Realizing a Source Authentic Internet*

Toby Ehrenkranz[1], Jun Li[1], and Patrick McDaniel[2]

[1] Department of Computer and Information Science
University of Oregon
Eugene, OR 97403 USA
tehrenkr,lijun@cs.uoregon.edu

[2] Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802 USA
mcdaniel@cse.psu.edu

Abstract. An innate deficiency of the Internet is its susceptibility to IP spoofing. Whereas a router uses a forwarding table to determine where it should send a packet, previous research has found that a router can similarly employ an incoming table to verify where a packet should come from, thereby detecting IP spoofing. Based on a previous protocol for building incoming tables, SAVE, this paper introduces new mechanisms that not only address a critical deficiency of SAVE when it is incrementally deployed (incoming table entries becoming obsolete), but can also push the filtering of spoofing packets towards the SAVE router that is closest to spoofers. With these new mechanisms, and under the assumption of incremental deployment, we further discuss the security of SAVE, evaluate its efficacy, accuracy, and overhead, and look into its deployment incentives. Our results show incoming-table-based IP spoofing detection is a feasible and effective solution.

Key words: IP spoofing, IP source address, IP spoofing detection, incoming table, pushback

1 Introduction

Attackers today can send packets pretending to be from any Internet address. Any host on the Internet can be a victim of such "spoofing" attacks. Even in today's botnet infested Internet, an attacker prefers to use IP spoofing whenever possible. While the attacker may simply spoof a victim's address to hide the real attack source, it is very likely the victim address is the focus of a targeted attack. For example, only through IP spoofing can an attacker perform DNS amplification (subverting DNS servers to perform a bandwidth-based DDoS attack [1, 2]),

* This material is based upon work supported by the USA National Science Foundation under Grant No. 0520326. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

reset a victim's TCP connections (sending spoofed TCP reset packets with in-window sequence numbers [3]), poison a DNS cache (transparently redirecting victims to the attacker's server [4]), or circumvent spam filters (getting around mail blocks an ISP may place on a botnet's zombie machines [5, 6]). Although the underlying threat of IP spoofing is not new, the problem continues to worsen: attackers persist in finding new ways of crafting attacks using spoofed IP packets, and attackers can spoof from a greater portion of the Internet than before [6].

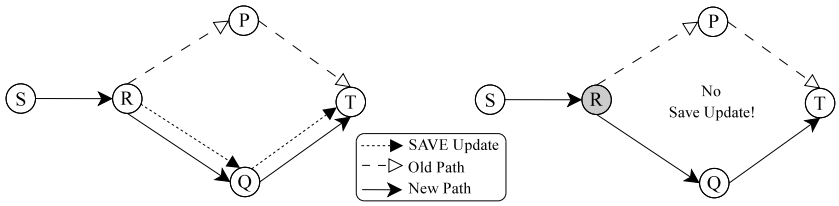
If every network in the world was able to coordinate a deployment of even simple ingress filtering [7] and unicast reverse path forwarding [8] checks, the threat would be all but eliminated. Unfortunately due to both technical and logistical reasons, this is an unattainable goal [9]. When even a small percentage of networks do not deploy such basic filtering methods, nobody is safe. Everyone's Internet address is still at risk of being spoofed. Researchers have proposed more sophisticated spoofing prevention mechanisms over the years [9], but all have failed to neutralize the threat of IP spoofing.

Fortunately there has been promising research, showing that if even a *small* percentage of routers on the Internet deploy a more finely grained filtering table to discard packets with a forged source address, a synergistic filtering effect can be achieved to stop a large fraction of spoofed IP packets [10, 11]. SAVE [12, 13] is a light-weight protocol to build such a filtering table, called an *incoming table*, at routers. As a router has multiple physical interfaces to receive incoming packets, every entry of the incoming table specifies the valid incoming interface for packets from a specific IP address prefix to arrive at the router. A recent survey [9] has further shown that compared to other IP spoofing prevention methods, using an incoming table to filter spoofed packets is the most effective.

Although SAVE provides the incoming tables necessary for effective filtering with only a small deployment, the SAVE protocol itself faces a serious challenge when incrementally deployed. A router's incoming table that is up-to-date at time t may become obsolete at some time after t because of routing changes on the Internet. We describe how a router's incoming table becomes obsolete below.

In the SAVE protocol, every SAVE-capable router that is in charge of a source address space *periodically* sends updates to its downstream routers about the *current* incoming interface for the source address space. Furthermore, when a routing change occurs at any downstream router, that router must send out a new update immediately to ensure routers further downstream learn the new valid incoming interface of the source address space in question. Otherwise, those routers will stay out of date until the next periodical update. However, legacy routers that do not run SAVE will simply do nothing when a routing change occurs; legacy routers will never initiate a SAVE update! As shown by the example in Figure 1(a) and 1(b), if R was a legacy router, the lack of SAVE update from R after its routing change will cause the SAVE-capable router T's incoming table about packets from router S to be out of date (until the next periodic SAVE update from S reaches T).

As SAVE is incrementally deployed, there will be many legacy routers, probably even outnumbering SAVE-capable routers for a long time; *it is highly likely*



(a) R is a SAVE router and initiates a SAVE update when the link from R to P is broken. T's incoming table is updated to show that packets from S should come from its lower left incoming interface.

(b) R is a legacy router so it does *not* initiate a SAVE update when the link from R to P is broken. T's incoming table is out of date, still showing packets from S should come through upper left incoming interface.

Fig. 1. An example showing how an incoming table can contain obsolete entries.

that incoming tables at SAVE-capable routers will often contain obsolete entries. While it is promising to use incoming tables to stop spoofed packets, it is also difficult to use them if they carry obsolete entries.

In this paper we make the following fundamental contributions: We study if we can introduce new mechanisms to enable SAVE-capable routers to reliably discard spoofing packets, even though their incoming table may be obsolete. In particular, we devise and evaluate three new elements that a SAVE-capable router can employ: a blacklist data structure, an on-demand-update mechanism, and a pushback mechanism.

- *Blacklist*: The blacklist complements the incoming table at a router in classifying an incoming packet, including determining if the packet is spoofed.
- *On-demand update*: A SAVE-capable router can request SAVE updates on demand to verify possibly incorrect or outdated information.
- *Pushback*: SAVE-capable routers along the way of a spoofing flow can push the filtering of spoofed packets to the router that is the closest to the spoofer.

In combination, these new elements allow SAVE to function properly even in the presence of legacy routers. Referring back to the example in Figure 1(b), T can request an on-demand update from S, essentially replacing the triggered SAVE update in Figure 1(a). The blacklist gives a router more state information, so the router does not need to request an on-demand update for every packet that does not match the incoming table. Finally, the pushback mechanism serves two purposes. First, it helps to reduce spoofing traffic by dropping spoofing packets as close to the attacker as possible. Second, it tells a router in charge of a source address space when downstream routers have incorrect information in their blacklists regarding its address space.

Also of great importance is the security of SAVE with these new mechanisms. SAVE must secure itself against all possible attacks. Not only may attackers try to evade the IP spoofing detection at SAVE routers, they may also attempt to introduce illegal control messages. For example, an attacker (or a bot machine it controls) could try to establish wrong incoming information at SAVE routers

by injecting a SAVE update about a source address it is going to spoof. In this paper, we also discuss how security can be addressed.

Our evaluation demonstrates the viability of these new mechanisms. We perform a detailed simulation to evaluate their effectiveness at detecting spoofing packets, and explore the relationship between efficacy and adoption rates. These Internet-scale simulations show that with as little as 0.08% deployment, attackers cannot spoof protected source addresses in over 90% of all cases. Moreover, we evaluate the storage, traffic, and computational overhead of SAVE with the new mechanisms, showing that the overhead is low.

The rest of the paper is organized as follows. We first describe how a SAVE router can introduce a blacklist alongside the incoming table to help classify incoming packets in Section 2. We then describe the on-demand-update mechanism and the pushback mechanism in Sections 3 and 4, respectively. Section 5 discusses how SAVE can secure itself. We present our evaluation in Section 6, with open issues discussed in Section 7. Finally, we discuss related work in Section 8, and conclude our paper in Section 9.

2 Incoming Table, Blacklist, and Packet Classification

In addition to an incoming table described above, we add to every SAVE-capable router a blacklist data structure. With both the incoming table and the blacklist, a router classifies incoming packets into several different types, and takes some action specific to each of those types. In this section we describe the blacklist data structure and classification mechanism.

2.1 Blacklist

Whereas the incoming table of a router specifies the legitimate incoming interfaces for different source address spaces, the blacklist indicates whether an incoming packet is *spoofing* based on its source address, destination address, and the incoming interface. More specifically, a router's blacklist is maintained through two separate blacklists corresponding to two different ways of matching an incoming packet against a blacklist entry:

SI: Matches the source address and the incoming interface of the packet.
SD: Matches the source address and the destination of the packet.

A router receiving spoofing packets that match its SI blacklist could be on the legitimate path from the spoofed source to the destination, but not from the spoofing packet's incoming direction. A router receiving spoofing packets that match its SD blacklist should not be on the legitimate path from the spoofed source to the destination, and so should never see a packet with such a source and destination.

2.2 Packet Classification

With both its incoming table and blacklist in place, a router classifies an incoming packet as:

- *Valid* if it matches the incoming table but not the blacklist.
- *Suspicious* if it matches neither the incoming table nor the blacklist.
- *Invalid* if it matches the blacklist.
- *Unknown* if there is no information regarding the packet's source address.

Only when the packet is classified as invalid will the router drop the packet. For the other three types the router will forward the packet.

Furthermore, if a packet is suspicious, the router will initiate the on-demand-update mechanism. The packet is suspicious either because the packet is spoofing, or because the packet is legitimate but the router's incoming direction information is outdated. As we described in Section 1, the routing change at a legacy router upstream will not lead to an immediate SAVE update for this router to update its incoming table.

If a packet is invalid, the router will initiate the pushback mechanism. No further actions are taken for valid or unknown packets.

We describe both on-demand-update and pushback in the following sections, including how they deal with obsolete incoming table entries.

3 On-Demand Update

When a router classifies a packet as suspicious, it still forwards the packet as usual, but it will also initiate an on-demand update. From the incoming table entry that matches the packet's source address, the router determines the source address space in question and the source router in charge of the source address space. It then requests that the source router sends an on-demand update—which is on behalf of the entire source address space—towards the destination of the suspicious packet.

Following the same design as in SAVE [12, 13], the on-demand update will travel the same path as the legitimate packets that originate from the source router's address space. When the on-demand update arrives at the router from a specific incoming interface, this interface is then also the legitimate interface for the source address space in question. The router then makes sure its incoming table records *this* interface as the legitimate incoming interface for the source address space.

If the on-demand update does *not* arrive from the same incoming interface as the suspicious packet, the suspicious packet was in fact spoofing. Furthermore, the router updates its blacklists. Denote the spoofing packet's spoofed source address *space* as S and its incoming interface as i. It adds to the SI blacklist a new entry $\langle S, i \rangle$. If in the future a packet matches the newly added blacklist entry, the router will classify it as invalid. The router does not add a new entry to the SD

blacklist, because the router could legitimately see packets from the suspicious packet's source to its destination—just not from the incoming interface that the suspicious packet used.

If the on-demand update *does* arrive from the same direction as the suspicious packet, the packet was not spoofing. Note the router already forwarded the packet earlier so no false positive occurs.

It is also possible the on-demand update never reaches the router. This could be because the router is not on the path from the source to the destination, or because congestion caused the update request or the update itself to be dropped. Since the router cannot know for sure, it takes no action. Assuming similar suspicious packets continue to arrive, the router will continue to request updates. We use a truncated binary exponential backoff scheme for subsequent requests.

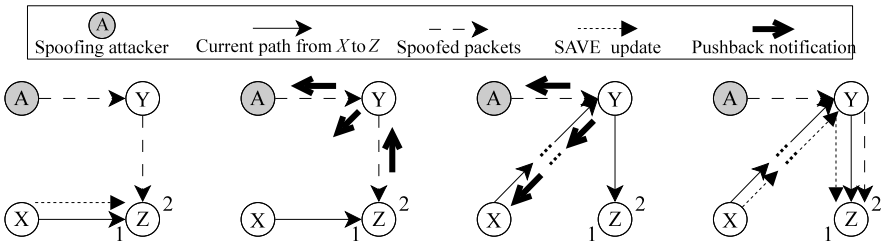
4 Pushback of Spoofed Packets

The aim of the pushback is to push the filtering of spoofed packets all the way toward the router that is the closest to the spoofer(s). The pushback procedure is packet-driven and it is triggered when a SAVE router receives an invalid packet.

Once the pushback procedure is triggered by an invalid packet, the router sends pushback messages to immediate upstream SAVE routers that the packet possibly passed through. (The router uses incoming SAVE updates to record upstream SAVE routers along every incoming interface, and can easily identify those upstream along the incoming interface of the packet.) Assume the packet is from source address space S to destination address d. When an upstream SAVE router receives a pushback message, it adds an entry $\langle S,\ d\rangle$ to its SD blacklist. The incoming interface does not matter—the upstream router is not on the legitimate path from the packet's inscribed source to its destination at all. Upon receiving subsequent packets that match this new blacklist entry, the upstream router will classify them as invalid and continue to propagate the pushback further upstream.

Blacklist entries can become outdated if a routing change causes the legitimate path from the spoofing victim to become the same as that of the spoofing packets. If that happens, the pushback procedure will finally reach the source router in charge of the victim source address space. The source router can in turn send out an update that travels along the path and reaches every SAVE router en route. Every SAVE router can then remove its outdated blacklist entries.

Fig. 2 shows a pushback example. An attacker at legacy router A sends spoofing packets with a source from space S_X (X's source address space) and a destination dst_Z (an address towards which Z is downstream from X and Y). The spoofing packets arrive at router Y along the same interface as legitimate packets from S_X. Y classifies the packets as valid and forwards them. Z however expects packets from S_X to arrive on interface 1 according to its incoming table, so it classifies the first spoofing packet arriving at incoming interface 2 as suspicious. Z requests an on-demand update. Upon receipt of the requested update, Z confirms its incoming table information was correct, and the suspicious packet



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

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

(c) After routing change, legitimate packets trigger pushback to X.

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

Fig. 2. A pushback example. An attacker, A, sends packets spoofing X's address space, S_X, towards dst_Z.

was in fact invalid (Fig. 2(a)). Z then adds a new entry to its SI blacklist: Based on source S_X and incoming interface 2 of the suspicious packet, the new blacklist entry is $\langle S_X, 2\rangle$.

Z classifies later spoofing packets as invalid, and initiates the pushback process (Fig. 2(b)). Z knows Y is its neighbor along the spoofing packet's incoming interface. Z sends Y a pushback message, instructing Y to add an entry to its SD blacklist for all packets from S_X to dst_Z. When Y receives a packet matching the new blacklist entry, it classifies the packet as invalid and continues the pushback. Y finds all neighbors along the spoofing packet's incoming interface, and propagates the pushback towards such neighbors. Y's neighbors do not see matching packets, so do not further propagate the pushback. Y is the closest SAVE router to the attacker.

Later, if there is a routing change at a legacy router which causes the originally invalid "$X \cdots Y \cdots Z$" path to become valid and legitimate packets begin to flow along the path, SAVE will quickly converge to correct the error. During the transient period, router Y and Z will misclassify valid packets from S_X towards dst_Z as invalid. But now that Y's upstream neighboring SAVE routers also see packets matching the pushback request, the upstream routers will relay the pushback all the way to the source router X (Fig. 2(c)). After X receives the pushback, it realizes that downstream routers have incorrect blacklist entries matching its legitimate traffic. X sends an update towards dst_Z, causing all routers along the newly valid "$X \cdots Y \cdots Z$" path to remove their incorrect blacklist entries (Fig. 2(d)).

5 Security Considerations

SAVE must also be secure. The security of SAVE encompasses securing SAVE itself against attack and keeping attackers from being able to use SAVE to launch attacks. In addition to basic security functions such as confidentiality, integrity, and replay prevention, we must consider (1) *origin authentication* to ensure

a router is authorized to speak for a source address space, and (2) *collusion prevention* to ensure attackers cannot collude to manipulate incoming direction information at SAVE routers.

5.1 Origin Authentication

Origin authentication ensures SAVE routers are authorized to speak for their corresponding source address space. This requires a trusted authority to sign a certificate that an address space owner can present. A public key infrastructure as described in [14] can provide such certificates of address ownership. The root certificate authority can be ICANN, with regional Internet registries such as ARIN or RIPE at the next level, and ISPs below. If SAVE is simply deployed inside an AS, the AS can simply use its self-signed certificates.

5.2 Collusion Prevention

Attackers may attempt collusion to manipulate the incoming table stored at a downstream SAVE router. They may collude by masquerading as each other or copying an update from an upstream space to each other, causing downstream routers to receive the update about a source address space along a wrong incoming interface.

Since the original update from the source address space still travels along the correct path[2], such manipulation by attackers can only be temporary. More importantly, such manipulation cannot cause a router to drop legitimate packets. If a router mistakenly classifies a legitimate packet as suspicious, an on-demand update will verify that it is in fact legitimate and fix the incoming direction information. Note the manipulation does not give the attackers further spoofing capabilities either, since attackers can only copy existing upstream updates.

5.3 Confidentiality, Integrity, and Replay Prevention

With a public key infrastructure, confidentiality and integrity is straightforward. If confidentiality is needed, a SAVE router can use the recipient's public key to encrypt its messages; or, it can establish a secure channel with the recipient, and use the session key associated with the channel to encrypt the messages. If integrity is needed, a SAVE router can uses its private key to create digital signatures for its messages.

Replay attacks must be prevented in order to ensure that a previous update cannot be copied and resent at a later time. Downstream routers must receive the most up-to-date incoming direction information. Replay attacks can be prevented by adding a counter to SAVE updates. The counter in an update must be greater than the counter of earlier updates.

[2] SAVE security does *not* encompass routing-level security, as that is the job of routing protocols—we assume routers *will* route a packet correctly towards its destination, and will *not* maliciously forward a packet in the wrong direction, nor maliciously drop a packet en route. Every SAVE update is encapsulated inside an UDP packet.

An implementation note. The security issues that SAVE faces are similar to those faced by BGP. Both need to ensure that a router can speak for an address space (SAVE's source address space and BGP's destination address space), both need to prevent conclusion of attackers, and both need to provide integrity, replay prevention, and sometimes confidentiality. In particular, to implement SAVE's security, we can borrow some ideas from IRV [15], an incrementally deployable BGP security solution. Basically, each network can contain a *validation server* to be responsible for security purposes, including keeping track of certificates and keys, performing signature creation and validation for SAVE messages, and managing security policies. Doing so would also maintain a lighter load on SAVE routers, allowing them to focus on its main purpose of receiving, validating, and forwarding packets.

6 Evaluation

In this section we discuss the performance of SAVE with the new mechanisms we introduced in this paper. First we present SAVE's efficacy in catching spoofed packets. We then evaluate SAVE's false positives. Finally we show that SAVE's storage, network, and computational overhead are reasonable.

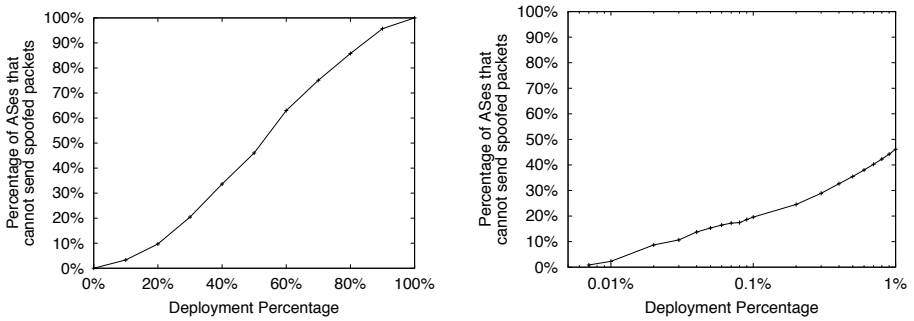
6.1 Efficacy

Methodology. For efficacy evaluation we use a modified static distributed packet filtering (DPF) [10] simulator. The DPF simulator allows us to evaluate the efficacy of SAVE on Internet-scale topologies by calculating efficacies based on the Internet AS graph and SAVE router locations. It uses Internet Autonomous System (AS) topologies from Route Views [16]. The efficacy metrics are similar to those in [10]. Specifically, we report:

- $\Phi_2(1)$ that represents the percentage of ASes that an attacker cannot send spoofing packets from—any spoofed packets from those ASes would be detected and filtered out;
- $\Phi_3(1)$ that represents the percentage of all attacker-victim AS pairs where the attacker *cannot* send spoofed packets to the victim; and
- $\Psi_1(\tau)$ that represents the fraction of target ASes which can narrow down an attacker's location to within τ possible attack ASes.

We consider a variety of placement strategies of SAVE routers. First, we deploy SAVE routers so they form a vertex cover (as in the original DPF work [10]). Then, we look at random deployments, with deployment percentages between 0% and 100% in 10% increments. Finally, we deploy SAVE routers at the top ASes by degree.

Results and Analysis. The efficacy of SAVE in catching spoofed packets depends upon both the deployment strategy and the percentage of deployment. With a random deployment, the efficacy increases along with the deployment



(a) Random deployments. (b) Deployments at highest-degree ASes.

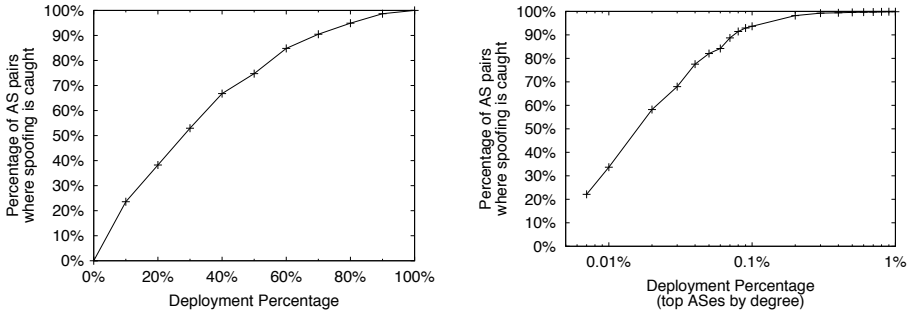
Fig. 3. $\Phi_2(1)$: The percentage of ASes on an Internet AS topology from which an attacker cannot send spoofed packets.

percentage. With deployment at high-degree ASes or using a vertex cover for deployment, the efficacy is much higher than a random deployment, even with a much lower percentage of deployment.

Fig. 3 shows $\Phi_2(1)$, the percentage of ASes on an Internet AS topology from which an attacker cannot send *any* packets that spoof a protected source address. We can clearly see that deployment strategies are an important factor. $\Phi_2(1)$ with a vertex cover deployment is around 99% (not shown), where the vertex cover consisted of around 14.5% of all routers. Fig. 3(a) shows $\Phi_2(1)$ for random deployments; with 15% or less deployment percentage, $\Phi_2(1)$ is even no more than 10%. Fig. 3(b) shows $\Phi_2(1)$ for deployments at ASes with the highest degree on the same topology; as a sharp contrast to random deployment, even with less than 1% of ASes deploying SAVE, over 40% of all ASes are unable to spoof *any* protected source.

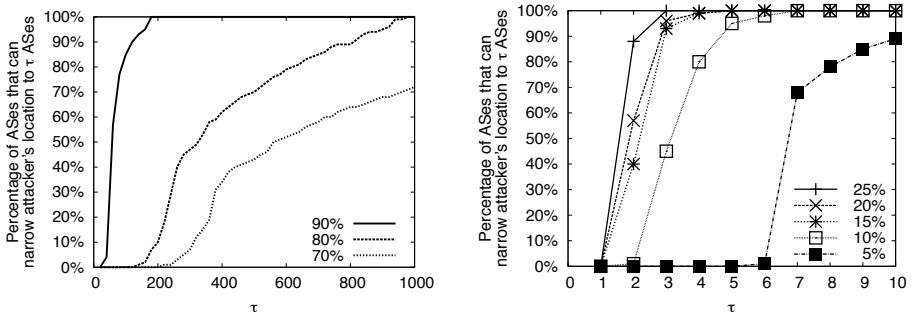
Fig. 4 shows $\Phi_3(1)$, the percentage of attacker-victim AS pairs where the attacker cannot send spoofed packets to the victim. Deployment strategy, again, plays an important role. A random deployment is not very effective—high efficacy requires high levels of deployment. More targeted deployments, however, can be extremely effective. With a vertex cover deployment the efficacy is over 99.9% (not shown). Even very small targeted deployments can be effective: With deployment at only the top 0.08% of ASes by degree (21 ASes in this case), efficacy is over 90%.

Fig. 5 shows $\Psi_1(\tau)$ with again the same deployments as above. $\Psi_1(\tau)$ is the percentage of destination ASes that can narrow down an attacker's location to within τ source ASes, when intermediate routers were unable to filter the spoofed packet. Note that this percentage is for instantly narrowing down an attacker's location based on the network topology, the location of SAVE routers, and the fact that the spoofed packet reached its destination. Vertex cover deployments (not shown) and high-degree AS deployments have excellent performance, generally being able to narrow down an attacker's actual location to within 5 locations or fewer. More random deployments are not able to reliably narrow down an attacker's location, with possible attacker locations measured in the



(a) Random deployments. (b) Deployments at highest-degree ASes.

Fig. 4. $\Phi_3(1)$: The percentage of attacker-victim AS pairs where the attacker cannot send spoofed packets to the victim.



(a) Random deployments. (b) Deployments at highest-degree ASes.

Fig. 5. $\Psi_1(\tau)$: The percentage of destination ASes that can narrow down an attacker's location to within τ source ASes.

hundreds. (SAVE includes additional location capabilities through the use of its pushback mechanism, which we plan to evaluate further.)

Finally, the high efficacy with the highest-degree ASes (as shown in Figs. 3(b), 4(b), and 5(b)) shows that such ASes—if they deploy SAVE—can filter spoofing packets and locate attackers very effectively. They thus will have a strong incentive to deploy SAVE. Moreover, doing so provides an incentive for other ASes to follow; with the highest-degree ASes deploying SAVE, when the followers also deploy SAVE they will protect their own address space much more effectively.

6.2 False Positives

False positives only occur when the following conditions are *all* met:

- A pushback path that spoofing packets travel becomes a valid path due to a sudden underneath routing change;
- Legitimate packets begin to flow along the path;

– The SAVE update from the source router of the legitimate packets has not reached SAVE routers along the path to void the blacklist entries that cause the legitimate packets to be dropped.

The transient time that all three conditions are met is short-lived. Assume S is the source router and R is a SAVE router on the path. False positives will occur at R from the time the first legitimate packet arrives along the new valid path to the time R's blacklist is updated. Assuming there is a steady stream of packets from S passing through R, it will take the following amount of time to update R's information:

$$rtt + \sum_{i=next(S)}^{R} C_i + \sum_{i=S}^{prev(R)} P_i + \sum_{i=next(S)}^{R} U_i \quad (1)$$

rtt is the round trip time between R and S, $next(S)$ is the SAVE router downstream from S towards R, C_i is the time it takes for router i to classify a packet, $prev(R)$ is the SAVE router upstream from R towards S, P_i is the time it takes for router i to process a pushback message, and U_i is the time it takes for router i to propagate an update.

The values of these parameters vary. Assuming values of 20ms for rtt, 100μs for C_i, P_i, U_i, and 10 SAVE hops from S to R, the transient period will be 50ms.

6.3 Overhead

Here we discuss SAVE's storage, network, and computational overhead.

Methodology. For overhead evaluation we use the J-Sim [17] network simulation framework. (Note the DPF simulator cannot calculate SAVE storage or network overheads.) The J-Sim framework simulates all routers, links, and messages in a network topology in order to conduct detailed overhead evaluation. This, however, limits the size of the topology to generally 5,000 nodes—even with our fairly high-end evaluation environment. (We performed all the evaluations on a computer with 16 GB of RAM, dual 2.6 GHz AMD dual-core Opteron 285 CPUs, running CentOS Linux 4.6.) To solve this problem, we note that SAVE can run at two separate levels: intra-AS level and inter-AS level, and we can evaluate the overhead at these two separate levels. At the intra-AS level, although a small number of ASes may have more than 5,000 routers, most ASes will fall into the range that the J-Sim framework can simulate. At the inter-AS level, all border routers of an AS can act as *one* "virtual router" with the entire AS as its source address space, and therefore SAVE's overhead at the inter-AS level can be analyzed using a topology of all virtual routers—which is equivalent to an Internet AS topology. As the Internet has approximately 26,000 ASes, a detailed J-Sim simulation with up to 5,000 nodes should be close enough for us to understand SAVE's overhead at a large scale.

The overhead analyses at intra-AS level and inter-AS level are similar, except that different topology models probably should be used. In this paper, we focus on the inter-AS level where each node is an AS (or a virtual router), and use

network topologies generated by shrinking AS topologies with Orbis [18]—such topologies are smaller than the AS topology of the real Internet but they have similar structure patterns.

Also, we again evaluate multiple placement strategies of SAVE routers. We evaluate vertex cover deployments, deployments at the top 1% of routers by degree, and biased 1% deployments consisting of a random half of the top 2% of routers by degree. They are all effective from our efficacy analysis above. We simulate networks ranging in size from 500 to 5,000 nodes (only up to 3,000 nodes for vertex cover deployments due to computation power limitation).

Storage Overhead: In this paper, the blacklist is the only new data structure. We now implement it as a cache of fixed size that runs the Least Recently Used (LRU) algorithm to replace old entries. Further work is needed to evaluate how the size affects the efficacy of the system. We do not worry about losing old blacklist entries; neighboring routers will still filter spoofing traffic, and the on-demand-update mechanism can recreate entries if necessary in any case.

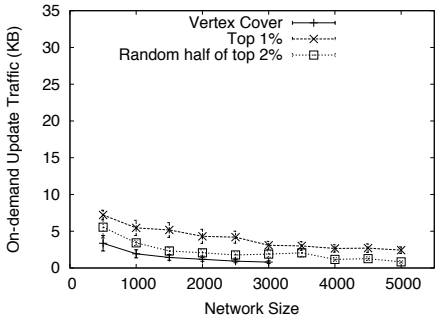
Network Traffic Overhead: Fig. 6 shows the per-router traffic during spoofing attacks. As the network size increases, network overhead decreases—because spoofing traffic, and thus control traffic, is more spread out (Figs. 6(a) and 6(d)). With a fixed network size (1,000 nodes) and increasing spoofing traffic, the network overhead increases linearly (Figs. 6(b), 6(c), 6(e), and 6(f)). This overhead is offset significantly as the SAVE system is also removing spoofing traffic from the network. Instead of the spoofing traffic overloading its target at the edge of the network, routers drop the spoofing traffic and SAVE's traffic overhead is spread out inside the network. Note that due to simulation limitations we cannot simulate a larger number of attackers, but results from Fig. 6 is still indicative about the network overhead effects from the size of the network, the number of attackers, and the number of spoofing packets.

Computational Overhead: SAVE's most crucial computational overhead is the time taken for a router to classify packets, which mainly consists of table lookup operations (using a router's incoming table and blacklist). We do not have actual measurements for computational overhead since the system is only implemented as a simulation, but we expect SAVE will impose only a minimal computational cost. Today's routers are designed for fast, efficient table lookups (a router's main function is forwarding table lookup).

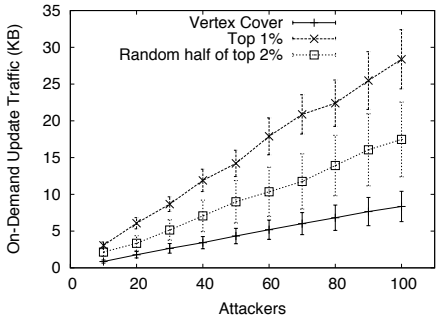
7 Open Issues

Several issues related to incoming-table-based IP spoofing detection warrant further investigation. These issues include incentives for deploying SAVE, spoofing strategies attackers can employ to avoid SAVE, and false positives.

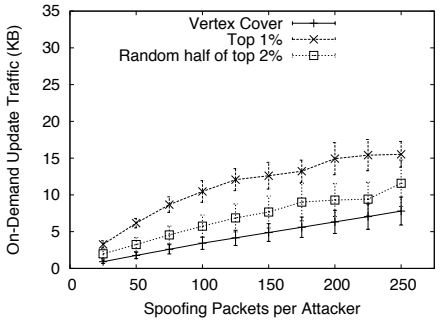
Incentives. As SAVE can only be deployed incrementally, for successful incremental deployment, domains must *want* to deploy SAVE. There must be incentives that include a clear benefit for the deploying domain. "Early adopters" of the protocol should be attracted when the deployment level is still low.



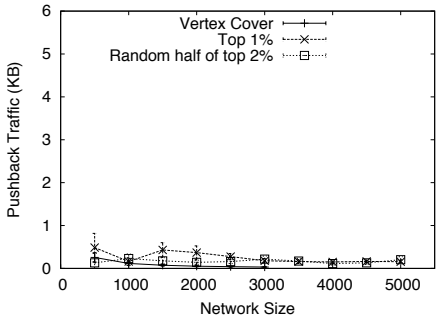
(a) On-Demand Update Traffic



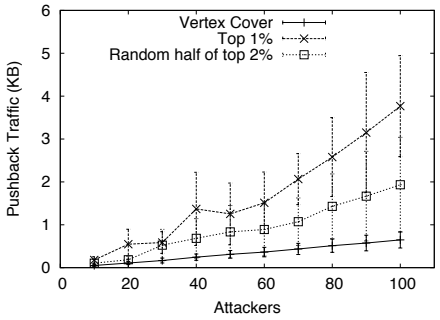
(b) On-Demand Update Traffic (varying attackers)



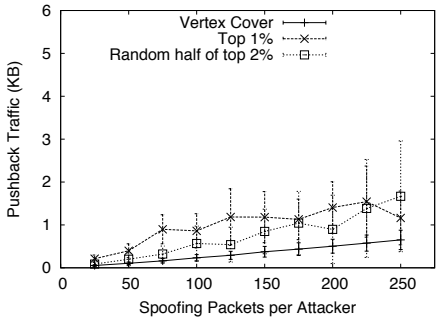
(c) On-Demand Update Traffic (varying packets per attacker)



(d) Pushback Traffic



(e) Pushback Traffic (varying attackers)



(f) Pushback Traffic (varying packets per attacker)

Fig. 6. Network traffic due to spoofing (with 95% confidence intervals).

We already know the following incentives for deploying SAVE on a network: Attackers are less likely to successfully spoof source addresses belonging to a SAVE-protected domain, protecting a domain from misplaced blame and reflection style attacks. A protected domain will also allow fewer spoofing packets to enter its network, protecting internal hosts from receiving spoofing packets. Furthermore, SAVE routers can assign higher priorities to packets from SAVE-protected source address spaces, giving clients with protected sources higher quality of service.

What needs to be further studied is how highest-degree ASes can become incentivized to deploy SAVE. From Section 6.1 we know that deployments at highest-degree ASes will be mostly effective while random deployments will be least. One driving force here could be that knowing the correct source of traffic may help large ISPs monitor and manage their traffic more reliably according to contractual agreements with their customers.

Spoofing Strategies. Attackers may employ peculiar spoofing strategies to evade SAVE's filtering. One such spoofing is random source spoofing in which the attacker stamps a random source address on every packet it sends out. When a SAVE router receives any such packet, it will locate the incoming table entry that matches the inscribed source of the packet. Since for each such packet the SAVE router probably has not seen its inscribed source before (i.e., no blacklist entry established), it will treat everyone of them as suspicious, and still forward them. The router will request on-demand updates, but only to confirm that its incoming table is up-to-date.

Fortunately, the damage an attacker could cause with random spoofing is limited. Random spoofing could hide the attacker's true identity, but random spoofing cannot be used in attacks such as DNS amplification [1, 2], DNS cache poisoning [4], in-window TCP resets [3], and spam filter circumvention [5, 6]. Any type of reflection attack cannot succeed, since traffic triggered by the spoofing packets will spread out through the network instead of becoming concentrated in one area. Similarly, the effect on the SAVE infrastructure is manageable since any requests for on-demand updates will also be spread throughout the network.

We are investigating the most cost-effective way of addressing this spoofing strategy. Our concern with random spoofing is the effect it might have on SAVE itself, so our solution focuses on minimizing the overhead it could generate. In our current solution, described only briefly for space considerations, a router does not request an on-demand update for every suspicious packet. Instead, routers use a truncated binary exponential back off strategy. Initially, a router will request an on-demand update after it sees $n = 1$ suspicious packet. If the requested update shows the incoming table was in fact correct, the router will not request another on-demand update until it sees $n = 2$ more suspicious packets. Every time a requested update arrives, if it agrees with the incoming table, the router subsequently waits for $n = 2n$ more suspicious packets before requesting another on-demand update. We do not allow n to increase over 1024. On the other hand, if the requested update shows the incoming table was incorrect, the router decreases the wait to $n = 1$ suspicious packet. In this manner, random

spoofing by attackers cannot cause too much network overhead nor fill up a router's blacklist; at the same time, SAVE can continue to quickly update its incoming table.

False Positives. As discussed in Section 6.2, our pushback mechanism is subject to false positives when certain conditions are met. Although the transient period for false positives to occur is very short, further minimizing them is important and we plan to study this issue further.

8 Related Work

Source address validation is comprised of end-host methods and router-based methods. To validate the source address of newly received packets, an end-host can either actively probe supposed sources, or passively observe the pattern of packets from them [19]. However, although end-host-based detection is easier to deploy, it cannot prevent spoofing packets from reaching their destinations. Router-based solutions can be classified as preventive approaches (e.g., filtering) or reactive approaches (e.g., traceback). Since SAVE is a router-based, filtering-oriented solution, we focus on these below.

Filtering approaches attempt to identify invalid packets by examining certain attributes of incoming packets at a router. Many approaches have been proposed. Network ingress filtering [7] can stop a spoofing attack at its source, but is useless against spoofing attacks once they enter the Internet. With unicast reverse path forwarding (uRPF) [8], a router drops any packet from an address that does not arrive on the interface that the router uses to reach that address. However, Internet routing is frequently asymmetric: the path from a given address is not necessarily the same as the path to that address [20].

SPM [21] proposes that packets from a source AS to a destination AS carry a key bound with that AS pair, but losing the key to an attacker will enable the attacker to successfully deliver spoofed packets from anywhere. As opposed to SAVE, SPM is also specific to BGP and cannot help intermediate routers gain source validity knowledge.

Passport [22] is also BGP specific and uses keys based on AS pairs. It uses multiple keys based on the packet's source AS and each AS along the path to its destination. This allows intermediate ASes to perform validation, in addition to the destination AS. Passport has problems with packets fragmented in the middle of the network; the fragmentation invalidates the Passport header. Routers therefore forward fragmented packets, both legitimate and spoofed, all the way to their destination. More importantly, intermediate ASes never drop invalid packets, only lower their priority. An attacker's spoofing packets will still reach the destination AS.

The authors of the route-based distributed packet filtering (DPF) [10] studied the benefits of DPF for attack prevention and traceback, as well as its partial deployment strategies. Unfortunately, the work did not specify how routers can learn the incoming direction for different source addresses. IDPF [23] attempts to address this gap. It relies on specific BGP forwarding policies and AS peering

relationships, but only to learn feasible paths, instead of actual paths, from a given source. BASE [24] is another similar work that relies on BGP, and has yet to effectively address commonly seen AS-level routing asymmetry.

Pi [25] and StackPi [26] provide a hybrid approach: routers mark each packet with an identifier for the path that the packet travels, and end hosts examine packets and classify which paths are attack paths and which are not. Pi/StackPi cannot handle fragmented packets correctly, and a spoofing packet must reach its destination before Pi/StackPi can detect it.

9 Conclusion

Research has shown that if a small percentage of routers throughout the Internet deploy a filtering table to discard packets with a forged source address, a synergistic filtering effect can be achieved to stop a large fraction of spoofed IP packets. Such an approach to IP spoofing has also been found to be the most effective. However, in building such a filtering table, specifically an incoming table, we have found that the previously designed SAVE protocol is susceptible to obsolete incoming table entries as it is incrementally deployed.

We introduce new mechanisms in this paper to address this deficiency. We introduce blacklists at SAVE routers and use both the blacklist and the incoming table to classify and filter incoming packets. Our on-demand mechanism enables a SAVE router to deal with suspicious packets and update its incoming table, and the pushback mechanism further pushes the filtering of spoofing packets toward the SAVE router that is the closest to spoofers. With these new mechanisms, and with both security and performance issues considered, we show that incoming-table-based IP spoofing detection is a viable approach to addressing the critical problem of IP spoofing, and that ASes (beginning with high-degree ASes) will have incentives to deploy such a solution. Simulations show that, for example, with deployment at only the top 0.08% of ASes by degree, the efficacy of catching spoofing packets is over 90%.

References

1. Paxson, V.: An analysis of using reflectors for distributed denial-of-service attacks. ACM Computer Communications Review (CCR) **31**(3) (July 2001) 38–47
2. Jackson, D.: DNS amplification variation used in recent DDoS attacks (February 2009) `http://www.secureworks.com/research/threats/dns-amplification/`.
3. Touch, J.: Defending TCP against spoofing attacks. RFC 4953 (July 2007)
4. US-CERT: Multiple DNS implementations vulnerable to cache poisoning (July 2008) Vulnerability Note VU 800113.
5. Morrow, C.: BLS FastAccess internal tech needed (January 2006) `http://www.merit.edu/mail.archives/nanog/2006-01/msg00220.html`.
6. Beverly, R., Berger, A., Hyun, Y., k claffy: Understanding the efficacy of deployed Internet source address validation filtering. In: Proceedings of the ACM Internet Measurement Conference. (November 2009)

7. Ferguson, P., Senie, D.: Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2827 (2000)
8. Baker, F., Savola, P.: Ingress Filtering for Multihomed Networks. RFC 3704 (2004)
9. Ehrenkranz, T., Li, J.: On the state of IP spoofing defense. ACM Transactions on Internet Technology **9**(2) (May 2009) 1–29
10. Park, K., Lee, H.: On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In: Proceedings of ACM SIGCOMM. (2001)
11. Mirkovic, J., Kissel, E.: Comparative evaluation of spoofing defenses. IEEE Transactions on Dependable and Secure Computing **99**(PrePrints) (2009)
12. Li, J., Mirkovic, J., Ehrenkranz, T., Wang, M., Reiher, P., Zhang, L.: Learning the valid incoming direction of IP packets. Computer Networks **52**(2) (February 2008) 399–417
13. Li, J., Mirkovic, J., Wang, M., Reiher, P.L., Zhang, L.: SAVE: Source address validity enforcement protocol. In: Proceedings of IEEE INFOCOM. (June 2002)
14. Kent, S., Lynn, C., Mikkelson, J., Seo, K.: Secure border gateway protocol (S-BGP) — real world performance and deployment issues. In: Proceedings of the Network and Distributed System Security Symposium. (2000)
15. Goodell, G., Aiello, W., Griffin, T., Ioannidis, J., McDaniel, P., Rubin, A.: Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In: Proceedings of the Network and Distributed System Security Symposium. (February 2003)
16. University of Oregon: Route Views Project. `http://www.routeviews.org/`
17. Tyan, H.Y., Sobeih, A., Hou, J.C.: Towards composable and extensible network simulation. In: Proceedings of the International Parallel and Distributed Processing Symposium. (2005)
18. Mahadevan, P., Hubble, C., Krioukov, D.V., Huffaker, B., Vahdat, A.: Orbis: rescaling degree correlations to generate annotated Internet topologies. In: Proceedings of ACM SIGCOMM. (2007)
19. Templeton, S.J., Levitt, K.E.: Detecting spoofed packets. In: Proceedings of the DARPA Information Survivability Conference and Exposition. Volume 1. (2003)
20. Paxson, V.: End-to-end routing behavior in the Internet. In: Proceedings of ACM SIGCOMM. (1996)
21. Bremler-Barr, A., Levy, H.: Spoofing prevention method. In: Proceedings of IEEE INFOCOM. (2005)
22. Liu, X., Li, A., Yang, X., Wetherall, D.: Passport: Secure and adoptable source authentication. In: Proceedings of USENIX Symposium on Networked Systems Design and Implementation. (2008)
23. Duan, Z., Yuan, X., Chandrashekar, J.: Constructing inter-domain packet filters to control IP spoofing based on BGP updates. In: Proceedings of IEEE INFOCOM. (2006)
24. Lee, H., Kwon, M., Hasker, G., Perrig, A.: BASE: An incrementally deployable mechanism for viable IP spoofing prevention. In: Proceedings of the ACM Symposium on Information, Computer, and Communication Security. (2007)
25. Yaar, A., Perrig, A., Song, D.: Pi: A path identification mechanism to defend against DDoS attack. In: Proceedings of the IEEE Symposium on Security and Privacy. (2003)
26. Yaar, A., Perrig, A., Song, D.: StackPi: New packet marking and filtering mechanisms for DDoS and IP spoofing defense. IEEE Journal of Selected Areas in Communications **24**(10) (October 2006) 1853–1863