Protecting BGP from Invalid Paths

Josh Karlin
University of New Mexico
karlinjf@cs.unm.edu

Stephanie Forrest
University of New Mexico
Santa Fe Institute
forrest@cs.unm.edu

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

ABSTRACT

The Internet's interdomain routing protocol, BGP, is vulnerable to a number of potentially crippling attacks. Many promising cryptography-based solutions have been proposed, but none have been embraced by the necessary communities to garner significant adoption. This is largely due to the difficulty of developing and maintaining the necessary PKI infrastructure and changes to the BGP protocol that the proposed solutions require. Alternative solutions such as anomaly detectors have been unable to provide the same level of security as the cryptographic mechanisms. In this paper we create an anomaly detector and response mechanism capable of automatically stopping the propagation of invalid path attacks, a difficult class of attacks to detect. Our solution provides comparable security to the cryptographic methods and could be readily deployed with a simple software upgrade in participating networks.

1. INTRODUCTION

The routing protocol that connects large IP networks to form a single Internet is the Border Gateway Protocol (BGP). This critical networking infrastructure has significant security vulnerabilities that are well documented but have been unresolved for many years. BGP's vulnerabilities arise from its trust model. By design, BGP trusts each individual network (Autonomous System) of the Internet to announce only legitimate and accurate routing information. BGP's trust model is critical to its success, because it allows networks to operate independently of external authority, thus enabling a truly distributed network. As evidenced by the current success of the Internet and widespread adoption of BGP, this level of trust is not entirely unwarranted.

In spite of BGP's success, inadvertent mistakes and malicious attacks can cause serious problems. A simple typographical error in configuring a router can cause an AS to inadvertently "hijack" traffic meant for other networks. We refer to these as *origin AS* attacks. For example, an AS may announce to its neighbors that it can route traffic destined to a block of addresses it does not own, and this false information may propagate to other networks. Such errors are common, and many examples have been documented [14, 6, 22, 19, 13, 25]. Hijacks range in severity from misrouting a small segment of address space, as happened in the 2005 Panix incident [25], to diverting a significant fraction of all Internet traffic to a small ISP for several hours [19, 5].

Proposals for protecting BGP from hijacking and other attacks fall into two categories, cryptographic protection and anomaly detection. Cryptographic approaches (notably Secure BGP [15] and Secure Origin BGP [20]) involve an authenticated registry that maps IP prefixes to their proper origin ASes. The registry would be secured and distributed using a Public Key Infrastructure (PKI). This approach requires global cooperation among the ASes to build and actively maintain the registries. To date, efforts to create such registries [3, 27, 4] have suffered from inaccuracy [18] and lack of trust by the operational community [32]. Other impediments include the need to change the basic BGP protocol (possibly requiring a universal "flag day" for the conversion) and the requirement for trusted parties to maintain the PKI and registries. Despite several credible proposals, cryptographic solutions have not yet been deployed.

Given the difficulty of deploying a secure interdomain routing protocol, it is worth asking how much security can be achieved without global cooperation and trusted third parties. In recent years, several anomaly-detection systems have been developed to detect origin AS attacks [35, 30, 16, 14, 22, 10, 34]. These systems work by adding information to BGP update messages [35, 30], comparing routing updates to out-of-band information [16, 10], or comparing new routes to a database of historically trusted routes [34, 14, 22, 16]. Because anomaly-detection methods can produce false positives, it is risky to discard all anomalous routes. However, it is also important to have an automated mechanism for responding to anomalies, to prevent the propagation (and use) of malicious routes.

Few methods have been proposed that pair anomaly detection with an automated response. One method, known as Pretty Good BGP (PGBGP) [14], avoids routes with unfamiliar origin ASes. PGBGP maintains a history of (destination prefix, origin AS) pairs and assigns a lower "preference" to routes that disagree with past history for a period of time. The lower preference prevents use (and further dissemination) of the route, and the time delay provides time for human

intervention or confirmation with external route-monitoring services, such as the Internet Alert Registry [12] or Renesys Routing Intelligence [26] (which many network operators already use). External monitoring services allow a network to compare its internal registries against alerts received from the external monitors without revealing their internal network information, often considered to be proprietary. Network operators are financially motivated to respond quickly to identified problems because clients might not be able to reach the operator's network.

Earlier anomaly-detection systems, including PGBGP, detect invalid origin AS attacks, but not problems in the rest of the AS-path attribute. Each AS that propagates a route is supposed to add its own AS number to the path and leave the remainder of the path intact. However, a router can be configured to manipulate the path or introduce an entirely bogus path. For example, an AS could remove some of the vertices from the path to make the path look shorter, making other ASes more likely to select it. These *invalid-path attacks* are more challenging to detect, both for cryptographic and anomaly-detection approaches, making them particularly attractive to intelligent adversaries. Invalid paths are difficult to detect because many organizations do not publish all their routing information (e.g., peering relationships) and consider such information to be proprietary. Thus, an authenticated registry of all edges and policies would face even more impediments than a registry of valid origin ASes.

Detecting invalid paths is challenging for anomaly detectors that rely on local information, because BGP routes do not contain enough information to verify the authenticity of each AS along the path. No single router can be aware of all of the valid edges on the Internet, because not all paths are advertised (e.g., for policy reasons). However, some progress can be made by using additional information [16, 22]. Kruegel *et al.*'s topology-based detector compares physical network location (obtained from the WHOIS registry [1]) to the BGP paths. If the paths do not follow a set of geographic constraints, they are considered suspicious and an alarm is sent to the operator. Hi-BGP [22] is a BGP security system that includes an anomaly detector. When the anomaly detector is used, the edges seen by the router are kept in a historical registry and a policy-inferencing algorithm is used to detect policy violations (one type of invalid path). We adopt a similar history-based approach in this paper, though our solution monitors *directed* edges and does not rely upon inferencing algorithms or external data.

In this paper we describe an anomaly detector and response mechanism that protects routers from invalid path attacks comparably to the well-known cryptographic approaches, Secure BGP and Secure Origin BGP. Invalid paths can be broken into three subclasses: spoofed AS numbers, spoofed edges, and policy violations. SBGP is vulnerable to policy violations [22] and soBGP is vulnerable to some AS number spoofing scenarios [20]. Our solution is capable of detecting all spoofed edges, equally capable of detecting AS number spoofing attempts as soBGP, and able to detect the same policy violations as soBGP except for the rare case in which two ASes are both customers and providers for one another. Our enhancement would allow PGBGP-enabled ASes to protect themselves from all known major BGP exploits comparably to SBGP or soBGP without a PKI, community consensus, or changes to the BGP protocol.

Route Learned From	*Should Export Route to*
Provider	All customers and siblings
Peer	All customers and siblings
Customer	All neighbors
Local	All neighbors

Table 1: Standard route export rules are based upon where the route was learned from. A route is never learned from a sibling, instead the route is said to have been learned from the nearest non-sibling relationship in the AS path to the recipient. If only sibling relationships exist, the route is considered to have been learned locally.

Our solution capitalizes on two main insights about the BGP routing system. First, almost every invalid path includes at least one *invalid (directed) edge* that a downstream AS on the path should not have seen before. This holds true for all spoofed edges, spoofed AS numbers where the victim and adversary do not share a neighbor, and policy violations where the victim AS is not both a customer and provider of its violating neighbor. Second, invalid edges can be detected by flagging unfamiliar directed edges. Together, these observations allow us to construct a simple anomaly-detection system that detects invalid edges, and, hence, invalid paths. By enhancing PGBGP with our simple invalid-edge detector, it is possible to respond to an anomalous edge by temporarily depreferencing the route, making the route less likely to propagate.

The paper makes four contributions to BGP security research. First, it describes an anomaly detector that is capable of detecting and stopping the propagation of most invalid-path attacks. Next, we enumerate important classes of invalid-path attacks and report simulation results that quantify the effectiveness of these attacks when the victim and adversary are randomly placed. Third, we compare the results of our edge-detection system to an idealized anomaly detector—one that achieves 100% true positives and 0% false positives, and discards all invalid routes. Finally, we show that under a partial deployment in large ASes, enhanced PGBGP provides nearly the same security benefit as the perfect detector. Our solution is incrementally deployable and offers substantial benefits to early adopters, making it an attractive alternative to cryptographic approaches.

2. BACKGROUND

This section reviews the BGP protocol, discusses two important classes of BGP attacks, and describes how Pretty Good BGP protects against one of these classes.

2.1 The Border Gateway Protocol

The Border Gateway Protocol protocol [24] is the standard interdomain routing protocol used today. It ensures that each participating network (Autonomous System) has a route for reaching every other IP block (prefix). It is a path-vector protocol in which each AS informs its neighboring ASes about its best available route to each destination via update messages. If a new prefix is added to the network, the owner

Exploit Name	*Category*	*Procedure*
Shortest Spoofed Path	Spoofed Edge	Erase AS path between adversary and origin AS before export
Shortest Path	Policy Violation	Replace AS path with shortest path of existing edges to origin AS
Redistribution [22]	Policy Violation	Export route learned from one provider or peer to another
Spoofed AS Number	Invalid AS Number	Erase AS path and prepend victim's AS number

Table 2: Invalid path exploits.

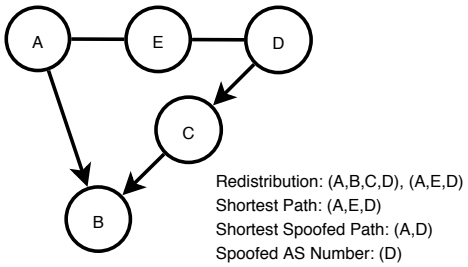

Figure 1: Examples of invalid paths. Here, Autonomous System A modifies its AS path when exporting routes to gain access to D's traffic. The paths listed in the figure are those that A could send to its neighbors for each type of invalid path attack. Arrows point to customers from providers, and undirected edges represent peer-peer relationships.

of the prefix announces it from her AS with an announcement update message. Similarly, if a prefix is retracted or a BGP session is dropped, ASes retract routes by propagating a withdrawal update.

If an AS knows about multiple routes to the same prefix, its router chooses the route which the operator has assigned highest preference to. If multiple routes tie for the highest preference, then a series of tie-breaker rules are applied (such as path length) to select a single route. Generally, all routes from a given neighbor are assigned the same preference, which is based upon the cost of using the link, not performance. The cost of the usage of a route can be determined by the contractual relationship between an AS and its neighbor.

There are three types of AS relationships [7]: customer-provider, peer-peer, and sibling-sibling. In a customer-provider relationship, the customer pays the provider for access to the rest of the Internet. This is economically beneficial to the provider, and the provider has an incentive to assign highest preference to routes that originate from its customers. Customers meanwhile have an incentive to avoid provider routes when possible, by assigning them low preference. Peer-peer relationships occur when two ASes agree to share routes and carry traffic for each other's customers at no charge. Because peer-peer routes are free, they are given higher preference than provider routes, but lower than customer routes. Sibling-sibling relationships are used to share all routes between two ASes, as if they were both providers for one another.

ASes are sometimes prevented by contractual agreements from forwarding (exporting) their best routes to all of their neighbors [7]. Routes that are exported in violation of contractual stipulations are considered policy violations, and are one type of invalid path. According to Gao *et al.* [7], an AS should only export routes learned from its peer-peer relationships and from its providers to its customers and siblings. Routes learned from customers and siblings should be exported to all neighbors. Therefore, an AS should not export a route learned from a provider or peer to another provider or peer, such routes are in violation of policy. Table 1 lists each of the export rules used in common practice for future reference.

Even accidental policy violations can cause serious network problems. Providers give highest preference to customer routes. When a customer AS C violates policy and exports a route learned from one provider or peer to another, P, then P will likely choose C's route. Since P will continue to export the route to its providers, a number of large provider ASes would be likely to route through C since it is a customer route. The increased traffic to C might overload C's capacity, preventing the traffic from reaching its destination.

2.2 Origin AS Attacks

There are two main classes of origin AS attacks: prefix hijacks and sub-prefix hijacks. Because BGP does not validate the origin AS of an update message, a BGP router can announce any prefix, even those it does not own (a prefix hijack). For example, a university could announce that it owns a prefix that actually belongs to a financial institution, such as a bank. Those ASes that selected the university's route would send their data to the wrong destination. The university could then use the data however it pleased: it could discard it (known as a blackhole); it could read the data and then forward it on to the intended destination; or, it could impersonate the bank's services to gain passwords (such as a website login page).

Because an AS can announce any prefix it wants, a network can accidentally or maliciously announce a subnet of another network's prefix rather than the whole prefix. This is known as a sub-prefix hijack. For instance, an AS could announce 12.0.0.0/9 which is a subnet of AT&T's 12.0.0.0/8. This is a particularly devastating form of attack because at packet forwarding time the router will forward to the smallest matching subnet.

An AS could also announce a larger network, or supernet, of its adversary's prefix. Although it has been shown that such hijacks could be used for sending spam from unused address space [23], it could not be used to divert traffic away from proper destinations since routers always forward packets to the smallest matching prefix, and in this paper we do not consider such attacks.

There are many examples of actual origin AS attacks, including the famous 1997 incident in which a single ISP sub-prefix

hijacked the first class-C subnet of every announced prefix causing reachability problems for a large number of networks. More recently, on January 22nd of 2005, AS 25706's announced [25] many prefixes it did not own including those owned by Panix (AS 2033). On November 30th of 2006, AS 4761 announced at least 4000 prefixes that it did not own [13], including specific prefixes owned by organizations such as Banks, Universities, and large corporations. It is generally thought that most of these attacks were accidental, but they still can cause damage and they occur routinely.

It is worth noting that origin AS attacks could be stopped by using only methods available to BGP today. BGP provides programmable filters, in which operators can program their routers to discard routes that violate certain conditions. Filters are used by some providers to ensure that their customers only announce routes for prefixes that they own. If all providers did this, the BGP network would be safe from origin AS attacks. However, many networks do not filter effectively, forcing neighboring ASes to infer which routes are invalid (e.g., routes that originate from many hops away), an impossible task without a complete registry. Even careful network operators make mistakes, allowing their customers to announce prefixes they do not own. For example, AS 2914 is well known to run carefully configured filters for its customers but it was one of the ASes that allowed its customer (AS 4761) to announce Panix's prefix in the previously mentioned attack [25].

2.3 Invalid Paths

Because BGP does not validate the path of an update message, an announced route might not be valid and could lead traffic to an incorrect destination. We define an *invalid path* as an AS path in which an edge (pair of consecutive ASes in an AS path) in the path is spoofed, the path violates a contractual policy, or at least one AS in the path has a spoofed AS number. This extends the definition introduced in [16] to include policy violations. Policy violations are difficult to identify because, as stated earlier, ASes are often unwilling to publicly reveal their peering and policy information.

When a route is exported, an AS is supposed to prepend its own AS number to the AS path. BGP routers are configured to do this to prevent routing loops. However, a malicious router could edit the AS path before exporting it. So long as the edited path contains real edges and does not violate any policies, it is considered valid. Secure Origin BGP also considers such paths to be valid. However, an adversarial AS might edit the path and insert spoofed edges or policy violations to make their route more likely to be selected, in which case our extension to PGBGP will detect it.

The most effective use of a spoofed edge is to erase the entire path between the adversarial AS and the origin AS, leaving the apparent edge (Adversary, Origin) in order to shorten the path. In this paper we refer to such an attack as a *shortest spoofed path attack*. To avoid having any spoofed edges in the path, an adversarial AS might erase the existing path and prepend the shortest valid path of actual edges between itself and the origin AS. If the shortest path contains a policy violation, then the path is considered a *shortest path attack*, otherwise it is considered a valid path.

If a BGP router is not correctly configured, it could accidentally export routes learned from providers or peers to other providers or peers, causing a policy violation. This is fairly common as many BGP routers export all learned routes to all neighbors by default. This form of attack is known as a *redistribution attack* [22]. The reason that accidental policy violations attacks are harmful is that the providers (and the provider's providers) that the customer might export the route to would be likely to select the customer route for the destination, but the customer might not be able to cope with such a large amount of traffic.

Finally, instead of prepending its own AS number before exporting the route, an adversarial AS might prepend another AS's number instead. This is considered an *AS Number (ASN) spoof attack*. An AS could use this to forge a prefix hijack. For instance, if an adversarial AS A wanted to steal the victim AS's (V) prefix P, then AS A could originate an announcement for P using V's AS number. This is a difficult attack to perform because A's peers would likely discard routes that do not have the correct next-hop AS number. Therefore, AS A must either convince its neighbors that it is indeed AS V, convince its neighbors to collude with it, or compromise its neighbor's routers.

Examples of all of the previously mentioned attacks are given in Figure 1, and a short description of each type of attack is listed in Tables 2.

2.4 Pretty Good BGP

In the remainder of the paper we describe extensions to Pretty Good BGP for detecting invalid paths. Before describing the extensions, we review how the original Pretty Good BGP protects against origin AS attacks. The extended system protects against both classes of BGP attacks.

Pretty Good BGP is an anomaly detection and response system. It could be deployed directly on any BGP router with a simple software upgrade, and it does not require any out-of-band communication. The basic mechanism monitors all of the router's available routes over time, treating routes with new origin ASes for a given prefix as *suspicious*. Suspicious routes are given a low preference for a fixed time period s, ensuring that the router is unlikely to use it until s expires.

PGBGP's depreferencing mechanism ensures reachability. If a prefix changes its origin AS (e.g. due to a move or use of a backup provider), then the new origin AS will be selected for forwarding when routes for the trusted origin AS have been withdrawn. Therefore, many of the routes that PGBGP incorrectly labels as suspicious (false positives) will propagate unhindered.

To detect origin AS attacks, the PGBGP algorithm maintains a database of all origin ASes found in received routing announcements and the routing table (RIB) within the last h days, known as the history period. All routes that contain a prefix not listed in the database are considered suspicious and depreferenced. For sub-prefix hijacks, a database of all prefixes within the RIB or update messages received within the last h days is kept. Any new sub-prefix is considered suspicious and is delayed from entering the RIB for s time. In both cases, if a trusted origin AS for the prefix (or its

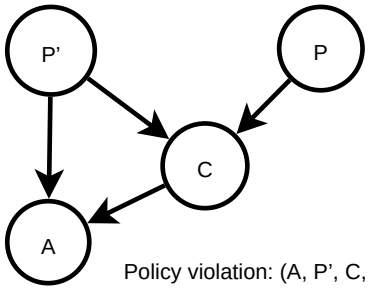

Figure 2: In this example AS A can legitimately see the edge (C,P) in path (A,C,P) since A is one of C's children. Invalid path (P',C,P) contains an invalid edge since P' should not see (C,P), but A would not recognize it as invalid.

supernet) is on the AS path, then the route is not considered suspicious.

While a suspicious route is depreferenced, a network operator has the opportunity to determine if the route is valid or invalid, either by manual inspection or by consulting an external monitoring system. Monitoring systems such as the Internet Alert Registry [12], Renesys Routing Intelligence [26], My ASN [28], or PHAS [17] can be used to inform victims of a hijack that a problem exists. These alerts can be automatically compared against the network's internal registry of origin information, and true positives sent directly to the operator for repair. Monitoring systems are currently used by networks even without PGBGP deployed, because it is important that network traffic be misdirected for as short a time as possible. PGBGP's contribution is to slow the propagation of new routes long enough for other methods to repair the problem. If PGBGP were deployed, the urgency would remain but the attack would not be able to propagate and misdirect traffic until after the suspicious period had ended.

3. DEPREFERNCING SUSPICIOUS PATHS

This section describes a simple anomaly detector based on invalid edge detection that protects BGP from invalid paths. First we discuss why invalid edges are a necessary component of most invalid paths and then we describe the detection algorithm and its overhead.

3.1 Common Characteristics of Invalid Paths

In most cases, invalid paths contain at least one *directed* edge that a recipient of the route should not have received in an update. We refer to such an edge as an *invalid edge*. To show why this is true for every type of invalid path, we consider three cases.

Case 1: The invalid path contains a spoofed edge. If a path contains a spoofed edge, then clearly that edge is invalid and no AS should receive an update containing such an edge.

Case 2: The invalid path violates policy. For this case, we assume that two ASes do not have a provider-customer relationship for some prefixes and a customer-provider relationship for others. Lixin Gao [7] showed that any path that violates policy must involve propagating a route learned from a provider or peer to another provider or peer. A route learned from a provider or peer contains the directed edge (C, P) in which AS C learns a route from its provider or peer P. As shown by the export rules given in Table 1, edge (C,P) should only be exported to C's customers and C's siblings. In turn, C's customers and siblings should only export the route to their own customers and siblings. All of the ASes that can legitimately receive the edge (C,P) are hereby referred to as C's children. In a policy-violating path, an edge of type (C,P) is exported to a non-child of C, which should not occur. Therefore edge (C,P) will be invalid to the peer or provider that receives edge (C,P) in an invalid path. Note that the edge direction matters; many routers could legitimately receive routes with the edge (P,C) when C announces a prefix, but only C's children should receive edge (C,P).

Case 3: The invalid path contains a spoofed AS number. When an adversarial AS A exports its spoofed route to its neighbors, the edge (Peer, A') (where A' represents the spoofed AS number) would be spoofed unless the victim AS has the same neighbor. In the case that a spoofed edge is formed, such an edge would never legitimately be propagated to its recipients, and thus it is invalid. Otherwise, no invalid edge exist in the path. As difficult as it is to perform an AS number spoofing attack, it is even less likely to occur with the victim's neighbor because the neighbor would risk a financial relationship with the victim and the neighbor can identify the adversary as fraudulent.

The analysis of these three cases shows that invalid paths contain invalid edges except in AS number spoofing cases in which the victim and adversary share a neighbor.

3.2 Detecting Invalid Paths

As the analysis of the three cases shows, an anomaly detector that can detect all invalid edges will also detect all invalid paths save for some AS number spoofing cases. As a proxy for identifying invalid edges directly, we adopt the PGBGP philosophy of treating all new information cautiously. In the case of edge detection this means that paths that contain new edges are depreferenced for a period of time known as the suspicious period s. We assume that all recently seen edges (that is, edges received as part of an update or that reside in the routing table within the last h days) are valid. New edges (Section 5 shows that these are rare) automatically become valid after they have remained in the routing table for s time.

There is a case in which our detector will fail to recognize an invalid edge if PGBGP is not ubiquitously deployed. In reference to Case 2 above, C's children will not recognize policy violations in which (C,P) is the invalid edge by monitoring for new edges because C's children can legitimately see edge (C,P). An example of this case is shown in Figure 2. Under full deployment, AS P' would detect the violation and not propagate it to AS A. We believe that this is a minor concern because all of the policy violating path's edges are edges that the child could legitimately use and our results in Section 4 show that this case does not inhibit PGBGP's ability to detect and stop the propagation of policy violations.

Upon detection of a suspicious edge, the system relies on the legitimate owners of the two ASes (on either end of the edge) to identify and remove the invalid path before s expires. An external monitoring service can warn the two ASes (and any others who might be interested) about the presence of a route with a suspicious edge, similar to earlier monitoring systems that warn an AS if its IP prefix has been hijacked. To ensure that the relevant operators are informed of each suspicious edge, the Internet Alert Registry [12] could be extended using this algorithm to monitor new edges and alert operators from both ASes for each suspicious edge. Alerts are sent to registered operators as emails, which can then be automatically parsed and compared to internal registry information. If the edge is invalid, the legitimate ASes can take actions to fix the problem, such as contacting other ASes in the path to have them filter the offending route. If the adversarial AS refuses to withdraw the route, then the victim AS can ask the rest of the operational community (easily reached through mailing lists [21, 2]) to pressure the offending AS to comply.

To summarize, the detection (identify new edges) and response (depreference for s time) algorithm is straightforward. In fact it is simple enough that the pseudo-code for the important components is provided in Appendix A. To initialize a BGP router with a valid edge database, all observed directed edges in update messages and the router's RIB are treated as valid for the first h days. After that time, directed edges found in updates that do not exist in the valid database are labeled as suspicious and depreferenced for s time. After s time (time for operators to respond to alerts), if the edge exists in the RIB then it is accepted into the valid database. In case of router downtime, the database can simply be stored and reused when the router comes back online, or in the case where it is offline for an extended period, it could use another local router's database or create a new one.

3.3 Algorithm Overhead

To be immediately useful, our system should run on today's routing hardware. In Appendix A we detail the necessary data structures for our enhancement to PGBGP. It shows that each edge in the database requires a 12 byte struct plus 8 bytes for the edge. For a full listing of roughly 50,000 edges, that would only require 977KB of memory. There is also a small amount of overhead for the map container and the queue used to restore the preference for suspicious routes after 24 hours.

Our algorithm's processor usage is also low. The running time per update is $O(elog(|E|))$ where $|E|$ is the number of edges in the edge database and e is the number of edges in the update's path.

4. SIMULATED ATTACKS – DETECTION AND RESPONSE

Any new version of BGP software is likely to be adopted incrementally. The experiments in this section quantify the effectiveness of our solution when only a subset of ASes participate. We simulate the propagation of both valid and invalid paths on the AS graph, based on routing policies derived from the business relationships between ASes.

4.1 Experimental Setup

To simulate PGBGP's defenses against attack, we used the BSIM [14] simulator. BSIM is a route propagation simulator under the GPL license. It loads a user-specified topology (complete with inferenced relationships) and models the propagation of route announcements across the network according to the export rules defined in Section 2. For our simulations we used the topology and relationships provided by the CAIDA AS Relationships Dataset [31] built on the 2nd of February 2007. The inferred topology describes 48,986 edges between 24,267 ASes.

We modified BSIM to support the PGBGP enhancements for invalid edge detection (the original PGBGP was already implemented in a modified BSIM). For the remainder of the paper, we refer to a router that has both the original PGBGP (for invalid AS origin attacks) and our edge monitor extensions enabled as a "PGBGP-enabled router." The pseudo-code for the algorithm we implemented is given in Appendix A, and our extensions to BSIM are available online [9]. We also implemented an idealized *perfect detector* in the form of a black box anomaly detection option, which is available to each simulated router. The black box detector discards all invalid routes, and never makes a mistake. Finally, we added all of the attack scenarios described in Table 2 into BSIM.

Within BSIM, an attack is simulated in two steps: initialization and attack. To initialize the network, each router's BGP routing table is cleared. At initialization time, the victim AS announces its address blocks to prime the history-based registry of each PGBGP-enabled router. For the second step, BSIM moves forward to time h, and the adversarial AS announces an invalid (bogus) route to steal the victim's traffic. After the routers have selected their best routes, ASes that select a path that includes the attacking AS are counted as having been hijacked. Note that before the attack, the adversary AS might have legitimately provided transit service for the prefix. For simplicity, we consider all routes that include the adversary's routers after the attack to be invalid. The adversarial and victim ASes are distinct and chosen uniformly at random from the network.

Our experiments report attack effectiveness—the fraction of ASes that erroneously select a route through the attacker—for varying levels of PGBGP deployment. In these experiments we systematically deploy PGBGP in ASes in order of decreasing node degree, starting with the AS with the most neighbors. This is because we believe that it would be easier to convince a small number of large ASes to adopt a new solution than 10,000 smaller ones. The large service providers are known to upgrade their routing software more frequently and follow the latest trends and best common practices. In addition, the larger ASes have greater influence on the Internet's security, because they have more customers to protect.

4.2 Unprotected Networks

We have simulated runs of the four forms of attack described in Section 2 as well as the two origin AS attacks described in the original PGBGP work on a completely unprotected BGP network. None of the routers perform any ingress filtering and none of them run PGBGP. This provides us with an

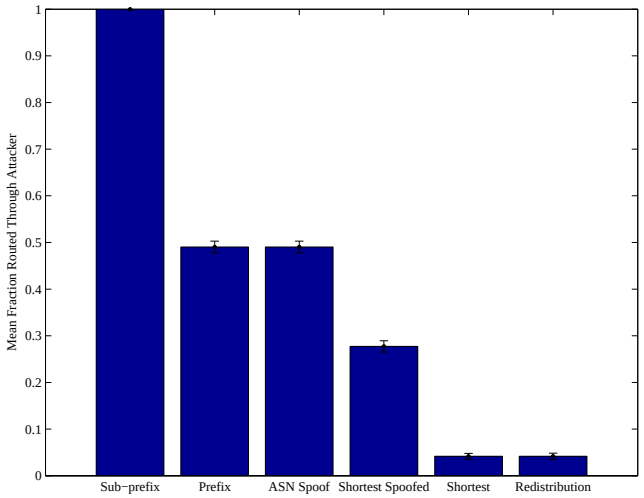

Figure 3: Effectiveness of each synthetic attack against a network of ASes without PGBGP deployed and no ingress filtering using the standard export rules. The x-axis describes the form of attack simulated while the y-axis represents the fraction of ASes that routed through the adversarial AS after 500 simulated attacks. Error bars represent standard error of the mean.

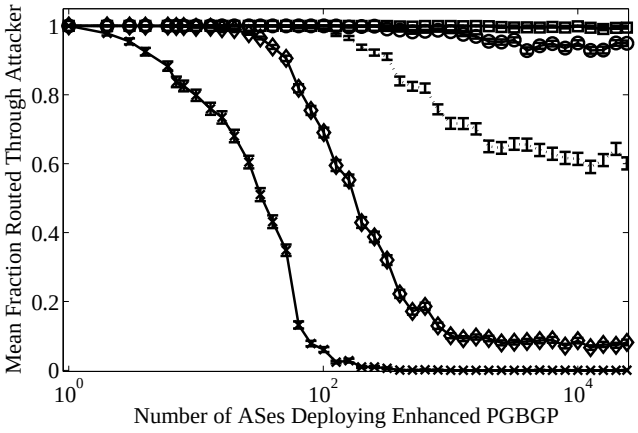

Figure 5: Effectiveness of each synthetic attack against a network of ASes over time. The X-axis is log-scaled (and shifted up by one to show $x = 0$) and represents the number of ASes that have deployed the security solution. The Y-axis is linearly scaled and represents the number of ASes that have selected a route that includes the adversary's AS after convergence. The bottom plot represents the time of the attack and each subsequent plot above it represents an increase of s simulated time. Error bars show the standard error of the mean.

upper bound on how damaging each attack could be. The results are shown in Figure 3, where the x-axis shows the type of attack, and the y-axis is the fraction of ASes that selected a route which traverses the adversarial AS. Surprisingly, policy-violation attacks (shortest path and redistribution, as summarized in Table 2) are not significant threats. Since a customer AS may have many providers, which in turn have have many large providers, and each of these providers prefer routes from customers, it seemed possible that such attacks could be quite significant on average. Instead, on average the adversary in each attack only convinced roughly 4% of the network to route through it. This is likely due to the extreme lengths of the adversary's paths.

We had expected that the shortest fake edge attack would not be as successful as a prefix hijack. This is because the fake edge attack has a longer path and is less likely to be selected. However, duping roughly 27% of ASes still poses a significant security threat. Finally, ASN spoofing is shown to be equivalent in impact to a prefix hijack. This is an expected outcome as the paths are of equal length and unprotected networks do not verify AS numbers.

4.3 PGBGP's Effectiveness

Here we show how effective PGBGP could be at stopping the attacks with a partial deployment. Figure 4 displays both PGBGP and the black box filter's ability to thwart the propagation of bogus routes. The x-axis shows the number of ASes (out of 24,267 total) that have deployed PGBGP, in order of decreasing node degree, and the y-axis shows the fraction of ASes that have chosen routes which pass through the adversarial AS. Though the PGBGP mechanism depreferences routes while the black box filter actually discards them, our results show that there is negligible difference when used on large ASes with many alternate routes available to them. This suggests that PGBGP's depreferencing mechanism could be as effective as discarding routes when deployed, while making the system robust to the occasional false positive.

For most attack scenarios, a PGBGP deployment on 100 (0.4%) of all ASes would suffice to protect the entire Internet from both invalid path and origin AS attacks. The same number is required for the perfect detector.

4.4 Propagation of Suspicious Routes

If a suspicious route is not corrected within time s, it will be considered normal by the PGBGP routers and spread to the next level of protected ASes. We show in Figure 5 how suspicious routes spread as simulated time increases for a subprefix hijack. The other simulated attacks have similar results which are not shown due to space limitations. Each line within the plot represents a simulated period of time s. The bottom line represents the initial announcement of the bogus route and each successive line above that line represents one extra simulated time period of s. Our simulations suggest that it could take three time periods on average for the route to fully propagate with an effective deployment of 100 large PGBGP ASes.

Figure 5 does not represent the necessary amount of time for a legitimate route to propagate. This is because the deprefrencing mechanism does not affect prefix reachability. If all paths for a prefix include suspicious edges, then the suspicious edges will propagate at normal speed. Therefore new edges (e.g., backup links) which become available due to link failure would not be hindered by PGBGP.

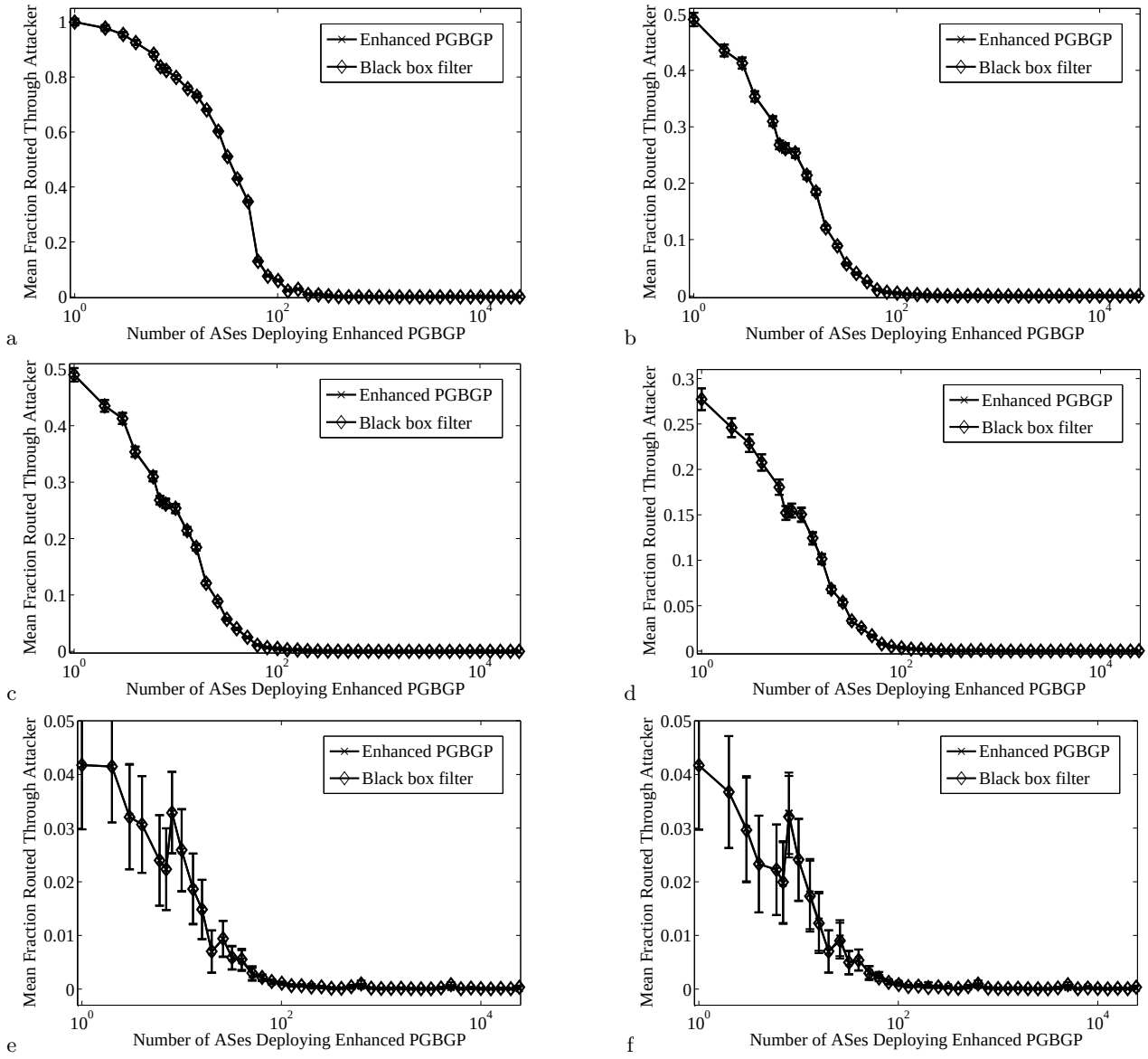

Figure 4: Effectiveness of each synthetic attack against networks protected by PGBGP on one plot and the black box ingress filters on the other. The x-axis is log-scaled (and shifted up by one to show $x = 0$) and represents the number of ASes that have deployed the security solution. The y-axis is linearly scaled and represents the number of ASes that selected a route that includes the adversary's AS after convergence. Error bars show the standard error of the mean. a) Sub-Prefix Hijack b) Prefix Hijack c) ASN Spoof d) Spoofed Edge e) Prepended Shortest Path f) Redistribution Attack

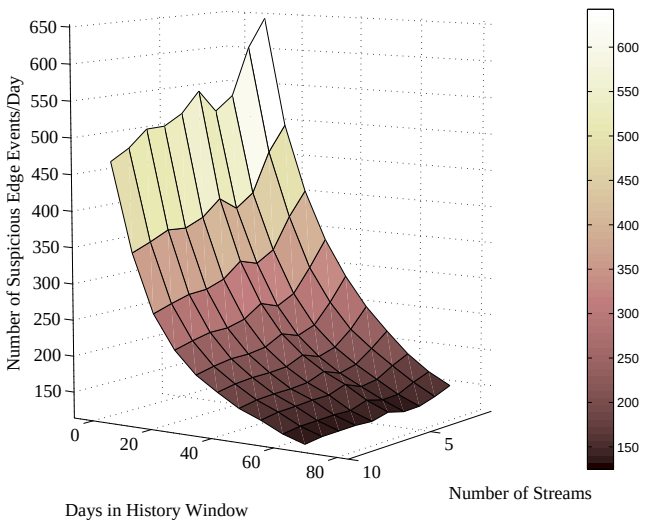

Figure 6: The number of edge events, or alerts, that our edge monitoring system observed during the 4 month time-period. The initial h days were used to initialize the normal database. The figure represents a parameter sweep of the number of BGP streams and the duration of the history period h.

5. MEASURING THE RATE OF POSITIVES

The frequency of suspicious edges depends not only on the likelihood of attacks, but also on the routing dynamics that cause an AS to see new *legitimate* edges. In this section, we quantify the rate at which suspicious edges are found, by analyzing feeds of BGP update messages.

5.1 Experimental Setup

Predicting the number of suspicious edges an AS would see is difficult without access to feeds of its BGP update messages. Instead, we ran our edge-monitoring algorithm against four months worth of publicly available BGP updates to get a rough estimate of how many routes might be labeled as suspicious per day based upon the size of the network (number of update streams) and history length. Since the Internet Alert Registry also relies upon feeds from public databases, the results from this experiment represent the number of suspicious edges that the IAR identifies per day and the how many alerts an AS can expect to receive based upon its size. By varying the number of BGP feeds (neighbors) and value of h (the history window) we show that larger ASes (those of high degree) are likely to have fewer suspicious edges than smaller ASes and that larger history windows incur fewer suspicious routes since.

Our BGP update streams were collected from the RouteViews [29] project at the University of Oregon. RouteViews collects BGP update messages from many routers scattered around the world, including backbone routers in large ASes. Our dataset consists of all BGP updates from September 1st 2006 through December 31st 3006 inclusive from the routeviews2 server, which includes over 40 BGP sessions.

We measured the rate at which suspicious events occurred over the four-month period with varying h values and number of sessions (neighbors). A suspicious event is the group of updates that are labeled suspicious due to the same new edge. Each suspicious event corresponds to a single alert from the Internet Alert Registry. To vary the number of BGP sessions available in the data set, we filtered out all but the necessary number of streams by removing those streams with the lowest number of updates over four months. The first h days were used to prime the historical registry, and the remaining days were used to monitor for suspicious updates. The program used to perform this measurement is available under the GPL license at the project's website [9].

5.2 Number of Suspicious Edges

Our anomaly detector has two tunable parameters, h and s. The history window h determines how recently an edge must be observed to be "trusted," whereas s determines how long a new edge is considered "suspicious." The original PGBGP work suggested a 24-hour suspicious period, which we also suggest. However, the original PGBGP work suggested an h value of 10 days for origin AS detection and we suggest a longer window for edge detection.

Figure 6 shows the number of suspicious edge events (new edges) seen per day with varying h values and number of neighbors. Fortunately, as the number of neighbors increases, the number of suspicious events decreases. This is likely because, over time, the router is exposed to more legitimate edges as routes change. Since we suggest deploying PGBGP on the largest ASes with the most neighbors, this will be beneficial. Similarly, as the length of h increases, the number of events decreases. Our analysis suggests that the history window for the edge detector should be set to 60 days (roughly two months).

With a value of h at 8 weeks and BGP 10 sessions, there are an average of roughly 150 events per day. Which means that for any given day, the router will try to avoid approximately 150 edges. Thankfully, most of these suspicious edges should come from the router's least profitable links (provider and peer-peer), since the majority of edges do not belong to customers of the router's network. Therefore our enhancement to PGBGP should not have a significant deleterious effect on profit.

Finally, we show that even though the alert rate for suspicious edges is relatively high, the number of alerts that each AS would receive from a monitoring service, such as the IAR, is very low. Analysis of our four-month stream of data (with 10 BGP sessions and $h = 60$) shows that the average AS would have received 0.01 alerts per day with a standard deviation of 0.075 for suspicious edges. Large ASes, such as the "Tier 1" providers (AS numbers 1668, 7018, 3549, 3356, 701, 2914, 209, 3561, and 1239) would have only received 2.1 alerts per day (with a standard deviation of 1.1). On the actual IAR, which monitors more than 10 sessions, these number would likely be even lower.

6. RELATED WORK

Improving BGP security has been an active research area in recent years. Some solutions use cryptographic signatures to authenticate BGP update messages [15, 20, 33, 11], while others rely on anomaly detection [30, 35, 16, 34, 10] or

introduce verification services [8]. In this section, we discuss the techniques most relevant to PGBGP.

Many solutions, such as SBGP [15], soBGP [20], and psBGP [33], require the use of an authenticated registry. If such registries were created and trusted by the network community, then our detector could use the registry information when available and use its own historical database for the rest of the network. Secure Origin BGP is similar to PGBGP. It first validates the origin AS of an update against its learned signature based registry of prefix ownership. It then uses a registry of known edges and policies to verify that the remainder of the path adheres to the list of known edges and does not violate any published policies. PGBGP also makes use of prefix ownership and edge registries but its databases are based on locally acquired historical data, rather than a cryptographically verifiable database.

Kruegel *et al.*'s [16] topology detector utilizes out-of-band techniques that we could also use to reduce the number of positives, at the risk of increasing false negatives. Their detector queries WHOIS registries to determine the geographic location of networks and ensures that the AS paths meets a list of proximity criteria. For instance, neighboring small networks should be physically close. The Hi-BGP anomaly detector [22] is one of the first BGP security protocols to focus on policy violations. During initial deployment, Hi-BGP depends upon a historical registry of known edges much like PGBGP. However Hi-BGP uses relationship inferencing methods to determine policy violations as opposed to PGBGP's simple edge detection. Also, Hi-BGP does not automatically prevent the propagation of suspicious routes. Instead, Hi-BGP alerts the operator after a delay period, allowing short attacks to propagate and finish before the operator is even made aware of the problem.

7. LIMITATIONS OF PGBGP

Our proposed enhancements to PGBGP provide a safer environment for the BGP network, but it still remains possible to subvert its security. In this section we list all of the known vulnerabilities our detector has when integrated with PGBGP.

Insecure Data Plane Like most BGP security mechanisms, PGBGP only protects the routing control messages (control plane), and does not verify that the traffic actually traverses the announced route (data plane). Hu *et al.* have started to explore the area of data plane route verification [10] by measuring destination characteristics such as the destination host OS, IP identifier probing, and TCP timestamps. Such techniques could be used to reduce the number of false positives in PGBGP.

Dirty Data PGBGP relies upon attentive operators. Not all operators will wish to monitor their address space and neighboring links and as such will have unsafe networks. We have shown in Section 5 that there are very few alerts that operators will have to deal with and the alerts are trivial to receive. If the adversarial AS has been contacted during the suspicious period but refuses to amend the problem, it will remain up to the adversary's providers and the operational community to prevent the bogus routes from propagating.

Under extreme circumstances, it is possible for a suspicious route to propagate unheeded by PGBGP, though alerts would still be distributed. This would occur if only suspicious routes were available for the prefix. For this to happen, the adversary would have to first cripple the victim's network by disabling its neighboring links, a noticeable event.

Collusion If two adversarial ASes agree to forge an edge between themselves (over a multi-hop path), known as a worm-hole attack, PGBGP will eventually allow the edge to propagate because the ASes will ignore IAR alerts. If a third party were to monitor these ASes edges it could be brought to the operational community's attention. The cryptographic security solutions are also vulnerable to worm-hole attacks.

Mixed Relationships If two ASes have both a customer-provider and a provider-customer relationship, PGBGP could miss a policy violation involving that edge. For instance, in North America AS A might be AS B's provider, but in Europe AS A could be B's customer. Both directed edges (A,B) and (B,A) could regularly be seen by non-children of A and B and might not be detected by PGBGP. Generally such a relationship mixture is rare, you would instead see a customer-provider and peer-peer mixture which PGBGP can detect.

Potential DoS Our invalid path detector is currently vulnerable to a potential denial-of-service attack. If an adversarial AS were to prepend a large number of fake edges to its paths, those fake edges could fill up PGBGP's historical registry. Since each new edge requires 20 bytes of memory plus the overhead of the tree, it would take a considerable number of edges to maximize the memory usage but it is possible. Such an attack would create many alarms and could be identified quickly by the operational community.

8. CONCLUSIONS

The Internet's IP routing infrastructure has a number of critical security vulnerabilities. Existing cryptographic solutions have not been deployed because they require a public key infrastructure, community consensus, and changes to the BGP protocol. Though anomaly-detection schemes are easier to deploy, they have traditionally been unable to offer the same level of protection. In this paper, we have shown that a simple anomaly detector, coupled with an automated response to suspicious paths, can offer protection comparable to the cryptographic solutions.

We have shown through simulation that our enhancements to PGBGP can largely eliminate the effects of invalid path attacks with a deployment on the largest 0.4% of ASes. We have also shown that our scheme is nearly as effective at thwarting attacks as an idealized detector. Finally, we have shown that PGBGP is incrementally deployable because it does not require global cooperation or changes to the BGP protocol. In addition, our solution has low overhead, generates an average of just two alerts per day to the largest ASes, and could be readily included in a routing software upgrade. In our ongoing work, we are building a prototype of PGBGP as an extension to open-source software routers and exploring opportunities to deploy our solution.

9. REFERENCES

[1] RIPE whois registry. http://www.ripe.net/whois.

[2] African Network Operators Group. http://www.afnog.org.

[3] American Registry for Internet Numbers. http://www.arin.net.

[4] Asia Pacific Network Information Centre. http://www.apnic.net.

[5] V. J. Bono. 7007 explanation and apology. http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html, Apr. 1997.

[6] P. Boothe, J. Hiebert, and R. Bush. How prevalent is prefix hijacking on the Internet? *NANOG 36* http://www.nanog.org/mtg-0602/boothe.html, Feb. 2006.

[7] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Trans. on Networking*, 9(6), December 2001.

[8] 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. Network and Distributed Systems Security*, February 2003.

[9] A. hidden for blind review. PGBGP project webpage. http://hidden.com/.

[10] X. Hu and Z. M. Mao. Accurate real-time identification of IP prefix hijacking. In *Proceedings of IEEE Security and Privacy*, 2007.

[11] Y.-C. Hu, D. McGrew, A. Perrig, B. Weis, and D. Wendlandt. (r)Evolutionary bootstrapping of a global PKI for securing BGP. In *Hot Topics in Networks Workshop*, Nov. 2006.

[12] Internet Alert Registry. http://cs.unm.edu/~karlinjf/IAR/.

[13] Internet Alert Registry forums. http://cs.unm.edu/~karlinjf/IAR/phpBB2/viewtopic.php?t=30, Nov. 2006.

[14] J. Karlin, S. Forrest, and J. Rexford. Pretty good BGP: Improving BGP by cautiously adopting routes. In *Proc. IEEE International Conference on Network Protocols*, Nov 2006.

[15] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol. *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, 2000.

[16] C. Kruegel, D. Mutz, W. Robertson, and FredrikValeur. Topology-based detection of anomalous BGP messages. In *Proc. Syposium on Recent Advances in Intrusion Detection*, volume 2820, pages 17–35, September 2003.

[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*, 2006.

[18] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *Proc. ACM SIGCOMM*, pages 3–16, 2002.

[19] S. A. Misel. Wow, AS7007! http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html, Apr. 1997.

[20] J. Ng. Extensions to BGP to support secure origin BGP (soBGP). *Internet Draft draft-ng-sobgp-bgp-extensions-02*, April 2004.

[21] North American Network Operators Group. http://www.nanog.org.

[22] J. Qiu and L. Gao. Hi-bgp: A lightweight hijack-proof inter-domain routing protocol. *University of Massachusetts Amherst TR*, 2006.

[23] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proc. ACM SIGCOMM*, pages 291–302, New York, NY, USA, 2006.

[24] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), Jan. 2006.

[25] Renesys Blog. Con-Ed Steals the 'Net. http://www.renesys.com/blog/2006/01/coned_steals_the_net.shtml.

[26] Renesys routing intelligence. http://www.renesys.com/products_services/routing_intelligence/.

[27] RIPE. http://www.ripe.net/.

[28] RIPE NCC MyASN service. http://www.ris.ripe.net/myasn.html.

[29] RouteViews. http://www.routeviews.org/.

[30] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz. Listen and Whisper: Security mechanisms for BGP. In *Proc. Networked Systems Design and Implementation*, March 2004.

[31] The CAIDA AS relationships dataset. http://www.caida.org/data/active/as-relationships/, Feb. 2007.

[32] W. Leibzon. Question on 7.0.0.0/8. http://www.merit.edu/mail.archives/nanog/msg05883.html, Apr. 2007.

[33] T. Wan, E. Kranakis, and P. van Oorschot. Pretty secure BGP, psBGP. In *Proc. Network and Distributed System Security*, 2005.

[34] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, and L. Zhang. Protecting BGP routes to top level DNS servers. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):851–860, 2003.

[35] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. Detection of invalid routing announcement in the Internet. In *Proc. Dependable Systems and Networks*, 2002.

APPENDIX
A. THE ALGORITHM

```
#define HISTORY_WINDOW 60 // days
#define SUSPICIOUS_PERIOD 24 // hours

struct Usage {
    uint32_t TimeLastSeen;
    uint32_t NumberInRib;
    uint32_t DepreferencedUntil;
    Usage() : TimeLastSeen(0), NumberInRib(0), DepreferenceUntil(0) {}
};

typedef pair<uint32_t, uint32_t> Edge;
map<Edge, Usage> EdgeDB;


void OnAnnouncement(uint32_t time, Route route) {
    uint DeprefUntil = 0;

    // Has any edge already been depreferenced?
    foreach edge in route.path {
        if (EdgeDB.count(edge) == 0) continue;
        if (EdgeDB[edge].DepreferencedUntil > time &&
            EdgeDB[edge].DepreferencedUntil > DeprefUntil)
            DeprefUntil = EdgeDB[edge].DepreferenceUntil;
    }

    // Are there any new edges?
    foreach edge in route.path {
        if (EdgeDB.count(edge) == 0) {
            EdgeDB[edge].DepreferencedUntil = time + SUSPICIOUS_PERIOD;
            DeprefUntil = time + SUSPICIOUS_PERIOD;
        } else {
            if (EdgeDB[edge].NumberInRib == 0 &&
                EdgeDB[edge].TimeLastSeen + HISTORY_WINDOW < time)
            {
                EdgeDB[edge].DepreferencedUntil = time + SUSPICIOUS_PERIOD;
                DeprefUntil = time + SUSPICIOUS_PERIOD;
            }
        }
    }
    // Depreference the route
    if (DeprefUntil > 0) {
        route.localpref = 0;
        ReprefRouteOnTimer(route, DeprefUntil);
        EraseEdgesNotInRIBOnTimer(route, DeprefUntil);
    }

    // Add usage information to DB
    foreach edge in route.path {
        EdgeDB[edge].NumberInRib++;
        EdgeDB[edge].TimeLastSeen = time;
    }
}

// Decrement counts for withdrawn or replaced routes
void OnExitFromRIB(uint32_t time, Route route) {
    foreach edge in route.path {
        EdgeDB[edge].NumberInRib--;
        EdgeDB[edge].LastSeen = time;
    }
}
```

Figure 7: Sample code for our edge monitoring and response mechanism. Some functions are removed for brevity.