An Incrementally Deployable Protocol for Learning the Valid Incoming Direction of IP Packets

Toby Ehrenkranz and Jun Li
{tehrenkr,lijun}@cs.uoregon.edu

ABSTRACT

Routers in today's Internet do not know which direction a packet with a given source address should come from. This problem not only allows IP spoofing to run wild—as routers cannot check the validity of a packet's source address based on its incoming direction—but also hinders the reliability of many source-relevant functions at routers, such as per-source fair queuing, source-based traffic management, source-based congestion control, or reverse path forwarding.

This research designs and evaluates an incrementally deployable protocol, **ID-SAVE**, that enables a subset of routers on the Internet to learn the valid incoming direction of packets from each other. With such knowledge, these routers can check whether a packet is from a valid direction based on its source address, thus determining whether the source address of the packet is valid—even when not all routers employ this new protocol. ID-SAVE not only makes source-based functions more reliable, but also addresses the root cause of IP spoofing prevalence. The evaluation also shows that ID-SAVE is effective and accurate in catching spoofed packets while incurring a low overhead.

1. INTRODUCTION

In the current Internet, there is no facility available for routers to learn which direction a packet with a given source address should come from. The lack of this "incoming direction" knowledge is the root cause of not only IP spoofing prevalence, but also the difficulty in reliably performing various source-based network functions, such as congestion control, fair queuing, and source-based traffic control schemes. Research [1, 2] has shown that IP spoofing is not only possible, but also still prevalent. Unfortunately, most studies so far focus on the problem—typically IP spoofing—through methods including spoofing detection and packet tracing (Section 2), instead of the cause of the problem—routers' lack of incoming information knowledge.

The only protocol previously proposed that provides incoming direction information is SAVE [3]. However, the system requires full deployment (i.e., every router in a network runs the protocol)—an unrealistic requirement in today's Internet.

Any solution must be incrementally deployable to be really useful. With only a subset of routers ready to run a new protocol, how to learn and utilize incoming information becomes a much bigger challenge than that in full deployment. In SAVE, *every* router can initiate updates—especially when routing changes occur—to notify other routers about the correct incoming direction of packets. However, when not every router participates, delivering such updates can fail since these updates are not simply data packets; and worse, routers which experience routing changes but do not participate will not initiate updates whatsoever, leaving incoming direction knowledge at participating routers obsolete.

Implementing an incrementally deployable solution is even more difficult considering such a solution should: offer strong deployment incentives; be independent from specific routing protocols; and operate efficiently and effectively no matter how an attacker spoofs IP packets or where routing changes occur in the network.

In this paper, we present a new protocol ID-SAVE (Incrementally Deployable Source Address Validity Enforcement) to tackle all these issues. We elaborate on its design, as well as evaluation. ID-SAVE is unique compared to other works in that it attacks the *root* cause of receiving packets with spoofed source addresses: routers do not always know the correct incoming direction for a given source address. As opposed to treating the symptom by detecting spoofed packets when they reach their destination host or destination network, ID-SAVE provides routers with the information they need to catch spoofed packets, often much closer to the attacker. Furthermore, by going after the root cause, ID-SAVE's benefits extend beyond simply detecting spoofed packets; the incoming direction information can also assist source-based protocols.

The rest of this paper is as follows: Section 2 describes related work, highlighting the compelling need of ID-SAVE. The design of ID-SAVE is covered in Sections 3 and 4, with certain issues further considered in Section 5. Section 6 evaluates ID-SAVE's correctness, efficacy, and overhead. We cover several major open issues in Section 7, and conclude our paper in Section 8.

2. RELATED WORK

Much network security research has focused on applying cryptographic operations in order to guarantee authenticity of packet information. IPsec is one representative at the IP layer [4]. The high computational cost of cryptographic operations prevent such approaches from being widely employed per packet.

Outside of cryptographic research, source address validity enforcement includes end-host methods and router-based methods. A variety of end-host-based detection methods can be found in [5]. These methods can be further classified as active or passive, depending on if end hosts actively probe to determine source address validity or if they simply observe the incoming packets. End-host detection is easier to deploy, but in general, routers are at the root of the problem. Router-based solutions include both preventive approaches and reactive approaches. Filtering is a preventive approach. Tracing is primarily reactive. We focus on router-based solutions in the rest of this section.

Baker [6] proposes a general filtering approach where many fields, including but not limited to the source address, can be used for filtering. Martian address filtering allows routers to discard packets if their source addresses are special addresses (loopback address, broadcast address, etc.) or are not unicast addresses.

Forwarding-table-based-filtering is one possible approach to validating source addresses [6]. It assumes that the outgoing interface that a router uses to reach a given address, as specified by its forwarding table, is the valid incoming interface for packets originating from that address. Unfortunately, routing asymmetry on the Internet is common [7], invalidating this assumption.

At the border of a stub network, both ingress [8] and egress [9] filtering ensure packets from the stub network have valid source IP addresses. However, research shows that unless such filtering is deployed almost everywhere, it is ineffective in preventing spoofing attacks [10].

SPM [11] utilizes keys associated with source-destination AS pairs. Unlike ID-SAVE, it cannot be used for router-based services that rely on source address validity. Designed only for spoofing prevention, SPM is specific to BGP and benefits participating ASes at the AS level. Perhaps most distressing, if attackers learn the correct key for a source-destination pair, they can successfully send spoofed packets from *anywhere* in the network without raising suspicion.

Hop-count filtering [12] relies on spoofed packets traveling over a different number of hops than legitimate traffic. Unfortunately, with only a small range of possible hop counts, it has limited efficacy.

Route-based distributed packet filtering (DPF) suggested in [10] studied benefits of DPF filtering for attack prevention and traceback, and partial deployment strategies. Unfortunately, the work assumed the existence of a DPF system without actually designing an approach for routers to learn the correct route for every source address. There have been other DPF-like works recently proposed, including IDPF [13] and BASE [14]. However, besides being very BGP-specific, IDPF only learns *feasible* paths, not *actual* paths. BASE is also tightly tied to BGP, and it cannot handle AS-level routing asymmetry well.

Finally, packet tracing has been widely studied [15, 16, 17], and often involves packet marking [18, 19, 20, 21, 22, 23, 24]. While complementary to each other, a fundamental difference between ID-SAVE and tracing is that tracing is typically performed after an attack is detected, possibly too late to avoid damage! Tracing IP packets with forged source addresses requires complex and often expensive techniques to observe the traffic at routers and reconstruct a packet's real path. Many tracing methods become ineffective when the volume of attack traffic is small or the attack is distributed [25].

Pi [26] and StackPi [27] mark each packet to identify the path that it traveled. End hosts use the path identifiers to filter out packets which travelled along an identified attack path. Detecting attack paths, and dropping attack packets is up to the end hosts. Routers cannot filter attack packets, nor can they discover incoming direction information.

3. DESIGNING ID-SAVE

3.1 Design Principles

The key of ID-SAVE is to learn valid incoming directions of packets. Using routing protocols to decide the path from every source to every destination, routers determine not only the outgoing direction for every destination, but also, albeit indirectly, the incoming direction of an IP packet from every source. Although outgoing directions are recorded in every router's forwarding table, the incoming direction knowledge is yet to be discovered.

In order for a protocol such as ID-SAVE to be successful, it must adhere to the following principles:

- It must be independent of underlying routing protocols, so that it can easily run on top of any of them.
- It should respond to routing changes and adjust incoming direction information in a timely manner.
- It must be lightweight in order to minimize router overhead and scale well.
- It must be easy to deploy, and provide benefits immediately.
- It must be secured or attackers could bypass any security it offers or even abuse the protocol.

We should also note that in this paper only end hosts will attempt attacks on the system; routers are not

malicious. Without this assumption, malicious routers would be able to wreak havoc on the Internet irregardless of ID-SAVE.

3.2 Utopian Solution

Understanding a utopian solution that assumes full deployment, as described in [3], can make it easier to understand the incrementally deployable solution we propose as ID-SAVE. We briefly present one such full deployment solution in this section.

Every router that is in charge of forwarding packets from a specific source address space (which can be configured as one or several address prefixes) periodically initiates updates on behalf of that space. For each address prefix in its forwarding table, also called a destination address space, the router will generate an update and forward it toward that space. The update contains the source address space of the originating router. Upon receipt of this update, every downstream router will (1) record the incoming direction of the update as the correct incoming direction for the source address space specified by the update—because the update traveled the same path as packets from these spaces, and (2) continue to forward the update toward its destination space, including splitting the update when the router's forwarding table has entries for *sub-spaces* of the destination space of the update.

Promptly reacting to routing changes, any change in a router's forwarding table will trigger new updates towards the destination space of the modified forwarding table entry. This helps downstream routers to update the incoming direction information for that router's source address space (a routing change may also affect the source address spaces of other routers and readers can refer to [3] for the handling of this issue).

3.3 Challenges of Incremental Deployment

While the utopian solution works in a world where all routers in the network run it, it begins to fail when there are **legacy routers**, or routers not running the protocol, in the network. The cause of the failure can essentially be broken down into two problems. Both of these problems lead to routers having outdated, missing, or incorrect incoming direction information:

- *Legacy routers do not correctly propagate, or split, updates.* While it is easy to encapsulate an update inside a normal data packet that a legacy router will forward, the legacy router will not be able to split the update when the update's destination space spans multiple forwarding table entries of the legacy router. As a result, updates may not travel along all of the paths that legitimate traffic follows. Downstream routers would not receive all of the updates from upstream routers.
- *Legacy routers do not initiate updates upon routing change.* Since legacy routers will not initiate ID-SAVE updates when their forwarding tables change, downstream routers cannot learn of the updated incoming directions. Thus, when a packet arrives from an unexpected direction, routers cannot be certain which is invalid: the router's incoming direction information or the packet's incoming direction.

3.4 ID-SAVE Solution

ID-SAVE establishes and utilizes packet incoming direction information based on the collaborative efforts of only a subset of routers in a network. The key contributions of ID-SAVE lie in two aspects: (1) successful dissemination of incoming direction information with the presence of legacy routers; and (2) correct packet handling with potentially incorrect incoming knowledge.

For the first problem raised in Section 3.3, ID-SAVE enables routers to *preemptively* split updates on behalf of downstream legacy routers. An ID-SAVE router maintains a **propagation table**, which tells the router whether or not to split an update into a finer grain than that provided by its forwarding table (Section 4.1.2).

For the second problem brought up in Section 3.3, we allow ID-SAVE routers to request updates, so they may confirm the correct incoming direction of any received packet. Routers also maintain a **blacklist** specifying the attributes of packets which ID-SAVE confirmed as invalid. A novel **pushback** mechanism assists in catching invalid packets, and in handling routing changes at legacy routers to minimize false positives. We present the details of these mechanisms in Section 4.3.

A smaller, more straightforward problem relates to the definition of *incoming direction*. Earlier work defined the incoming direction to be the incoming interface of a packet [3]. Without full deployment, however, this definition cannot effectively differentiate different incoming directions. One example case is when an ID-SAVE router receives packets from different paths, but all go through a legacy router right before reaching the ID-SAVE router. ID-SAVE redefines the incoming direction of a packet to be the packet's previous ID-SAVE-hop, or the **p-hop**. The p-hop is a powerful and logical concept; as deployment percentages increase, the p-hop reduces to the previous physical hop, or incoming interface. In order for routers to know the p-hop of a packet, an ID-SAVE router embeds its identification into the packets which pass through it. We leave the embedding mechanism as a future work.

4. ID-SAVE DETAILS

In this section we describe ID-SAVE's design in more detail. First, we describe the data structures and methods that ID-SAVE uses to communicate incoming direction information. Then we describe how ID-SAVE stores and maintains this incoming direction information. Finally, we present the methods for catching and

handling invalid packets. (Section 5 will address additional design issues related to deploying ID-SAVE.)

Note: *For the remainder of this paper "router" will refer to "ID-SAVE router" unless otherwise noted.*

4.1 Communicating Incoming Direction Information

4.1.1 Updates

An **ID-SAVE update**, on behalf of a set of source address spaces, travels along the same path as data packets from these spaces, informing routers along its path of the valid incoming direction for these source address spaces.

ID-SAVE routers send updates in essentially the same way as routers from the utopian solution (Section 3.2). Every update has a **destination address space** field that specifies the final destination address space of this update. It also has an **address space vector (ASV)** field to record a chain of **source address spaces (SAS)** and router IP address pairs that are associated with the ID-SAVE routers crossed by the update. A router's source address space includes all address spaces for which it is a source router. In general, if an update originated from R_1 crosses ID-SAVE routers R_2, R_3, ..., and R_n, the ASV field will have the form $< (S_1, r_1), (S_2, r_2), \cdots, (S_n, r_n) >$, where S_i and r_i are the source address space and IP address of R_i, respectively. Any data packets that are from address space S_i will cross R_{i+1}, R_{i+2}, ..., and R_n (and perhaps routers beyond R_n) to reach the destination space.

4.1.2 Update Propagation

As discussed in Section 3.3, legacy routers cannot assist in splitting updates. In order to solve this problem, ID-SAVE routers *preemptively* split ID-SAVE updates. To do so, every ID-SAVE router maintains a **propagation table** to keep track of spaces which need to be split into a finer grain than provided by the forwarding table. For every update that a router issues or propagates according to its forwarding table, it checks the update's destination address space against the propagation table. For every address space β in the propagation table that this destination address space contains, the router creates a new update with β as the destination address space (with the other fields unchanged), and sends it toward β.

For a router to populate and maintain its propagation table, it uses a probing mechanism. Once a router discovers another ID-SAVE router is downstream toward a specific IP address, the former will intelligently probe addresses surrounding that address in order to discover the *largest destination space* for which the latter router is guaranteed to be downstream. More specifically, the probing process begins when an ID-SAVE router, say Q, discovers (through ID-SAVE updates or by using the p-hop embedded in packets) that it is downstream toward a destination address, say d, of an upstream ID-SAVE neighbor, say P. Q notifies P that it has a new neighbor downstream towards d. Then, P sends a probing message towards $d/32$ (an address range of size 1, or the IP address of d). When P receives Q's response, it knows that Q is at least downstream towards that single address, d. P then adds $d/32$ to the propagation table, and sends another probe—this time towards $d/31$ (an address range of size 2, including d and one new address), sending the probe towards the "new" address. Each time Q responds, P overwrites the propagation table entry with the new, larger address space, and sends a new probe towards an address range twice as large as the last (making certain to send the probe to an address in the "new" half of the address range). P stops probing when either Q stops responding, or the address range that P would probe is larger than that associated with P's forwarding table entry for d.

This packet-driven neighbor discovery and probing process leads to a very simple bootstrapping process: there is none. A router simply initiates and forwards updates according to its forwarding table, waiting for downstream routers to request propagation table probes.

4.2 Storing Incoming Direction Information

After receiving updates containing incoming direction information, a router uses an incoming tree and incoming table to keep this information.

4.2.1 Incoming Tree

The **incoming tree** at an ID-SAVE router is to derive the incoming table of that router. Every node on the tree represents a source address space and maps to the valid incoming direction for that address space (except the root of the tree). The incoming tree is also key to maintaining the relationship of different source address spaces. It has the following properties:

- If—and only if—an update crosses router E and then router D before later reaching a router A, node S_E will be the child of node S_D on A's incoming tree. Here S_E and S_D are E's and D's source address space, respectively. Note that other ID-SAVE routers could separate D and A, but only legacy routers can separate E and D.
- Because of the first property, the incoming direction a parent node maps to determines—and is the same as—the incoming direction a child node maps to.
- Recursively applying the second property, all nodes from a sub-tree directly below the root map to the same incoming direction, making building an incoming table even more straightforward.

Consider the incoming tree of router A in Figure 1.

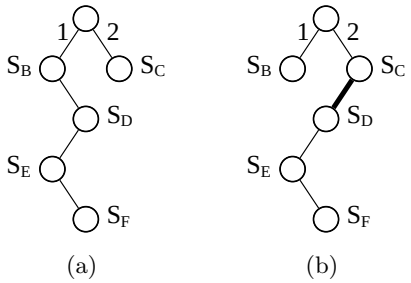


Figure 1: Router A's incoming tree before (a) and after (b) routing change at router D.

Each node S_i represents the SAS of router i. The incoming tree before a routing change at D is shown in Figure 1(a), where S_D is the parent of S_E, and S_E is the parent of S_F. The routing change at D triggers an update for S_D, which will cross C before reaching A, causing A to modify its incoming tree so that S_D becomes the child of S_C, and all source address spaces of D, E, and F will now correctly map to incoming direction 2 (Figure 1(b)).

4.2.2 Incoming Table

The **incoming table** contains the incoming direction information for protected source address spaces. It does not contain more information than the incoming tree, but it is built to be faster and more compact than the incoming tree. Recall routers are highly optimized for table lookup functions. Furthermore, in cases where routing symmetry exists, ID-SAVE takes advantage of this symmetry: If the *incoming* direction *from* a given address is the same as the *outgoing* direction *to* that address, the router does not need to store the information twice. Instead of creating incoming table entries for *every* source address space, a router only creates entries for those whose incoming direction is different from the outgoing direction. The router can set a flag in its forwarding table to indicate which address spaces have entries in its incoming table. This way, when looking for the incoming direction for an address space, the router can first check the forwarding table, and then, only if necessary, consult the incoming table.

4.3 Catching and Handling Invalid Packets

With the introduction of legacy routers, packet classification in ID-SAVE becomes more difficult. Routers cannot assume they have complete or up-to-date information (Section 3.3) and easily classify packets as either valid or invalid. Using their incoming tables and blacklists, an ID-SAVE router classifies and acts upon a packet as follows:

- *Validated*: The packet's source address corresponds to an incoming table entry, and the incoming direction (p-hop) matches. The router forwards the packet.
- *Unknown*: The packet's source address does not correspond to any incoming table entry. The router forwards the packet.
- *Illegitimate*: The packet's source address corresponds to an incoming table entry, but the packet lacks any p-hop information. The router catches the packet.
- *Suspicious*: The packet's source address corresponds to an incoming table entry, but the p-hop does not match. The packet may be invalid, or the router may have outdated information. The router forwards the packets, and requests an on-demand update.
- *Invalid*: This packet matches a blacklist entry. The router catches the packet, and issues a pushback message.

Note, we also assume that an ID-SAVE router performs ingress/egress filtering.

ID-SAVE routers can easily categorize the validated, unknown, and illegitimate packets. Suspicious and invalid packets, however, need further explanation.

When a router classifies a packet as suspicious, it is not sure if the packet is an invalid packet, or if the router itself has outdated information. Before coming to a decision, the router must confirm its information. It requests that the ID-SAVE router in charge of the packet's source address space send an **on-demand update** towards the destination of the suspicious packet. After requesting the update, ID-SAVE may or may not classify the packet as invalid, depending on which one of the following three situations occur:

- The requesting router receives the on-demand update, and it *confirms* that the incoming table was up-to-date and correct—the suspicious packet was invalid.
- The requesting router receives the on-demand update, and the incoming table was *not* correct. It updates its incoming table. If the correct incoming direction matches that of the suspicious packet, the packet was *valid*; otherwise, the packet was *in*valid.
- The requesting router does *not* receive the on-demand update. This could be because it is not on the path from the source to the destination, in which case the suspicious packet was invalid, but it could also be that either the request or the on-demand update was dropped because of a congested link. Since the router cannot know for sure, it takes no action. After waiting for the update a reasonable amount of time, the router simply requests another update if it sees another similar suspicious packet.

If the ID-SAVE router concludes that the suspicious packet was invalid, it creates an entry in its blacklist. The entry will consist of the following attributes: the spoofed source address space of the invalid packet, the destination address of the invalid packet, and the incoming direction that the router verified as invalid.

	source space	destination	p-hop
At router Q:	α	dst	P
At router P:	α	dst	-

Figure 2: Example blacklist entries. The entry at router P does not contain a p-hop.

4.3.1 Blacklists

Every ID-SAVE router maintains a **blacklist** of attributes that previous invalid packets have used, and checks every incoming packet against all blacklist entries to see if it is invalid. Using the blacklist, a router does not need to request on-demand updates for invalid packets. However, it may lead to false positives after a routing change at legacy routers, which we address in Section 4.3.3.

In addition to the above method of creating blacklist entries, routers may also add entries based on pushback messages (Section 4.3.3). Such an entry contains the source address space of an invalid packet and its destination, but no incoming direction (p-hop). Figure 2 shows two example blacklist entries.

Matching a packet to a blacklist entry can be done in two ways. The *first method* is straightforward: every attribute listed in the blacklist entry must match the corresponding packet attribute. If a blacklist entry has three attributes, all three must match; if it has two attributes, two must match. We describe the second method in Section 4.3.2.

In order to limit the storage and processing overhead, the number of entries kept in a router's blacklist can be bounded. Once the limit is reached, the router can remove the entry that has not matched a packet for the longest amount of time.

4.3.2 Catching Sophisticated Spoofing

Some attackers may not spoof a single source to a single destination. They may randomize the spoofed source if they only care about which destination address their packets reach. Or, they may randomize the destination if they want the spoofed source to remain constant—perhaps for a reflection attack. Such spoofing styles prevent the spoofed packets from matching blacklist entries using the first matching method above.

In order to improve ID-SAVE's efficacy against such attacks, we introduce a second, less rigorous, matching method. Disregarding the source address allows a router to catch spoofed packets with randomized source addresses. Similarly, disregarding the destination address allows a router to catch those with randomized destination addresses. But, will legitimate packets be misclassified if ID-SAVE simply classifies all packets that match the p-hop, along with the source or destination, as invalid? Yes! Consider the following: Q and

```
boolean match( Packet p, BLEntry bl ){
  if( (bl.phop == null || bl.phop == p.phop) &&
       bl.source == p.source &&
       bl.dest == p.dest ){
    //matching method 1
    return true;
  }else if( p.suspicious() && bl.phop == p.phop &&
            ( bl.source == p.source ||
              bl.dest == p.dest ) ){
    //matching method 2
    return true;
  }else
    return false;
}
```

Figure 3: Pseudo-code of matching a packet against a blacklist entry.

P are ID-SAVE routers, and Q is downstream from P towards dst. Q has a blacklist entry containing source address space α, destination dst, and p-hop P. When P sends a legitimate packet from its source address space γ towards dst, Q will find the packet matching the blacklist entry with this simplified classification, although the packet is legitimate.

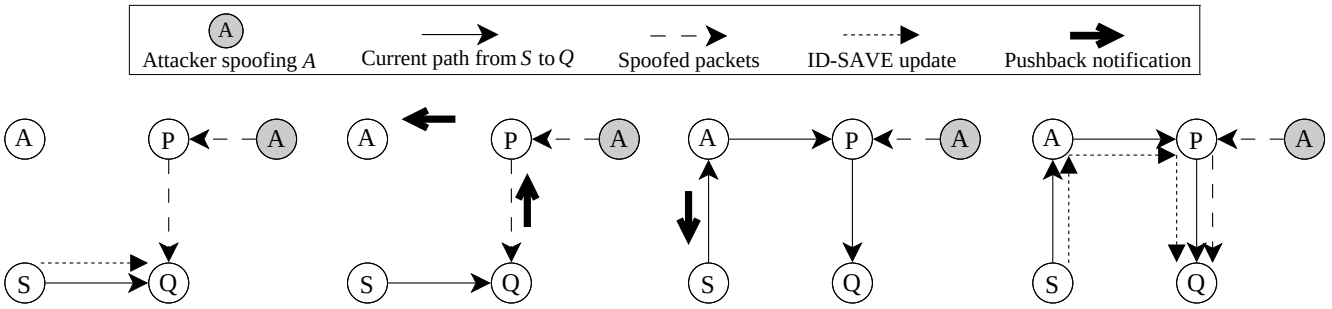
ID-SAVE's second matching method strikes a balance between being aggressive in catching spoofed packets, and taking care not to misclassify legitimate packets. In using the *second method*, not only must a packet's p-hop and either its source *or* destination match the blacklist, but it also must be a suspicious packet. Note, requiring the p-hop to match disqualifies any blacklist entry without a p-hop attribute from matching using this method. Figure 3 shows the pseudo-code of the two matching methods.

When a router receives a packet that matches a blacklist entry, using either method, it classifies the packet as invalid and initiates the pushback mechanism.

4.3.3 Pushback

ID-SAVE's **packet-driven pushback** mechanism not only enlists the help of upstream routers in catching invalid packets, as in [28], but also assists in keeping false positives to a minimum by quickly detecting when a router mistakes packets from a valid legitimate path as invalid, such as when a previously invalid path becomes valid after routing changes at legacy routers.

The pushback procedure begins when an ID-SAVE router receives an invalid packet. The router sends a pushback message to the previous ID-SAVE-hop of the invalid packet (recall that the p-hop is embedded in the packet). This message will instruct the previous ID-SAVE-hop router to add a blacklist entry that includes the invalid packet's source address space and destination address, but does not include the incoming direction. In fact, it does not matter what the incoming direction is at the upstream router—according to the on-demand update, the upstream router is not on the



(a) Q receives a suspicious packet and requests an on-demand update.

(b) Q receives an invalid packet and begins the pushback process.

(c) After a routing change, a legitimate packet causes A to continue the pushback.

(d) S discovers legitimate packets are misclassified and sends an update.

Figure 4: An on-demand update and pushback example. An attacker masquerades as router A and sends spoofed packets from α to dst. (Legacy routers are not shown.)

legitimate path from the packet's (claimed) source to its destination. The upstream router should classify all packets matching that source and destination as invalid.

Figure 4 shows an example of the pushback process. Here, router Q first requests an on-demand update when it receives a suspicious packet from P with a source from space α (S's source address space) and a destination IP address dst. Upon receipt of the requested update, Q confirms that the suspicious packet was in fact invalid (Figure 4(a)). Q then puts an entry into its blacklist that says to drop packets from α to dst with P as the incoming direction.

If Q continues to receive spoofed packets, it initiates the pushback process (Figure 4(b)). Q sends a pushback message to the previous ID-SAVE-hop router that was forwarding the invalid packets, P. The message instructs P to add a blacklist entry for all packets from α to dst, since P should not be on the path from α to dst. At this time, P and Q have the blacklist entries in Figure 2. P waits for a packet to match its new blacklist entry before continuing the pushback. Then P propagates the pushback upstream, where it reaches A. A never sees any packets matching its newly created blacklist entry, since it is not along the spoofing path of the attacker, and does not issue any pushback messages. In this manner, ID-SAVE is able to catch spoofed packets as close to the attacker as possible.

Later, if there is a routing change at a legacy router which causes the originally invalid path to become valid and legitimate packets begin to flow along the path, a small amount of false positives will occur. Router A will misclassify legitimate packets from α towards dst as invalid—but A will also relay the pushback upstream towards the source router S (Figure 4(c)). Upon receipt of the pushback, S realizes that downstream routers have blacklist entries matching legitimate traffic from α to dst. Therefore, S sends an update towards the destination address listed in the pushback notification so that all of the downstream routers along the newly valid path can erase the corresponding blacklist entry and update their incoming tables and trees (Figure 4(d)).

Note, in order to prevent reflection attacks, a router will not send a pushback message for *every* invalid packet received. After a router sends a reasonable number of pushback messages to a given upstream neighbor, the router will set a timer so that the router does not send additional pushback messages to that neighbor until the timer has expired.

5. FURTHER DESIGN CONSIDERATIONS

While ID-SAVE overcomes significant incremental deployment challenges, its design remains clean and straightforward: Routers use their forwarding tables and propagation tables to send updates, communicating incoming direction information. Incoming trees and incoming tables allow routers to easily store, maintain, and look up this information. ID-SAVE can then classify packets using a router's incoming table and blacklist. And finally, legacy routing changes are easily handled by combining the blacklist and a novel pushback mechanism.

Three important issues, all related to deploying ID-SAVE in real world, still must be addressed:

Incentive: Deploying ID-SAVE can only happen incrementally. For successful incremental deployment, domains must *want* to deploy ID-SAVE. There must be incentives that include a clear benefit for the deploying domain. "Early adopters" of the protocol should be attracted when the deployment level is still low.

The incentives for deploying ID-SAVE on a network are similar to the incentives of SPM [11]. Specifically, it will be less likely for attackers to successfully spoof source addresses belonging to an ID-SAVE-protected network. This not only protects such a network from reflection attacks, but—perhaps more importantly—it protects it from misplaced blame. A protected domain will also allow fewer spoofed packets to enter its net-

work, protecting internal hosts from receiving spoofed packets. Furthermore, ID-SAVE routers can choose to assign higher priorities to packets from ID-SAVE source address spaces, giving clients with protected sources higher quality of service.

Unlike SPM, ID-SAVE gives routers more information than simply which packets are spoofed and which are legitimate—it gives routers incoming direction knowledge. This allows for additional benefits other spoofing prevention protocols cannot offer. For instance, reverse path forwarding [29], used in some multicast implementations, could easily know the correct reverse path.

Intra/Inter-AS: Today's Internet functions as a hierarchy, both for administrative reasons and for performance considerations. Organizations prefer not to share their internal networking and routing information, yet they want to be able to communicate with everyone. Nobody wants every router to deal with the overhead of knowing about every destination space, either. The intra/inter-AS concept used in the Internet solves these problems quite well, and any protocol purporting to supply incoming direction information must also work with similar hierarchical concepts. ID-SAVE functions at both the *intra*-AS and the *inter*-AS level, while maintaining the hierarchical separation between the local domain and the AS-level network. It not only hides incoming direction information regarding the internal network from other ASes, but also hides the incoming direction information regarding other ASes from routers in the internal network. ID-SAVE's scalability also improves by running in a hierarchical fashion, resulting in less control traffic and smaller incoming tables.

The modifications needed for intra/inter-AS functionality are simple and straightforward. We assume all the border routers of an AS will deploy ID-SAVE[1] and consider all the border routers as one virtual router that uses the address space of the entire AS as its source address space. When issuing updates to (border) routers of other ASes, every border router embeds the same p-hop identification created for the virtual router, and acts on behalf of the same source address space. For internal routers of an AS, they simply know that border routers (or the corresponding virtual router) are the correct incoming direction for everything except the internal address space (this never changes no matter how packets travel outside of the AS). Incoming directions for inter-AS packets will be built based on inter-AS updates—which will not go inside of an AS, and those for intra-AS packets based on intra-AS updates—which will not go outside of an AS.

Multihoming and multipath routing: When multihoming or multipath routing is in use, a downstream router may receive multiple updates for the same source address space, but from different incoming directions. Without any modifications, the incoming tables of downstream routers would only consider packets matching the incoming direction for the *last* update the router receives to be valid.

We introduce a simple, optional tag to the (SAS, IP) pair in the ASV of ID-SAVE updates (Section 4.1.1). This tag identifies which home or path the update originated from or passed through. The incoming tree of a downstream router may thus contain nodes that represent source address spaces that are further differentiated by tags. We have verified this solution, with detailed discussion in our technical report [30].

[1]Without this assumption ID-SAVE still works (as it is designed to be incrementally deployable), but there would be no clear separation of inter-AS vs. intra-AS operations.

6. EVALUATION

In this section we discuss the performance of ID-SAVE, including what we measured, the methods we used, and the results of the evaluation.

6.1 What We Measure

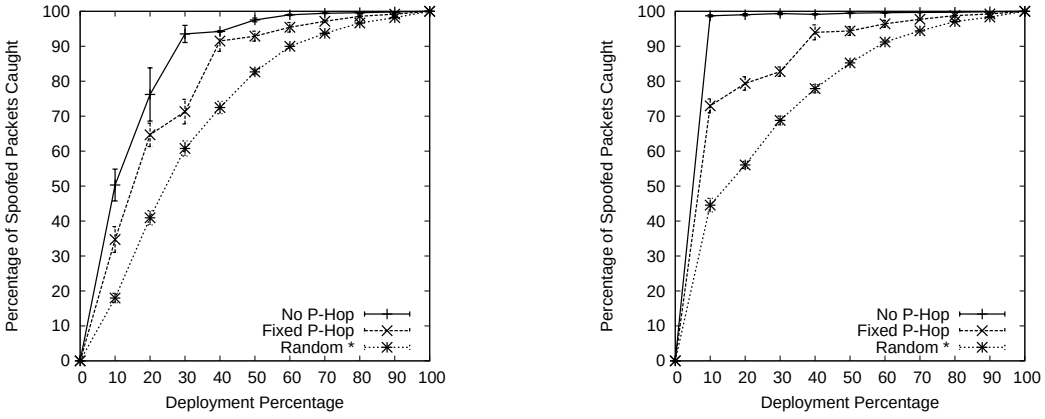
One must make proper measurements in order to understand the performance of any system. We evaluate ID-SAVE by measuring the following:

- *Efficacy* – How well is ID-SAVE able to catch spoofed packets?
- *False Positives* – How often does ID-SAVE misclassify legitimate packets?
- *Overhead*
  - Computational: How much processing overhead does ID-SAVE incur?
  - Storage: How much space do ID-SAVE's data structures require?
  - Network: How much traffic does ID-SAVE incur?

6.2 Methodology

In order to evaluate the ID-SAVE system on a moderately sized network, we use the J-Sim [31] network simulation framework, along with the GT-ITM [32] topology generator. Simulations ran on ten unique transit-stub topologies with 260 nodes (other topology types are left as future work). More importantly, we generate both spoofed packets and legitimate packets with different percentages of ID-SAVE deployment and with or without routing changes in order to measure how effective and efficient ID-SAVE is in catching different types of spoofed packets and in making sure that legitimate packets are not mistakenly flagged as spoofed.

Note that spoofed packets and legitimate packets can both be broken down into packets with a **protected source** (a source address that belongs to an ID-SAVE router's source address space) and packets with a **legacy source** (a source that does not belong to an ID-SAVE



(a) Efficacy (b) Adjusted efficacy

Figure 5: Efficacy and adjusted efficacy of ID-SAVE (± std. error). Adjusted efficacy does not count packets which did not encounter an ID-SAVE router.

router's source address space). Unless otherwise noted, both "spoofed packet" and "legitimate packet" below will refer to packets with a protected source.

Attackers use various spoofing methods to generate and send spoofed packets. In our simulation, we generate and test the following types of spoofed packets:

- *No P-Hop*: An attacker sends packets with a spoofed source address, but without a spoofed p-hop. This would be similar to present-day spoofing methods.
- *Fixed P-Hop*: Spoofed packets include a fixed spoofed p-hop. This represents an attacker who knows to add a fake p-hop, but does not randomize them.
- *Random P-Hop*: An attacker sends out some "Fixed P-Hop" spoofed packets, then randomizes the p-hop of every subsequent spoofed packet. This represents the best an attacker can do when wanting to spoof a single source to a single destination.
- *Random Source & P-Hop*: After sending out some "Fixed P-Hop" spoofed packets, an attacker randomizes subsequent spoofed source addresses and p-hops. This represents an attacker who does not care which source is spoofed, but targets a single destination.
- *Random Destination & P-Hop*: After an attacker sends out some "Fixed P-Hop" spoofed packets, he randomizes the destination and p-hop of subsequent packets. In this attack, the source matters, but the destination does not—such as in a reflection attack.

We will also use "Random *" to refer to the various "Random" types as a group.

We randomly deploy ID-SAVE routers throughout the network, and vary the deployment percentage between 0% and 100% in 10% increments. Varying the deployment percentage can show us how the metrics from Section 6.1 increase or decrease as more routers begin to run ID-SAVE.

We introduce routing changes into the system by randomly disconnecting five routers from the network. We then re-evaluate every metric from Section 6.1, in order to understand how routing changes may affect the efficacy and false positive rates, along with how traffic overhead relates to routing changes.

6.3 Results

We now present the performance of ID-SAVE as measured against the metrics described in Section 6.1. We have found that ID-SAVE effectively catches spoofed packets, even with a low percentage of deployment; with only 20% deployment ID-SAVE is able to catch a majority of the "Random *" spoofed packets that encounter an ID-SAVE router. The system also maintains an acceptable false positive rate and overhead levels.

6.3.1 *Efficacy*

Figure 5(a) shows the percentage of spoofed packets ID-SAVE catches at various deployment percentages. For instance, if, at some random point in a network with 40% ID-SAVE deployment, an attacker sends out "Fixed P-Hop" spoofed packets, there is a 90% chance that ID-SAVE will catch these spoofed packets. Note that if an attacker continuously spoofs a particular source to a particular destination it does not mean ID-SAVE will catch 90% of his spoofed packets; assuming there are no routing changes, packets that escape will continue to escape, and packets that are caught will continue to be caught. This means that for a given location in the network, some source-destination pairs are simply not spoofable.

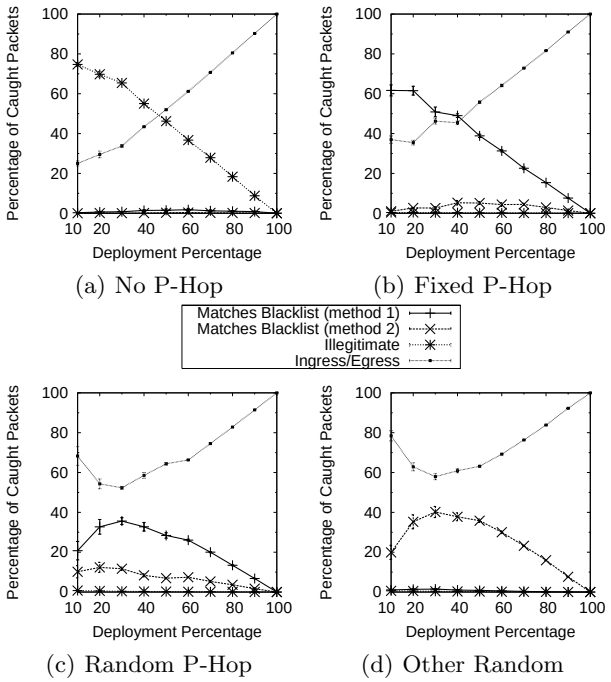
Not surprisingly, ID-SAVE caught the "No P-Hop" type of spoofed packet most easily. Spoofed packets of the "Fixed P-Hop" type were the second easiest to catch. ID-SAVE's effectiveness against the "Random *"

spoofed packet types was very similar, and so they are grouped together for clarity. ID-SAVE is the least effective against the "Random *" types.

When the deployment percentage is low, many spoofed packets may not even pass through a single ID-SAVE router. This of course means that ID-SAVE does not even have the chance to catch some of the spoofed packets. Figure 5(b) shows the efficacy when we only consider spoofed packets that pass through at least one ID-SAVE router. This "adjusted" efficacy is more accurate in reflecting ID-SAVE's usefulness since, otherwise, it will be considered ID-SAVE's "fault" when spoofed packets reach their destination without encountering any ID-SAVE routers. With this measurement, we can see that even when only 20% of the routers run ID-SAVE, already more than 50% of the spoofed packets fail to escape, no matter which spoofing method an attacker employs. Furthermore, this is achieved with a 20% *random* deployment; we did not arrange ID-SAVE routers at strategic locations.

We now look more closely at what mechanisms ID-SAVE uses in catching different types of spoofed packets (Figure 6). The first thing we notice is common to all of the spoofing types: At higher percentages of deployment, spoofed packets are more likely to encounter an ID-SAVE router at their first hop, causing simple ingress/egress filtering to account for the majority of caught packets. At lower percentage of deployment, however, a large portion of packets are caught because ID-SAVE classifies them as illegitimate or invalid. In another words, the effectiveness of ID-SAVE is more obvious at a low deployment percentage. Combining results from Figures 5 and 6, we see that ID-SAVE catches spoofed packets well, even at a low deployment percentage. Early ID-SAVE adopters can effectively enjoy the benefits of ID-SAVE. Below, we analyze how packets of each spoofing type are caught by ID-SAVE aside from using ingress/egress filtering:

- *No P-Hop*: Caught because they are classified as illegitimate. Since every packet with a protected source is expected to carry an embedded p-hop, these packets will be considered illegitimate and caught at the first ID-SAVE router they encounter.
- *Fixed P-Hop*: Caught mostly because they match blacklist entries using the first matching method. These packets keep the same source, destination, and p-hop; they easily fully match blacklist entries.
- *Random P-Hop*: Caught because they match blacklist entries, more often by using the first method than the second method. These packets always go towards the same destination with the same spoofed source, and after they pass through their first ID-SAVE hop they also carry the same p-hop. As in "Fixed P-Hop," later packets will fully match blacklist entries created for earlier packets.
- *Random Source & P-Hop* and *Random Destination & P-Hop*: Caught because they match blacklist entries using the second matching method. With their source or destination field changing, they cannot match blacklist entries using the first matching method. But, once they are classified as suspicious, the second matching method can take effect: An ID-SAVE router will still see them travel from the same upstream ID-SAVE router (i.e., these packets will carry the same p-hop), and they carry either the same source or destination address.



(a) No P-Hop (b) Fixed P-Hop

(c) Random P-Hop (d) Other Random

Figure 6: How much different aspects of packet classification contribute to catching spoofed packets (± std. error).

In addition to the percentage of spoofed packets that ID-SAVE can catch, another useful measure of ID-SAVE's efficacy is how many ID-SAVE routers a spoofed packet can cross before it is caught.

Figure 7 shows results from the "Random *" spoofing type scenarios. As the density of ID-SAVE routers increases, escaped spoofed packets cross more ID-SAVE hops. (Note that some of these escaped packets will not encounter ID-SAVE routers at all, especially when ID-SAVE deployment percentage is low, so Figure 7 also shows the results for escaped packets that encountered at least one ID-SAVE router.) In contrast, those packets that are caught only cross one or two ID-SAVE hops, no matter what the deployment percentage of ID-SAVE is. This shows ID-SAVE is sharp at recognizing spoofed packets early, and spoofed packets are caught close to

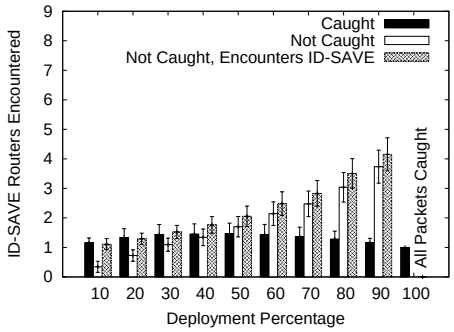


Figure 7: Number of ID-SAVE routers encountered by caught and escaped spoofed packets (± std. error).

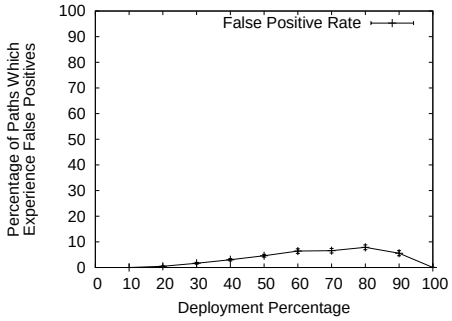


Figure 8: False positive rate (the percentage of end-to-end paths that experienced transient misclassification of legitimate packets, ± std. error).

the attacker.

After routing changes are introduced, the efficacy of ID-SAVE remains essentially unchanged. The details of the results are therefore omitted for brevity.

6.3.2 False Positives

False positives occur when legitimate packets match blacklist entries on ID-SAVE routers. When evaluating the false positives of ID-SAVE, there are mainly two situations to consider: legitimate packets sent *before* routing changes; and legitimate packets sent *after* routing changes. In the first case, there are *no false positives* since all packets are sent along the exact same path as the updates. In the latter case, there may be false positives in some circumstances as described below.

For a legitimate packet to be misclassified, it must match a blacklist entry. If an attacker previously sent a spoofed packet that carries a source address from the legitimate packet's source address space, or travels towards the legitimate packet's destination, this spoofed packet may cause certain routers to set up a blacklist entry that match the legitimate packet. When such routers receive the legitimate packet, they will then misclassify the packet as invalid, causing false positives.

We define the false positive rate of ID-SAVE as the percentage of end-to-end paths that would experience transient misclassification of legitimate packets. Figure 8 shows our results, where we used "Random *" spoofing attacks to populate the blacklists that were later obsoleted by routing changes. Remember, if any legitimate packets match blacklists after a routing change, the packet-driven pushback mechanism will activate. The source router of legitimate packets will quickly discover that its legitimate packets are misclassified, and send out an update to remedy the situation.

6.3.3 Overhead

There are three types of overhead that stem from ID-SAVE: computational, storage, and network overhead.

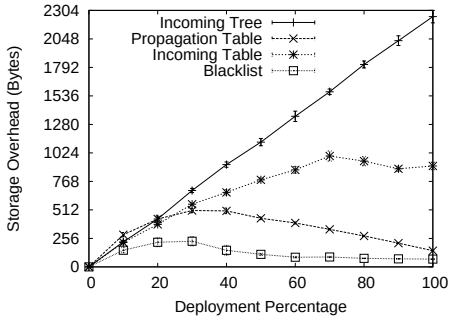


Figure 9: Storage required for ID-SAVE's data structures (± std. error).

Computational Overhead: ID-SAVE's most crucial computational overhead is the time taken for a router to classify packets. As packet classification is mainly a function of table lookup operations (using a router's incoming table and blacklist), and today's routers have been designed for fast, efficient table lookups (the main function of today's routers is forwarding table lookup), we expect ID-SAVE will impose only a minimal computational cost in classifying incoming packets.

We did not take actual measurements for computational overhead since currently the system has only been implemented as a simulation. We plan to measure ID-SAVE's computational overhead once it is implemented outside of simulation.

Storage Overhead: ID-SAVE maintains four data structures: the incoming tree, propagation table, incoming table, and blacklist. The incoming table and blacklist should be stored in cache memory for quick access, but the incoming tree and propagation table are good candidates for storage in a slower medium.

Figure 9 shows the average per-router storage requirements of ID-SAVE, after the "Random *" spoofing attacks occur. As the deployment percentage increases, we can see that the propagation table and blacklist sizes both first increase, but then decrease. For the propaga-

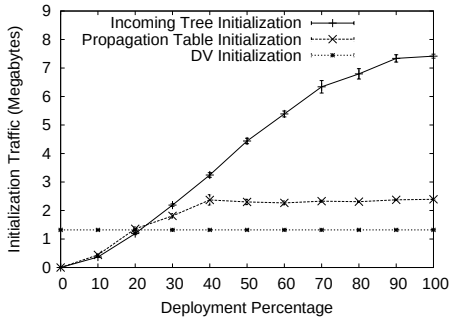


Figure 10: Total initialization traffic in the network (± std. error). Periodic traffic overhead would be similar.

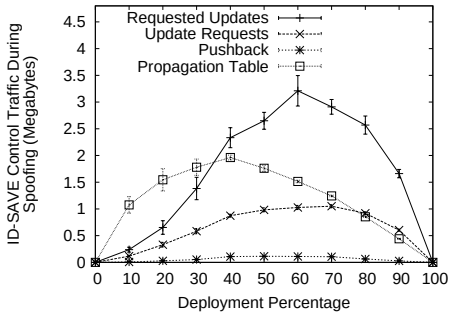


Figure 11: Total network traffic caused by ID-SAVE under spoofing attack (± std. error).

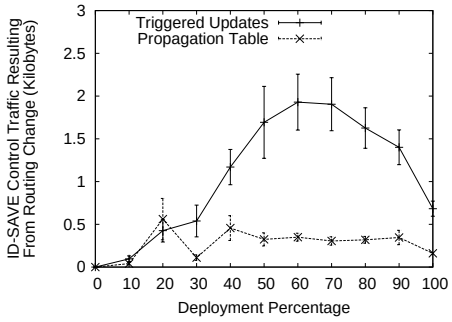


Figure 12: Total network traffic from ID-SAVE caused by routing changes. Normalized to be per forwarding entry change (± std. error).

tion table, as there are more ID-SAVE routers, every ID-SAVE router will have more ID-SAVE neighbors, and a larger propagation table is generally needed; but after the deployment is dense enough, ID-SAVE neighbors are more often direct physical neighbors, negating the need for propagation table entries. For the blacklist, recall their entries are created when attackers spoof addresses from protected spaces. As there are more ID-SAVE routers, there will be more protected spaces and thus more spoofed packets from protected spaces, leading to more blacklist entries created. As the deployment is more dense, however, routers are more likely to catch spoofed packets at their source, negating the need to set up a corresponding blacklist entry.

The incoming tree and incoming table increase with the deployment percentage. Clearly, for every new ID-SAVE address space, a router's incoming tree needs another corresponding node. The incoming table must similarly grow, however once the deployment is high enough, a router takes advantage of routing symmetry and simply uses the forwarding table in many cases (Section 4.2.2).

Network Traffic Overhead: ID-SAVE incurs a reasonable amount of traffic for all topologies we measured. There are three time periods of interest regarding ID-SAVE's traffic usage: initialization, during a spoofing attack, and during routing changes. (The periodic control traffic of ID-SAVE would be similar in size to the initialization traffic.)

Figure 10 shows the initialization traffic of ID-SAVE. The figure also includes the initialization traffic of the Distance-Vector-based routing algorithm used in J-Sim as a reference. Initially, as the deployment increases, the traffic for both the incoming tree and propagation table initialization increase. When the deployment percentage increases past 40%, propagation table traffic levels off. Neighboring ID-SAVE routers are separated by fewer physical hops, so propagation table probes do not travel as far, using less bandwidth—but meanwhile there are more routers probing, which balances it out and keeps the traffic from decreasing. The incoming tree traffic continues to increase with deployment, since every new ID-SAVE router must tell all the other routers about its protected source address space.

There is room for improvement of the initialization and periodic traffic. A router can be configured so that if it recently propagated updates from an upstream router towards a destination space β (therefore adding its own SAS to the ASV), it does not have to initiate its own update towards β.

Figure 11 shows ID-SAVE's traffic usage during the "Random *" spoofing attacks. Similar to some of the storage overhead results, we see the bandwidth usage first increase and then decrease. As the deployment percentage increases, spoofed packets are more likely to include protected sources, leading to more control traffic to deal with such packets; but as ID-SAVE neighbors become physically closer, all the control traffic travels shorter distances. Additionally, with increased deployment, ID-SAVE catches more and more spoofed packets using ingress/egress filtering (recall Figure 6), lessening the amount of control traffic routers send.

Figure 12 shows the average amount of ID-SAVE control traffic that a single forwarding table change causes during routing changes. When adding or modifying a forwarding table entry at an ID-SAVE router, at least

one update will be triggered in order to keep downstream routers' incoming direction information fresh. These triggered updates may then in turn trigger propagation table probing. As the deployment percentage increases, more routers not only append their address spaces to the triggered updates, but also split the triggered updates more times, creating more traffic. Eventually, when the deployment becomes dense enough to require less splitting of updates, traffic levels decrease.

7. OPEN ISSUES

As this paper is focused on the feasibility for a subset of routers on the Internet to build incoming information knowledge, we leave certain issues open and plan to address them in our future work.

ID-SAVE's scalability requires further evaluation, including simulating networks of various sizes and topologies. We must find out how the size of a network affects the performance of ID-SAVE, especially its storage and traffic overhead. Trying different topology styles can also further verify ID-SAVE's efficacy and accuracy. Evaluating ID-SAVE with a facility such as PlanetLab [33] can also improve the evaluation of ID-SAVE.

Illegitimate or spoofed packets sent by attackers at end hosts is the main attack we deal with in this paper. ID-SAVE can further learn from state-of-the-art security solutions for routing protocols in order to secure itself. For instance, S-BGP [34] has studied the ownership of address spaces and update authentication. Although not fully explored in this paper, it is clear that it is critical for ID-SAVE routers to be able to authenticate each other, be able to authenticate ID-SAVE messages, and be resilient against various attacks.

Before deciding upon a method of embedding p-hop information, we must research various methods and weigh the advantages and disadvantages of each, including such aspects as deployability and security. As our evaluation shows, the initial p-hop value of a packet has no significant impact on ID-SAVE's efficacy. No matter what p-hop an attacker embeds in a packet, it will be overwritten by the first ID-SAVE router the packet passes through. This is in stark contrast to previous work [11], where attackers can spoof from anywhere in the network with the correct key.

Additional issues still remain regarding deployment in the real world. One step is essentially an engineering problem—making ID-SAVE run on real routing hardware. We can also use real-world topologies to discover more effective deployment locations. One problem is that not all deployment strategies are feasible. Previous studies [10] found that a vertex cover of an ID-SAVE-like system would be extremely effective, but such a deployment would be difficult if not impossible to actually implement. The relative incentives for a domain to deploy ID-SAVE would of course also change depending on the deployment locations throughout the network.

8. CONCLUSION

With only a subset of routers on the Internet communicating with each other according to a certain protocol, can they successfully learn the valid incoming direction of packets from each other?

Answering this question is crucial to solving a distressing problem facing today's Internet: Routers know where to forward packets, but not where they should come from. The former ensures packets can be delivered; unfortunately, the latter causes *any* packet to be delivered, including those with spoofed source addresses. Without this incoming direction information, the consequences are severe.

Indeed, there have already been various approaches targeting this issue at the edge of the Internet, trying to determine whether or not the source address field of a packet is valid. They probe suspected sources, observe statistical features of incoming packets (such as their expected hop counts), or require packets carry an identifier associated with a specific source-destination pair or with a particular path.

But, is there a clean, effective solution possible at the core of the Internet? One which can potentially benefit the design of a next-generation network, while contributing to the incremental evolution of the Internet? After all, the root cause of the problem is not at the edge.

In answering this question, we proposed and measured ID-SAVE, a protocol that allows a portion of Internet routers to distribute and utilize incoming direction information. Whereas open issues certainly exist, ID-SAVE takes a key step in demonstrating the feasibility of an incrementally deployable source address validity enforcement solution. It shows that it is feasible for interested parties (i.e., ID-SAVE routers) to (1) effectively communicate with each other to build up their knowledge of incoming directions for different sources—even if they are not directly connected to each other since not every router is ID-SAVE-capable; and (2) maintain their incoming direction knowledge while handling routing changes at either interested parties (i.e., ID-SAVE routers) or disinterested parties (i.e., non-ID-SAVE routers).

9. REFERENCES

[1] R. Beverly and S. Bauer, "The Spoofer Project: Inferring the extent of source address filtering on the Internet," in *Proc. USENIX SRUTI*, 2005.

[2] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," *ACM Trans. Computer Systems*, vol. 24, no. 2, pp. 115–139, 2006.

[3] J. Li, J. Mirkovic, M. Wang, P. L. Reiher, and L. Zhang, "SAVE: Source address validity

enforcement protocol," in *Proc. IEEE INFOCOM*, New York, June 2002.
[4] S. Kent and K. Seo, "Security architecture for the Internet protocol," RFC 4301, IETF, 2005.
[5] S. J. Templeton and K. E. Levitt, "Detecting spoofed packets," in *DARPA Information Survivability Conference and Exposition*, vol. 1, 2003.
[6] F. Baker, "Requirements for IP Version 4 routers," RFC 1812, IETF, 1995.
[7] V. Paxson, "End-to-end routing behavior in the Internet," in *Proc. ACM SIGCOMM*, 1996.
[8] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing," RFC 2827, IETF, 2000.
[9] T. Killalea, "Recommended Internet service provider security services and procedures," RFC 3013, IETF, 2000.
[10] K. Park and H. Lee, "On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets," in *Proc. ACM SIGCOMM*, 2001.
[11] A. Bremler-Barr and H. Levy, "Spoofing prevention method," in *Proc. IEEE INFOCOM*, 2005.
[12] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: An effective defense against spoofed DDoS traffic," in *Proc. Conference on Computer and Communications Security*, 2003.
[13] Z. Duan, X. Yuan, and J. Chandrashekar, "Constructing inter-domain packet filters to control IP spoofing based on BGP updates," in *IEEE Infocom*, 2006.
[14] H. Lee, M. Kwon, G. Hasker, and A. Perrig, "BASE: An incrementally deployable mechanism for viable IP spoofing prevention," in *Proc. ACM Symposium on Information, Computer, and Communication Security*, 2007.
[15] S. M. Bellovin, "ICMP traceback messages," March 2000, work-in-progress Internet Draft: draft-bellovin-itrace-00.txt.
[16] H. Burch and W. Cheswick, "Tracing anonymous packets to their approximate source," in *Proc. USENIX LISA*, 2000.
[17] R. Stone, "CenterTrack: An IP overlay network for tracking DoS floods," in *USENIX Security Symposium*, 2000.
[18] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson, "Network support for IP traceback," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, pp. 226–237, 2001.
[19] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer, "Single-packet IP traceback," *IEEE/ACM Trans. Networking*, vol. 10, no. 6, pp. 721–734, 2002.
[20] M. Ma, "Tabu marking scheme for IP traceback," in *IPDPS*, 2005.
[21] D. Dean, M. K. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *ACM Transactions on Information and System Security*, vol. 5, no. 2, pp. 119–137, 2002.
[22] M. Adler, "Trade-offs in probabilistic packet marking for IP traceback," *J. of the ACM*, vol. 52, no. 2, pp. 217–244, 2005.
[23] M. T. Goodrich, "Efficient packet marking for large-scale IP traceback," in *Proc. Conference on Computer and Communications Security*, 2002.
[24] A. Yaar, A. Perrig, and D. Song, "FIT: Fast Internet traceback," in *Proc. IEEE INFOCOM*, 2005.
[25] K. Park and H. Lee, "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack," in *Proc. IEEE INFOCOM*, 2001.
[26] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against DDoS attack," in *Proc. IEEE Symposium on Security and Privacy*, 2003.
[27] A. Yaar, A. Perrig, and D. Song, "StackPi: New packet marking and filtering mechanisms for DDoS and IP spoofing defense," *IEEE J. Selected Areas in Communications*, vol. 24, no. 10, pp. 1853–1863, October 2006.
[28] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 3, pp. 62–73, 2002.
[29] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets," *Communications of the ACM*, vol. 21, no. 12, pp. 1040–1048, 1978.
[30] Anonymous, "Multipath and multihoming with ID-SAVE," Tech. Rep., 2007.
[31] H.-Y. Tyan, "Design, realization and evaluation of a component-based compositional software architecture for network simulation," Ph.D. dissertation, Ohio State University, 2002, adviser-Chao-Ju Jennifer Hou.
[32] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM*, 1996.
[33] PlanetLab, http://www.planet-lab.org.
[34] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (S-BGP)," *IEEE J. Selected Areas in Communications*, vol. 18, no. 4, pp. 582–592, April 2000.