and R. Bush. Visual-based anomaly detection for bgp origin as change (oasc) events. In *IFIP/IEEE DistributedSystems: Operations and Management (DSOM)*, pages 155–168, October 2003.

[20] 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 *Proceedings of the ICDCS 2003*, 2003.

[21] Jianhong Xia and Lixin Gao. On the evaluation of AS relationship inferences. In *Proc. of IEEE GLOBECOM*, December 2004.

[22] Beichuan Zhang, Raymond Liu, Daniel Massey, and Lixia Zhang. Collecting the internet as-level topology. *ACM SIGCOMM Computer Communications Review (CCR)*, 35(1):53–62, January 2005.

[23] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Wu, and L. Zhang. An Analysis BGP Multiple Origin AS(MOAS) Conflicts. In *Proceedings of the ACM IMW2001*, Oct 2001.

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