On the State of IP Spoofing Defense

TOBY EHRENKRANZ and JUN LI
University of Oregon

IP source address spoofing has plagued the Internet for many years. Attackers spoof source addresses to mount attacks and redirect blame. Researchers have proposed many mechanisms to defend against spoofing, with varying levels of success. With the defense mechanisms available today, where do we stand? How do the various defense mechanisms compare? This article first looks into the current state of IP spoofing, then thoroughly surveys the current state of IP spoofing defense. It evaluates data from the Spoofer Project, and describes and analyzes host-based defense methods, router-based defense methods, and their combinations. It further analyzes what obstacles stand in the way of deploying those modern solutions and what areas require further research.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms: Performance, Security

Additional Key Words and Phrases: IP spoofing, spoofing defense, spoofing packet, packet filtering

ACM Reference Format:

Ehrenkranz, T. and Li, J. 2009. On the state of IP spoofing defense. ACM Trans. Internet Technol. 9, 2, Article 6 (May 2009), 29 pages. DOI = 10.1145/1516539.1516541
http://doi.acm.org/10.1145/1516539.1516541

1. INTRODUCTION

In today's Internet, attackers can forge the source address of IP packets to both maintain their anonymity and redirect the blame for attacks. When attackers inject packets with spoofed source addresses into the Internet, routers forward those packets to their destination just like any other packet—often without checking the validity of the packets' source addresses. These spoofing packets[1] consume network bandwidth en route to their destinations, and are often part of some malicious activity, such as a DDoS attack. Unfortunately, routers on

[1]In this article we use *spoofing packets* instead of *spoofed packets* as such a packet is from an attacker, and it does harm to the network instead of being a victim.

Authors' address: T. Ehrenkranz and J. Li, University of Oregon, Computer and Information Science, Eugene, OR 97403-1202; email: {tehrenkr,lijun}@cs.uoregon.edu.

DOI 10.1145/1516539.1516541 http://doi.acm.org/10.1145/1516539.1516541

the Internet today cannot effectively filter out spoofing packets. They either do not know what distinguishes legitimate packets from spoofing packets, or do not leverage such knowledge.

For many years the research community has been working hard to combat IP spoofing, starting with early works such as ingress/egress filtering, and continuing through the present with modern solutions such as Spoofing Prevention Method [Bremler-Barr and Levy 2005] and Distributed Packet Filtering [Park and Lee 2001]. Many proposed solutions have shown great promise, but, has the problem been solved?

In order to understand the spoofing abilities of attackers today, we first examine results from the Spoofer Project [MIT Advanced Network Architecture Group 2007; Beverly and Bauer 2005]. This project attempts to measure the ability of hosts throughout the Internet to send spoofing packets. We will show that IP source address spoofing remains a severe problem on the Internet. Although many may feel the network community solved the spoofing problem through the widespread adoption of ingress and egress filtering, attackers can still spoof a significant portion of existent IP addresses, often *any* IP address.

We then inspect and compare various IP spoofing defense solutions. Our goal is to provide a comprehensive study of the state-of-the-art, and meanwhile analyze what obstacles stand in the way of deploying those modern solutions and what areas require further research. We will compare spoofing defense mechanisms in terms of three features: *identifying spoofing packets*, *mitigating spoofing attacks*, and *pinpointing an attacker's real location*. Note that identifying spoofing packets and mitigating a spoofing attack are not equal. For example, with a bandwidth-based denial-of-service attack, even if we are able to identify spoofing packets, we cannot mitigate an attack they cause if the identification is done at or close to the victim. Furthermore, identifying and mitigating an attack does not mean we can identify the actual attacker. Without being able to locate an attacker, there is no deterrent for attackers; their attacks may be stopped, but as long as they can continue to attack in anonymity there is no risk to themselves or their resources. Not all spoofing defense mechanisms implement all three features, and those that do may have implementations of varying effectiveness.

Spoofing defense mechanisms should also maintain a set of desired properties. They cannot rely on traffic characteristics that attackers can easily manipulate and spoof the correct values. They should also be easy to deploy, and preferably independent of routing protocols in order to ensure deployability across *all* current and future intra-AS and inter-AS networks. And finally, of course, an ideal defense mechanism must incur low overhead so as not to affect network performance.

Note that this article looks into IP spoofing and not IP prefix hijacking. Although they both involve attackers pretending to have a false identity, the problems are inherently unique. When an attacker successfully hijacks a prefix, hijacked IP addresses will be effectively co-owned by both the attacker and the legitimate owner; although some packets toward the hijacked IP addresses may still reach the legitimate owner, many packets will reach the attacker. When an attacker uses IP spoofing, however, the spoofed source addresses are entirely out of the attacker's control. Many of the IP spoofing defense mechanisms assume that attackers cannot receive responses, and would not be effective in defending against attackers that employ IP prefix hijacking. Prefix hijacking is an important network security problem, but it requires solutions different from those for IP spoofing.

The remainder of this article is organized as follows. In Section 2 we discuss why IP spoofing is still a serious problem today. Then in Section 3 we categorize the spoofing defense mechanisms from a high level. Sections 4, 5, and 6 describe the defense mechanisms in detail, focusing on host-based ones, router-based ones, and their combination, respectively. In Section 7 we analyze the pros and cons of all these defense mechanisms, addressing their capabilities and characteristics as well as overhead. We conclude our survey with a discussion on the future of spoofing defense in Section 8.

2. SPOOFING TODAY

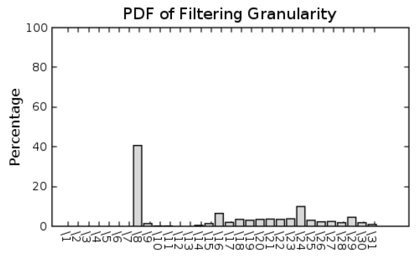
With the prevalence of ingress/egress filtering, it may be concluded that attackers are not able to spoof many addresses. Also, some readers may feel that IP spoofing is no longer a problem. With the increase in botnets, it may seem that attackers no longer need spoofing. To see that these conclusions are not valid, we have only to look at the results from the Spoofer Project [MIT Advanced Network Architecture Group 2007; Beverly and Bauer 2005], as well as how attackers use botnets. In fact, IP spoofing continues to be a prospective tool used by malicious users for attack and misdirection.

2.1 Spoofer Project

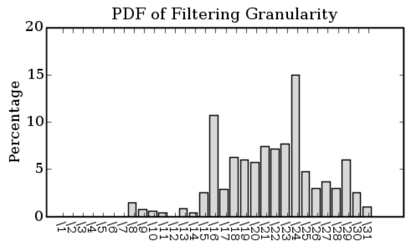
The Spoofer Project measures the ability of hosts throughout the Internet to send spoofing IP packets and the granularity of any ingress/egress filtering that packets from those hosts encounter.

Volunteers throughout the Internet participate in the Spoofer Project by downloading and running a testing program on their end-hosts. This program sends IP packets with forged source addresses towards a Spoofer Project server, where the forgery can use different allocated addresses. When picking allocated addresses to impersonate, the program tries to iterate through neighboring netblocks of the volunteer host—all the way from a neighboring /32 to a /9 netblock. For example, consider a host with IP address 208.77.188.166 that runs the testing program. The program will send a packet with spoofed source address 208.77.188.167 (the "neighbor" /32 block in 208.77.188.166/31). Then it will send packets with source addresses 208.77.188.164, 208.77.188.160, 208.77.188.175, etc., to test the "neighbor" blocks in 208.77.188.166/30, /29, /28, etc. Iterating through neighboring netblocks gives the filtering granularity, or how large of a netblock a volunteer end-host can successfully spoof.

The Spoofer Project keeps track of both how many hosts can successfully spoof at least a single address, and the filtering granularity for such hosts. Assuming the volunteer end-hosts make up a representative sample of all Internet hosts, this project can then estimate the amount of spoofing possible on the Internet.



(a) Results before the change in reporting methodology (August 2006).



(b) Current results from the Spoofer Project website (February 2008).

Fig. 1. Filtering granularity results from the Spoofer Project. Older results had a large spike at /8 granularity because hosts which did not encounter any filtering were counted as hosts with /8 filtering granularity. Figures from [MIT Advanced Network Architecture Group 2007].

The results from the Spoofer Project indicate that hosts able to perform spoofing make up a significant percentage of Internet addresses. As of February 7, 2008, the Spoofer Project estimates hosts at 20.3% of all Internet addresses can perform spoofing, or approximately $464,000,000$ out of $2,290,000,000$ addresses. Note that as 20.3% can perform spoofing, it does not mean that attackers can spoof 20.3% of all Internet addresses. In fact, from some locations, attackers can spoof 100% of all source IP addresses.

Looking at the filtering granularity helps showcase how many source IP addresses attackers can spoof. A finer granularity means the range of addresses an attacker can spoof is smaller. A coarser granularity means the range of addresses an attacker can spoof is larger. If we compare the original results [Beverly and Bauer 2005] to the current online results [MIT Advanced Network Architecture Group 2007], we see that around 40% of the filtering was originally occurring at /8 granularity (Figure 1(a)), but currently less than 2% of the filtering occurs at /8 granularity (Figure 1(b)). This would indicate a drastic increase in filtering efficacy: Hosts that could spoof any address within a /8 netblock are now limited to spoofing a much smaller netblock. Unfortunately, however, this difference in granularity is actually because of a change in the reporting methodology.[2] The original results counted hosts that could spoof *any* address as encountering an /8 filtering granularity. The current filtering granularity results simply ignore such hosts. Most of the hosts previously reported as having an /8 filtering granularity could in fact spoof any valid Internet address, and now do not show up in the results at all.

If we consider the sampling used in the Spoofer project, the situation may be even worse. Only $7,980$ unique hosts participated in the measurement. Figure 2 shows the locations of those hosts which contributed to the Spoofer Project. It is not clear that this sampling is representative of the Internet as a whole. The United States and Europe appear well represented, but since few hosts from other areas of the world contributed measurements, the results may be strongly biased towards U.S. and European networks. If the networks in less-developed

[2]Discussion relating to this change can be found in the Spoofer Project's mailing list: `https://lists.csail.mit.edu/pipermail/spoofer/2006-August/000009.html`

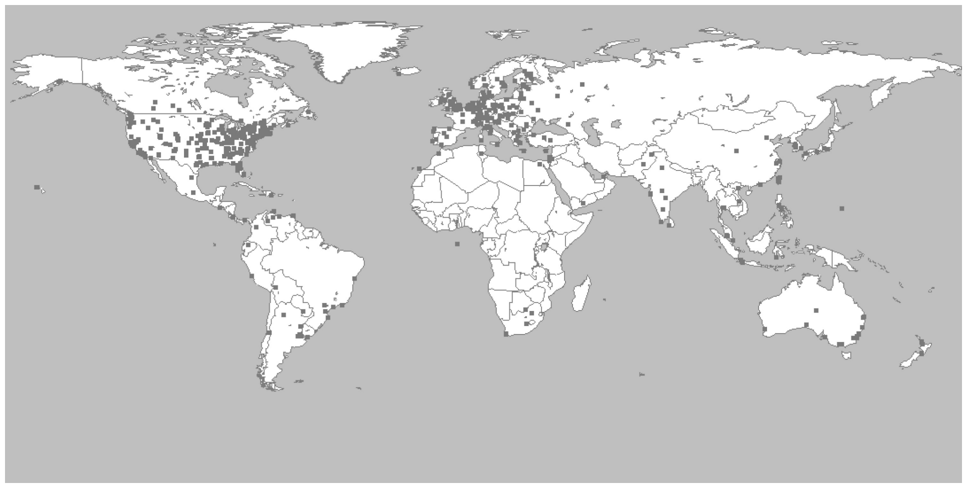


Fig. 2. Locations of hosts that participated in the Spoofer Project. Figure from MIT Advanced Network Architecture Group [2007].

areas are less secure, the actual amount of spoofable addresses would be much greater.

2.2 Botnets

With the growing popularity of botnets, some believe attackers no longer need to use IP spoofing. When attackers can control hundreds of thousands of zombie nodes, why should we care about spoofing?

In fact, when considering botnets, IP spoofing remains a problem for defenders and an asset for attackers. In some botnet-based attacks, such as a DNS DDoS amplification attack [Piscitello 2006], IP spoofing is vital to the attack's success. In other attacks, IP spoofing may be used but not necessary. Even when not necessary, IP spoofing gives botnet owners another layer of anonymity, and protects their botnet. Botnet owners would prefer to keep the identity of their zombies anonymous as long as possible, in order to prevent defenders from identifying or perhaps even disconnecting zombies. Larger botnets can more effectively perform their tasks, such as stealing identities or producing spam [Messmer 2007; Martin 2006], equaling more money for the botnet owner.

Even if botnets did not use IP spoofing, the threat of IP spoofing would still exist. As network defenders and network security researchers, we should not only research how to defend against known attacks. We should learn how to mitigate any possible threat and deal with any underlying flaw of the network infrastructure.

3. OVERVIEW OF IP SPOOFING DEFENSES

As Section 2 shows, IP spoofing remains a severe problem in the Internet today. Unfortunately, although many defense mechanisms have been proposed, none have eradicated the spoofing problem. In this section, we present an overview of these mechanisms.

Spoofing defense solutions can essentially be broken down into three categories:

(1) *End-Host-Based Solutions*. These solutions are implemented on end-hosts, and aim to allow an end-host to recognize spoofing packets. Although some such solutions can be deployed at routers as well, end-host-based solutions are designed with end-hosts in mind, and do not rely upon any special router functionality. In general, these solutions do not need to change networking infrastructure and are the easiest to deploy. On the other hand, they may act too late since the spoofing packets must reach end-hosts before they are detected. (As one seldom deploys an end-host-based solution at source end-hosts, which cannot even come close to preventing IP spoofing from occurring unless every source host is uncompromised and honest. In this article we will only survey those deployed at destination end-hosts.)

(2) *Router-Based Solutions*. These solutions are meant to be implemented by routers at the core of the Internet, the edge of the Internet, or both. These solutions generally face more hurdles to deployment, but can be the most effective since they can stop spoofing packets from even reaching end-hosts. Seldom can router-based mechanisms be ported to end-hosts. (Routers may deploy some reactive mechanisms such as tracing where a malicious packet is from. However, reactive mechanisms are triggered only after spoofing packets are detected, so in this article we focus on router-based mechanisms with the main objective of preventing the delivery of spoofing packets.)

(3) *Solutions Requiring the Use of Both Routers and End-Hosts*. Routers and end-hosts must work together in order for these solutions to work.

An obvious difference between host-based and router-based mechanisms relates to the end-to-end argument [Saltzer et al. 1984]. Host-based mechanisms clearly adhere to end-to-end principles while router-based mechanisms do not. This allows for host-based mechanisms to be deployed much more easily. Host-based mechanisms can generally be deployed even on a single host, without requiring the cooperation of any other host or router. Host-based mechanisms, adhering to end-to-end principles, also avoid increasing the complexity of routers. This can be seen as an argument in favor of host-based mechanisms, but router-based mechanisms should not be discounted so easily.

In contrast to host-based solutions, router-based defense mechanisms are able to address the fundamental weakness which allows IP spoofing. Packets with a forged source address can successfully reach their destination because routers only use the destination address of packets to deliver them and do not verify the source address of packets. Preventive router-based solutions can either provide a way for routers to verify the source address of packets based on their incoming direction, or use some marking to identify the true source of a packet. This allows routers to detect and drop spoofing packets closer to an attacker, before the packets even reach end-hosts.

Table I. Spoofing Defense Mechanisms

Host-based Solutions	Router-based Solutions	Combination
Active — *Cryptographic:* IPsec *Probing:* OS fingerprinting, IP ID field probing, TCP probing *Other:* SYN cookies, IP puzzles	**Basic —** Martian address filtering, ingress/egress filtering, reverse path forwarding	Pi, StackPi
Passive — Hop-count filtering	**Distributed —** SPM, Passport, DPF, SAVE, IDPF, BASE	

Next, in Sections 4, 5, and 6 we will briefly describe the various spoofing defense mechanisms, and then in Section 7 we will analyze their capabilities, characteristics, as well as performance. Table I lists the specific spoofing defense mechanisms we describe and analyze.

4. HOST-BASED DEFENSE METHODS

We can categorize host-based defense mechanisms into active and passive types. Active mechanisms require the end-host to perform some sort of active probing, or other pro-active actions. Passive mechanisms on the other hand rely solely on information which a host can gather locally without probing the supposed source of a packet.

4.1 Active

Active host-based mechanisms include cryptographic solutions, active probing solutions, and IP puzzles.

Cryptographic solutions, such as IPsec [Kent and Seo 2005], require a handshaking operation to set up secret keys between two hosts. Further communication between the two hosts can be encrypted or signed to ensure any message received by one host was sent by the other host. An attacker would not be able to successfully spoof packets to create a connection, because the attacker would not receive the replies necessary to complete the handshaking process. Similarly, an attacker attempting to spoof packets of an existing connection would fail, because the attacker could not know the secret key. Although IPsec is useful in many situations to prevent spoofing, and provide confidentiality and integrity guarantees, it is unrealistic to use in *all* situations. First of all, it is not feasible to require all end-hosts connect through IPsec. Furthermore, the computational cost of encrypting or signing every packet would mean hosts cannot maintain as many active connections. The computational cost also precludes routers from forming IPsec tunnels for all connections.

Active probing solutions can involve a number of types of probes, including active operating system fingerprinting, IP identification field probes, and TCP-specific probes. Note that these probing mechanisms are not designed to be used for spoofing defense, but they can be used for spoofing defense.

Probing a host to determine its operating system, using tools such as NMAP [Fyodor 2006], can prove valuable in detecting spoofing packets. These tools send carefully crafted packets to an end-host and observe its response. Although most operating systems generally follow standard protocol specifications, their specific implementations may measurably differ; these implementation differences act as an operating system fingerprint. If a host can actively fingerprint the supposed source host and find that it is running operating system X, while passive fingerprinting [Zalewski 2006; Beverly 2004; Taleck 2003] on the original received packet suggests it is running Y, it is likely the original packet was spoofing. Unfortunately, fingerprinting involves a high amount of overhead. A host must send numerous probes and buffer suspicious packets if the host requires sources be verified before processing the suspicious packets. Furthermore, fingerprinting cannot identify spoofing packets if the attacker uses the same operating system as the host located at the spoofed address.

Probing the identification field of IP packets can also identify spoofing packets. After receiving a suspicious packet, a host can send packets to the supposed source to observe the identification field in its response. Different IP stacks may set the identification field of IP packets differently. For instance, some hosts may choose a random identification number, and others may simply increment the identification number for every packet. Assuming the source host simply increments the identification field for every packet, the identification number in the probe response should be near the identification number of the suspicious packet; otherwise, the suspicious packet was a spoofing packet. Unfortunately, if the source host sets the identification field in a more complicated manner, it may be difficult or impossible to decide whether or not a suspicious packet carries a spoofed source address.

Using TCP-specific probes is another clever way of defending against spoofing packets. Simply requiring a TCP handshake may not be enough to prevent attackers from spoofing TCP packets, since attackers may be able to predict TCP sequence numbers. Although many operating systems use a random sequence number selection, the pseudo-random number generators they use may not be random enough [Zalewski 2002; Zalewski 2001]. TCP-specific probes intelligently craft TCP acknowledgment messages to add another layer of protection. Since the sender of spoofing packets is often unable to see any replies, a recipient host can send acknowledgments that should change the TCP window size or cause packet retransmission, and then observe whether or not the supposed source responds correctly. If the supposed source does not change the window size or does not retransmit the packet, the recipient host considers the packet's source to be spoofed.

Some servers use SYN cookies [Bernstein 1996] to prevent opening connections to spoofed source addresses. The main reason to use SYN cookies, however, is to mitigate the effects of SYN floods by making the TCP handshake stateless [Aura and Nikander 1997]. When a server uses SYN cookies it does not allocate resources to a connection until the 3-way TCP handshake completes. First the server sends a SYN + ACK packet with a specially encoded initial sequence number, or cookie, that includes a hash of the TCP headers from the client's initial SYN packet, a timestamp, and the client's Maximum Segment Size (MSS). Then when it receives the client's response, the server can check the sequence number and create the necessary state only if the client's sequence number is the cookie value plus one. Because the cookie uses a hash involving the server's secret key, attackers should not be able to guess the correct cookie values. However, because of performance concerns and some incompatibilities with TCP extensions, such as large windows, operating systems generally do not activate the SYN cookie mechanism until the host's SYN queue fills up. An attacker sending spoofing traffic at a low rate may avoid triggering the SYN cookie mechanism. Administrators may be able to forcibly enable SYN cookies for all connections, but should be aware of the side effects.

IP puzzles [chang Feng et al. 2005] are another mechanism hosts can employ to actively defend against spoofing. A server sends an IP puzzle to a client, then the client needs to "solve" the puzzle by performing some computational task. Only after the server receives the puzzle solution from the client will the server allow the client to connect. While the main goal of IP puzzles is to make it prohibitively expensive for malicious hosts to send large numbers of packets, a side effect is preventing attackers from successfully sending spoofing packets. Since the IP puzzle would be sent to the listed source and not the attacker, an attacker could not send a puzzle solution, thus preventing the attacker from spoofing.

4.2 Passive

Passive host-based spoofing defense mechanisms decide whether or not a packet is spoofing by passively observing incoming traffic.

Hop-Count Filtering [Jin et al. 2003; Wang et al. 2007], or HCF, observes the hop-count of packets arriving at a given host. First, by measuring the hop-counts during normal times, HCF creates a mapping of IP addresses to hop-counts. Then, if an attacker sends a spoofing packet to the host, it is likely the hop-count of the packet will not match the expected hop-count for packets from the spoofed source address. Because legitimate hop-counts may change due to routing changes, strictly filtering all packets that do not match would lead to false positives. In order to minimize false positives, HCF only begins filtering traffic if some threshold amount of packets do not match their expected hop-counts. This threshold protects against mistakenly filtering legitimate packets, but it also makes HCF ineffective against low amounts of spoofing packets that do not reach the threshold. Furthermore, because the range of expected hop-counts on the Internet is narrow, around 10% of the spoofing packets can be expected to have the correct hop-count [Jin et al. 2003].

An earlier paper [Templeton and Levitt 2003] also covered many of these active and passive host-based methods, and they have remained largely unchanged. For more detailed descriptions we refer readers to Templeton and Levitt [2003].

5. ROUTER-BASED DEFENSE METHODS

Router-based spoofing defense methods generally take a different approach from host-based mechanisms. Although in principle most host-based methods could also be used by routers, researchers generally only consider a few, such as IPsec or IP puzzles, for use at the router level. Other host-based methods generally require too much overhead that would impact router performance.

Router-based defense mechanisms are all similar in that they perform some sort of filtering to prevent spoofing packets from reaching their intended destinations. The various mechanisms' differences lie in what information they use to decide whether a packet contains a spoofed source address, and where in the network the filtering takes place. We will discuss more basic, traditional mechanisms along with state-of-the-art distributed filtering solutions.

5.1 Basic Filtering

One of the earliest and simplest methods of filtering packets with spoofed source addresses is "Martian" filtering [Baker 1995]. Martian filtering simply involves examining IP header fields and looking for invalid IP addresses, for instance, nonunicast source addresses, loopback addresses, or some other "special" addresses. By design, this mechanism can only stop the most obvious and simple types of IP spoofing.

Ingress [Ferguson and Senie 2000] and egress [Killalea 2000] filtering are the most well-known filtering methods. Run on a router at the border of a network, ingress/egress filtering checks the addresses of packets flowing into and out of an edge network. For a packet originating from the edge network, if the source address does *not* belong to the edge network, the packet is a spoofing packet. Similarly, for a packet originating outside the edge network, if the source address *does* belong to the edge network, the packet is a spoofing packet. The border router is easily able to filter out any packet it identifies as a spoofing packet. If this simple filtering mechanism were implemented on all networks, attackers would be limited to only spoofing addresses within their own local network. Unfortunately, as the Spoofer Project shows us, such universal deployment is not a reality. Furthermore, there is not much incentive for a network to deploy ingress/egress filtering; when a network deploys ingress/egress filtering, it has only a minimal effect on how many other networks are able to spoof its source addresses. But incentive is not the only problem; without nearly 100% deployment, ingress/egress filtering is ineffective [Park and Lee 2001]. Even if only a small number of networks do not implement ingress/egress filtering, hosts within those unprotected networks would still be able to spoof nearly any source address.

Another basic filtering method is Reverse Path Forwarding, or RPF [Baker and Savola 2004]. Similar to ingress/egress filtering, routers running RPF attempt to filter packets depending on where a packet with a given source address should originate from. But whereas ingress/egress filtering only considered two directions—into an edge network or out of an edge network—RPF attempts to deal with traffic passing through a given router from any direction to any direction. RPF uses the forwarding table information at a router. It assumes that whichever direction, or interface, a packet with destination address γ should be forwarded to is the same direction a packet with source address γ should arrive from. Unfortunately, with the high amount of routing asymmetry in the Internet today [He et al. 2005; 2004], this assumption is not valid. Without this assumption holding true, RPF cannot be deployed in many places.

5.2 Distributed Defense Methods

In distributed methods of spoofing defense, routers cooperate in order to discover information for distinguishing valid and spoofing packets. The information may be related to a key which only valid packets will carry, or to the incoming direction for packets from a given source. First we discuss Spoofing Prevention Method (SPM) and Passport, and then cover Distributed Packet Filtering (DPF) and other path-based filtering works including Source Address Validity Enforcement (SAVE), Inter-Domain Packet Filters (IDPF), and BGP Anti-Spoofing Extension (BASE). Routers using SPM and Passport mark outgoing packets with secret keys, while routers using DPF and other path-based filtering systems need to learn the correct incoming direction of packets for a given source.

5.2.1 *Spoofing Prevention Method.* Routers implementing Spoofing Prevention Method, or SPM [Bremler-Barr and Levy 2005], validate a packet by checking for a secret key embedded into the packet. A source Autonomous System (AS), s, decides upon a key for every (s, d) pair, where d is a destination AS. When a packet reaches a router in AS d, the router checks for the presence of the secret key. Any packet with the key is valid, and any packet without the key is spoofing. Packets from ASes not deploying SPM do not have associated keys, so a router cannot know if a packet purporting to be from an unprotected AS is spoofing nor not. Packets from these unprotected ASes are allowed through, but when a router's network is under attack the router gives preferential service to legitimate packets from ASes deploying SPM.

In order to disseminate the secret keys that a source router uses to mark valid packets, SPM can use either a passive or active key distribution protocol. Using a passive distribution protocol, SPM routers must infer the correct key based on connections they observe. Active distribution uses a mechanism similar to the Inter-domain Routing Validator (IRV) [Goodell et al. 2003] used for securing BGP. To maintain a lighter load on routers, each AS has a server which is in charge of all key-related communication. The server keeps track of all keys which routers should check on incoming packets, and disseminates to other ASes' servers the keys that its routers will embed in outgoing packets.

Passive distribution requires a router at a destination AS to monitor connections that require some handshaking process, such as TCP connections, since it is regarded as difficult to successfully initiate such connections using spoofing packets. In the case of routing asymmetry, the router may not be able to see both sides of a connection and must utilize TCP intercept [Cisco Systems Inc. 2007] to verify source validity. If a router can only see the incoming side of a supposed connection, it cannot know if the connection actually exists, or if an attacker is simply pretending to receive valid responses. The key that is embedded into a verified valid connection is taken to be the key that the source AS uses to send to the destination router's AS. Since it is possible, albeit difficult, to complete a TCP handshake with spoofing packets by predicting TCP sequence numbers, an attacker could try to trick SPM into storing a false key. SPM would then identify spoofing traffic as legitimate and legitimate traffic as spoofing. Such an attack would be highly difficult, since SPM only observes a single connection to passively learn a new key: not only would an attacker need to complete the TCP handshake, but it would also need to do it at just the right time for SPM to observe that spoofed handshake instead of some other legitimate handshake.

Active distribution requires each AS to maintain a server that keeps track of all key-related information. The server knows how to contact key servers in all other ASes, and propagates key information to the other ASes. Each server also maintains all the necessary incoming key information it receives, and ensures local routers have up-to-date incoming key information. The local routers use the incoming key information to verify incoming packets.

In order to maintain reasonable storage requirements, a router only stores key information for source/destination key pairs in which the router's AS is either the source or destination AS. Unfortunately, this means intermediate routers cannot assist in filtering spoofing packets; only the source or destination AS can filter a spoofing packet.

5.2.2 *Passport.* Routers running the Passport [Liu et al. 2008] system also use secrets embedded in packets to verify source addresses. Instead of embedding a single secret key in the IP header, Passport defines its own header, to allow for a much larger space to hold secrets. This header can be thought of as a "passport" that contains multiple "visas," with each visa corresponding to a Passport-enabled AS along the path that a packet will travel. As the packet travels towards its destination, Passport-enabled ASes verify the visas. Each visa in the passport is a Message Authentication Code (MAC). A packet's source AS computes the MACs using secret keys shared between itself and each AS along the path to the packet's destination. Each MAC covers the packet's source address, destination address, IP identification field, packet length field, and the first 8 bytes of the payload. When a downstream AS-level router encounters a packet with a passport, it can verify the passport by recalculating the MAC value using the secret key it shares with the source AS. If the verification fails, the router will then demote or drop the packet.

In order for each pair of ASes to have a unique shared secret key, each participating AS performs a Diffie-Hellman [Diffie and Hellman 1976] key exchange. Each AS tells all other ASes its public key by piggybacking its public key onto BGP update messages. When one AS learns the public key of another AS, it can construct a shared secret key from its own private key and the other AS's public key. The other AS can construct the same shared secret key using a similar process.

5.2.3 *DPF.* Distributed Packet Filtering, or DPF [Park and Lee 2001], proposed having routers throughout the network maintain incoming direction knowledge (knowledge of which interface a packet from a given source to a given destination should arrive on). When a packet with a spoofed source address arrives at an incorrect interface, the router can detect this and filter the packet.

When a portion of routers in the network have such incoming direction knowledge, the set of addresses that an attacker can successfully spoof decreases. For instance, if an attacker behind some router, X, wants to successfully spoof addresses behind some other router, Y, packets from behind X and those from behind Y must arrive on the same interface at every DPF-enabled router en route to the destination.

The major omission from the DPF research was the actual method for routers to learn the incoming direction information. Instead, the research assumed such knowledge was available to routers and focused on defining efficacy metrics, and analyzing the effects of using various topologies and deployment distributions.

Next, we present works which provide the missing piece of how routers can learn incoming direction knowledge.

5.2.4 *SAVE.* The Source Address Validity Enforcement protocol, or SAVE [Li et al. 2002], while not designed with DPF in mind, operates similarly in that routers filter packets based on their incoming direction. Whereas the original DPF work did not provide a means of discovering valid incoming direction knowledge (DPF simply assumes routers have that knowledge), SAVE is the first protocol that helps routers learn the information and leverage the knowledge for filtering spoofing packets.

The SAVE protocol operates alongside any routing protocol by having the router (or set of routers) in charge of a given source address space send SAVE updates corresponding to each forwarding table entry. Each SAVE update travels through the network along the same path as normal traffic from that source address space. When a downstream router receives the update, it records the incoming interface of the update as the valid incoming direction for the corresponding source address space. Updates are sent both periodically and whenever a router changes its forwarding table.

Another contribution from SAVE is its invention of an "incoming tree." Every router maintains its own incoming tree that keeps track of the topological relationships of upstream source address spaces. Thus, when one routing change affects the incoming direction of many spaces, a router can automatically update the information for every affected space.

SAVE assumes that all routers deploy SAVE. For intradomain operation such an assumption is feasible. A single domain can easily deploy a new protocol like SAVE across all its routers. For interdomain operation, however, SAVE is yet to be designed for correct functionality when some routers do not run SAVE.

5.2.5 *IDPF.* Inter-Domain Packet Filters, or IDPF [Duan et al. 2006], attempts to provide an implementation of the DPF principles. Learning from BGP updates, and assuming that BGP routers adhere to a specific set of exporting rules, routers running IDPF can discover AS relationship information and then use this information to build packet filtering rules. In general, ASes can be in a provider-customer, peer-peer, or sibling-sibling relationship. These relationships put constraints on which AS paths are feasible, and which are not feasible. Packets which arrive from neighbors along an infeasible path can be filtered out.

Whereas the original DPF research specifies that routers know the actual valid incoming direction of packets, routers running IDPF only know *feasible* incoming directions, not actual incoming directions. With this limited knowledge, IDPF is not as effective as a precise DPF implementation; attackers are able to successfully spoof more source address spaces. Also, with IDPF's dependence on AS relationship information gathered from BGP updates, it is limited to only functioning alongside BGP.

5.2.6 *BASE*. BGP Anti-Spoofing Extension, or BASE [Lee et al. 2007], is another path-based packet filtering mechanism. As its name implies, BASE's implementation relies on BGP. The basic operation of BASE is similar to that of SAVE; it sends updates that routers use to learn the correct incoming direction of packets. However, instead of treating the incoming interface as the incoming direction, each BASE router marks packets with a unique key and uses the key as the incoming direction. Using markings as opposed to a physical interface is useful for incremental deployment when BASE routers are not physical neighbors. Another important difference is that BASE-enabled routers send updates by piggybacking them on top of BGP updates, as opposed to sending updates on their own. These piggybacked updates include marking information and control messages. Updates for distributing marking information include a source AS and a corresponding 16-bit marking. When a BASE router receives such an update, the router records the enclosed marking as the "incoming direction" for packets from the specified source prefix. Relying on BGP updates means BASE updates must travel the same path as BGP updates. BGP updates do not always travel the same path as normal traffic, however. The path a BGP update for prefix P takes to reach AS X does not define the path that packets from P follow to reach AS X, rather it defines the path that AS X can use to forward traffic towards P. The path of normal traffic from P to AS X may be different. When updates and normal traffic travel different paths, routers will expect the incorrect marking and misidentify legitimate packets as spoofing packets. To minimize such false positives, BASE uses control messages to enable or disable filtering. Thus, BASE routers are only able to filter spoofing packets after receiving instructions to filter.

6. COMBINATION-BASED DEFENSE MECHANISMS

Spoofing defense methods which utilize both routers and hosts generally mark packets. First, routers mark packets as they travel through the network. Then, when a packet reaches an end-host, the end-host can take action by using the marking, such as tracing the true origin of a packet regardless of its source address field.

6.1 Pi and StackPi

Path Identifier, or *Pi* [Yaar et al. 2003], originally designed to defend against denial-of-service attacks, also provides an IP spoofing defense solution. Pi reuses the fragmentation field of an IP packet to identify the path the packet traveled. As a packet travels the network, each router it encounters sets a bit in the fragmentation field. When the packet reaches its destination, the fragmentation field will contain a marking that is (almost) unique to the path the packet traveled. The end-host does not know what path the packet traveled, but if multiple packets have the same marking it is highly likely that they traveled the same path. If an end-host can identify a packet as an attack packet, then the end-host can filter out subsequent packets which have the same path identifier. Initially identifying which packet is an attack packet is left as a separate problem for the end-host to solve on its own.

StackPi [Yaar et al. 2006], essentially an improved version of Pi that functions better than Pi with incremental deployment, also added mechanisms to protect against spoofing packets. To protect against IP spoofing, an end-host can remember markings it sees in legitimate packets from different source addresses. Then, when under attack, the end-host can filter out any packet which has a marking that does not match the stored marking for the source of that packet. Note, there is no technical reason end-hosts in the Pi system cannot perform a similar comparison of markings; the authors simply did not evaluate such usage in the original Pi paper.

In both Pi and StackPi, since the marking is not entirely unique for each path, and routes are dynamic, a router may mistakenly drop legitimate packets if the router simply drops all packets with an “attack path” marking. In order to minimize any such false positives, the authors recommend only dropping packets when a host is under attack.

7. ANALYSIS

With a basic understanding of how the various spoofing defense mechanisms work, we now analyze the capabilities and characteristics of each mechanism. We first look at how well they can identify spoofing packets, and what sort of deployability issues they may have. We then consider what sort of traffic characteristics the spoofing defense mechanisms rely on, and their routing protocol independence. Next we look at how well they can mitigate attacks and discover an attacker’s true location. Finally we compare the overhead of these mechanisms.

For each capability and characteristic there is at least one spoofing defense mechanism that excels in that area, but there is no single mechanisms that excels in all areas. For instance, some mechanisms may be able to identify all spoofing packets when deployed across the entire Internet, but would not work in the real world because of deployability issues.

7.1 Identifying Spoofing Packets

All of the spoofing defense mechanisms can of course identify *some* spoofing packets, but they often cannot identify *all* spoofing packets. Table II shows how the different spoofing defense mechanisms perform at identifying spoofing packets.

IPsec can consistently identify spoofing packets. An attacker cannot set up a connection because it will not be able to receive a response. And after the connection setup, an attacker cannot know the key needed to spoof a packet.

OS Fingerprinting can detect spoofing packets if the spoofed source can be actively fingerprinted *and* the resulting fingerprint is different from the passive

Table II. Identifying Spoofing Packets

Mechanism	Efficacy	Note
Hop-count filtering	$\approx$ 90%	
TCP/IP Probes & OS Fingerprinting	unknown	Has not been analyzed
SYN cookies	100%	Only for use with TCP connections when SYN queue is full
IPsec	100%	Requires client to use IPsec as well
IP Puzzles	100%	Requires client to understand IP puzzles as well
Martian filtering	100%	Only when attackers spoof a few special addresses
Ingress/egress filtering	100%	Requires 100% deployment, poor otherwise
RPF	unknown	Has not been analyzed
SPM	$\approx$ % of deployment	Spoofing packets only identified if destination runs SPM
Passport	$\approx$ % of deployment	
DPF	$\approx$ 96% of AS pairs w/ vertex cover	$\approx$ 76% with 50% random placement
	$\approx$ 88% of ASes w/ vertex cover	$\approx$ 65% with 50% random placement
SAVE	100% w/ full deployment	Yet to be analyzed without full deployment
BASE	$\approx$ 90% w/ 10% priority placement	Top 10% of ASes by degree
	$\approx$ 20% w/ 10% random placement	
IDPF	$\approx$ 80% of ASes w/ vertex cover	$\approx$ 60% w/ 50% random placement
StackPi	$>$ 99% w/ full router deployment	Yet to be analyzed without full deployment

fingerprint of the spoofing packet. Even then, results can be complicated by a firewall between the target and the spoofed source, if the firewall filters the fingerprinting probes, or alters the responses. Fingerprinting is not reliable enough to depend on.

IP identification field probing can identify spoofing packets, but only if the spoofed source does not use any sophisticated methods for identification number assignment. Furthermore, results can be complicated by a firewall if it either filters the probing, or alters the response; this mechanism cannot be relied upon.

TCP-specific probes can identify spoofing packets fairly easily and reliably. An attacker will not receive the TCP control messages, and thus cannot react correctly. Of course, this mechanism is useless against UDP or other non-TCP packets.

IP puzzles could also identify spoofing packets fairly easily, for the same reason: The attacker will not receive the puzzle and thus cannot send a puzzle solution. Since IP puzzles work at the IP layer, below UDP or TCP, all Internet traffic could be validated.

Hop-count filtering can identify spoofing packets when the hop-count from attacker to destination is different from the hop-count from spoofed source to destination. Since the range of likely hop-counts is narrow, many spoofing packets may arrive at the destination with the correct hop-count.

Basic router-level filtering methods can effectively identify all spoofing packets (assuming the spoofed source belongs to a different network), but only if all routers employ filtering. As the Spoofer Project shows (Section 2), this has not happened. When considering filtering only deployed at a subset of routers, RPF can provide greater efficacy than ingress/egress filtering, but such a strategy has yet to be analyzed.

SPM can identify all spoofing packets whose destination and spoofed source both belong to different SPM-protected domains, but only if the keys used for marking remain secret. Packets with a spoofed source belonging to a non-SPM-protected domain will not be identified.

Passport can identify a spoofing packet when the spoofed source's AS is Passport-enabled, and at least one AS on the path of the spoofing packet is also Passport-enabled. Packets with a spoofed source belonging to a non-Passport-enabled AS will not be identified.

DPF, assuming a vertex-cover deployment and an oracle-based method of building incoming direction information, cannot identify all spoofing packets—but it can identify a large majority. When AS-level routers deploying DPF form a minimum-size vertex cover, DPF can identify all spoofing packets from around 88% of all ASes, and if we consider all source-destination AS pairs, only 4% are feasible attack pairs, or source and destination AS pairs between which DPF cannot identify spoofing packets.

SAVE can identify all spoofing packets where the attacker and destination are in different networks. Without being fully deployed on all routers, however, SAVE could not identify all spoofing packets; it has yet to function correctly when not all routers run SAVE.

IDPF is not an ideal implementation of DPF, and it cannot identify as many spoofing packets as an optimal DPF. IDPF filters spoofing packets by checking against feasible paths that packets may travel, whereas DPF knows the actual paths. When AS-level routers deploying IDPF form a vertex cover, IDPF can identify all spoofing packets from around 80% of all ASes. Further details regarding the effectiveness of IDPF at identifying spoofing packets are unavailable.

BASE also performs similarly to DPF, and is able to identify most spoofing packets. With the top 30% of ASes (according to AS-connectivity degree) BASE-enabled, BASE can identify around 97% of spoofing packets. Measurements are not available that are more directly comparable to DPF and IDPF. To avoid false positives because of AS-level routing asymmetry, BASE only enables filtering *after* a spoofing attack is detected.

Pi and StackPi can identify spoofing packets as long as the target host has received legitimate packets from the spoofed source network (otherwise it cannot know what the correct marking should be). It cannot identify all spoofing packets since some markings will overlap. Assuming full deployment, and that the target host has seen legitimate packets from every network, the probability that an attacker (with a random IP address) could successfully send a spoofing packet (with a random source IP address) is less than one percent. Efficacies with lower deployment levels are unknown.

7.2 Deployability

In general, host-based mechanisms are easier to deploy than router-based methods. It is easier to install new software at end-hosts than on routers. Also, host-based methods generally do not need as much cooperation as router-based methods. For instance, active probing and passive host-based methods can offer protection for a given host, even when that host is the only host using such a mechanism. However, IPsec can only be relied on by an end-host if all end-hosts it communicates with also implement IPsec. Similarly, in order for administrators to rely on IP puzzles, IP puzzles must first be standardized and deployed so that all network nodes can understand the puzzles.

Some router-based mechanisms, including SPM, Passport, BASE, Pi, and StackPi, rely on packet markings. Deployment problems arise when using IP header fields reserved for other purposes. Excluding Passport, all of the other proposed packet markings use the IP identification field, which is needed to properly reassemble fragmented packets. Additionally, there are many IP traceback mechanisms that want to re-use the identification field for their own purposes, such as Snoeren et al. [2002], Savage et al. [2000], Adler [2005], and Dean et al. [2002]. There is no standard for any alternative usage of the IP identification field, and at the present time, we cannot know which usage will prevail. Passport uses the IP Options field to insert multiple markings in a packet. This avoids direct incompatibilities with the IP identification field, but fragmentation will still invalidate the markings, since an intermediate router cannot recalculate the correct markings. Therefore, Passport treats all fragmented packets with a lower priority. Using IP Options also makes every packet's IP header require more space. IDPF does not use any packet markings and can be deployed by even a single router, but there is not a large incentive for administrators to deploy the protocol; deploying IDPF does not directly benefit the deploying AS more than another AS. SAVE is more deployable in that it uses no packet markings and can run alongside any routing protocol, but it has yet to be designed to function without full deployment; work is ongoing to allow SAVE to work with incremental deployment [Ehrenkranz and Li 2007]. Basic router filtering also uses no markings and is easy to deploy, but lacks incentive for deployment.

7.3 Essential Traffic Characteristics

In order to be resilient against more intelligent attackers, spoofing defenses cannot rely on traffic characteristics that an attacker can easily manipulate and spoof the correct values. First we will discuss the resiliency of router-based and combination-based systems to manipulation, then we will discuss the resiliency of host-based systems.

SAVE, basic router filtering, DPF, and IDPF validate packets using their physical incoming interface. An attacker cannot manipulate which interface attack packets enter at a router.

The rest, including SPM, Passport, BASE, Pi, and StackPi, rely on packet markings that an attacker can attempt to manipulate, which attackers can manipulate to attempt falsifying marking values. Using SPM, routers do not change a packet's marking once it leaves the source AS, and only the destination AS checks the marking. Furthermore, all packets from a given source AS to a given destination AS contain the same marking. Thus, if an attacker sniffs or otherwise learns a valid marking, the attacker can use that marking to successfully spoof that source to that destination from any non-SPM location. In contrast, markings in Passport change for every packet; an attacker cannot use markings from a legitimate packet to craft markings for spoofing packets. With BASE, Pi, and StackPi, routers along a packet's path will modify the packet's markings; even if an attacker knows the legitimate marking values set at the source and seen at the destination, setting the correct initial marking is much more difficult. Since the path from the attacker to the destination is different from the legitimate source to the destination, the marking will be modified in a different manner.

Router-based and combination-based solutions that use packet marking also need to be careful in dealing with packet fragmentation. SPM, BASE, Pi, and StackPi use the IP identification field to store their markings. But the IP identification field is needed to properly reassemble fragmented packets. Passport uses the IP identification field and the length field to calculate its markings. But when a router fragments a packet, the router changes the length field, making any previously calculated marking invalid. In order to avoid affecting (or being affected by) fragment reassembly, a mechanism can avoid marking and processing fragmented packets, but then an attacker can break through the defense mechanism simply by making fragmented packets. One might choose to simply drop all fragmented packets when under attack, but then the question is: How does the defense mechanism decide when an attack is occurring? Furthermore, even if one knows an attack is occurring, how large must an attack be before the cost of losing fragmented legitimate traffic is outweighed by the benefit of dropping spoofing traffic?

Regarding host-based defenses, attackers can also manipulate certain traffic characteristics. If using hop-count filtering and OS fingerprinting, for example, attackers can easily set a packet's initial TTL value to any value, and craft packets which seem to originate from any operating system. While it is difficult for an attacker to know the correct TTL value or OS type to set for a specific source address, if they have a way of probing to see when an attack succeeds they can try multiple values until the attack succeeds. Once they find the correct value, they can continue to use that value.

Attackers can hardly manipulate the host-based defense methods such as IPsec, IP puzzles, SYN cookies, and TCP probes, that rely on end-hosts knowing a secret key or receiving secret control messages. Although these secrets are somewhat similar to markings router-based methods use (both involve the source knowing a secret), it is not as problematic as markings used in router-based methods. Since the secrets are unique for every connection, the danger of an attacker learning a secret is not as great. Furthermore the secrets do not interfere with fragmentation.

7.4 Routing Protocol Independence

Although BGP version 4 is the de facto inter-AS routing protocol of the Internet, we cannot know how future versions will change. There are also a number of intra-AS routing protocols currently in use. It would be ideal to have a solution that would work across any routing protocol, be it BGP [Rekhter et al. 2006], OSPF [Moy 1998], IS-IS [Oran 1990], EIGRP [Albrightson et al. 1994], or some other routing protocol.

All host-based and combination-based defense mechanisms are routing protocol independent, but some router-based mechanisms are dependent on a specific routing protocol. IDPF relies on BGP to learn AS relationships and which AS-level paths are feasible and which are not. BASE piggybacks its own updates on top of BGP updates to send control messages and marking information. SPM does not use BGP directly, but it limits key granularity to the AS level, and routers within an AS share a server to handle all the key maintenance operations. Passport piggybacks public keys on top of BGP updates to facilitate a Diffie-Hellman key exchange, and requires routers to know the AS path outgoing packets will travel. For router-based mechanisms, only basic filtering mechanisms and SAVE can function alongside any routing protocol. SPM may be modified to use prefix-level granularity, but how such a modification would alter the performance and operation of the protocol has yet to be researched. BASE and Passport may also be modified to piggyback keys and control messages on top of another routing protocol, or to send keys and control messages independently of a routing protocol, but such modifications would be changing operations central to BASE and Passport, and have yet to be researched.

7.5 Mitigating Attacks

Filtering spoofing packets may not do much good if the attacker can still achieve his goal. Even if a defense mechanism may detect and drop a spoofing packet, additional spoofing packets may keep coming. In some cases, an attack may be mitigated by a host simply by ignoring the spoofing packets. But in case the attack actually targets an end-host's bandwidth, ignoring the packets when they reach the end-host is of no use—the spoofing packets already did their damage. We now look at how well the spoofing defense mechanisms mitigate attacks. Table III shows how they compare.

In general, all of the *host-based* mechanisms fail at mitigating bandwidth-based denial-of-service attacks. The end-host may not waste resources responding to the spoofing packets, but since the packets cannot be stopped before reaching the end-host, the spoofing packets still waste the end-host's bandwidth.

IPsec is completely effective at preventing an attacker from interfering with an existing connection, and preventing an attacker from successfully initiating a connection. But it is ineffective at mitigating a bandwidth-based

Table III. Mitigating Spoofing Attacks

Mechanism	Efficacy
Host-based systems	Poor, reaches end-host
Ingress/egress filtering	Good, filtered either by source or destination router
Martian filtering and RPF	Excellent, filtered by intermediate routers
SPM	Good, reaches target AS
Passport	Better, priority lowered by intermediate ASes, filtered by target AS
DPF	Excellent, filtered by intermediate ASes
SAVE	Excellent, filtered by intermediate routers
BASE	Excellent, filtered by intermediate ASes
IDPF	Excellent, filtered by intermediate ASes
StackPi	Poor, reaches end-host

denial-of-service attack, and may in fact exacerbate the problem, since initializing a connection may require more resources when using IPsec than when not.

OS Fingerprinting and *IP identification field probing* can prevent a host from processing spoofing packets, but with a high cost. First, actively probing the spoofed source will require resources. Second, in order to avoid processing the spoofing packets before verifying the source, packets may require buffering. Even with this cost many spoofing packets may remain unidentified, as reported previously.

TCP-specific probes offer some protection by preventing a host from processing spoofing packets. But if it is necessary to check every packet, the probing overhead could pose a problem, and even more so when defending against a bandwidth-based denial-of-service attack.

IP puzzles offer some protection against attack. Targets would not waste resources processing spoofing packets, but the packets would still reach the target host. Bandwidth-related attacks would still succeed. If puzzles are sent by intermediate routers instead of end-hosts, the spoofing packets may be detected closer to an attacker and IP puzzles could then mitigate bandwidth-based attacks as well.

Hop-count filtering can prevent the target from wasting resources responding to spoofing packets. However, since the packets have already arrived at the target, bandwidth resources would have already been wasted. Hop-count filtering does not protect against spoofing packets which attack the bandwidth of a target.

Basic router-level filtering, if deployed everywhere, would protect against nearly all spoofing attacks since any spoofing packet would be dropped at the attacker's network. If we assume spoofing packets escape the attacker's network, RPF and Martian filtering could catch the packets at intermediate routers; however, ingress/egress filtering would fail to catch the spoofing packets, unless the packets were spoofing an address from the destination router's internal network.

SPM would offer some protection to SPM-protected networks. Identifiably spoofing packets would not reach the end-hosts, but would still reach the target AS. Unfortunately, this means any bandwidth-related attacks would still succeed if the bottleneck link is an inter-AS link, not an intra-AS link.

Table IV. Locating Attackers

Mechanism	Efficacy
Host-based systems	None
Basic router-level filtering	Minimal
SPM	Only if attacker is in SPM-enabled AS
Passport	Only if attacker is in Passport-enabled AS
DPF	Narrow down to 5 or fewer ASes with vertex cover
SAVE	100% with full deployment Incremental deployment not analyzed
BASE	Not analyzed
IDPF	Narrow down to 28 or fewer ASes with vertex cover
StackPi	Possible if end-host has a record of the spoofing packet's marking

Passport offers better protection against most spoofing attacks, assuming the attacker spoofs a source address belonging to a Passport-enabled AS, and that there are Passport-enabled ASes between the attacker and destination. Note that intermediate routers merely put spoofing packets into a lower priority queue; only the last Passport-enabled AS actively drops spoofing packets. This would mitigate bandwidth-based attacks, but not entirely, since legitimate traffic from non-Passport-enabled ASes is not in the same high priority queue as Passport-verified traffic.

DPF, *IDPF*, and *BASE* could mitigate most spoofing attacks, given proper deployment locations. Spoofing packets would normally be dropped before reaching the target AS, even offering protection against bandwidth-based attacks.

SAVE would also be able to protect against all spoofing attacks, including bandwidth-based attacks, since any spoofing packet would be dropped at the attacker's network or at an intermediate router.

Pi and *StackPi* offer some protection against attack. Targets would not waste resources processing spoofing packets, but the packets would still reach the target host. Bandwidth-related attacks would still succeed.

7.6 Locating Attackers

When spoofing packets are received, can the spoofing defense mechanisms identify where an attacker is located? Without being able to locate an attacker, an attacker has no risk of being caught. The more likely that defenders can locate an attacker or an attacker's zombies, the less likely that an attacker will risk mounting an attack. Table IV shows how well the various spoofing defense mechanisms can locate an attacker.

All of the *host-based* mechanisms cannot identify where an attacker is located. When using a purely host-based defense mechanism, a spoofing packet will not contain any information providing a hint to an attacker's true location.

Basic router-level filtering would only be able to locate an attacker if the attacker's local router captured the spoofing packets.

SPM may or may not identify where an attacker is located. If the attacker is not in an SPM-protected AS, then the spoofing packets will have no marking and SPM has no way of identifying the true source. If the attacker is in an SPM-protected AS, SPM can identify the attacker's location: If the attacker's AS performs ingress/egress filtering, the attacker is located by the attacker's SPM-enabled router; if the attacker's spoofing packet reaches the target network, the SPM-enabled router there can identify the attacker's network by looking up the key embedded in the packet.

Passport could identify an attacker's location in a manner similar to SPM. If an attacker's spoofing packets originate from a Passport-enabled AS, and that AS does not drop the spoofing packet using ingress/egress filtering, a downstream Passport-enabled router may be able to identify the attacker's location. The difference is that a Passport-enabled router cannot perform a simple lookup operation on the embedded MAC. The router would have to compare the packet's MAC with MACs computed using the shared secret key with every possible source AS.

DPF cannot identify exactly where an attacker is located, but it can narrow down the possible locations. When AS-level routers deploying DPF form a vertex cover, DPF can narrow down the possible locations of an attacker to 5 or fewer.

SAVE can identify where an attacker is located, since the router at the attacker's network would catch the spoofing packet. SAVE's ability to locate an attacker without full deployment has not been analyzed but should be similar to DPF, as SAVE provides routers with exactly the information DPF requires.

IDPF functions similarly to DPF, and can narrow down possible locations of an attacker. When AS-level routers deploying IDPF form a vertex cover, IDPF can narrow down the possible locations of an attacker to 28 or fewer.

BASE also functions similarly to DPF, and should be able to narrow down possible locations of an attacker, but this aspect of BASE has not been analyzed. The ability to locate an attacker would be hindered, however, if an attacker is able to spoof BASE markings.

Pi and *StackPi* can possibly narrow down the possible locations of an attacker. If an end-host keeps track of which Pi markings it has seen in legitimate packets, then that end-host might be able to narrow down the location of an attacker. When a spoofing packet arrives, the end-host can identify which networks it has received packets from with the same marking as that of the spoofing packet. If the end-host never received a legitimate packet from the attacker's network, this would not work. There may also be multiple networks which produce the same Pi marking, since the markings are not entirely unique.

7.7 Overhead

All of the spoofing defense mechanisms require at least some level of overhead. They may incur some storage cost, computational cost, and bandwidth cost. The overhead cost of a defense mechanism must be acceptable for administrators to consider deploying it. Host-based mechanisms generally have higher overhead costs than router-based mechanisms, but this is a conscious design decision: End-hosts can afford to be more heavily loaded than routers.

Nearly all of the defense mechanisms have a storage cost. Table V shows the storage overhead of the various spoofing defense mechanisms. Cryptographic

Table V. Storage Overhead

Mechanism	Storage Overhead
Hop-count filtering	16MB for /24 granularity hop-count tables (+ ≈20% for higher granularity clustering)
SYN cookies	Negligible
Other host-based systems	O(# of active flows)
Basic router-level filtering	Essentially none, routers already have necessary information
SPM	O(# of SPM-enabled ASes)
Passport	O(# of Passport-enabled ASes)
DPF	Unknown (no actual implementation)
SAVE	O(# of SAVE-enabled network prefixes)
BASE	16 bits per protected prefix (≈ 0.5 MB with 275,000 prefixes)
IDPF	O(size of AS-path graph)
StackPi	16 bit marking for each known IP at end-host

solutions such as IPsec must keep track of any negotiated keys for each protected connection. Active host-based solutions must also incur a storage cost, such as results from probing or state information regarding which hosts provided a puzzle solution. Passive host-based methods maintain information such as IP-to-hop-count mappings. Router-based methods maintain valid incoming direction or key information. Basic router filtering mechanisms such as ingress/egress filtering obtain this information essentially for "free," since a border router must already know what addresses are internal. SPM maintains a list of incoming and outgoing keys, one of each for every SPM-enabled AS. Passport routers maintain a list of shared secret keys and key contexts, one of each for every Passport-enabled AS. SAVE maintains interface-based incoming direction information for all address spaces, while BASE maintains marking-based "incoming direction" information for ASes. IDPF keeps track of AS relationships and incoming direction information for ASes. Pi and StackPi not only require a small amount of state in routers, but also require end-hosts to keep track of markings seen in packets from different IP addresses.

The computational cost of the defense mechanisms is more diverse. Some have only a negligible amount of computation required, such as sending out probing messages or comparing expected TTL values to actual TTL values. In contrast, IP puzzles are specifically meant to incur a computational cost on clients, acting as a deterrent to flooding the network. The cryptographic computations of IPsec also incur a high cost on communicating hosts, and in fact will decrease the number of packets a host can process during a given period of time. The computational costs of router-based and combination-based mechanisms are carefully considered by protocol designers, since nobody wants to propose a mechanism which will increase latencies in the Internet. Disregarding computation triggered by control messages (which generally only occurs during routing changes), SAVE and IDPF simply need to compare a packet's incoming direction with the valid incoming direction. Mechanisms which mark packets incur a higher cost, since routers must take time to alter the packet markings. With SPM, routers at edge networks mark keys on outgoing packets and check the keys of incoming packets. Passport has a higher cost since both edge routers and intermediate routers check packet markings. Additionally, each

Table VI. Spoofing Defense Mechanism Recap

Mechanism	Identifying Spoofing Packets	Deploy-ability	Essential Characteristics	Routing Protocol Independence	Attack Mitigation	Locating Attackers
Hop-count filtering	⋆⋆⋆	⋆⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆	—
TCP/IP Probes & OS Fingerprinting	?	⋆⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆	—
SYN cookies	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆	—
IPsec	⋆⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆	—
IP Puzzles	⋆⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆	—
Martian filtering	⋆	⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	—
Ingress/egress filtering	⋆	⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆	—
RPF	?	⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	—
Passport	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆	⋆⋆⋆	⋆
SPM	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆⋆	⋆
DPF	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	?	⋆⋆⋆⋆	⋆⋆⋆⋆
SAVE	⋆⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆
BASE	⋆⋆⋆⋆	⋆⋆	⋆⋆⋆	⋆	⋆⋆⋆⋆	?
IDPF	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	—	⋆⋆⋆⋆	⋆⋆⋆
StackPi	⋆⋆⋆⋆	⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	⋆	⋆⋆

source router has to digitally sign all outgoing packets with multiple MACs. BASE also has a high cost since not only edge routers but any BASE router will set and check packet markings. Pi and StackPi also mark packets at each router, but each router only sets one or two bits of the marking. End-hosts using Pi or StackPi only need to map an IP address to a marking.

The bandwidth cost of the defense mechanisms is also very variable. Some mechanisms do not incur any bandwidth overhead: Pi, StackPi, IDPF, basic router filtering, and passive host-based methods. Active host-based mechanisms and other router-based mechanisms, however, incur a bandwidth cost. For example, IPsec not only increases the bandwidth needed to initiate a connection, but also adds an additional header to every packet. Passport, BASE, and SAVE use bandwidth for control messages to update key or incoming direction information. Passport and BASE piggyback on top of BGP updates and so BGP updates will increase in size. SAVE sends its own update messages separate from any routing protocol, and the bandwidth overhead is comparable to that of a routing protocol. SPM only requires bandwidth when an AS informs other ASes of its corresponding keys. Routers would send and receive around 120 kilobytes every few hours when keys are updated.

7.8 Recap

Table VI gives an overview of how the spoofing defense mechanisms stack up against each other with respect to the aforesaid characteristics. For details on each characteristic and capability, refer to the relevant preceding sections. Each mechanism is given zero to four stars for each characteristic and capability. We use a question mark to indicate an unknown value, and a dash to indicate a value of zero stars. For brevity we do not list ratings for overhead.

8. CONCLUSION

IP spoofing remains a severe problem in today's Internet. Not only are there still many areas where spoofing is possible, but attackers also have motivation for performing IP spoofing. Researchers have developed numerous spoofing defense mechanisms, all with advantages and disadvantages. All the spoofing defense mechanisms are able to identify some amount of spoofing traffic, but they show a variety of efficacies, including when considering their capabilities of locating an attacker or mitigating an attack.

In this article, we surveyed host-based solutions, router-based solutions, and their combination. None of the host-based methods can identify an attacker's location, nor defend against bandwidth-based denial-of-service attacks. They may be used to offer extra protection to a specific service, but they cannot be viewed as a final solution. They are more easily deployable than other mechanisms, and may be useful while we wait for more complete solutions to become available. Meanwhile, current router-based methods are promising yet inadequate. Basic router-level filtering, especially ingress/egress filtering, would be effective in all aspects, but its full deployment requirement is prohibitive. SAVE would also be very effective, but suffers from a similar requirement, namely not being able to function correctly without full deployment. SPM offers hope with the possibility of incremental deployment. It also has better attack mitigation and attacker identification capabilities than the host-based methods, but still allows spoofing traffic to reach the target AS. Passport can catch spoofing packets at intermediate routers, but in order to avoid false positives due to routing changes, intermediate routers do not drop the packets until they reach the final Passport-enabled AS. Furthermore, Passport requires BGP for key exchange, and it remains to be seen if the cryptographic calculations required by Passport can be done at high-speed routers. DPF, along with its relatives IDPF and BASE, offer the best performance all around when incremental deployment is required. They can identify spoofing packets, narrow down possible attacker locations, and effectively mitigate even bandwidth-based spoofing attacks. However, both DPF implementations (IDPF and BASE) rely on BGP to function, affecting their portability to wherever or whenever a different routing protocol is used. Furthermore, IDPF requires routers follow a specific set of exporting rules in order to function. BASE relies on packet markings and only enables filtering after an attack is discovered in order to avoid false positives. Finally, Pi and StackPi, requiring both routers and end-hosts, bring some capabilities to locate attackers, but still cannot stop spoofing packets from reaching end-hosts.

For deploying a defense mechanism today, we can recommend some host-based mechanisms. SYN cookies should be enabled to defend against both IP spoofing and SYN flooding. However, SYN cookies only work with TCP connections, and are generally only enabled when a host's SYN queue fills up;

additional protection may be required. Hop-count filtering would be a good choice for protection against spoofing with any type of IP traffic. Although the host-based mechanisms may not be as effective as a well-deployed, router-based defense mechanism, we cannot recommend a router-based mechanism at this point since there is no single mechanism clearly superior to the others.

We believe a "future-proof," router-based solution should be developed which is not only incrementally deployable, but also has no reliance on manipulatable traffic characteristics or a specific routing protocol, and is proactive without suffering from false positives. Incremental deployment is required since we cannot assume all routers, or hosts, could implement a new protocol at the same time. Note that incentive is particularly necessary for incremental deployment; if the incremental benefit of deploying a defense mechanism is too low, networks will be hesitant to deploy it. Defenses cannot rely on traffic characteristics that an attacker can easily manipulate and spoof the correct values, since intelligent attackers may be able to pass through the defense mechanism. Routing protocol independence is required since, although BGP is the de facto routing protocol in the Internet today, there is no guarantee that will always be the case. Relying on BGP could also deny protection to most intra-AS networks. It would be a mistake to assume that IP spoofing only happens at the inter-AS level. And finally, since an attack can happen at any time, it is not acceptable to activate a solution only after an attack is detected. The system should automatically accurately detect and effectively mitigate an attack, instead of waiting for instructions to detect and mitigate an attack.

One recent trend some researchers are looking into is attempting to piece together different existing works. For example, Source Address Validation Architecture [Wu et al. 2007], or SAVA, uses different mechanisms at different levels to address IP spoofing in IPv6. At the subnet level, every IP address is bound to a specific MAC address and switch port; Within an AS, SAVA requires ingress/egress filtering; Between ASes, SAVA is either similar to IDPF by building filtering rules between SAVA-enabled ASes, or similar to SPM by embedding a signature in packets when non-SAVA-compliant ASes are involved.

Whereas we have yet to see if simply using different mechanisms in different situations is the best way to create effective defense mechanisms, there is no question we should build upon the lessons learned from past research. Borrowing ideas and methods from each of these mechanisms may prove beneficial to future development of improved defenses. As a starting point for a future spoofing defense mechanism, we suggest taking concepts from two existing works: DPF and SAVE. A DPF-like system will work well with incremental deployment, and a SAVE-like system can provide routers the necessary information to create DPF tables—without relying on any specific routing protocol and without using any manipulatable characteristics.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

ADLER, M. 2005. Trade-offs in probabilistic packet marking for IP traceback. *J. ACM 52*, 2, 217–244.

ALBRIGHTSON, B., GARCIA-LUNA-ACEVES, J., AND BOYLE, J. 1994. EIGRP—A fast routing protocol based on distance vectors. In *Proceedings of the Networld/Interop*.

AURA, T. AND NIKANDER, P. 1997. Stateless connections. In *Proceedings of the International Conference on Information and Communication Security*, Y. Han et al., Eds. Lecture Notes in Computer Science, vol. 1334. Springer, 87–97.

BAKER, F. 1995. Requirements for IP Version 4 routers. RFC 1812.

BAKER, F. AND SAVOLA, P. 2004. Ingress Filtering for Multihomed Networks. RFC 3704.

BERNSTEIN, D. J. 1996. SYN cookies. `http://cr.yp.to/syncookies.html`.

BEVERLY, R. 2004. A robust classifier for passive TCP/IP fingerprinting. In *Proceedings of the Passive and Active Measurement Conference*, 158–167.

BEVERLY, R. AND BAUER, S. 2005. The Spoofer Project: Inferring the extent of source address filtering on the Internet. In *Proceedings of the USENIX Work-Shop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 53–59.

BREMLER-BARR, A. AND LEVY, H. 2005. Spoofing prevention method. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*.

CHANG FENG, W., KAISER, E. C., CHI FENG, W., AND LUU, A. 2005. Design and implementation of network puzzles. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*. 2372–2382.

CISCO SYSTEMS INC. 2007. Configuring TCP intercept. `http://www.cisco.com/en/US/docs/ios/security/configuration/guide/sec_cfg_tcp_intercpt.pdf`

DEAN, D., FRANKLIN, M. K., AND STUBBLEFIELD, A. 2002. An algebraic approach to IP traceback. *ACM Trans. Inf. Syst. Secur. 5*, 2, 119–137.

DIFFIE, W. AND HELLMAN, M. E. 1976. New directions in cryptography. *IEEE Trans. Inf. Theory 22*, 6, 644–654.

DUAN, Z., YUAN, X., AND CHANDRASHEKAR, J. 2006. Constructing inter-domain packet filters to control IP spoofing based on BGP updates. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*.

EHRENKRANZ, T. AND LI, J. 2007. An incrementally deployable protocol for learning the valid incoming direction of IP packets. Tech. rep. CIS-TR-2007-05, University of Oregon. March.

FERGUSON, P. AND SENIE, D. 2000. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2827.

FYODOR. 2006. Remote OS detection. `http://nmap.org/book/osdetect.html`.

GOODELL, G., AIELLO, W., GRIFFIN, T., IOANNIDIS, J., MCDANIEL, P., AND RUBIN, A. 2003. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In *Proceedings of the Network and Distributed System Security Symposium*.

HE, Y., FALOUTSOS, M., AND KRISHNAMURTHY, S. V. 2004. Quantifying the routing asymmetry in the Internet at the AS level. In *Proceedings of IEEE Conference and Exhibition on Global Telecommunications (GlobeCom)*.

HE, Y., FALOUTSOS, M., AND KRISHNAMURTHY, S. V. 2005. On routing asymmetry in the Internet. In *Proceedings of IEEE Conference and Exhibition on Global Telecommunications (GlobeCom)*.

JIN, C., WANG, H., AND SHIN, K. G. 2003. Hop-count filtering: An effective defense against spoofed DDoS traffic. In *Proceedings of the Conference on Computer and Communications Security*, 30–41.

KENT, S. AND SEO, K. 2005. Security architecture for the Internet Protocol. RFC 4301.

KILLALEA, T. 2000. Recommended Internet service provider security services and procedures. RFC 3013.

LEE, H., KWON, M., HASKER, G., AND PERRIG, A. 2007. BASE: An incrementally deployable mechanism for viable IP spoofing prevention. In *Proceedings of the ACM Symposium on Information, Computer, and Communication Security*.

LI, J., MIRKOVIC, J., WANG, M., REIHER, P. L., AND ZHANG, L. 2002. SAVE: Source address validity enforcement protocol. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*. 1557–1566.

LIU, X., LI, A., YANG, X., AND WETHERALL, D. 2008. Passport: Secure and adoptable source authentication. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*.

MARTIN, K. 2006. Stop the bots. *Security Focus*.

MESSMER, E. 2007. Report says identity thieves working hand in hand with 'bot herders'. *Network World*.

MIT ADVANCED NETWORK ARCHITECTURE GROUP. 2007. ANA Spoofer Project. `http://spoofer.csail.mit.edu/`.

MOY, J. 1998. OSPF Version 2. RFC 2328 (Standard).

ORAN, D. 1990. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Informational).

PARK, K. AND LEE, H. 2001. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proceedings of ACM SIGCOMM Data Communications Festival*, 15–26.

PISCITELLO, D. M. 2006. Anatomy of a DNS DDoS amplification attack. `http://www.watchguard.com/infocenter/editorial/41649.asp`.

REKHTER, Y., LI, T., AND HARES, S. 2006. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard).

SALTZER, J. H., REED, D. P., AND CLARK, D. D. 1984. End-to-end arguments in system design. *ACM Trans. Comput. Syst. 2*, 4, 277–288.

SAVAGE, S., WETHERALL, D., KARLIN, A. R., AND ANDERSON, T. E. 2000. Practical network support for IP traceback. In *Proceedings of ACM SIGCOMM, Data Communications Festival*, 295–306.

SNOEREN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., SCHWARTZ, B., KENT, S. T., AND STRAYER, W. T. 2002. Single-packet IP traceback. *IEEE/ACM Trans. Netw. 10*, 6, 721–734.

TALECK, G. 2003. Ambiguity resolution via passive OS fingerprinting. In *Proceedings of the Symposium on Recent Advances in Intrusion Detection*, 192–206.

TEMPLETON, S. J. AND LEVITT, K. E. 2003. Detecting spoofed packets. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, vol. 1. 164–175.

WANG, H., JIN, C., AND SHIN, K. G. 2007. Defense against spoofed IP traffic using hop-count filtering. *IEEE/ACM Trans. Netw. 15*, 1, 40–53.

WU, J., REN, G., AND LI, X. 2007. Source address validation: Architecture and protocol design. In *Proceedings of the Annual International Conference on Network Protocols (ICNP'07)*.

YAAR, A., PERRIG, A., AND SONG, D. 2003. Pi: A path identification mechanism to defend against DDoS attack. In *Proceedings of the IEEE Symposium on Security and Privacy*, 93–107.

YAAR, A., PERRIG, A., AND SONG, D. 2006. StackPi: New packet marking and filtering mechanisms for DDoS and IP spoofing defense. *IEEE J. Selected Areas Commun. 24*, 10, 1853–1863.

ZALEWSKI, M. 2001. Strange attractors and TCP/IP sequence number analysis. `http://lcamtuf.coredump.cx/oldtcp/`.

ZALEWSKI, M. 2002. Strange attractors and TCP/IP sequence number analysis—One year later. `http://lcamtuf.coredump.cx/newtcp/`.

ZALEWSKI, M. 2006. Passive OS fingerprinting tool. `http://lcamtuf.coredump.cx/p0f.shtml`.

Received September 2007; accepted May 2008