Controlling IP Spoofing Through Inter-Domain Packet Filters*

Zhenhai Duan†, Xin Yuan, and Jaideep Chandrashekar

Abstract

The Distributed Denial of Services (DDoS) attack is a serious threat to the legitimate use of the Internet. Prevention mechanisms are thwarted by the ability of attackers to forge, or spoof, the source addresses in IP packets. By employing IP spoofing, attackers can evade detection and put a substantial burden on the destination network for policing attack packets. In this paper, we propose an inter-domain packet filter (IDPF) architecture that can mitigate the level of IP spoofing on the Internet. A key feature of our scheme is that it does not require global routing information. IDPFs are constructed from the information implicit in BGP route updates and are deployed in network border routers. We establish the conditions under which the IDPF framework works correctly in that it does not discard packets with valid source addresses. Based on extensive simulation studies, we show that even with partial deployment on the Internet, IDPFs can proactively limit the spoofing capability of attackers. In addition, they can help localize the origin of an attack packet to a small number of candidate networks.

Keywords: IP Spoofing, DDoS, BGP, Network-level Security and Protection, Routing Protocols

1 Introduction

Distributed Denial of Service (DDoS) attacks pose an increasingly grave threat to the Internet, as evidenced by recent DDoS attacks mounted on both popular Internet sites and the Internet infrastructure [14]. Alarmingly, DDoS attacks are observed on a daily basis on most of the large backbone networks [17]. One of the factors that complicate the mechanisms for policing such attacks is *IP spoofing*, the act of forging the source addresses in IP packets. By masquerading as a different

*A preliminary version of this paper appeared in the Proc. IEEE INFOCOM 2006 with the title "Constructing Inter-Domain Packet Filters to Control IP Spoofing Based on BGP Updates".

Zhenhai Duan and Xin Yuan are with the Computer Science Department, Florida State University, Tallahassee, FL 32306 USA (email: {duan,xyuan}@cs.fsu.edu).
Jaideep Chandrashekar is with the Intel Research/CTL, Santa Clara, CA 95054 USA (email: jaideep.chandrashekar@intel.com)

†Corresponding author.

host, an attacker can hide its true identity and location, rendering source-based packet filtering less effective. It has been shown that a large part of the Internet is vulnerable to IP spoofing [2].

Recently, attackers are increasingly staging attacks via *bot-nets*[1] [16]. In this case, since the attacks are carried out through intermediaries, i.e., the compromised "bots", attackers may not utilize the technique of IP spoofing to hide their true identities. It is tempting to believe that the use of IP spoofing is less of a factor. However, recent studies [14, 20, 22] show that IP spoofing is still a common phenomenon: it is used in many attacks, including the high-profile DDoS attacks on root DNS servers in early February 2006 [14]. In the response to this event, the ICANN security and stability advisory committee made three recommendations [14]. The first and long-term recommendation is to adopt source IP address verification, which confirms the importance of the IP spoofing problem.

It is our contention that IP spoofing will remain popular for a number of reasons. First, IP spoofing makes it harder to isolate attack traffic from legitimate traffic—packets with spoofed source addresses may appear to be from all around the Internet. Second, it presents the attacker with an easy way to insert a level of indirection, which shifts the burden to the victim; substantial effort is required to localize the source of the attack traffic [26]. Finally, many popular attacks, such as man-in-the-middle attacks [30, 28], reflector-based attacks [24], and TCP SYN flood attacks [4], use IP spoofing and require the ability to forge source addresses.

Although attackers can insert arbitrary source addresses into IP packets, they cannot, however, control the actual paths that the packets take to the destination. Based on this observation, Park and Lee [23] proposed the *route-based packet filters* as a way to mitigate IP spoofing. The idea is that, assuming single-path routing, there is exactly one single path $p(s, d)$ between source node s and destination node d. Hence, any packets with source address s and destination address d that appear in a router not in $p(s, d)$ should be discarded. The challenge is that constructing such a route-based packet filter requires the knowledge of global routing information, which is hard to reconcile on the current BGP-based Internet routing infrastructure [25].

The current Internet consists of approximately 15,000 autonomous systems (ASes), each of which

[1]collections of hundreds or thousands of compromised hosts, "recruited" by worm or virus infection

is a logical collection of networks with common administrative control. Each AS communicates with its neighbors using the Border Gateway Protocol (BGP), the de-facto inter-domain routing protocol, to exchange information about its own networks and others that it can reach [25]. BGP is a *policy-based* routing protocol in that both the selection and the propagation of the best route to a destination at an AS are guided by some locally defined routing policies. Given the insular nature of how policies are applied at individual ASes, it is impossible for an AS to acquire the complete knowledge of routing decisions made by all other ASes. Hence, constructing route-based packet filters as proposed in [23] is an open challenge in the current Internet routing regime.

Inspired by the route-based packet filters [23], we propose an Inter-Domain Packet Filter (IDPF) architecture, a route-based packet filter system that can be constructed solely based on the locally exchanged BGP updates, assuming all ASes employ a set of routing policies that are commonly used today [10, 12, 13]. The key contributions of this paper are as follows. First, we describe how to practically construct IDPFs at an AS by only using the information in the *locally* exchanged BGP updates. Second, we establish the conditions under which the proposed IDPF framework works correctly in that it does not discard packets with valid source addresses. Third, to evaluate the effectiveness of the proposed architecture, we conduct extensive simulation studies based on AS topologies and AS paths extracted from real BGP data provided by the Route-Views project [21]. The results show that, even with partial deployment, the architecture can proactively limit an attacker's ability to spoof packets. When a spoofed packet cannot be stopped, IDPFs can help localize the attacker to a small number of candidate ASes, which can significantly improve the IP traceback situation [26]. In addition, IDPF-enabled ASes (and their customers) provide better protection against IP spoofing attacks than the ones that do not support IDPFs. This should give network administrators incentives to deploy IDPFs.

The rest of this paper is organized as follows. We discuss related work in Section 2. We provide an abstract model of BGP in Section 3. Section 4 presents the IDPF architecture. Section 5 discusses practical deployment issues. We report our simulation study of IDPFs in Section 6. We conclude the paper in Section 7.

2 Related Work

The idea of IDPF is motivated by the work carried out by Park and Lee [23], which was the first effort to evaluate the relationship between topology and the effectiveness of route-based packet filtering. The authors showed that packet filters that are constructed based on the *global* routing information can significantly limit IP spoofing when deployed in just a small number of ASes. In this work, we extend the idea and demonstrate that filters that are built based on *local* BGP updates can also be effective.

Unicast reverse path forwarding (uRPF) [1] requires that a packet is forwarded only when the interface that the packet arrives on is exactly the same used by the router to reach the source IP of the packet. If the interface does not match, the packet is dropped. While simple, the scheme is limited given that Internet routing is inherently asymmetric, i.e., the forward and reverse paths between a pair of hosts are often quite different. The uRPF loose mode [6] overcomes this limitation by removing the match requirement on the specific incoming interface for the source IP address. A packet is forwarded as long as the source IP address is in the forwarding table. However, the loose mode is less effective in detecting spoofed packets. In Hop-Count Filtering (HCF) [15], each end system maintains a mapping between IP address aggregates and valid hop counts from the origin to the end system. Packets that arrive with a different hop count are suspicious and are therefore discarded or marked for further processing. In Path Identification [32], each packet along a path is marked by a unique Path Identifier (Pi) of the path. Victim nodes can filter packets based on Pi carried in the packet header. StackPi [33] improved the incremental deployment property of Pi by proposing two new packet marking schemes. In [18], Li *et al.*, described SAVE, a new protocol for networks to propagate valid network prefixes along the same paths that data packets will follow. Routers along the paths can thus construct the appropriate filters using the prefix and path information. Bremler-Barr and Levy proposed a spoofing prevention method (SPM) [3], where packets exchanged between members of the SPM scheme carry an authentication key associated with the source and destination AS domains. Packets arriving at a destination domain with an invalid authentication key (w.r.t. the source domain) are spoofed packets and are discarded.

In the Network Ingress Filtering proposal described in [8], traffic originating from a network is forwarded only if the source IP in the packets belongs to the network. Ingress filtering primarily prevents a specific network from being used to attack others. Thus, while there is a collective social benefit in everyone deploying it, individuals do not receive direct incentives. Finally, the Bogon Route Server Project [29] maintains a list of *bogon* network prefixes that are not routable on the public Internet. Examples include private RFC 1918 address blocks and unassigned address prefixes. Packets with source addresses in the bogon list are filtered out. However, this mechanism cannot filter out attack packets carrying routable but spoofed source addresses.

3 Border Gateway Protocol and AS Interconnections

In this section, we briefly describe a few key aspects of BGP that are relevant to this paper (see [27] for a comprehensive description). We model the AS graph of the Internet as an *undirected* graph $G = (V, E)$. Each node $v \in V$ corresponds to an Autonomous System (AS), and each edge $e(u, v) \in E$ represents a BGP session between two neighboring ASes $u, v \in V$. To simplify the exposition, we assume that there is at most one edge between neighboring ASes.[2]

Each node owns one or multiple network prefixes. Nodes exchange BGP route updates, which may be announcements or withdrawals, to learn of changes in reachability to destination network prefixes. A route announcement contains a list of *route attributes* associated with the destination network prefix. Of particular interest to us are the path vector attribute, `as_path`, which is the sequence of ASes that this route has been propagated over, and the `local_pref` attribute that describes the *degree of local preference* associated with the route. We will use `r.as_path`, `r.local_pref`, and `r.prefix` to denote the `as_path`, the `local_pref`, and the destination network prefix of r, respectively. Let $r.\texttt{as\_path} = \langle v_k v_{k-1} \ldots v_1 v_0 \rangle$. The route was originated (first announced) by node v_0, which owns the address space described by `r.prefix`. Before arriving at node v_k, the route was carried over nodes $v_1, v_2, \ldots, v_{k-1}$ in that order. For $i = k, k-1, \ldots, 1$, we say that edge $e(v_i, v_{i-1})$ is on the AS path, or $e(v_i, v_{i-1}) \in r.\texttt{as\_path}$.

When there is no confusion, route r and its AS path $r.\texttt{as\_path}$ are used interchangeably. For convenience, we also consider a specific destination AS d; all route announcements and withdrawals

[2]This is merely for convenience; our scheme is correct without this assumption.

are specific to the network prefixes owned by d. For simplicity, notation d is also used to denote the network prefixes owned by the AS d. As a consequence, a route r that can be used to reach the network prefixes owned by destination d may simply be expressed as a route to *reach destination* d.

3.1 Policies and Route Selection

Each node only selects and propagates to neighbors a single *best* route to the destination, if any. Both the selection and the propagation of best routes are governed by locally defined routing policies. Two distinct sets of routing policies are typically employed by a node: *import* policies and *export* policies. Neighbor-specific import policies are applied upon routes learned from neighbors, whereas neighbor-specific export policies are imposed on locally-selected best routes before they are propagated to the neighbors.

In general, *import* policies can affect the "desirability" of routes by modifying route attributes. Let r be a route (to destination d) received at v from node u. We denote by $\mathbf{import}(v \leftarrow u)[\{r\}]$ the possibly modified route that has been *transformed* by the import policies. The transformed routes are stored in v's routing table. The set of all such routes is denoted as $\mathbf{candidateR}(v, d)$:

$$\mathbf{candidateR}(v, d) = \{r : \mathbf{import}(v \leftarrow u)[\{r\}] \neq \{\}, r.\mathbf{prefix} = d, \forall u \in N(v)\}. \tag{1}$$

Here, $N(v)$ is the set of v's neighbors.

Among the set of candidate routes $\mathbf{candidateR}(v, d)$, node v selects a single best route to reach the destination based on a well defined procedure (see [27]). To aid in description, we shall denote the outcome of the selection procedure at node v, i.e., the best route, as $\mathbf{bestR}(v, d)$, which reads the *best route to destination d at node v*. Having selected $\mathbf{bestR}(v, d)$ from $\mathbf{candidateR}(v, d)$, v then exports the route to its neighbors after applying neighbor specific *export policies*. The export policies determine if a route should be forwarded to the neighbor, and if so, modify the route attributes according to the policies (Section 3.2). We denote by $\mathbf{export}(v \rightarrow u)[\{r\}]$ the route sent to neighbor u by node v, after node v applies the export policies on route r.

BGP is an incremental protocol: updates are generated only in response to network events. In the absence of any events, no route updates are triggered or exchanged between neighbors, and we say that the routing system is in a stable state. Formally,

Definition 1 (Stable Routing State) *A routing system is in a stable state if all the nodes have selected a best route to reach other nodes and no route updates are generated (and propagated).*

3.2 AS Relationships and Routing Policies

The specific routing policies that an AS employs internally is largely determined by economics: connections between ASes follow a few commercial relations. A pair of ASes can enter into one of the following arrangements [10, 13]:

• *provider-customer:* In this arrangement, a customer AS pays the provider AS to carry its traffic. It is the most common when the provider is much larger in size than the customer.

• *peer-peer:* In a mutual peering agreement, the ASes decide to carry traffic from each other (and their customers). Mutual peers do not carry transit traffic for each other.

• *sibling-sibling:* In this arrangement, two ASes provide mutual transit service to each other. Each of the two sibling ASes can be regarded as the provider of the other AS.

An AS's relationship with a neighbor largely determines the neighbor-specific import and export routing policies. In this paper we assume that each AS sets its import routing policies and export routing policies according to the rules specified in Table 1 [12] and Table 2 [10, 13], respectively. These rules are commonly used by ASes on the current Internet. In Table 1, r_1 and r_2 denote the routes (to destination d) received by node v from neighbors u_1 and u_2, respectively; and $customer(v)$, $peer(v)$, $provider(v)$, and $sibling(v)$ denote the set of customers, peers, providers, and siblings of node v, respectively. The import routing policies in Table 1 state that an AS will prefer the routes learned from customers or siblings over the routes learned from peers or providers.

In Table 2, the columns marked with **r1-r4** specify the export policies employed by an AS to announce routes to providers, customers, peers, and siblings, respectively. For instance, export rule **r1** instructs that an AS will announce routes to its own networks, and routes learned from customers and siblings to a provider, but it will not announce routes learned from other providers and peers to the provider. The net effect of these rules is that they limit the possible paths between each pair of ASes. Combined together, the import and export policies also ensure the propagation of valid routes on the Internet. For example, combining the import and export policies, we can guarantee that a provider will propagate a route to a customer to other ASes (customers, providers, peers, and siblings). If an AS does not follow the import policies, it may prefer an indirect route via a peer instead of a direct route to a customer. In this case, based on export rule $r3$, the AS will not propagate the route (via a peer) to a customer to a peer, since the best route (to the customer) is learned from a peer. This property is critical to the construction and correctness of IDPFs (see Sections 4.2 and 4.3). The routing policies in Tables 1 and 2 are incomplete. In some cases, ASes may apply less restrictive policies. For the moment, we assume that all ASes follow the import and export routing policies specified in Tables 1 and 2 and that each AS accepts legitimate routes exported by neighbors. More general cases will be discussed at the end of the next section.

```
if $((u_1 \in customer(v) \cup sibling(v))$ and $(u_2 \in peer(v) \cup provider(v)))$ then
  $r_1$.local_pref $>$ $r_2$.local_pref
```

Table 1: Import routing policies at an AS.

Export rules		**r1**	**r2**	**r3**	**r4**
Export routes to		provider	customer	peer	sibling
Learned from	provider	**no**	**yes**	**no**	**yes**
	customer	**yes**	**yes**	**yes**	**yes**
	peer	**no**	**yes**	**no**	**yes**
	sibling	**yes**	**yes**	**yes**	**yes**
Own routes		**yes**	**yes**	**yes**	**yes**

Table 2: Export routing policies at an AS.

If AS b is a provider of AS a, and AS c is a provider of AS b, we call c an *indirect* provider of a, and a an *indirect* customer of c. Indirect siblings are defined in a similar fashion. The import and export routing policies in Tables 1 and 2 imply that an AS will distribute the routes to direct or indirect customers/siblings to its peers and providers. If $e(u, v) \in \mathbf{bestR}(s, d)$.as_path, we say that u is the best upstream neighbor of node v for traffic from node s to destination d, and denote u as $u = \mathbf{bestU}(s, d, v)$. For ease of exposition, we augment the AS graph with the relationships between neighboring ASes. We refer to an edge from a provider to a customer AS as a provider-to-customer edge, an edge from a customer to provider as a customer-to-provider edge, and an edge connecting sibling (peering) ASes as sibling-to-sibling (peer-to-peer) edge. A *downhill* path is a sequence of edges that are either provider-to-customer or sibling-to-sibling edges, and an *uphill path* is a sequence of edges that are either customer-to-provider or sibling-to-sibling edges. Gao [10] established the following theorem about the candidate routes in a BGP routing table.

Theorem 1 (Gao [10]) *If all ASes set their export policies according to* **r1-r4***, any candidate route in a BGP routing table is either (a) an uphill path, (b) a downhill path, (c) an uphill path followed by a downhill path, (d) an uphill path followed by a peer-to-peer edge, (e) a peer-to-peer edge followed by a downhill path, or (f) an uphill path followed by a peer-to-peer edge, which is followed by a downhill path.*

4 Inter Domain Packet Filters

In this section, we discuss the intuition behind the IDPF architecture, describe how IDPFs are constructed using BGP route updates, and establish the correctness of IDPFs. After that, we discuss the case where ASes have routing policies that are less restrictive than the ones in Tables 1 and 2. We shall assume that the routing system is in the *stable routing state* in this section. We will discuss how IDPFs fare with network routing dynamics in the next section.

Let $M(s, d)$ denote a packet whose source address is s (or more generally, the address belongs to AS s), and destination address d. A packet filtering scheme decides whether a packet should be forwarded or dropped based on certain criteria. One example is the route-based packet filtering [23]:

Definition 2 (Route-Based Packet Filtering) *Node v accepts packet $M(s, d)$ forwarded from node u if and only if $e(u, v) \in \mathbf{bestR}(s, d)$. Otherwise, the source address of the packet is spoofed, and the packet is discarded by v.*

In the context of preventing IP spoofing, an ideal packet filter should discard spoofed packets while allowing legitimate packets to reach the destinations. Since even with the perfect routing information, the route-based packet filters cannot identify all spoofed packets [23], a valid packet filter should focus on not dropping any legitimate packets while providing the ability to limit spoofed packets. Accordingly, we define the correctness of a packet filter as follows.

Definition 3 (Correctness of Packet Filtering) *A packet filter is* correct *if it does not discard packets with valid source addresses when the routing system is stable.*

Clearly, the route-based packet filtering is correct, because valid packets from source s to destination d will only traverse the edges on $\mathbf{bestR}(s, d)$. Computing route-based packet filters requires the knowledge of $\mathbf{bestR}(s, d)$ on every node, which is impossible in BGP. IDPF overcomes this problem.

4.1 IDPFs overview

The following concepts will be used in this section. A *topological route* between nodes s and d is a loop-free path between the two nodes. Topological routes are implied by the network connectivity. A topological route is a *feasible route* under BGP if and only if the construction of the route does not violate the routing policies imposed by the commercial relationship between ASes (Tables 1 and 2). Formally, let $\mathbf{feasibleR}(s, d)$ denote the set of feasible routes from s to d, then $\mathbf{feasibleR}(s, d)$ can be recursively defined as follows:

$$\mathbf{feasibleR}(s, d) =$$
$$\{\langle s \oplus \cup_{u : \mathbf{import}(s \leftarrow u)[\{r\}] \neq \{\}, r.\mathtt{prefix} = d, u \in N(s)} \mathbf{feasibleR}(u, d)\rangle\}, \tag{2}$$

where $\oplus$ is the concatenation operation, e.g., $\{s \oplus \{\langle ab\rangle, \langle uv\rangle\}\} = \{\langle sab\rangle, \langle suv\rangle\}$. Notice that $\mathbf{feasibleR}(s, d)$ contains *all* the routes between the pair that does not violate the import and export routing policies specified in Tables 1 and 2. Obviously, $\mathbf{bestR}(s, d) \in \mathbf{candidateR}(s, d) \subseteq \mathbf{feasibleR}(s, d)$. Each of the feasible routes can potentially be a candidate route in a BGP routing table. Theorem 1 also applies to feasible routes.

Definition 4 (Feasible Upstream Neighbor) *Consider a feasible route* $r \in \mathbf{feasibleR}(s, d)$*. If an edge* $e(u, v)$ *is on the feasible route, i.e.,* $e(u, v) \in r.\mathtt{as\_path}$*, we say that node* u *is a feasible upstream neighbor of node* v *for packet* $M(s, d)$*. The set of all such feasible upstream neighbors of* v *(for* $M(s, d)$*) is denoted as* $\mathbf{feasibleU}(s, d, v)$*.*

The intuition behind the IDPF framework is the following. First, it is possible for a node v to infer its feasible upstream neighbors using BGP route updates. The technique to infer feasible

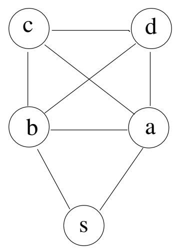
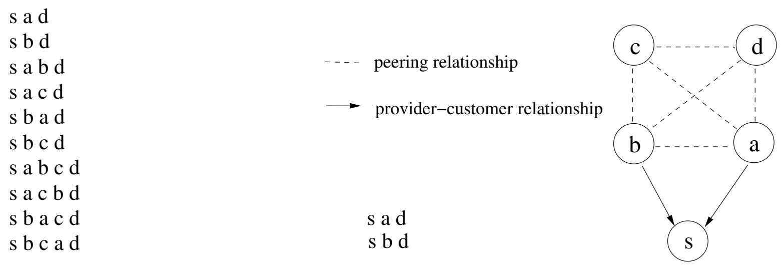


Figure 1: An example network topology.



s a d
s b d
s a b d
s a c d
s b a d
s b c d
s a b c d
s a c b d
s b a c d
s b c a d

(a) Topological routes implied by connectivity

s a d
s b d

(b) Feasible routes constrained by routing policies

Figure 2: Routes between source s and destination d.

upstream neighbors is described in the next sub-section. Since $\mathbf{bestR}(s, d) \in \mathbf{candidateR}(s, d) \subseteq \mathbf{feasibleR}(s, d)$, a node can only allow $M(s, d)$ from its feasible upstream neighbors to pass and discard all other packets. Such a filtering will not discard packets with valid source addresses. Second, although network connectivity (topology) may imply a large number of topological routes between a source and a destination, commercial relationship between ASes and routing policies employed by ASes act to restrict the size of $\mathbf{feasibleR}(s, d)$. Consider the example in Figure 1. Figures 2(a) and (b) present the topological routes implied by network connectivity and feasible routes constrained by routing policies between source s and destination d, respectively. In Figure 2(b) we assume that nodes a, b, c, and d have mutual peering relationship, and that a and b are providers to s. We see that although there are 10 topological routes between source s and destination d, we only have 2 feasible routes that are supported by routing policies. Of more importance to IDPF is that, although network topology may imply all neighbors can forward a packet allegedly from a source to a node, feasible routes constrained by routing policies help limit the set of such neighbors. As an example, let us consider the situation at node d. Given that only nodes a and b (but not c) are on the feasible routes from s to d, node d can infer that all packets forwarded by node c and allegedly from source s are spoofed and should be discarded.

It is clear that IDPF is less powerful than route-based packet filters [23] since the IDPF filters are computed based on $\mathbf{feasibleR}(s, d)$ instead of $\mathbf{bestR}(s, d)$. The point is that $\mathbf{feasibleU}(s, d, v)$ can be inferred from local BGP updates while $\mathbf{bestU}(s, d, v)$ cannot.

4.2 Constructing IDPFs

The following lemma summarizes the technique to identify the feasible upstream neighbors of node v for packet $M(s,d)$.

Lemma 2 *Consider a feasible route r between source s and destination d. Let $v \in r.\mathtt{as\_path}$ and u be the feasible upstream neighbor of node v along r. When the routing system is stable, $\mathbf{export}(u \to v)[\{\mathbf{bestR}(u,s)\}] \neq \{\}$, assuming that all ASes follow the import and export routing policies in Tables 1 and 2 and that each AS accepts legitimate routes exported by neighbors.*

Lemma 2 states that if node u is a feasible upstream neighbor of node v for packet $M(s,d)$, node u must have exported to node v its best route to reach the *source* s.

Proof: Since Theorem 1 applies to feasible routes, a feasible route can be one of the six types of paths in Theorem 1. In the following, we assume the feasible route r is of type (f), i.e., an uphill path followed by a peer-to-peer edge, which is followed by a downhill path. Cases where r is of types (a)-(e) can be similarly proved. To prove the lemma, we consider the possible positions of nodes u and v in the feasible route.

Case 1: Nodes u and v belong to the uphill path. Then node s must be an (indirect) customer or sibling of node u. From the import routing policies in Table 1 and the export routing policy **r1** and the definition of indirect customers/siblings, we know u will propagate to (provider) node v the reachability information of s.

Case 2: $e(u,v)$ is the peer-to-peer edge. This case can be similarly proved as case 1 (based on the import routing policies in Table 1 and the export routing policy **r3**).

Case 3: Nodes u and v belong to the downhill path. Let $e(x,y)$ be the peer-to-peer edge along the feasible route r, and note that u is an (indirect) customer of y. From the proof of case 2, we know that node y learns the reachability information of s from x. From the export routing policy **r2** and the definition of indirect customers, node y will propagate the reachability information of s to node u, which will further export the reachability information of s to (customer) node v. ∎

Based on Lemma 2, a node can identify the feasible upstream neighbors for packet $M(s,d)$ and conduct inter-domain packet filtering as follows.

Definition 5 (Inter-Domain Packet Filtering (IDPF)) *Node v will accept packet $M(s,d)$ forwarded by a neighbor node u, if and only if* $\mathbf{export}(u \rightarrow v)[\{\mathbf{bestR}(u,s)\}] \neq \{\}$. *Otherwise, the source address of the packet must have been spoofed, and the packet should be discarded by node v.*

4.3 Correctness of IDPF

Theorem 3 *An IDPF as defined in* Definition *5 is correct.*

Proof: Without loss of generality, consider source s, destination d, and a node $v \in \mathbf{bestR}(s,d).\texttt{as\_path}$ such that v deploys an IDPF filter. To prove the theorem, we need to establish that v will not discard packet $M(s,d)$ forwarded by the best upstream neighbor u, along $\mathbf{bestR}(s,d)$.

Since $\mathbf{bestR}(s,d) \in \mathbf{candidateR}(s,d) \subseteq \mathbf{feasibleR}(s,d)$, u is also a feasible upstream neighbor of node v for packet $M(s,d)$. From Lemma 2, u must have exported to node v its best route to source s. That is $\mathbf{export}(u \rightarrow v)[\{\mathbf{bestR}(u,s)\}] \neq \{\}$. From *Definition* 5, packet $M(s,d)$ forwarded by node u will not be discarded by v. ■

Notice that the destination address d in a packet $M(s,d)$ plays no role in an IDPF node's filtering decision (*Definition* 5). By constructing filtering tables based on source address alone (rather than *both* source and destination addresses), per-neighbor space complexity for an IDPF node is reduced from $O(N^2)$ to $O(N)$, where $N = |V|$ is the number of nodes in the graph (the route-based scheme can achieve the same complexity bound [23]).

It is worth noting that IDPFs filter packets based on whether the reachability information of a network prefix is propagated by a neighbor, not on how the BGP updates are propagated. As long as ASes propagate network reachability information according to rules in Tables 1 and 2, IDPFs should work correctly. Moreover, the effectiveness of IDPFs is determined largely by the size of $\mathbf{feasibleR}(s,d)$, which is a function of the (relatively static) AS relationships. Hence, how the BGP updates are propagated does not affect both the correctness and the performance of IDPFs. For example, the multiple path advertisement supported by MIRO [31] will not affect IDPFs' correctness and effectiveness.

4.4 Routing Policy Complications

As discussed earlier, the import routing policies and the export routing policies specified in Tables 1 and 2 are not complete. In particular, multi-homed ASes may employ less restrictive routing policies for traffic engineering or other purposes [11]. In this section we first present two traffic engineering examples that do not follow the import and export routing policies specified in Tables 1 and 2. Then, we discuss how ASes employing these special traffic engineering practices should control the forwarding of their traffic to ensure the delivery of their traffic in the IDPF framework.

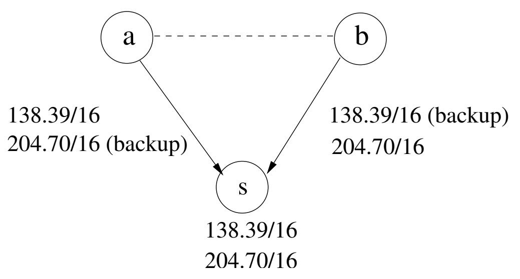


Figure 3: Automatic backup route.

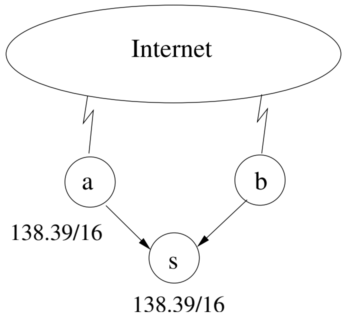


Figure 4: Conditional route advertisement.

In the first example (Figure 3), taken from [27], ASes a and b are providers of AS s, and s has two prefixes `138.39/16` and `204.70/16`. The link between a and s is used as the primary link for `138.39/16` and backup link for `204.70/16`; while the link between b and s is used in a reverse manner. To achieve this traffic engineering goal, s informs a to assign the direct customer route r_1 between a and s a lower local preference over the peering route r_2 learned from b to reach the network prefix `204.70/16`. That is r_1.`local_pref` < r_2.`local_pref`. This local preference assignment at node a does not follow the import routing policies defined in Table 1, which requires that an AS should prefer a direct route over an indirect route (through a peer) to reach a customer.

Now consider the example in Figure 4. Customer s has a primary provider a and a backup provider b. AS s realizes this goal using a technique called conditional route advertisement; prefix `138.39/16` is announced to the backup provider b only if the link to the primary provider a fails. This asymmetric advertisement does not follow the export routing policy **r1** defined in Table 2, which states that a customer will always export to its providers the routes to its own prefixes.

In the examples, the customer s controls the route propagation either by manipulating the local preference of the routes in providers (Figure 3) or by conditional route advertisement (Figure 4). As long as the customer AS does not forward packets through the backup route while the primary route is still available, the IDPF architecture will not discard any valid packets. This requirement is not hard to meet since the customer controls both the route propagation and traffic delivery. The same observation applies to other cases when the routing policies specified in Tables 1 and 2 are not followed. We have the following restricted traffic forwarding policy for the ASes that do not follow the routing policies specified in Tables 1 and 2.

1. **Restricted traffic forwarding policy:** If an AS does not follow the import and export routing policies in Tables 1 and 2, as long as the primary route is available, the AS should not forward traffic along other (backup) routes.

If each AS on the Internet follows the import routing policies in Table 1 and the export routing policies in Table 2 or the restricted traffic forwarding policy, we can establish the correctness of IDPFs as defined in *Definition* 5 on the Internet. The proof is similar to the one of Lemma 2 and Theorem 3 and we omit it here.

5 Practical Deployment Issues of IDPFs

5.1 Incremental Deployment

From the description in Section 4, it should be clear that the IDPFs can be deployed independently in each AS. IDPFs are deployed at the border routers so that IP packets can be inspected before they enter the network. By deploying IDPFs, an AS constrains the set of packets that a neighbor can forward to the AS: a neighbor can only successfully forward a packet $M(s, d)$ to the AS after it announces the reachability information of s. All other packets are identified to carry spoofed source addresses and discarded at the border router of the AS. In the worst case, even if only a single AS deploys IDPF and spoofed IP packets can get routed all the way to the AS in question, using an IDPF perimeter makes it likely that spoofed packets will be identified, and blocked, at the perimeter. Clearly, if the AS is well connected, launching a DDoS attack upon the perimeter itself takes a lot more effort than targeting individual hosts and services within the AS. In contrast, ASes that do not deploy IDPF offer relatively little protection to the internal hosts and services. Therefore, an AS has direct benefits to deploy IDPFs. In general, by deploying IDPFs, an AS can also protect other ASes to which the AS transports traffic, in particular, the customer ASes. This can be similarly understood that, an IDPF node limits the set of packets forwarded by a neighbor and destined for a customer of the AS.

5.2 Handling Routing Dynamics

So far, we have assumed that the AS graph is a static structure. In reality, the graph does change, triggering the generation of BGP updates and altering the paths that ASes use to reach each other. In this subsection, we examine how routing dynamics may affect the operation of IDPFs. We consider two different types of routing dynamics: 1) those caused by network failures; 2) and those caused by the creation of a new network (or recovery from a fail-down network event). Routing dynamics caused by routing policy changes can be similarly addressed and we omit them here.

IDPF filters are completely oblivious to the specifics of the announced routes. Following a network failure, the set of feasible upstream neighbors will not admit more members during the period of routing convergence assuming AS relationships are static, which is true in most cases. Hence, for the first type of routing dynamics (network failure), there is no possibility that the filters will block a valid packet. We illustrate this as follows: consider an IDPF enabled AS v that is on the best route from s to d. Let $u = \mathbf{bestU}(s, d, v)$, and let $U = \mathbf{feasibleU}(s, d, v)$. A link or router failure between u and s can have three outcomes: 1) AS u can still reach AS s, and u is still chosen to be the best upstream neighbor for packet $M(s, d)$, i.e, $u = \mathbf{bestU}(s, d, v)$. In this situation, although u may explore and announce multiple routes to v during the path exploration process [5], the filtering function of v is unaffected. 2) AS u is no longer the best upstream neighbor for packet $M(s, d)$; another feasible upstream neighbor $u' \in U$ can reach AS s and is instead chosen to be the new best upstream neighbor (for $M(s, d)$). Now, both u and u' may explore multiple routes; however, since u' has already announced a route (about s) to v, the IDPF at v can correctly filter (i.e., accept) packet $M(s, d)$ forwarded from u'. 3) No feasible upstream neighbors can reach

s. Consequently, AS v will also not be able to reach s, and v will no longer be on the best route between s and d. No new packet $M(s, d)$ should be sent through v.

The other concern of routing dynamics relates to how a newly connected network (or a network recovered from a fail-down event) will be affected. In general, a network may start sending data immediately following the announcement of a (new) prefix, even before the route has had time to propagate to the rest of the Internet. During the time for the route to be propagated, packets from this prefix may be discarded by some IDPFs if the reachability information has not propagated to them. However, the mitigating factor here is that in contrast to the long convergence delay that follows failure, reachability for the new prefix will be distributed far more speedily. In general, the time taken for such new prefix information to reach an IDPF is proportional to the shortest AS path between the IDPF and the originator of the prefix and independent of the number of alternate paths between the two. Previous work has established this bound to be $O(L)$, L being the diameter of the AS graph [5]. We believe that in this short timescale, it is acceptable for IDPFs to potentially behave incorrectly (discarding valid packets). It must be noted that during BGP route convergence periods, without IDPF, BGP can also drop packets. One alternative solution is to allow a neighbor to continue forwarding packets from a source within a grace period, after the corresponding network prefix has been withdrawn by the neighbor. In this case, during this short time of period, IDPFs may fail to discard spoofed attack packets. However, given that most DDoS attacks require a persistent train of packets to be directed at a victim, not discarding spoofed packets for this short period of time should be acceptable. We plan to further investigate the related issues in the future.

In short, IDPFs can handle the routing dynamics caused by network failures, which may cause long route convergence times. IDPFs may, however, drop packets in the network recovery events. We argue that this is not a big problem since (1) the network recovery events typically have a short convergence time; and (2) such events can also cause service disruptions in the original BGP without IDPF.

5.3 Overlapping Prefixes

In the IDPF architecture, all ASes along the path from s to d can spoof the source address of s and reach d without being filtered out. The route-based packet filtering has a similar behavior. Due to this property, IDPF is most effective when different ASes own non-overlapping prefixes. For example, let s be 1.2/16, all ASes along the path from s to d can spoof this prefix. Now, if there is a more specific address $s' = 1.2.3/24$ somewhere in the network, all these ASes can now also spoof s' since a more specific prefix also matches a more general prefix. This situation does not happen when prefixes are not overlapped. Hence, statistically, IDPF is more effective when prefixes are not overlapped. However, due to the ubiquitous use of classless addressing, CIDR [9], the prefixes owned by different ASes may overlap. The effect of overlapping prefixes will be studied in the next section.

6 Performance Studies

In this section, we first discuss the objectives of our performance studies and the corresponding performance metrics. We then describe the data sets and specific settings used in the simulation studies. Finally, detailed results obtained from simulations are presented.

6.1 Objectives and Metrics

We evaluate the effectiveness of IDPFs in controlling IP spoofing based DDoS attacks from two complementary perspectives [23]. First, we wish to understand how effective the IDPFs are in *proactively* limiting the capability of an attacker to spoof addresses of ASes other than his own. IDPFs do not provide complete protection and spoofed packets may still be transmitted. Thus, the complementary, *reactive* view is also important; we study how the deployed IDPFs can improve IP traceback effectiveness by localizing the actual source of spoofed packets. Since the (incremental) deployment of IDPFs directly affects the effectiveness, various deployment scenarios are considered. The last dimension of our simulation studies concerns the issue of incentive, i.e., how an individual AS will benefit from deploying IDPF on its routers.

We use the performance metrics introduced in [23] in our study. Given any pair of ASes, say a and t, $S_{a,t}$ is the set of ASes, from which an attacker in AS a can forge addresses to attack t. For any pair of ASes, s and t, $C_{s,t}$ is the set of ASes, from which attackers can attack t using addresses belonging to s, without such packets being filtered before they reach t.

To establish a contrast: $S_{a,t}$ quantifies the *pool of IP addresses* that may be forged by an attacker in a to send packets to t without being stopped. On the other hand, $C_{s,t}$ is defined from the victim's perspective. This quantifies the size of the set of ASes that can forge an address belonging to s in sending packets to t without being discarded along the way. Thus, the latter is a measure of the *effort* required, at AS t, to trace the packets to the actual source (there are $|C_{s,t}|$ locations that the packet could have originated from).

6.1.1 Proactive Prevention Metrics

Given the AS graph $G = (V, E)$, we define the *prevention* metric from the point of view of the victim as follows:

$$VictimFraction(\tau) = \frac{|\{t : \forall a \in V, |S_{a,t}| \leq \tau\}|}{|V|}$$

$VictimFraction(\tau)$, redefined from [23], denotes the proportion of ASes that satisfy the following property: if an arbitrary attacker intends to generate spoofed packets, he can successfully use the IP addresses of at most τ ASes (note that this includes the attacker's own AS). Thus, $VictimFraction(\tau)$ represents the effectiveness of IDPFs in *protecting* ASes against spoofing-based DDoS attacks—the fraction of ASes that can be attacked by attackers who can spoof addresses of at most τ networks. For instance, $VictimFraction(1)$, which should be read as *the fraction of ASes that can be attacked with packets from at most 1 AS*, describes the immunity to *all* spoofing based attacks.

Next, we define a metric from the attacker's perspective. Given $G = (V, E)$, $AttackFraction(\tau)$, defined in [23], describes the fraction of ASes from which an attacker can forge addresses belonging to at most τ ASes (including the attacker's own), in attacking any other ASes in the graph.

$$AttackFraction(\tau) = \frac{|\{a : \forall t \in V, |S_{a,t}| \leq \tau\}|}{|V|}$$

Intuitively, $AttackFraction(\tau)$ is the strength of IDPFs in *limiting* the spoofing capability of an arbitrary attacker. For instance, $AttackFraction(1)$ quantifies the fraction of ASes from which an attacker cannot spoof any address other than his own.

6.1.2 Reactive IP Traceback Metrics

To evaluate the effectiveness of IDPFs in reducing the IP traceback effort, i.e., the act of determining the true origin of spoofed packets, $VictimTraceFraction(\tau)$ is defined in [23], which is the proportion of ASes being attacked that can localize the true origin of an attack packet to be within τ ASes.

$$VictimTraceFraction(\tau) = \frac{|\{t : \forall s \in V, |C_{s,t}| \leq \tau\}|}{|V|}$$

For instance, $VictimTraceFraction(1)$ is simply the fraction of ASes, which when attacked, can correctly identify the (single) source AS that the spoofed packet was originated from.

6.1.3 Incentives to Deploy IDPF

To formally study the gains that ASes might accrue by deploying IDPFs on their border routers, we introduce a related set of metrics, $VictimFraction\_IDPF(\tau)$, $AttackFraction\_IDPF(\tau)$, and $VictimTraceFraction\_IDPF(\tau)$. Let T denote the set of ASes that support IDPFs.

$$\begin{aligned} VictimFraction\_IDPF(\tau) &= \frac{|\{t \in T : \forall a \in V, |S_{a,t}| \leq \tau\}|}{|T|} \\ AttackFraction\_IDPF(\tau) &= \frac{|\{a \in V : \forall t \in T, |S_{a,t}| \leq \tau\}|}{|V|} \\ VictimTraceFraction\_IDPF(\tau) &= \frac{|\{t \in T : \forall s \in V, |C_{s,t}| \leq \tau\}|}{|T|} \end{aligned}$$

Note that these are similar to the metrics defined earlier, i.e., $VictimFraction(\tau)$, $AttackFraction(\tau)$, and $VictimTraceFraction(\tau)$, respectively. However, we restrict the destinations to the set of IDPF enabled ASes, rather than the entire population of ASes.

6.2 Data Sets

In order to evaluate the effectiveness of IDPFs, we construct four AS graphs from the BGP data archived by the Oregon Route Views Project [21]. The first three graphs, denoted G_{2003}, G_{2004}, and G_{2005} are constructed from single routing table snapshots (taken from the first day in each of the years). While these provide an indication of the evolutionary trends in the growth of the Internet AS graph, they offer only a partial view of the existing connectivity [10]. In order to obtain a more comprehensive picture, similar to [7], we construct G_{2004c} by combining G_{2003} and *an entire year of BGP updates* between G_{2003} and G_{2004}. Note that the Slammer worm attack [19], which caused great churn of the Internet routing system, occurred during this period of time. This had the side effect of exposing many more edges and paths than would be normally visible.[3] Nonetheless, it is worth pointing out that, even with this effort, the AS graphs we constructed still may only represent a partial view of the Internet AS-level topology, and may not capture all the feasible routes between a pair of source and destination. Thus, we may over-estimate the performance of IDPFs, especially for G_{2003}, G_{2004}, and G_{2005}.

Table 3 summarizes the properties of the four graphs. In the table we enumerate the number of nodes, edges, and AS paths that we could extract from the datasets. We also include the size of the vertex cover for the graph corresponding to individual datasets (the construction is described later). From the table we see that, G_{2004c} has about 22000 more edges compared to G_{2004}, or a 65.9% increase. Also, the number of observed AS paths in G_{2004c} is an order of magnitude more than the observed paths in the G_{2004} data.

Table 3: Graphs used in the performance studies.

Graph	# of Nodes	# of Edges	# of AS paths	VC size (%)
G_{2003}	14516	27406	373350	2124 (14.6%)
G_{2004}	16566	34217	731240	2422 (14.6%)
G_{2005}	18949	39879	811342	2734 (14.4%)
G_{2004c}	18684	56763	7489979	3319 (17.8%)

[3]Given the lengthy period over which we applied the updates, it is likely that our AS graph includes “stale-edges”, i.e., edges that no longer exist. We ignore this effect in our study, noting that AS relationships are quite stable, and thus the number is likely to be very small.

6.2.1 Inferring Feasible Upstream Neighbors

In order for each AS to determine the feasible upstream neighbors for packets from source to destination, we also augment each graph with the corresponding AS paths used for constructing the graph [21]. We infer the set of feasible upstream neighbors for a packet at an AS as follows. In general, if we observe an AS path $\langle v_k, v_{k-1}, \ldots, v_0 \rangle$ associated with prefix P, we take this as an indication that v_i announced the route for P to v_{i+1}, i.e., $v_i \in \mathbf{feasibleU}(P, v_{i+1})$, for $i = 0, 1, \ldots, k-1$.

6.2.2 Determining Routes between Two Nodes

Given an AS graph $G = (V, E)$ and a subset of nodes $T \subseteq V$ deploying the IDPFs, the route that a packet takes from source node s to destination node t will determine the IDPFs that the packet will encounter on the way. Consequently, *in order to compute the described performance metrics*, we require the exact routes that will be taken between any pairs of nodes. Unfortunately, there is simply no easy way to get this knowledge accurately. In this paper, as a heuristic, we simply use the shortest path on G. When there are multiple candidates, we arbitrarily select one of them. As a consequence, in addition to AS paths, we also include the selected shortest path as a feasible route, if it has not been described in the routing updates observed. Note that this knowledge, i.e., the best path from an AS to another, is only required in the simulation studies to determine the IDPFs that a packet may encounter on the way from source to destination. It is *not required in the construction of the IDPFs.* Note also that due to the way that feasible neighbors are computed, the effectiveness of IDPFs may be artificially inflated since the the set of feasible neighbors of a node in our simulations is a subset of feasible neighbors of the node in reality (with the complete Internet topology).

6.2.3 Selecting IDPF Nodes

Given a graph $G = (V, E)$, the effectiveness of IDPF depends heavily on the the *filter set*, i.e., nodes in V to support IDPF. We consider two methods for selecting IDPF nodes, which represents two ways that IDPFs can be incrementally deployed. In the first method, denoted as *Top*, we aggressively select the nodes with the highest degree to deploy IDPF. An special case of this method, denoted as VC, is selecting the IDPF nodes until a vertex cover of G is formed. The number of nodes to form the VC for each data set is shown in Table 3. In the second method, denoted as *Rnd*, we randomly (uniformly) choose the nodes from V until a desirable proportion of nodes are chosen. We will use the notion *RndX* and $TopX$ to denote the selection of X percent of all nodes for deploying IDPFs using the *Rnd* and Top methods, respectively. For example, *Rnd*30 represents selecting 30% nodes to be IDPF nodes using the *Rnd* method. Notice that the *Rnd* method represents more realistic IDPF deployment scenario where ASes decide whether to deploy IDPF independently.

6.3 Results of Performance Studies

The studies are performed with the *Distributed Packet Filtering (dpf)* simulation tool [23]. We extended *dpf* to support our own filter construction based on BGP updates and to deal with overlapping prefixes. We evaluated the performance of IDPFs using the three performance metrics ($VictimFraction(\tau)$, $AttackFraction(\tau)$, and $VictimTraceFraction(\tau)$) under different situations. In addition, we also studied the impact of using BGP updates instead of precise routing information to construct packet filters, investigated the effect of overlapping prefixes in the Internet, and considered IDPFs with and without network ingress filtering. Before we describe the simulation results in detail, we briefly summarize the salient findings.

- IDPFs can significantly limit the spoofing capability of an attacker. For example, with the VC IDPF coverage on the 2004c data set, an attacker in more than 80% of ASes cannot successfully launch any spoofing-based attack on the Internet (assuming no overlapping prefixes are announced). Moreover, with the same configuration, the AS under attack can localize the true origin of an attack packet to be within 28 ASes, therefore, greatly reducing the effort of IP traceback. In this summary, unless specified otherwise, all example data are based on the VC IDPF coverage on the 2004c data set with the assumptions that IDPF nodes are also capable of ingress filtering and that there are no overlapping prefixes.

- The placement of IDPFs plays a key role in the effectiveness of IDPFs in controlling spoofing-based attacks. It is much more effective to deploy IDPFs on ASes with high connectivity (such as tier-1 ISPs) than deploying IDPFs on random ASes. For example, deploying IDPFs on 5% of ASes selected by the *Top* method is more effective than deploying IDPFs on 30% of ASes selected by the *Rnd* method in all of the three performance metrics.

- In comparison to constructing filters with precise routing information, constructing filters with BGP updates does not significantly degrade the IDPF performance in limiting spoofed packets. However, the IDPF traceback capability is affected fairly substantially. For example, the number of nodes that cannot launch any spoofing-based attacks drop from 84% to 80% (a slight decrease) while the number of ASes that an AS can pinpoint as the potential true origin of an attack packet increases from 7 to 28 (a fairly large increase).

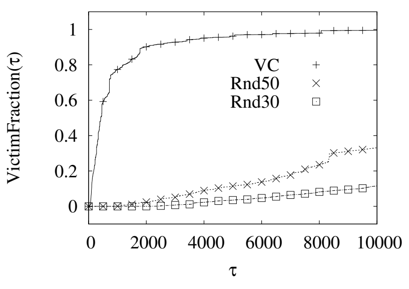
- Overlapping prefixes have a detrimental effect on the performance of IDPFs. However, IDPFs still work reasonably well with overlapping prefixes announced on the Internet. For example, in this case, an attacker in about 50% ASes cannot launch any spoofing-based attacks; and for the majority of attack packets, the AS under attack can pinpoint the true origin to be within 79 ASes.

- Network ingress filtering [8] helps improve the performance of IDPFs. However, even without network ingress filtering, IDPF is still effective. For example, an attacker still cannot launch any spoofing-based attacks from within more than 60% of ASes. Moreover, the AS under attack can localize the true origin of an attack packet to be within 87 ASes.

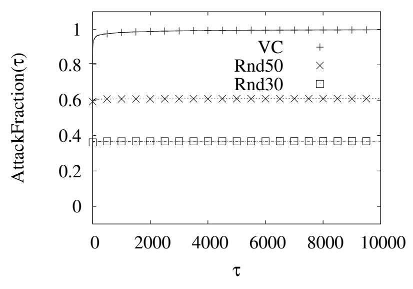
Next, we will present the experimental results. In all experiments except the ones in Section 6.3.5, we assume that ASes that deploy IDPFs, being security conscious and network-savvy, also implement network ingress filtering [8].

6.3.1 IDPFs with BGP Updates and Non-Overlapping Prefixes

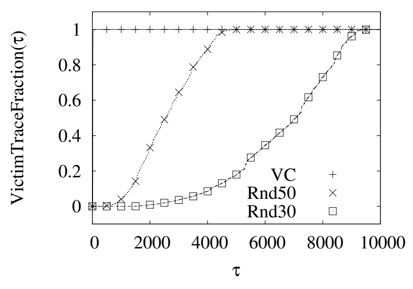
To begin with, we study the performance of IDPFs with BGP updates and non-overlapping prefixes. Figure 5 shows the results on G_{2004c} with different IDPF node coverages while Figure 6 shows the results of the IDPF VC coverage on different data sets.



(a) $VictimFraction(\tau)$



(b) $AttackFraction(\tau)$



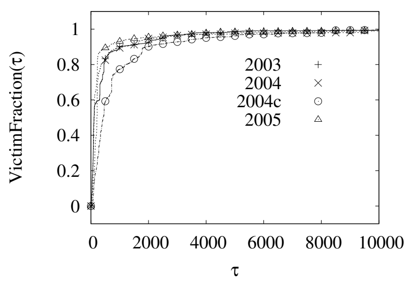
(c) $VictimTraceFraction(\tau)$

Figure 5: Results for G_{2004c} with different IDPF node coverages

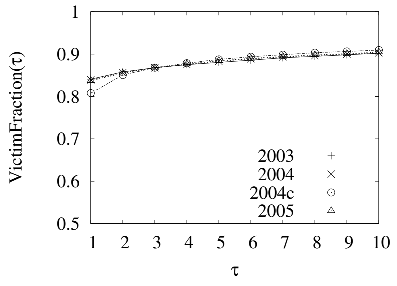
Figure 5(a) presents the values of $VictimFraction(\tau)$ for three different ways of selecting the IDPF node on the G_{2004c} graph: vertex cover (VC) and random covers (*Rnd*50 and *Rnd*30). Note that $VictimFraction(\tau)$ indicates the proportion of nodes that may be attacked by an attacker that can spoof the IP addresses of at most τ nodes. As we discussed earlier, IDPFs cannot *completely protect* ASes from spoofing-based attacks. Hence, we focus on its ability to *limit* the spoofing capability of attackers. This figure shows that IDPF is effective in controlling $VictimFraction(\tau)$, especially with the IDPF VC coverage. The figure shows that the placement of IDPFs plays a key role in the effectiveness of IDPFs in controlling spoofing-based attacks. For example, with only 17.8% of nodes supporting IDPFs, *VC* outperforms both *Rnd*30 and *Rnd*50, although they recruit a larger number of nodes supporting IDPFs. In general, it is more preferable for nodes with large degrees (such as big ISPs) to deploy IDPFs. Figure 6(a) shows $VictimFraction(\tau)$ for the graphs from 2003 to 2005 (including G_{2004c}) with the VC coverage. We see that, overall, similar trends hold for all the years examined. However, it is worth noting that G_{2004c} performs worse than G_{2004}. This is because G_{2004c} contains more edges and more AS paths by incorporating one-year BGP updates.

$AttackFraction(\tau)$ illustrates how effective IDPFs are in limiting the spoofing capability of attackers. In particular, $AttackFraction(1)$ is the proportion of nodes from which an attacker cannot launch any spoofing-based attacks against any other nodes. Figure 5(b) shows that IDPFs are very effective in this regard. For G_{2004c}, $AttackFraction(1)$ = 80.8%, 59.2%, and 36.2%, for VC, *Rnd*50, and *Rnd*30, respectively. Similar trends hold for all the years examined (Figure 6(b)).

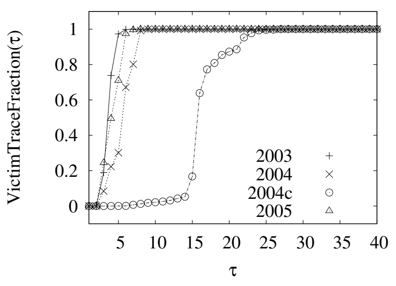
This indicates that IDPFs are very effective in limiting the spoofing capability.



(a) $VictimFraction(\tau)$



(b) $AttackFraction(\tau)$

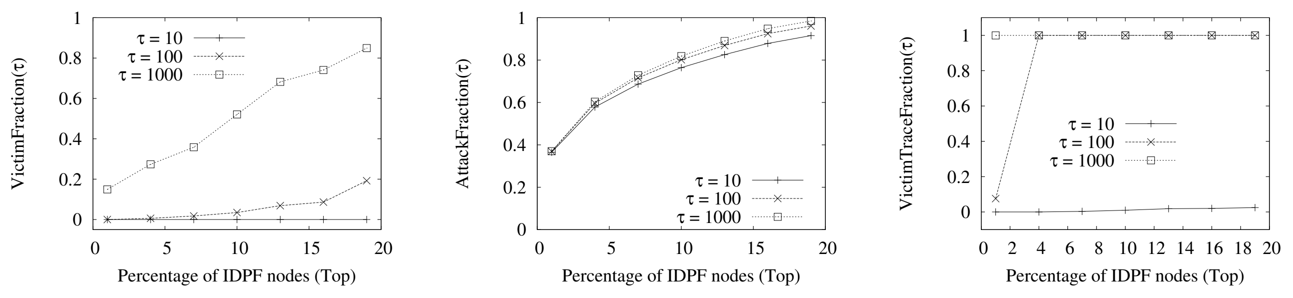


(c) $VictimTraceFraction(\tau)$

Figure 6: Results for G_{2003}, G_{2004}, G_{2004c}, and G_{2005} with the VC coverage

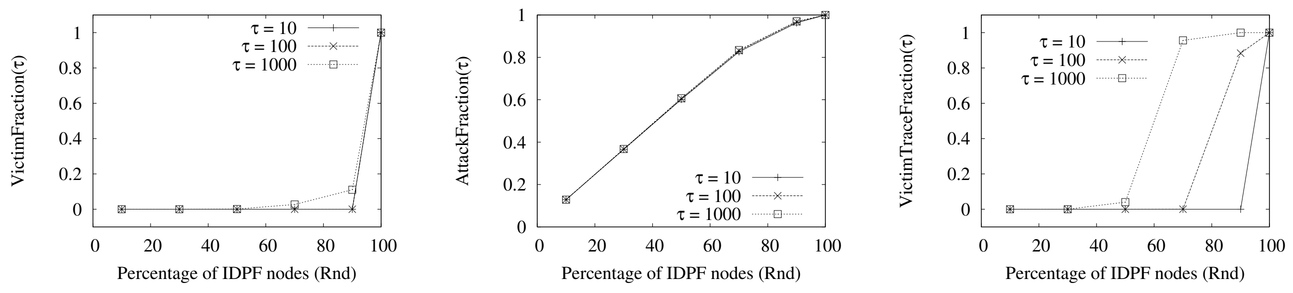
Recall that $VictimTraceFraction(\tau)$ indicates the proportion of nodes that, under attack by packets with a source IP address, can pinpoint the true origin of the packets to be within at most τ nodes. Figure 5(c) shows that all nodes can localize the true origin of an arbitrary attack packet to be within a small number of candidate nodes (28 nodes, see Figure 6(c)) for the VC cover. For the other two, i.e., $Rnd30$ and $Rnd50$, the ability of nodes to pinpoint the true origin is greatly reduced. From Figure 6(c) we also see that G_{2003}, G_{2004}, and G_{2005} can all pinpoint the true origin of attack packets to be within 10 nodes. However, it is important to note that such graphs are less-complete representations of the Internet topology compared to G_{2004c}. Nonetheless, the trend in the results for G_{2003}, G_{2004} and G_{2005} is quite similar to that in the results for G_{2004c}. In the rest of the section, we will mostly show results in G_{2004c} since this data set is more complete than others.

Figure 7 and Figure 8 show the performance as functions of the percentages of IDPF nodes, selected with the Top and Rnd methods, respectively. As expected, in both cases, the effectiveness of IDPF increases as a larger number of nodes deploy IDPF. However, these two figures show that the Top method is significantly more effective than the Rnd scheme, which argues strongly for the deployment of IDPFs in large ISPs with more connectivity. As shown in the figure, even with deployed only on 1% of the most connected nodes, IDPFs can significantly limit the spoofing capability of the attackers and increase the traceback accuracy. Moreover, the performance of IDPFs with 5% of all nodes selected by the Top method is never worse than that with 30% of all nodes selected by the *Rnd* method in terms of all of the three performance metrics. When the IDPF nodes are randomly selected, they can still significantly limit the spoofing capability (Figure 8 (b)).



(a) $VictimFraction(\tau)$ (b) $AttackFraction(\tau)$ (c) $VictimTraceFraction(\tau)$

Figure 7: The *Top* method with different percentages of IDPF nodes



(a) $VictimFraction(\tau)$ (b) $AttackFraction(\tau)$ (c) $VictimTraceFraction(\tau)$

Figure 8: The *Rnd* method with different percentages of IDPF nodes

6.3.2 Impacts of Precise Routing Information

In this section, we study the impact of the precise global routing information on the performance of IDPFs. The goal is to determine the performance difference between IDPFs and the ideal route based packet filters [23] with precise global route information. Notice that in a sense, SAVE [18] is a way to realize route based packet filtering on the Internet. Its packet filtering performance should be close to route based packet filtering with precise global routing information. As discussed in Section 6.2.2, we use the shortest path on AS graph for a given pair of source and destination

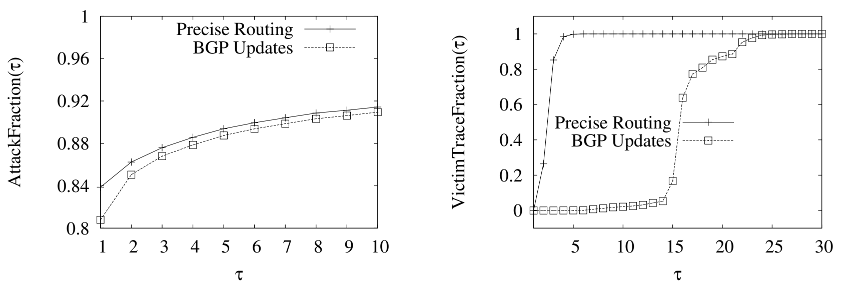


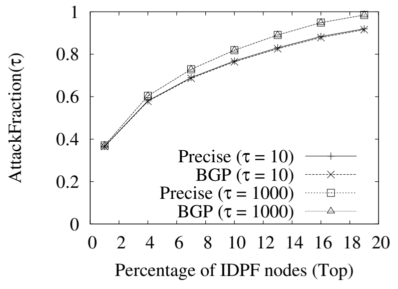
Figure 9: Precise routing information vs. BGP update information (G_{2004c}, VC)

to approximate the precise route between the pair. As shown in Figure 9, the availability of the precise routing information between any pair of source and destination only slightly improves the $AttackFraction(\tau)$ of IDPFs in comparison to the case where BGP update information is used. For example, while about 84% of nodes cannot be used by attackers to launch any spoofing-based attacks by relying on the precise routing information, there are still about 80% of ASes where an attacker cannot launch any such attacks by solely relying on BGP update information. However, the traceback ability is affected more significantly. By only relying on BGP update information, an arbitrary AS can still pinpoint the true origin of an attack packet be within 28 ASes, compared to 7 if precise global routing information is available.

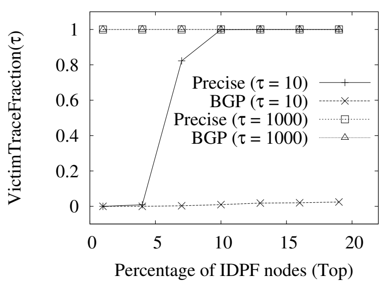
Figure 10 and Figure 11 show the results when the IDPF nodes are selected with the Top and Rnd methods respectively. For both IDPF node selection schemes, the precise routing information (versus BGP updates) has little impact on $AttackFraction$ and has significant impact on $VictimTraceFraction$. These results indicate that using local BGP updates does not significantly affect the IDPFs' ability to limit the spoofing capability of attackers, but may affect the traceback accuracy. This conclusion applies to both Top and Rnd deployment scenarios.

6.3.3 Impacts of Overlapping Prefixes

Figure 12 shows the impacts of overlapping prefixes. From Figure 12(a), we see that overlapping prefixes only have a relatively moderate impact on *limiting* the spoofing capability of attackers. For example, an attacker on about 50% nodes cannot spoof IP addresses of any other nodes.

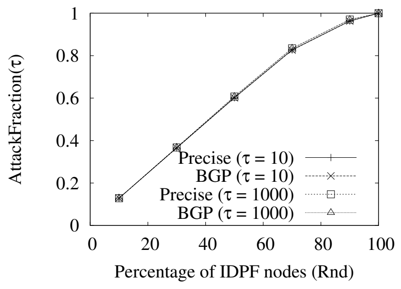


(a) $AttackFraction(\tau)$

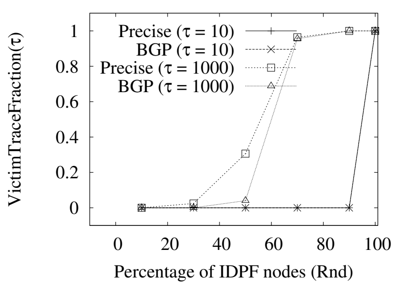


(b) $VictimTraceFraction(\tau)$

Figure 10: The *Top* method with different percentages of IDPF nodes



(a) $AttackFraction(\tau)$



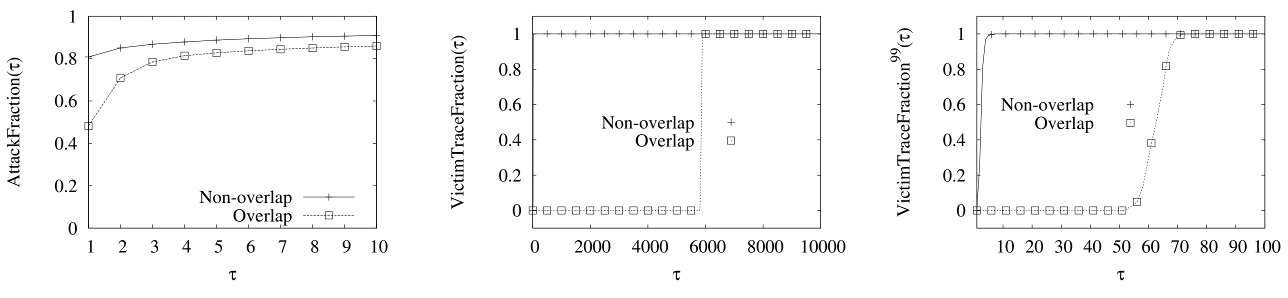
(b) $VictimTraceFraction(\tau)$

Figure 11: The *Rnd* method with different percentages of IDPF nodes

Figure 12(b) demonstrates that overlapping prefixes may significantly affect the ability of nodes in pinpointing the true origin of an attack packet. However, we speculate that this is caused by ISPs that announce less specific prefixes that contain more specific prefixes announced by other ASes. To verify this, we introduce another metric, $VictimTraceFraction^{99}(\tau)$, which is defined with respect to the 99th percentile of $|C_{s,t}|$. Formally,

$$VictimTraceFraction^{99}(\tau) = \frac{|\{t : \forall s \in V, P(|C_{s,t}| \leq \tau) = 99\%\}|}{|V|}$$

$VictimTraceFraction^{99}(\tau)$ can be interpreted as follows: For an attack packet with an arbitrary IP source address, with 99% probability, we can pinpoint the true origin of the packet to be within τ ASes. Figure 12(c) presents the values of $VictimTraceFraction^{99}(\tau)$. From the figure we see

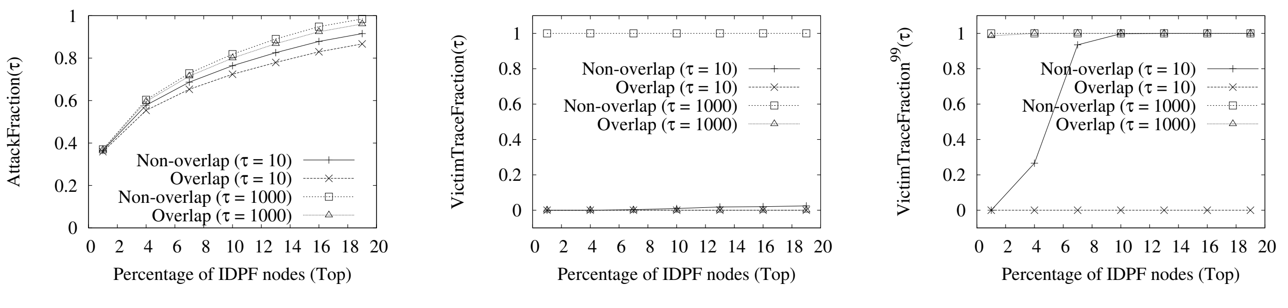


(a) $AttackFraction(\tau)$ (b) $VictimTraceFraction(\tau)$ (c) $VictimTraceFraction^{99}(\tau)$

Figure 12: Impacts of overlapping prefixes (G_{2004c},VC. Note that scales are different).

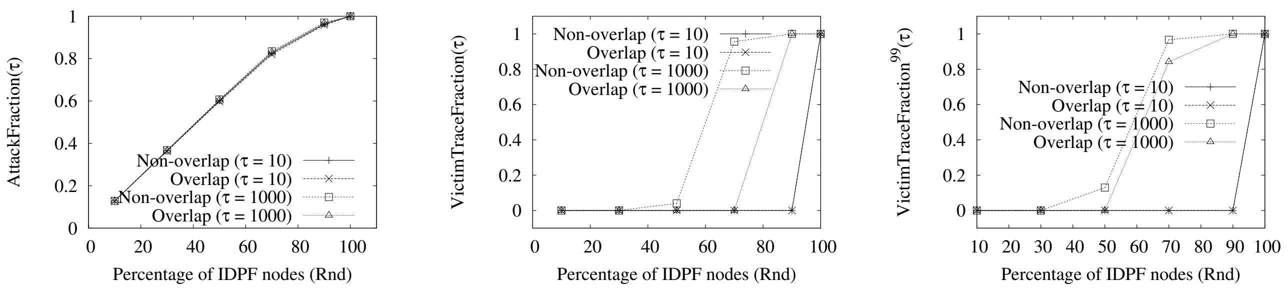
that for more than 99% of IP addresses of attack packets, a node can pinpoint the true origin to be within 79 nodes.

Figure 13 and Figure 14 show the results when the IDPF nodes are selected with the Top and Rnd methods respectively. For the Top method, overlapping prefixes slightly affect $AttackFraction(\tau)$, but may significantly change $VictimTraceFraction(\tau)$. For example, $VictimTraceFraction(1000)$ changes from 100% with non-overlapping prefixes to 0% with overlapping prefixes for all the percentages plotted in the figure. For the Rnd method, as shown in Figure 14, the impact on $AttackFraction$ is negligible while the impact on $VictimTraceFraction$ is significant. These results are in line with the results for the VC coverage, which indicates that the conclusion applies to both IDPF node selection schemes.



(a) $AttackFraction(\tau)$ (b) $VictimTraceFraction(\tau)$ (c) $VictimTraceFraction^{99}(\tau)$

Figure 13: The Top method with different percentages of IDPF nodes



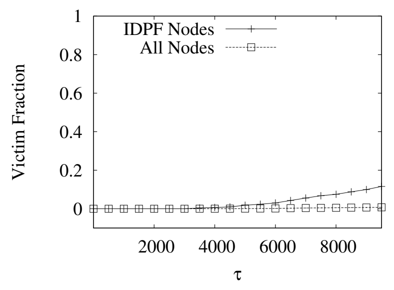
(a) $AttackFraction(\tau)$ (b) $VictimTraceFraction(\tau)$ (c) $VictimTraceFraction^{99}(\tau)$

Figure 14: The *Rnd* method with different percentages of IDPF nodes

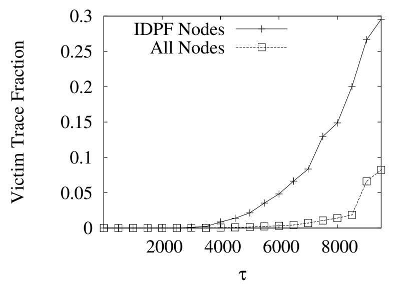
6.3.4 Deployment Incentives

This section studies the incentives for an AS to deploy IDPFs. The deployment incentive is the key factor that is responsible for the slow deployment of network ingress filtering. Figures 15 and 16 show the incentive for an AS to deploy IDPF: the ASes that deploy IDPFs are better protected than those that do not deploy IDPFs. Figure 15 shows the results when only 5 percent of all nodes (randomly selected) deploy IDPFs while Figure 16 shows the results when 30 percent of all nodes are IDPF nodes. We show the values of $VictimFraction\_IDPF(\tau)$ (curve marked with *IDPF Nodes*) and $VictimFraction(\tau)$ (marked with *All Nodes*). From the figures we see that in the *Rnd*30 (Figure 16) case while only about 5% of all nodes on the Internet cannot be attacked by attackers that can spoof IP addresses of more than 6000 nodes, that percentage increases to higher than 11% among the nodes that support IDPFs. Moreover, as the value of τ increases, the difference between the two enlarges. Similarly, while only about 18% of all nodes on the Internet can pinpoint the true origin of an attack packet to be within 5000 nodes, more than 33% of nodes supporting IDPFs can do so (Figure 16(b)). Comparing Figure 15 and Figure 16, we can see that the relative benefit for deploying IDPF is larger when a less number of nodes deploy IDPFs: there is more incentive to deploy IDPFs when a less number of ASes in the Internet are IDPF nodes.

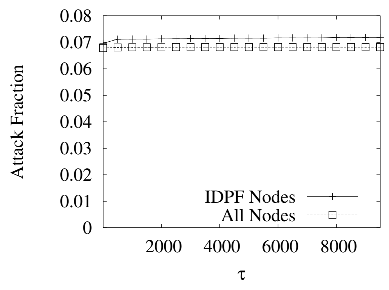
Figures 15(c) and 16(c) compare the spoofing capability of attackers in attacking a general node on the Internet and that supporting IDPFs. We see that networks supporting IDPFs only gain slightly in this perspective. This can be understood by noting that, by deploying IDPFs, an AS not only protects itself, but also those to whom the AS transports traffic.



(a) $VictimFraction\_IDPF(\tau)$ vs. $VictimFraction(\tau)$

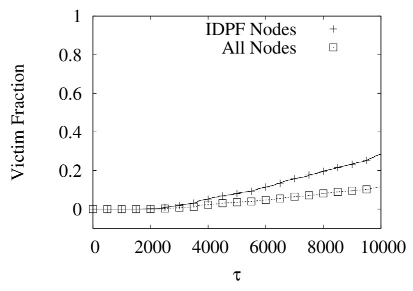


(b) $VictimTraceFraction\_IDPF(\tau)$ vs. $VictimTraceFraction(\tau)$

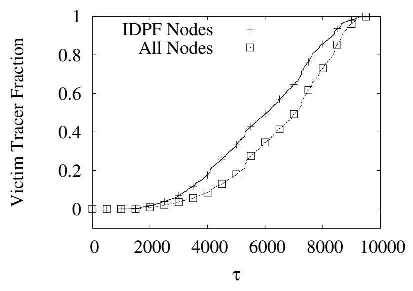


(c) $AttackFraction\_IDPF(\tau)$ vs. $AttackFraction(\tau)$

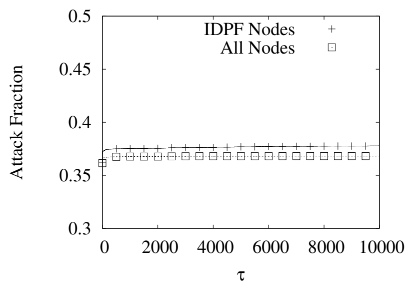
Figure 15: Deployment incentives (G_{2004c}, Rnd5).



(a) $VictimFraction\_IDPF(\tau)$ vs. $VictimFraction(\tau)$



(b) $VictimTraceFraction\_IDPF(\tau)$ vs. $VictimTraceFraction(\tau)$



(c) $AttackFraction\_IDPF(\tau)$ vs. $AttackFraction(\tau)$

Figure 16: Deployment incentives (G_{2004c}, Rnd30).

6.3.5 IDPF with and without Network Ingress Filtering

So far we have assumed that networks supporting IDPFs also employ network ingress packet filtering [8]. In this section we examine the implications of this assumption.

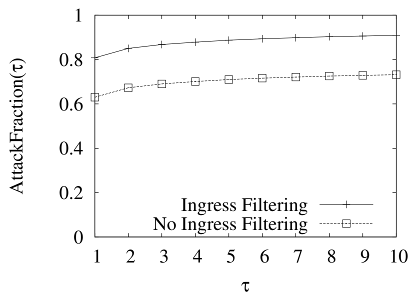


Figure 17: IDPF with and without ingress filtering (G_{2004c}, VC).

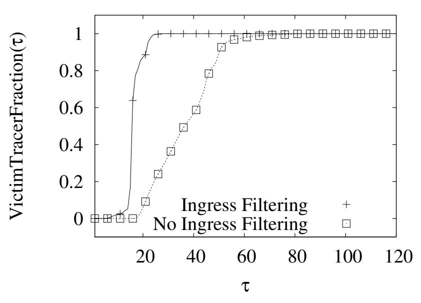


Figure 18: IDPF with and without ingress filtering (G_{2004c}, VC).

From Figure 17, we can see that ingress packet filtering indeed has an impact on the effectiveness of IDPFs in limiting the spoofing capability of attackers. However, without network ingress filtering, we still have more than 60% of nodes from which an attacker cannot launch any spoofing-based attacks, compared to 80% when ingress filtering is enabled at nodes supporting IDPFs. As shown in Figure 18, the impact of network ingress filtering on the effectiveness of IDPFs in terms of reactive IP traceback is not very large. Without ingress filtering, an arbitrary node can pinpoint the true origin of an attack packet to be within 87 nodes, compared to 28 when networks supporting IDPFs also employ ingress filtering. We have also perform simulations with different IDPF node selection schemes, the trend in the results are similar to those displayed in Figure 17 and Figure 18.

7 Conclusion

In this paper we proposed and studied an inter-domain packet filter (IDPF) architecture as an effective countermeasure to the IP spoofing-based DDoS attacks. IDPFs rely on BGP update messages exchanged on the Internet to infer the validity of source address of a packet forwarded by a neighbor. We showed that IDPFs can be easily deployed on the current BGP-based Internet routing architecture. We studied the conditions under which the IDPF framework can work correctly without discarding any valid packets. Our simulation results showed that, even with partial deployment on the Internet, IDPFs can significantly limit the spoofing capability of attackers. Moreover, they also help pinpoint the true origin of an attack packet to be within a small number of candidate networks, therefore, simplifying the reactive IP traceback process.

Acknowledgment

We thank Kihong Park, Heejo Lee, and Ali Selcuk for providing us with the *dpf* simulation tool, and the Oregon Route Views Project for making BGP routing tables and updates publicly available. The authors are also grateful to Nick Feamster and Jennifer Rexford for the insightful comments on an early version of the paper. Zhenhai Duan was supported in part by NSF Grant CCF-0541096. Xin Yuan was supported in part by NSF Grants ANI-0106706, CCR-0208892, CCF-0342540, and CCF-0541096. Jaideep Chandrashekar was supported in part by NSF Grants ITR-0085824 and CNS-0435444, and a Cisco URP grant. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of National Science Foundation or Cisco Systems.

References

[1] F. Baker. Requirements for ip version 4 routers. RFC 1812, June 1995.

[2] R. Beverly and S. Bauer. The Spoofer Project: Inferring the extent of Internet source address filtering on the internet. In *Proceedings of Usenix SRUTI*, Cambridge, MA, July 2005.

[3] A. Bremler-Barr and H. Levy. Spoofing prevention method. In *Proc. IEEE INFOCOM*, Miami, FL, March 2005.

[4] CERT. Cert advisory ca-1996-21 TCP SYN flooding and IP spoofing attacks, 1996. `ttp://www.cert.org/advisories/CA-1996-21.tml`.

[5] J. Chandrashekar, Z. Duan, Z.-L. Zhang, and J. Krasky. Limiting path exploration in BGP. In *Proc. IEEE INFOCOM*, Miami, FL, March 2005.

[6] Cisco Systems, Inc. Unicast reverse path forwarding loose mode. `http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newf%t/122t/122t13/ft_urpf.pdf`.

[7] X. Dimitropoulos, D. Krioukov, and G. Riley. Revisiting internet as-level topology discovery. In *Passive and Active Measurement Workshop (PAM)*, Boston, MA, March 2005.

[8] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. RFC 2267, January 1998.

[9] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless inter-domain routing (CIDR): an address assignment and aggregation strategy. RFC 1519, September 1993.

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

[11] L. Gao, T. Griffin, and J. Rexford. Inherently safe backup routing with bgp. In *Proc. IEEE INFOCOM*, 2001.

[12] L. Gao and J. Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6), December 2001.

[13] G. Huston. Interconnection, peering and settlements-part I. *The Internet Protocol Journal*, March 1999.

[14] ICANN/SSAC. ICANN SSAC Advisory SAC008 DNS Distributed Denial of Service (DDoS) Attacks, March 2006.

[15] C. Jin, H. Wang, and K. Shin. Hop-count filtering: an effective defense against spoofed ddos traffic. In *Proceedings of the 10th ACM conference on Computer and communications security*, October 2003.

[16] S. Kandula, D. Katabi, M. Jacob, and A. Berger. Botz-4-Sale: Surviving Organized DDoS Attacks that Mimic Flash Crowds. In *NSDI*, 2005.

[17] Craig Labovitz, Danny McPherson, and Farnam Jahanian. Infrastructure attack detection and mitigation. SIGCOMM 2005, August 2005. Tutorial.

[18] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang. Save: source address validity enforcement protocol. In *INFOCOM*, June 2002.

[19] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security and Privacy*, 2003.

[20] D. Moore, C. Shannon, D. Brown, G. Voelker, and S. Savage. Inferring internet Denial-of-Service activity. *ACM Transactions on Computer Systems*, 24(2), May 2006.

[21] University of Oregon. Route Views project. http://www.routeviews.org/.

[22] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of internet background radiation. In *Proceedings of ACM Internet Measurement Conference*, October 2004.

[23] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proc. ACM SIGCOMM*, San Diego, CA, August 2001.

[24] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *ACM Computer Communications Review (CCR)*, 31(3), July 2001.

[25] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). RFC 1771, March 1995.

[26] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. *ACM SIGCOMM Computer Communication Review*, 30(4), October 2000.

[27] J. Stewart. *BGP4: Inter-Domain Routing In the Internet.* Addison-Wesley, 1999.

[28] J. Stewart. DNS cache poisoning - the next generation. Technical report, LURHQ, January 2003.

[29] Team Cymru. The team cymru bogon route server project. http://www.cymru.com/BGP/bogon-rs.html.

[30] P. Watson. Slipping in the window: TCP reset attacks. In *Cansecwest/core04 Conference*, 2004.

[31] W. Xu and J. Rexford. Miro: multi-path interdomain routing. *SIGCOMM Comput. Commun. Rev.*, 36(4), October 2006.

[32] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against DDoS attacks. In *IEEE Symposium on Security and Privacy*, May 2003.

[33] A. Yaar, A. Perrig, and D. Song. StackPi: New packet marking and filtering mechanisms for DDoS and IP spoofing defense. *IEEE Journal on Selected Areas in Communications*, 24(10), October 2006.