

Packet Forwarding with Source Verification

Craig A. Shue	Minaxi Gupta	Matthew P. Davy
<i>Computer Science Department</i>	<i>Computer Science Department</i>	<i>Chief Network Engineer</i>
<i>Indiana University</i>	<i>Indiana University</i>	<i>Indiana University</i>
cshue@cs.indiana.edu	minaxi@cs.indiana.edu	mpd@grnoc.iu.edu

Abstract

Routers in the Internet do not perform any verification of the source IP address contained in the packets, leading to the possibility of IP spoofing. The lack of such verification opens the door for a variety of vulnerabilities, including denial-of-service (DoS) and man-in-the-middle attacks. Currently proposed spoofing prevention approaches either focus on protecting only the target of such attacks and not the routing fabric used to forward spoofed packets, or fail under commonly occurring situations like path asymmetry. With incremental deployability in mind, this paper presents two complementary hop-wise packet tagging approaches that equip the routers to drop spoofed packets close to their point of origin. Our simulations show that these approaches dramatically reduce the amount of spoofing possible even under partial deployment.

Keywords:

IP spoofing, denial-of-service, security.

Correspondence Author:

Craig Shue (cshue@cs.indiana.edu)
Lindley Hall 215
150 S. Woodlawn Ave.
Bloomington, IN 47405-7104
001.812.361.7319 (Voice)
001.812.855.4829 (Fax)

I. INTRODUCTION

Packet forwarding in the Internet is based only on the destination IP address contained in the packet. This permits forging of the source IP address, commonly referred to as *IP spoofing*. With as much as a quarter of the Internet able to spoof [1], IP spoofing is a boon for miscreants. Perhaps the most well-known misuse of IP spoofing is in launching denial-of-service (DoS) attacks on critical infrastructure such as Web and DNS servers, as evidenced by backscatter analysis [2], [3]. Another avenue made possible by spoofing is that of illegal content distribution. UDP-based peer-to-peer (p2p) applications that exploit IP spoofing to mask the identity of the sender already exist [4], [5] and it is only a matter of time before illegal content distribution with spoofing becomes an attractive proposition.

In light of the legal and security-related violations that are possible with spoofing, the prevention of spoofing in the Internet is an urgent need. Present approaches to curb IP spoofing can be broadly divided into two categories: *traceback* techniques and prevention techniques. The traceback techniques [6], [7], [8], [9], [10], [11], [12] can trace packet paths and help in identifying the perpetrators of the DoS attacks with a high probability. These can be useful forensic tools in law enforcement but do nothing to prevent the occurrence of IP spoofing. Among the spoofing prevention techniques, many focus on shielding the destination from IP spoofing [13], [14], [15], [16], [17]. Their shortcoming lies in the observation that they fail to protect the Internet routing fabric from being misused in forwarding spoofed packets. The rest of the spoofing prevention techniques possess the ideal goal of preventing spoofing near its source. However, among these, the filtering techniques [18], [19], [20], [21] suffer from two main drawbacks: 1) they can potentially drop valid packets in the face of *routing asymmetries*, a common occurrence in the Internet and 2) they are primarily a goodwill gesture and fail to protect the implementing domains from being spoofed. The remaining techniques either require full deployment [22] or the presence of a key distribution infrastructure [23].

With an expectation of prolonged partial deployment in mind, we present a comprehensive solution to prevent IP spoofing in the Internet. Our work is motivated by two main goals. The first is to *drop spoofed packets as close to their origin as possible, while protecting all valid packets, even in the face of routing asymmetries*. The second goal is to *incentivize deployment and accomplish the above filtering without requiring any infrastructure beyond what is used in the Internet for routing packets today*. It is the latter goal that caused us to choose practical security over provable security and impacted many of the choices made in the work presented here. An example of one of the choices is our decision to not use cryptographic primitives in our solution.

Our solution consists of two approaches. In the first approach, *definitive packet tagging*, implementing routers close to the end-hosts guarantee that hosts in their domain are not spoofing anyone from outside their domain. They signal this

by tagging un-spoofed packets from the network prefixes they originate. These tags are used by the subsequent implementing routers on the path to the destination, which verify the tags and drop packets that are either incorrectly tagged or lack tags when they should be tagged. This hop-wise verification process protects all valid packets while dropping all other packets that attempt to spoof IP addresses from domains that implement definitive packet tagging. After verification, each implementing router, except for the last one before the destination, re-tags the packets with a tag of its own before forwarding the packets towards their destinations. The re-tagging process keeps the number of tags each implementing router has to remember to just the neighboring implementing routers and also limits the damage in the event such a router is compromised. Finally, the last implementing router toward the destination is responsible for stripping off any tags contained in the packets to ensure that the end-hosts do not steal them. We propose two methods to avoid requiring the presence of any additional infrastructure to support tag exchange among neighboring implementing routers. The first method utilizes border gateway protocol (BGP) route announcements to distribute the tags for various network prefixes. This method has minimal additional overhead but fails to provide correct tags in cases where routes are asymmetric or suppressed due to routing policies. For such cases, we develop a second technique, called the *recursive router challenge*, that facilitates the learning of tags for prefixes on-demand. Under this technique, which is modeled after the *traceroute* [24] utility, a router issues challenges of increasing depths upon receiving an unexpected tag.

The second spoofing prevention approach, *deductive packet tagging*, complements the definitive packet tagging approach and is useful under partial deployment scenarios. Under deductive packet tagging, deploying routers can verify and tag traffic from nearby legacy domains that do not deploy definitive packet tagging. We develop a technique called *host probing* to allow routers deploying deductive packet tagging to learn the validity of traffic before tagging it as valid. The host probing technique is a variant of the *TCP intercept* [13] technique which many routers implement today. In our technique, the implementing router interferes with the TCP handshake process of a randomly chosen host to verify tags for a network prefix. However, unlike the TCP intercept, a host probing router does not attempt to stitch the subsequent part of the TCP connection.

Our approaches drop spoofed traffic near its origin and prevent attacks not just on the end-hosts, but also on the Internet infrastructure. Further, they incentivize deployment by protecting the deploying domains from being spoofed. We examine various aspects of our solution using GT-ITM [25] topologies and Skitter data [26]. In particular, we find for GT-ITM that when a mere 10% of the domains deploy definitive packet tagging, 44% of networks are prevented from spoofing a deploying system. The results were even more encouraging for Skitter data, where at 10% deployment, 83% of the networks could not spoof a deploying network. When deductive packet

tagging was used as well, the legacy networks that encountered a deploying router were further limited in their ability to spoof.

The rest of this paper is organized as follows. In Sections II and III, we describe the proposed approaches. Deployment considerations are addressed in Section V and Section VI highlights related work. Evaluation of the proposed approaches is described in Section IV. Finally, Section VII concludes the paper.

II. DEFINITIVE PACKET TAGGING

Routers deploying the definitive packet tagging approach verify that the source address contained in each incoming packet is not spoofed before they forward them towards their destination. (These, and the other changes required to the router functionality, are summarized in Section V.) The rest of the routers in the Internet, including some even in deploying domains, continue to forward packets as they do today. We now describe various aspects of this approach. Throughout this paper, we assume an adversarial model where any number of end-hosts can be compromised but the routers, in general, are trustworthy. The implications of untrustworthy routers in are discussed in Section V.

A. Decision Process at a Deploying Router

A domain deploying definitive packet tagging first has to select the routers whose functionality will be enhanced. The set of routers should be selected so as to guarantee that every packet that either originates from that domain or enters it will pass through at least one deploying router. In a typical domain, routers running external BGP (eBGP) act as border routers. Thus, eBGP routers are a practical choice for the enhanced functionality. Each deploying router independently picks a bit string for each of its interfaces. This string is used to tag packets leaving the router¹.

Figure 1 depicts the decision process at a deploying router. For traffic originating from within the domain², the router uses knowledge about the prefix ranges belonging to its own domain to drop spoofed packets. For packets arriving from other domains, the deploying router first checks if it has any tag records stored for neighboring routers, possibly multiple records for each [source address prefix, incoming interface] pair. (Multiple tags can exist for each [source address prefix, incoming interface] combination due to multi-homing and load balancing. Our approach allows routers to learn and store all such combinations, ensuring correct operation even in the case of path asymmetry, multi-homing, load balancing or pathologies, such as route flapping. We discuss how tags of neighboring BGP routers are learned and where they are stored in Section II-B.) If no tag records exist, the router has no means of distinguishing a spoofed packet from a packet that comes from a domain that does not implement definitive

packet tagging. Such packets are forwarded as they would be today. If at least one tag record exists, the packet must contain a valid tag. The packet is considered spoofed (and is dropped) if the tag contained in the packet either does not match any of the tags stored at the router or if the packet lacks a tag when it should have one³.

Each packet that passes the above check is forwarded. Before forwarding, the router strips the existing tag. Also, in the event the router knows that the packet would encounter another implementing router while en-route to the destination, the local tag corresponding to the outgoing interface is placed in the packet. The process of tag stripping and conditional re-tagging ensures that the end-hosts receiving the packets do not steal router tags, which can be misused for spoofing. In Section II-B, we discuss how the routers find out whether another implementing router would be encountered in the path toward destination and where this information is stored.

B. Learning Tags and Deployment Status of Neighbors

Each deploying router needs to know two pieces of information for processing incoming packets: 1) the tags of neighboring deploying routers (these are the tags incoming packets would contain) and 2) whether the packets would encounter another deploying router on the path to the destination (for deciding whether or not to put its own tag in outgoing packets). Various options exist for learning tags and deployment status of neighboring routers, such as those proposed for securing BGP [27]. However, they all require additional infrastructure, which could hinder deployment. Consequently, we propose two methods for gathering the required pieces of information. These methods are described in Sections II-B.1 and II-B.2.

An important consideration is that of storing the information gathered about deployment status and tags of neighboring routers. Various options exist: The routers can store the deployment status for each destination prefix in the forwarding information base (FIB) by simply incorporating an extra bit. The FIB is an obvious choice because this information is needed while forwarding packets, during which the FIB is consulted anyway. However, the FIB is not suited for storing tags of neighboring routers because multiple tags can exist for each [source address prefix, incoming interface] combination. This will complicate the current FIB forwarding, which expects just one entry for each prefix unless load balancing is in use. For storing the tags of neighboring routers, we introduce an additional data structure, called the *tag table*, which is consulted for each incoming packet. This table is very similar to the FIB and even indexed by network prefixes. The only difference is that the tag table stores incoming interfaces and tags corresponding to each prefix instead of the outgoing interface as the FIB does.

1) Leveraging BGP Route Announcements: BGP announcements are primarily used to convey the autonomous system (AS) path information for the announced pre-

¹This process may require a packet to be tagged twice while traversing the same AS. While this may seem unnecessary, this results in a limited scope of each tag, which results in greater security.

²Edge routers have different interfaces for traffic originating from within the domain and the traffic entering the domain from outside. Thus, these routers can easily distinguish between these two types of traffic.

³Packets containing source or destination IP addresses from invalid prefix ranges should be dropped by each router and it is possible to do it today.

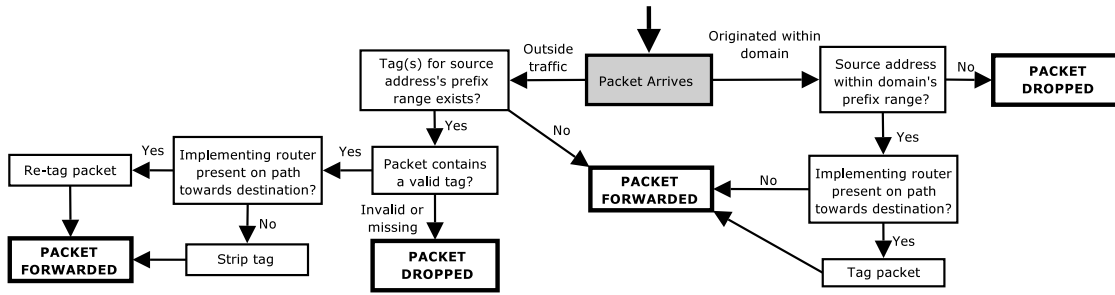


Fig. 1. Flow chart of the definitive packet tagging process.

fixes. However, BGP also supports the COMMUNITIES attributes [28], which are widely used for implementing policy routing. These attributes could be used to distribute tags and 1-bit deployment status among neighboring deploying routers. Unfortunately, the basic COMMUNITIES attribute is only 32 bits in size and many of the values are already standardized. This leaves us with very few usable bits and hence very little security. As a result, it is not a compelling option for tag distribution.

Fortunately, BGP also allows for EXTENDED COMMUNITIES attributes [29] which are now supported by all major vendors. This allows creation of an *opaque extended community* with a 48-bit value field. This value field could contain the tag used by the last deploying router and its 1-bit deployment status. If better security is desired beyond the 47-bit tags, a deploying router could use the value field to send its own IP addresses instead, allowing the next deploying router to send a special packet directly to learn the tag.

The use of EXTENDED COMMUNITIES attributes allows for a peaceful co-existence with legacy routers. Because these community attributes are optional and transitive, legacy systems would simply ignore this extra information, but still propagate it further when re-announcing routes.

Though elegant and simple, the BGP route announcements fail to distribute the required information among neighboring deploying routers in two scenarios. First, under asymmetric routing conditions, the tags and arrival interfaces learned would be incorrect (though the deployment status will still be learned correctly). Second, the suppression of route announcements due to route export policies can prevent routers from announcing certain routes to other routers. This can prevent deploying routers from receiving the required information about many prefixes. To overcome these limitations, we have developed the *recursive router challenge* technique, which we describe next.

2) **Recursive Router Challenges:** The recursive router challenge technique allows a router to learn the deployment status and tags of its neighboring deploying routers on-demand. Short of knowing the routing policies of its neighbors, the best way to learn this information is at the granularity of the source address prefixes contained in incoming IP packets.

We model the recursive router challenge technique after the *traceroute* [24] utility. In order to verify the tag corresponding

to a [source address prefix, incoming interface] pair, a router creates a UDP packet with a *depth* field and nonce value. The source IP address in the packet is that of the initiating router and the destination address is that of a random host in the prefix range whose tag is to be verified. Choosing a random host minimizes the risk of an attacker trying to fool routers into learning tags when no deploying router exists in the path. The random nonce value is used to prove the liveness of the challenged router's response. Each deploying router that encounters this UDP packet decrements the depth field before sending it towards its destination. However, if the depth becomes zero when decremented, the receiving router swaps the source and destination IP addresses in the packet and routes the packet as it normally would. When the swapped packet travels to the initiating router, it follows the path packets whose tag is to be verified would have taken and thus contains the tag of the last deploying router. This allows the initiating router to validate the intended tag even when routes are asymmetric.

Just as in the case of *traceroute*, the initiating router starts with a depth field of one and increments up to a fixed maximum configured value⁴ if the tag it receives in reply to the challenge does not match the tag it attempted to verify. Beyond the maximum possible value of the depth field, the initiator assumes the lack of an implementing router and assumes that the tag it set out to validate was incorrect.

Clearly, while a router is verifying the tag contained in a packet using the recursive router challenge technique, it *cannot hold the incoming packet* because this could incur large delays. To avoid this situation, definitive packet tagging operates in two modes: a *start-up mode*, in which no packets are dropped and each tagged packet is re-tagged, and a *standard mode* where the router has learned all the relevant tags and deployment statuses. The start-up mode allows a router to avoid dropping valid traffic immediately after booting. The standard mode can be used to prevent spoofing once the router has been operating long enough to accumulate an extensive amount of information.

Determining when a deploying router can transition from start-up to standard mode would largely be an engineering decision. An approach similar to the *hot standby* protocol [30]

⁴The maximum value would be configured much like how hosts choose an initial TTL value in their packets: the value should be big enough to reach the destination but small enough to give up after a reasonable number of tries.

can help a faster transition to the standard mode if routes are booted up with the necessary information about tags and deployment status (such as from a previous run in case of a reboot). Alternately, the core routers could track the percentage of packets (later confirmed as valid) that would be discarded by switching to standard mode. When this percentage drops below a certain percentage (e.g., 0.01%), the router would transition to standard mode.

We note that recursive router challenges are required only when a route changes. Typically, when a route change occurs, the new route is selected from a small number of alternate routes, such as when route flapping occurs [31]. If our approach learns each of these alternate routes, due to previous route changes, when future route changes occur, they will not result in the loss of legitimate packets. Additionally, when route changes are detected by BGP, routers could temporarily drop into start-up mode to learn any new tags that result from the change.

An important concern is malicious hosts issuing router challenges to learn router markings. Routers deploying definitive packet marking prevent such occurrences from hosts on their networks by dropping packets containing recursive router challenges. They also drop such packets if they know they are the last deploying router on the path to the requester’s prefix. Hosts not covered by these scenarios can issue challenges. However, as our simulations later show, information gathered through such challenges are not useful in a vast number of cases.

C. Tag Theft

Our discussion thus far indicates that tags need to be available only to legitimate routers. While ideal, there are circumstances in which tags can be stolen by malicious users. We now briefly explore ways tags can be stolen.

Tag theft can occur in a few different ways. First, a compromised or malicious router could publicly share all of the tags it sees. Second, end-hosts in edge networks without a deploying router may be able to see some tags. When sending packets, edge network routers will tag packets unless they know the destination network lies on a path without a deploying router. This is required since edge routers do not have full deployment status information. If another deploying router exists on the path to the destination, it can remove the tag. Otherwise, the packet reaches the destination network with the tag in place. At this point, it is trivial for the host to “steal” the tag. A variant on this second approach allows end-hosts to proactively issue recursive router challenges to these destination networks. But again, the challenge must traverse an all-legacy path before reaching a compliant router in a different edge network.

While tag theft is clearly undesirable, it would not immediately undermine the approach as a whole, as we show in Section IV-C. Intuitively, tags have only local significance, since each tag is specific to a given router’s outgoing interface and because each compliant router performs re-tagging. Accordingly, the network topology and packet routing

significantly curtails the number of hosts that can abuse a tag, should it be stolen.

III. DEDUCTIVE PACKET TAGGING

The definitive packet tagging approach reduces spoofing by preventing deploying networks from being spoofed and from being able to spoof addresses. However, it does nothing to curb the spoofing of IP addresses belonging to legacy networks, which can happen when definitive packet tagging is partially deployed. To limit spoofing attempts by legacy networks, we propose an optional and complementary approach called *deductive packet tagging*, to be deployed by the same routers that would deploy definitive packet tagging. The deductive packet tagging approach allows near-by upstream deploying routers to verify and tag a legacy network’s traffic. This approach can provide significant spoofing protection by exploiting the tree-like branching of edge networks; a single edge router near the core can provide most of the benefits of the definitive approach to numerous edge networks.

Before a router can tag outgoing packets deductively, it needs to determine the valid prefixes from which un-spoofed packets can arrive. This task was trivial under definitive packet tagging because routers knew which prefixes could originate from their domains. However, it is not simple to determine this information for traffic originating in other domains. To infer valid prefixes belonging to legacy domains, we propose a technique called *host probing*, which we describe in Section III-A. After verification, the router adds the [verified source address prefix, incoming interface] combination to its tag table with a blank tag. Any un-tagged packets arriving from a prefix and interface that has a blank tag table entry would subsequently be tagged deductively. Subsequent deploying routers would verify this tag as if it were a definitive tag.

A. Host Probing Technique

The host probing technique is a variant of the TCP intercept technique [13] which is deployed by many routers in the Internet today. In this discussion, we focus on TCP for ease of exposition; however, we note that our approach can be adopted to any protocol that has a request-response nature. In TCP intercept, a local router at the receiver mimics the destination host and performs a TCP handshake with the sender. If the handshake succeeds, it becomes confident that the sender is indeed interested in a two-way connection. At that time, the TCP intercepting router “stitches” the connection between the sender and the receiver. The stitching process requires the router to be involved throughout the duration of the connection. Host probing differs from TCP intercept in that the router does not perform connection stitching, avoiding the costs of long-term connection mapping. Instead, it immediately resets successfully verified connections after the TCP handshake stage. This interferes minimally with connection performance because the source can simply retry the connection.

In particular, in host probing, for each selected TCP SYN packet, a deploying router interferes with the connection establishment by mimicking the packet’s destination: it does

not forward the original handshake packet to the destination and instead sends a reply packet with both the SYN and the ACK flags set. The router then awaits a reply to the packet on the interface the SYN originally arrived on. If a reply returns within a timeout, and the acknowledgment number correctly corresponds to the sequence number the router previously sent, the router issues a RST packet to the source and records a blank tag as valid for the prefix on the given interface. A TCP connection would then attempt to retry the connection. In [32], the authors find resets in 15-25% of the TCP connections. Since host probing is designed to be a rare event, our technique is unlikely to significantly impact this rate.

Deploying routers probabilistically decide which destination address to target for host probing. This is to avoid learning incorrect prefixes from packets generated by a determined attacker. Finally, the host probing technique relies upon a protocol handshake to succeed. While it is possible that a network will never send traffic with handshakes, this scenario is unlikely given the current ubiquity of connection-oriented protocols. If no handshaking protocols exist, echo requests, such as those using ICMP, would be sufficient to verify a host. However, if such approaches failed, manual configuration of valid prefixes would be required.

Finally, when performing a host probe, a router must store a small amount of connection information for a short period. This information includes the source IP address and the sequence number the router placed in its SYN+ACK packet. This 8 byte value would be stored for a few round-trip times from the initiating host before being expired. Even if several thousand host probes are being conducted simultaneously, the memory required to store the relevant information is only of the order of a few tens of KBytes. Further, to completely avoid saving state, the SYN cookie approach could be used [33].

While the neighbors of a deductively tagging router would often be able to use the BGP communities and recursive router challenges to verify the deductively tagging router, there are cases where this could fail and the neighboring routers would need to resort to using the host probing approach. Specifically, if a legacy AS is multi-homed, a recursive router challenge may be unable to elicit a response that returns using the secondary route. In this case, challenging router must instead resort to a host-probe on packets arriving using the secondary route.

We note that the host probing technique is a last resort; it is used only used with legacy domains and only when BGP does not provide sufficient information about the available prefixes. Further, the host probe only needs to be issued once per prefix to confirm availability. Because host probing will be infrequent and because it exploits connection establishment, we believe it would be low overhead and would not pose a significant burden for end-hosts.

IV. EVALUATION

We evaluated the proposed approaches for 1) their effectiveness in preventing spoofing, 2) the extent of tag theft possible and the extent to which stolen tags can be used

for spoofing, and 3) the overheads incurred by deploying routers in learning tags of other deploying routers. The key results from our analysis are: 1) deductive and definitive packet tagging approaches effectively prevent legacy networks from spoofing deploying networks even at low deployment, 2) even if networks steal tags (and share them), only a small percentage is actually able to misuse stolen tags, and 3) the overheads for learning tags are proportionate to the number of prefixes to which routers send traffic.

A. Simulation Configuration

We conducted our simulations on transit-stub GT-ITM [25] graphs of sizes 10,500, 21,000, and 31,500 routers; and Internet topology maps of size 70,000 routers, obtained from the ARIN site of the Skitter project [26]. The GT-ITM tool grouped the routers into domains, which we used to make autonomous systems (ASes). For the Skitter graphs, we grouped the routers into ASes by their respective /24 prefixes. For each topology, we varied the percentage of the compliant ASes to simulate different levels of partial deployment in the Internet. We used an AS-level granularity because adoption is likely to occur at this granularity. All routers within a deploying AS were assumed to be deploying. All routers used a basic shortest-path first algorithm to construct their routing tables.

Since the functionality of core and edge routers differs at times (e.g., edge routers do not store the deployment status beyond their FIB entries, implying that they re-tag all packets when using their default routes), we labeled some routers as *core* in our simulations. The rest were assumed to be edge routers. The core was defined dynamically based on a percentage of the path length from the source to the destination. We varied the core percentage. The non-core routers on the path were equally divided as source and destination edge routers. In particular, we assumed that the first set of routers on the path to destination were source edge routers, the next set was core routers, and the final set was destination edge routers.

B. Effectiveness in Preventing Spoofing

We examined the effectiveness of both the definitive and deductive packet tagging approaches for both the GT-ITM and Skitter topologies. We classified networks into three categories: those that can spoof *no one*, those that can spoof *legacy* networks (but not deploying networks), and those that can spoof both legacy and deploying networks (*all*).

Figure 2 shows the effectiveness of the definitive packet tagging approach for a 10,500 node GT-ITM topology when the core is estimated at 60% of the network. The results are based on 3,200 random samplings of source and destination pairs. The 95% confidence intervals for the mean values presented in Figure 2 varied between $\pm 0.00\%$ and $\pm 1.03\%$ of the mean. In particular, *at 10% deployment, about 44% of the networks are prevented from spoofing deploying networks*. Of these 44% of networks, 10% (the topmost region in Figure 2), can spoof *no one* because they deploy. This is the same percentage as would be prevented from spoofing in the case of *ingress filtering*, a simple scheme that filters packets with

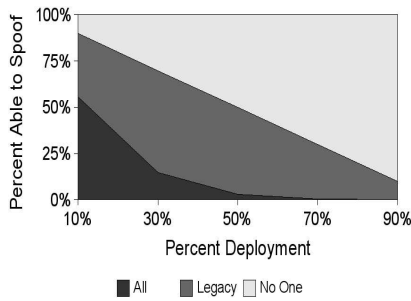


Fig. 2. Percentage of networks that are able to spoof *no one*, only *legacy* networks, and *all* networks under definitive packet tagging (10,500 node GT-ITM topology).

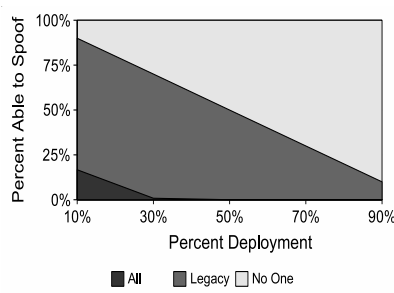


Fig. 3. Percentage of networks that are able to spoof *no one*, only *legacy* networks, and *all* networks under definitive packet tagging (70,000 node Skitter topology).

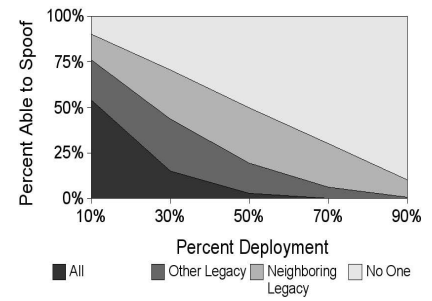


Fig. 4. Percentage of networks that are able to spoof *no one*, only *neighboring legacy* networks, *other legacy* networks, and *all* networks under deductive packet tagging (10,500 node GT-ITM topology).

forged source IP addresses near the packet’s source. The rest of the 34% of the networks (the middle region in Figure 2) can only spoof *legacy* networks, not the networks that deploy definitive packet marking. Even with such limited deployment, deploying networks attain significant protection from reflector attacks. Comparatively, ingress filtering would only prohibit implementing domains from launching such attacks.

The results improve at higher deployment percentages. For example, at 30% deployment, 85% of networks are unable to spoof a deploying network and at 50% deployment, 97% of networks cannot spoof a deploying network. Higher than that, almost no networks can spoof deploying networks. The results were similar for other GT-ITM topologies and different estimates of the core.

Next, we evaluate the effectiveness of definitive packet tagging on Skitter topology data. The results of this simulation on a graph with a 60% core estimate are shown in Figure 3. The graph was much larger in size, requiring us to limit our sampling to 1,600 (source, destination) samples for each data point⁵. The results on the Skitter topology were better than those obtained for GT-ITM graphs: *At a mere 10% deployment, about 83% of the networks cannot spoof a deploying network.* This is obtained by summing the percentage of networks that can spoof *no one* because they deploy and those that can spoof just the *legacy* systems. At 30% deployment, 99% of the networks are unable to spoof deploying networks. At greater levels of deployment, almost no networks are able to spoof deploying networks. The reason definitive packet tagging is more effective on the Skitter graphs than on the GT-ITM graphs has to do with the number of ASes in the topologies. The mask we applied to the Skitter graphs resulted in an average of 2.4 routers per AS, causing more ASes to be picked on an average for each deployment percentage than the GT-ITM graphs. The GT-ITM graphs on the other hand had about 10 routers per transit AS and about 4 routers per stub AS.

We also evaluate the effectiveness of the deductive tagging approach, when it is applied in conjunction with the definitive

tagging. Figure 4 shows its effectiveness in spoofing prevention in a 10,500 node GT-ITM graph with a core estimated at 60% of the network. These results are based on 3,200 random samplings. At 10% deployment, 44% of the networks are prevented from spoofing deploying networks. This result is the same as that for definitive packet marking with a subtle difference. Of the 44% of networks, 10% (the topmost region in Figure 4), can spoof *no one* because they deploy. Another 15% (the second from the top region in Figure 4) of the legacy networks can only spoof *neighboring legacy* networks. These are the legacy networks whose traffic is aggregated before reaching a router that tags packets deductively. Further, another 19% can spoof all *other legacy* networks. These are the legacy networks that either do not encounter a deploying router, or encounter one that does not tag deductively. This is an improvement over definitive packet tagging approach where 34% of the legacy networks could spoof other legacy networks. The results improve for higher deployments: At 30% deployment, about 30% of the networks can spoof *no one*, 25% can spoof only *neighboring legacy* routers, and 30% can spoof all *other legacy* networks. The corresponding numbers for 50% deployment were 50%, 35%, and 10% respectively. The trends were similar for the Skitter topology and the corresponding graphs are omitted due to space constraints.

In Figure 5, we show the benefits on a 10,500 node GT-ITM topology when ingress filtering and our deductive approach are applied cooperatively. In this graph, 70% of ASes are randomly selected to deploy ingress filtering. Additionally, we independently randomly select varying percentages of deployment of our deductive approach. Where the two groups overlap, we just consider them to be deploying our approach, since ingress filtering is a component in our technique. These results are based on 32,000 random samplings of source and destination pairs. We note that a substantial ingress filtering deployment simply results in a greater percentage of systems unable to spoof at all. When combined with 70% ingress filtering deployment, at 10% deployment more than 83% of networks are unable to spoof a deploying system. At the same deployment, this percentage was 44% under just deductive filtering. Further, at 30% deployment, more than 95% of networks are unable to spoof a deploying system.

⁵While our analysis was limited to fewer runs than in the GT-ITM graph, the results were largely unchanged when increasing from 240 to 1,600 sample points. Accordingly, we believe that future trials are unlikely to significantly impact the results.

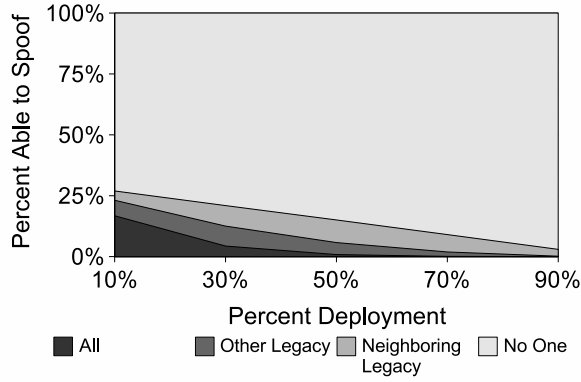


Fig. 5. Percentage of networks that are able to spoof *no one*, only *legacy* networks, and *all* networks under deductive packet tagging with 70% of the network deploying ingress filtering (10,500 node GT-ITM topology).

C. Extent of Tag Theft and Exploitation of Stolen Tags

Since edge routers do not maintain deployment status for packets that travel on the default routes, end hosts can steal tags. We now measure how often the end hosts will be able to learn tags for deploying networks under definitive packet tagging. To be realistic about how much deployment information edge routers will be able to maintain, we assume that if the path to the destination is less than four hops in length, the destination is close enough that the source network would not have to rely on its default route to reach it.

Figure 6 shows the results of the simulation on a 10,500 node GT-ITM graph for various percentages of core routers. Each data point represents an average of 3,200 random samplings. At 10% deployment, 55% to 67% of the networks can steal a tag. At 30% deployment, 14% to 25% of the networks can steal a given tag. At 50% deployment, only 2% to 7.5% of the networks can steal tags. At greater deployments, tag theft is possible less than 1.5% of the time.

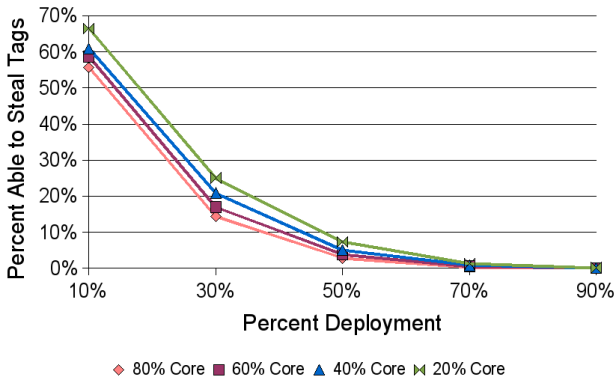


Fig. 6. Percentage of networks that can steal a tag.

Not all stolen tags can be exploited. Even if a host successfully steals a tag used by a given network, it must be topologically situated so that its own packets (containing the forged tags) would be aggregated with the legitimate network's traffic before encountering a deploying router. We

now examine the percentage of networks that can successfully send forged packets with stolen tags. We do so by assuming varying amount of collusion among networks that possess stolen tags.

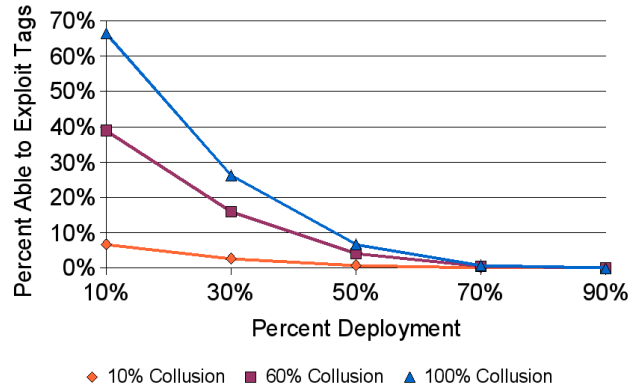


Fig. 7. Percentage of networks that can abuse a tag, assuming a given percentage know the tags.

Figure 7 shows the percentage of networks that can abuse tags for 10,500 node GT-ITM graph. Assuming 100% collusion among tag stealers, at 10% deployment, 66% of networks can abuse a given tag. This drops to 26% at 30% deployment and under 7% at 50% deployment. At 70% or higher deployment, less than 1% of prefixes can effectively abuse a given tag. The abuse drops considerably under more realistic abuse settings. For example, if 10% of the Internet colludes to exchange information about stolen tags, the amount of tag abuse that can occur is under 7%, even at only 10% deployment. These results lead us to conclude that while tag abuse will occasionally be possible, it does not pose a risk of undermining our solution.

D. Overhead Analysis

We now analyze the overheads associated with learning tags under definitive packet marking, assuming that the routers learn deployment status only via BGP announcements. We assume the following inputs:

- percent asymmetry (A) present in the network,
- degree of asymmetry (DEG) - the average number of hops that are asymmetric when asymmetry occurs,
- average path length (L) between a given router and a destination host,
- and percent deployment (DEP).

Assuming no route suppression, the percentage of time that BGP announcements fail to distribute the tags due to asymmetry can therefore be expressed as $A \times \frac{DEG}{L} \times DEP$. This failure is less than 2% for each level of deployment. Each time router announcements fail, a recursive router challenge must be attempted. This must happen for each core router and for each network prefix. Again, this can be modeled analytically given the following additional inputs:

- number of prefixes (P) - the number of prefixes to which a router can send traffic,

- and number of routers (R) - the total number of BGP routers present in the Internet.

The recursive router challenges comprise the main overheads since the tags learned through BGP route announcements are essentially free. Using the above inputs, the overheads of the recursive router challenges can be expressed as $A \times \frac{DEG}{L} \times DEP \times P \times R$. Table I shows the Internet-wide estimated number of recursive router challenges (RRCs) under varying degrees of deployment using this equation. We choose 14% and 30% as estimates of asymmetry, based on prior work analyzing asymmetry [31], [34]. We consider a weighted average of the degree of asymmetry in the Internet from [34] for our DEG input. Further, we used an estimate on the number of FIB prefixes indicated in [35] to estimate the number of network prefixes. While the overheads for learning tags at higher deployment may seem high, the average number of challenges per router are relatively small (about 2,858 challenges at 90% deployment) and can be distributed over a period of time to prevent bursty traffic.

A	DEG	L	DEP	P	R	# RRCs
14%	1.89		10%			31,752,000
14%	1.89		30%			95,256,000
14%	1.89		50%			158,760,000
14%	1.89	15	70%	180,000	100,000	222,264,000
14%	1.89		90%			285,768,000
30%	1.89		10%			68,040,000
14%	2.50		10%			42,000,000

TABLE I

NUMBER OF RECURSIVE ROUTER CHALLENGES FROM VARIOUS INPUTS.

Another concern about recursive router challenges is the amount of time that will be required while waiting for a response. Assuming they are processed quickly by the routers, the amount of time required to perform a challenge is related to the number of links they must traverse. The number of hops a challenge must travel is characterized by $\frac{DEG}{DEP}$. For example, with a degree of asymmetry of 1.89 and 30% deployment, a challenge would have to traverse 6.31 links before reaching a router past the point of asymmetry. Since a router challenge will be successful typically before a round-trip time between end-hosts, connectivity will be restored to legitimate prefixes before a time-out occurs in any connection-oriented protocols at the end-host. Therefore, the lost connectivity could simply be interpreted as temporary congestion by the end-hosts. This reduces the impact of the recursive router challenge approach.

V. DEPLOYMENT AND OTHER PRACTICAL CONSIDERATIONS

A. Changes to Router Functionality

Our approaches require several changes to the functionality of deploying routers. Exchanging deployment status and tags using BGP requires the introduction of new BGP COMMUNITY values. Learning tags and deployment status using the recursive router challenges and host probing techniques requires maintenance of new data structures. Each of these changes are in the control plane, which is not directly involved

in packet forwarding. Thus, they are unlikely to require any changes in router hardware.

While our scheme requires changes to BGP, the alterations are minimal. Further, only deploying systems have to make any of those changes. The rest of the Internet can continue to function as it does today. Several other research projects have proposed changes to BGP as well, with many being much more invasive than our own [27], [36], [37], [38], [39], [40]. Additionally, the BGP COMMUNITY and Extended COMMUNITY values were added to BGP for communicating optional information through the protocol, so it is reasonable to leverage them for this purpose.

The routers must also incorporate changes in the data plane in order to incorporate the required per-packet processing. In our scheme, routers determine the deployment status of a destination prefix by checking an extra bit in their FIB. If set, the router must write the tag associated with the outgoing interface into the packet. Since this information is small, this can be accomplished using fast, dedicated memory. Finally, router lookup functionality needs to be enhanced as well. In particular, the source IP address of each incoming packet is tested for spoofing, using a tag table, before making a forwarding decision. This feature is similar to the other currently proposed techniques that filter spoofed IP addresses near their origin [18], [19], [20], [21]. Each of these changes will likely require changes in router hardware.

Some core routers may simply be unable to bear the costs associated with a tag table lookup. These routers could instead be only partial participants: they would not verify the tags in packets or re-tag packets. Instead, they would simply strip existing tags from packets in which the associated deployment status bit is not set. This would help in reducing tag theft while posing only a minimal burden.

B. Threat Model and Router Compromise

When we consider the threat of IP spoofing, we focus on an attacker attempting to overwhelm a given target. This behavior is consistent with the goal of DoS attacks. The proliferation of worms and botnets makes collusion among end-hosts increasingly likely. Accordingly, in Section IV, we modeled the extent that end-hosts can steal tags from routers that leak this information due to incorrect deployment assumptions. Further, we showed the extent in which malicious hosts can exploit this information, with varying degrees of collusion. We note that even if all the router tags are completely public, at 30% deployment, only 26% of hosts can abuse a tag.

We stated that our adversarial model assumes that hosts could be compromised but routers, in general, are not compromised. If however, malicious routers are present, our approaches degrade gracefully. In particular, malicious routers could divulge tags of their neighbors and/or add valid tags to spoofed packets.

Exposing tags of the neighboring routers to malicious users can only do limited damage because the tags only have local significance due to hop-wise tagging. Therefore, if a router, A, gives the tag router B uses when sending via interface X

to a group of malicious hosts, these malicious hosts would have to produce packets that would traverse a legacy path up to where they are aggregated with traffic leaving router B's interface X. Our simulation results show that this is a difficult task, especially as deployment increases.

Tagging spoofed packets with legitimate tags also does not permit arbitrary spoofing. This is because subsequent routers toward the destination check the validity of any new [source address prefix, tag] pairs and a malicious router could only add valid tags for the legitimate prefixes it forwards.

C. Tag Format and Location

To avoid cryptographic overheads, we assume that a tag is a pseudo-random bit pattern that each deploying router generates in advance for each of its interfaces. This tag is used for each packet sent through the interface. Though tags do not need to be globally unique, they have to be large enough to avoid being guessed by adversaries. As an example, a 64 bit tag would force an adversary to enumerate 9.22×10^{18} tags, on average, before correctly guessing the tag.

The most obvious place for tag placement is the the IP options field in IPv4 and IPv6 headers. Router vendors have been using ASICs extensively for speeding up router lookups and it is now possible for them to incorporate faster processing of specific IP options in the next design cycle. Another option is to reuse unused IPv4 header fields. This has been done by other researchers [12] but requires redefinition of fields by the IETF to be deployable.

D. Efficient Storage of Extra Information and Tag Theft Ramifications

As mentioned in Section II, the 1-bit deployment status of neighboring routers is maintained in the FIB. This scheme is well suited to core routers since their FIBs already contain entries for all Internet prefixes and the addition of 1-bit deployment status would not alter the number or size of FIB entries. However, the edge routers normally only contain a few FIB entries to specific (nearby) network prefixes and rely on one *default route* for the rest of the prefixes. If they actively seek deployment status corresponding to all Internet prefixes, their FIB table size could grow prohibitively large, hurting lookup speeds. An option to avoid this scenario is to have the edge routers restrict storing the deployment information to only the specific network prefixes contained in their FIB. This choice implies that the edge routers sending traffic via a default route cannot be sure whether to tag/re-tag the packets. If they do tag their packets, they cannot be guaranteed that their packets will encounter a deploying router before reaching the end-host. And if none is encountered, the end-hosts can steal router tags. We find in our simulations that the possibility of exploiting tags stolen in this manner is difficult, making the savings worth the small amount of risk. The size of the tag table depends on the prefixes that edge and core routers receive traffic from.

E. Changing Tags

Routers should frequently change their tags for security purposes. A simple approach would be to switch from the old tag to the new tag. In time, adjacent routers would learn the validity of the new tag for each prefix on the given interface. Unfortunately, this means that for routers operating in standard mode, all traffic with the new tag would be dropped until the receiving routers successfully verified the packets. A more compelling approach is for routers to use one-way hash chains [41], [42] to facilitate more rapid tag changes. Hash chains allow a deploying router to generate a practically infinite chain of tags. The end of the hash chain is then used as the first tag. Changing tags is easy under this scheme because the sending router can simply switch to the next tag in the chain, while the routers receiving traffic can still use the previous element in the chain to verify the new value.

The hash chain technique raises the possibility of an attacker guessing the next element in the hash chain, causing downstream routers to incorrectly believe a tag change has already occurred, resulting in legitimate packets being dropped. A simple heuristic could foil this attack: routers accept both the old and new for a short period of time; if the vast majority of the traffic using the chain would be dropped by the transition, the router can continue to use the old tag. Further, as indicated earlier, the chances of guessing the next tag are negligible for a 64 bit tag (it would require 9.22×10^{18} attempts). Smaller tags would be more easily guessed; reusing unused fields in IP options would allow for up to 40 bits for tags (requiring 5.49×10^{11} attempts). While IP options processing in IPv4 is more difficult to do in the fast path, IPv6 options have been redesigned to allow faster processing.

F. Prefix Granularity for Tags

When routers learn tags using BGP route announcements, the question "At what granularity of network prefixes should the tags be stored?" has an obvious answer: the router should add a new entry in the tag table for each network prefix with a BGP route announcement containing the EXTENDED COMMUNITIES attribute. However, when using the recursive router challenges (as described in Section II-B.2) or the host probing (as described in Section III-A) techniques, it is unclear what network prefixes should be used for storing the tags corresponding to each interface. An option is to use the same network prefixes as those appearing in core BGP routing tables. This appears to be a good balance between accuracy of information and the overheads of learning tags and corresponding storage requirements. However, if higher accuracy is desired, the tag table could be made to contain tags at higher granularity, such as that of class C network prefixes.

G. Impact of Routing Changes

Route changes may require deploying routers to learn the validity of new tags arriving via the new routes. This is not an issue for tags learned through BGP announcements because the tag table can be updated along with the routing change.

However, for tags learned using the recursive router challenge technique or the host probing technique, a routing change will cause the deploying router to consider traffic containing the new tags to be spoofed until the validity of the new route is established. To minimize the number of valid packets dropped, deploying routers can probabilistically attempt to verify tags using recursive router challenges upon a packet drop.

H. DoS Considerations

The issue of DoS attacks on routers through exploitation of recursive router challenge and host probing techniques deserves a careful consideration. First, DoS attacks can be launched without the use of IP spoofing. In particular, miscreants can simply send large numbers of unspoofed packets in order to bring the routers down. Thus, mechanisms, such as those prescribed in [43], [44], [45] need to be deployed in the Internet in addition to those that prevent spoofing. With IP spoofing curtailed, [44] and [45] can utilize source IP addresses without fear of collateral damage due to spoofing. Further, both recursive router challenge and host probing techniques could be made a function of link utilization to avoid DoS attacks.

I. Incentives for Adoption

Any IP spoofing prevention scheme comes at some cost to the organizations deploying them. Accordingly, it is essential that these schemes provide benefits to the deploying organizations to incentivize adoption.

Our approach provides several incentives to deploying systems. The approach reduces the amount of malicious traffic leaving the deploying network, reducing the amount of traffic the network must carry. Ingress filtering provides only this benefit, yet has enjoyed widespread adoption [1]. Additionally, it becomes increasingly difficult for malicious hosts to spoof addresses from deploying networks, reducing the threat of reflector-based attacks. Next, packets from deploying networks carry a tag indicating the legitimacy of the source address. This can be used by destination networks for prioritizing scarce resources. Finally, systems employing deductive packet marking can also sell their marking as a service to the legacy systems, providing the associated benefits to the covered systems without requiring additional infrastructure.

When only one system deploys our scheme, it has the same risks and benefits as the ingress filtering scheme. However, if more organizations collaboratively deploy, our scheme offers benefits to all those who deploy. There is precedent for such collaborative deployment. DomainKeys [46], an approach to combat spam by enabling better filtering, also hinges on collaborative deployment.

VI. RELATED WORK

IP spoofing can be of two types: inter-domain spoofing and intra-domain spoofing. Approaches to contain intra-domain spoofing, in which attackers spoof only the addresses of other machines in their respective domains, are available. For example, work in [47], describes one such deployed solution

for spoofing prevention in an Ethernet network. Our paper focuses on the problem of inter-domain spoofing, in which attackers spoof addresses of machines outside their domain.

The approaches to curb IP spoofing can broadly be divided into two categories: *traceback* techniques, that trace the path spoofed packets took and spoofing prevention techniques. The traceback solutions seek to discover the path taken by spoofed packets and are a useful forensic tool. Extensive work has been done in this area [6], [7], [8], [9], [10], [11], [12], resulting in a variety of approaches for implementing traceback that trace the path with fewer number of packets, increased accuracy, and lower overheads. In the latest work in this category of solutions [12], the deploying routers mark the offset field of IP header to identify themselves. This, along with the TTL (time to live) value contained in the packet determines the position of the tagging routers even in the presence of non-deploying routers. Destination hosts can use this information to identify the actual path spoofed packets used.

Several of the spoofing prevention techniques attempt to shield the destination from IP spoofing. They discard invalid packets at the destination network. They go a step further in spoofing prevention but fail to protect the routing fabric from spoofing. In TCP intercept [13], routers near the destination can be configured to mimic the destination in order to ensure the packet is not spoofed. If a valid connection is established, the router stitches both sides of the connection for the entire duration of the connection. The work in [15] associates TTL values in the incoming packets with their source. When an attack begins, the TTL values of arriving packets are compared to the values stored for that domain. Packets not matching their stored TTL values are regarded as spoofed and could be filtered. In [16], the authors propose a deterministic marking approach in which each router adds a fingerprint to each packet by marking the packet's IP Identification based on the packet's TTL value. Victims can use this fingerprint to group packets that took the same path, rather than relying on the source IP addresses. This approach strengthens the victim's packet filtering ability. In [17], each pair of source and destination networks share a secret that is included as a marking in the IP header of all packets exchanged between them. The source network inserts the marking and the destination network verifies and removes the marking before delivering it to the hosts, thus discarding packets with incorrect marks. This prevents spoofing if both source and destination networks deploy, but unlike our approach, it does not prevent non-deploying domains from spoofing IP addresses of deploying domains when sending traffic to other non-deploying domains.

Another category of spoofing prevention techniques, the filtering techniques, have the ideal goal of either blocking spoofed traffic from entering the Internet or filtering it as early as possible. Among the very first techniques in this category of solutions is ingress filtering [18]. This technique prevents spoofing by checking the validity of the source IP address near packet origination. The extent of deployment is critical to the success of ingress filtering because it offers no protection from spoofed packets that escape into the Internet through

legacy routers. Another filtering approach, reverse path forwarding (RPF), uses BGP routing information to determine the possible interfaces from which un-spoofed prefixes can originate. RPF is comprised of three techniques [19], [20], [21]: *strict RPF*, *loose RPF*, and *feasible RPF*. Strict RPF and loose RPF consult the forwarding information base (FIB) to determine the legitimacy of arriving packets. Feasible RPF instead consults a table containing all the accepted routes before tie-breakers are applied to select the best route. This extra information allows a feasible RPF router to overcome some of the limitations of strict and loose RPF. However, each of the RPF approaches will drop valid packets if a router does not receive a routing advertisement for a given prefix, but nonetheless receives traffic originating from that prefix.

The work in [14] uses full routing information to perform filtering and provides valuable metrics for evaluating filtering effectiveness, which we leverage in our work. However, it only addresses intra-domain routing, while our work applies to inter-domain routing. In [48], the authors present a work that utilizes some approaches similar to our own. However, this work requires the distribution of a hash chain to other deploying routers. While they leverage BGP COMMUNITIES in a method similar to our own, they do not properly account for asymmetry in their distribution. Additionally, while they make provisions for storing multiple entries for asymmetry, they do not describe any automated approaches for detecting asymmetry or adding these entries. Our work overcomes these limitations with the introduction of our recursive router challenge and host probing techniques.

Work in [22] proposes to prevent spoofing via a protocol that allows routers to announce how they will send traffic. This protocol is used to construct a table of valid source IP addresses which can be used to filter invalid sources. Per the authors, reaping the benefits of this protocol under partial deployment is challenging. Finally, work in [23] prevents spoofing by the use of hash-based message authentication codes that indicate the AS path traversed by a router. Secret keys are used as input to the hashes and must be communicated using a Diffie-Hellman exchange, in turn relying upon a public key infrastructure for validation of the exchange. Our approach has the same goal as this proposal but does not require hashing on a per-packet basis and requires less data storage in the packet header, yielding better performance. We also avoid relying on a public key infrastructure and expensive public key operations.

VII. CONCLUSION

In this paper, we presented a practical hop-wise packet tagging solution to prevent IP spoofing in the Internet. Our approaches, definitive and deductive packet tagging, allow routers to drop spoofed packets close to their origination. The design of the proposed approaches was guided by deployability concerns, which led us to choose practical security over provable security.

Beyond their use in IP spoofing prevention, the proposed approaches offer an additional advantage: the tags contained

in un-spoofed packets can be used for *prioritizing* traffic from legitimate deploying systems over best effort traffic under attack conditions. The prioritization can be done by configuring firewalls to drop all un-tagged traffic.

ACKNOWLEDGMENTS

Our evaluation used topology data collected by CAIDA's Skitter initiative. The simulations were conducted on the departmental infrastructure that was made possible by the NSF grant EIA-0202048.

REFERENCES

- [1] R. Beverly and S. Bauer, "The spoofer project: Inferring the extent of Internet source address filtering on the Internet," *USENIX Workshop on Steps to Reduce Unwanted Traffic in the Internet (SRUTI)*, 2005.
- [2] D. Moore, G. Voelker, and S. Savage, "Inferring internet denial-of-service activity," in *USENIX Security Symposium*, 2001.
- [3] D. Moore, C. Shannon, D. Brown, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service activity," *ACM Transactions on Computer Systems (TOCS)*, May 2006.
- [4] J. Connelly, "SUMI: A fast anonymous file transfer program website," http://sumi.berlios.de/wiki/Main_Page.
- [5] RODI, "Rodi website," <http://rodi.sourceforge.net/>.
- [6] S. Bellovin, M. Leech, and T. Taylor, "ICMP traceback messages," IETF Draft, Oct. 2001.
- [7] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to IP traceback," *ACM Transactions on Information and System Security*, 2002.
- [8] M. Goodrich, "Efficient packet marking for large-scale IP traceback," in *ACM Conference on Computer and Communications Security (CCS)*, 2001.
- [9] J. Li, M. Sung, J. Xu, , and L. Li, "Large-scale IP traceback in high-speed Internet: Practical techniques and theoretical foundations," in *IEEE Symposium on Security and Privacy*, May 2004.
- [10] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *ACM SIGCOMM*, Aug. 2000.
- [11] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer, "Hash-based IP traceback," in *ACM SIGCOMM*, 2001.
- [12] A. Yaar, A. Perrig, and D. Song, "FIT: Fast internet traceback," in *IEEE INFOCOM*, 2005.
- [13] "Configuring TCP intercept," <http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed.cr/secur.c/scpr3/scdenial.htm#xtcid254812/>.
- [14] H. Lee and K. Park, "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack," in *IEEE INFOCOM*, Apr. 2001.
- [15] C. Jin, H. Wang, and K. Shin, "Hop-count filtering: an effective defense against spoofed DDoS traffic," in *ACM Conference on Computer and Communications Security (CCS)*, 2003.
- [16] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against DDoS attacks," in *IEEE Symposium on Security and Privacy*, 2003.
- [17] A. Bremner-Barr and H. Levy, "Spoofing prevention method," in *IEEE INFOCOM*, 2005.
- [18] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing," IETF RFC 2827, May 2000, updated by RFC 3704.
- [19] F. Baker and P. Savola, "Ingress filtering for multihomed networks," IETF RFC 3704, Mar. 2004.
- [20] *Configuring Unicast Reverse Path Forwarding*, Cisco Systems, Mar. 2004.
- [21] *Unicast Reverse Path Forwarding Enhancements for the Internet Service Provider-Internet Service Provider Network Edge*, Cisco Systems, 2005.
- [22] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang, "SAVE: Source address validity enforcement protocol," in *IEEE INFOCOM*, 2002.
- [23] X. Liu, X. Yang, D. Wetherall, and T. Anderson, "Efficient and secure source authentication with packet passports," in *USENIX Workshop on Steps to Reduce Unwanted Traffic in the Internet (SRUTI)*, 2006.
- [24] G. Malkin, "Traceroute using an IP option," IETF RFC 1393 (Experimental), Jan. 1993.

- [25] "Georgia Tech Internetwork Topology Models," <http://www.cc.gatech.edu/projects/gtitm/>.
- [26] C. A. for Internet Data Analysis (CAIDA), "Skitter project website," 2005, <http://www.caida.org/tools/measurement/skitter/>.
- [27] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, 2000.
- [28] R. Chandra and P. Traina, "BGP communities attribute," IETF RFC 1997 (Proposed Standard), Aug. 1996.
- [29] S. Sangli, D. Tappan, and Y. Rekhter, "BGP extended communities attribute," IETF RFC 4360 (Proposed Standard), Feb. 2006.
- [30] T. Li, B. Cole, P. Morton, and D. Li, "Cisco Hot Standby Router Protocol (HSRP)," IETF RFC 2281, Mar. 1998.
- [31] V. Paxson, "End-to-end routing behavior in the internet," *IEEE Transactions Of Networking*, 1997.
- [32] M. Arlitt and C. Williamson, "An analysis of TCP reset behaviour on the internet," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 1, 2005.
- [33] D. J. Bernstein, "Syn cookies," 1997, <http://cr.yp.to/syncookies.html>.
- [34] Y. He, M. Faloutsos, and S. Krishnamurthy, "Quantifying routing asymmetry in the internet at the AS level," *IEEE GLOBECOM*, 2004.
- [35] "CIDR report," <http://www.cidr-report.org/>.
- [36] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz, "Listen and whisper: security mechanisms for BGP," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.
- [37] Y. Hu, A. Perrig, and M. Sirbu, "SPV: secure path vector routing for securing BGP," in *ACM SIGCOMM*, 2004.
- [38] T. Wan, E. Kranakis, and P. van Oorschot, "Pretty secure BGP (psBGP)," in *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2005.
- [39] J. Karlin, S. Forrest, and J. Rexford, "Pretty good BGP (pgBGP)," in *IEEE ICNP*, 2006.
- [40] N. Kushman, S. Kandula, D. Katabi, and B. Maggs, "R-BGP: Staying connected in a connected world," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.
- [41] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, 24(11):770-772, November 1981.
- [42] Y. Hu, M. Jakobsson, and A. Perrig, "Efficient constructions for one-way hash chains," in *International Conference on Applied Cryptography and Network Security (ACNS)*, 2005.
- [43] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *ACM SIGCOMM*, 2005.
- [44] J. Ioannidis and S. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2002.
- [45] K. Argyraki and D. Cheriton, "Active Internet traffic filtering: Real-time response to denial-of-service attacks," in *USENIX Annual Technical Conference (USENIX)*, 2005.
- [46] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas, "DomainKeys identified mail (DKIM) signatures," IETF RFC 4871 (Proposed Standard), May 2007.
- [47] *Cisco Catalyst Integrated Security - Enabling the self-defending network*, Cisco Systems, 2004, White Paper.
- [48] H. Lee, M. Kwon, G. Hasker, and A. Perrig, "BASE: An incrementally deployable mechanism for viable IP spoofing prevention," in *ACM Symposium on Information, Computer and Communication Security (ASIACCS)*, Mar. 2007.