Using Router Stamping to Identify the Source of IP Packets

Thomas W. Doeppner*
Brown University
twd@cs.brown.edu

Philip N. Klein†
Brown University
klein@cs.brown.edu

Andrew Koyfman
Oracle Corporation
akoyfman@us.oracle.com

ABSTRACT

Denial of Service and *Distributed Denial of Service* attacks have cost millions of dollars to online companies. Unfortunately, these attacks are particularly difficult to stop since hackers are able to hide their IP address by IP spoofing, so that it is often impossible to identify their location. Our proposal, *Router Stamping*, would help to identify the source of *Denial of Service* attacks, provided that a significant percentage of packets are sent from one subnet. In accomplishing this, *Router Stamping* imposes only a small increase in the size of the packet header. In addition, it is easy to implement and maintain, and it can function in the presence of some noncompliant or malicious routers.

1. INTRODUCTION

A number of *Denial of Service* (*DoS*) and *Distributed Denial of Service* (*DDoS*) attacks have occurred in 2000, of which the attacks on Yahoo!, eBay, Amazon.com, and other dot-coms launched in February 2000 are the most notorious. These attacks prevented service to millions of users over a period of 2-3 days and cost the companies millions of dollars in lost revenues.

Many *DoS* attacks rely on IP spoofing at some stage of the attack. By hiding his IP address, the hacker makes it very difficult to find the source of the malevolent IP packets. In addition, IP spoofing allows the hacker to make it appear that an attack comes from a completely different and innocent network. When the attacked network's administrator attempts to solve the problem by blocking all traffic from the apparent attack source, he is in fact taking action against a non-hostile system, thereby contributing to the denial of service.

*Research partially supported by a grant from Sun Microsystems

†Research partially supported by NSF Grant CCR-9700146.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CCS '00, Athens, Greece.
Copyright 2000 ACM 1-58113-203-4/00/0011 ..$5.00

1.1 Current Defenses

Attacks with a valid IP address are easier to defend against, and in fact simple prevention tools and operating system fixes that limit the damage from these attacks already exist. However, they are not enough.

To help combat IP spoofing, Ferguson proposes [2] that ingress traffic filters be used to prohibit *DoS* attacks that use a forged IP address. In addition, Baker suggests [3] a method for detecting packets with spoofed addresses without relying on ingress routing.

Unfortunately, these approaches are meeting much resistance in the ISP community. Some security experts believe that such filters may not work in institutions such as universities, where the network configuration changes quickly. In addition, ISPs are concerned about the costs associated with maintaining the filters on thousands of their routers [5].

1.2 Motivation for Router Stamping

Clearly, an effective algorithm that addresses IP spoofing in *DoS* attacks is required. At the very least this algorithm should identify with high probability some subnet containing the source of an attack, unwitting or otherwise. Furthermore, the identification must be possible despite the presence of some non-compliant routers and ISPs. In addition, the algorithm needs to address the concerns of the ISP community and be easy to implement and maintain. Finally, there should be incentives for ISPs to adopt these security measures.

Emerging approaches that address these issues attempt to track or to trace the flood of malicious IP packets. Recent interest in this idea produced a number of new systems. Stone's CenterTrack[14] is one such example. Another proposal, similar to ours, was done independently by Savage [12] and inspired the TRACE work of the IETF [13].

Our proposal, *Router Stamping*, is a simple and effective method for locating the source of a *DoS* attack even if it employs IP spoofing. First we present a deterministic version of our algorithm in Section 2. In Section 3 of this paper, we address the shortcomings of the deterministic version by allowing the algorithm to function probabilistically. We explore the costs associated with this approach in Section 4. Finally, in Sections 5 and 6 we present some guidelines for a distributed defensive algorithm that can be used in response to the *DoS* attack once the approximate location of the source is known. Although *Router Stamping* is not a complete solution to *DoS* attacks, it is a promising approach that could provide one component of Reddy's self

Cloudflare, Inc. Exhibit 1007 Page 1

monitoring Internet [8].

2. DETERMINISTIC ROUTER STAMPING

Before we present the actual *Router Stamping* proposal, let us consider a simplified deterministic version of the algorithm. *Deterministic Router Stamping* is analogous to the *Record Route* option of IP [1]. For the purpose of this algorithm, however, we assume that the IP header has enough space to record all IP addresses of all relevant routers, rather than just the 36 bytes allocated for *Record Route*.

Each router R_i that implements *Deterministic Rout er Stamping* follows the following algorithm:

1. For each packet routed through R_i
2. R_i fills in the first empty slot with the IP address of the router that forwarded the packet to R_i
3. R_i routes the packet

2.1 Example of Deterministic Router Stamping

Consider the sample network presented in Figure 1. A packet is sent from S to T. When R_1 processes the packet, it records S's IP address in the first available slot and forwards the packet to R_2. R_2 records R_1's address in the second slot and forwards the packet to R_4, which in turn records R_2's address in the third slot and forwards the packet to T.

In case T does not trust the source address in the IP header, it can verify the source of the packet: assuming that routers' entries are accurate, T looks at the recorded route that the packet has traveled.

2.2 Drawbacks of Deterministic Router Stamping

Unfortunately, *Deterministic Router Stamping* is impractical and infeasible. First and foremost, it requires too much space. A typical packet goes through as many as 30 routers before it reaches its destination [6]; if space were allocated to hold all 30 router entries the standard TCP/IP header would expand 400% to 160 bytes. Clearly this is unacceptable.

Even if enough space could be allocated for routers' entries, however, the deterministic algorithm would still fail to identify the source of malicious packets. The problem with a fixed-length header is that each router relies on the information recorded in the packet in order to determine where to write its entry. A smart attacker can exploit this dependency.

For example, the attacker can send a packet with all of the slots filled with fake router entries. Since a given router does not know how many routers processed the packet before it, it must rely on the entries in the packet. Since all the slots are filled with fake entries, the router will assume that the packet has already traveled through many routers and will not record its entry. Thus the attacker avoids detection.

The above problem could be avoided by using a variable length field, whereby each router appends its entry to the end. However, this would make the IP header expand out of control. One could easily imagine IP packets that contain more header information than data.

The considerations above imply that a router cannot decide on a course of action on the basis of the information already contained in the packet. For similar reasons, routers should not act according to any pre-determined patterns, since such patterns can be exploited by an ingenious attacker. In order to avoid these patterns we add a probabilistic component to the *Deterministic Router Stamping* algorithm. In addition, the probabilistic element allows *Router Stamping* to function using significantly less space than the deterministic algorithm.

3. ROUTER STAMPING ALGORITHM

Router Stamping is a probabilistic algorithm with a storage parameter s and a constant probability p, where s is the number of slots available for routers' *stamps*. Section 3.2 addresses the choice of s and p. The *stamp* is the IP address of the router and the interface on which the packet was received.

Each compliant router R_i implements the following algorithm:

1. For each packet routed through R_i
2. $x \leftarrow$ random number between 0 and 1
3. if $x < sp$
4. R_i places its *stamp* into slot $\lfloor x/p \rfloor$
5. R_i routes the packet

3.1 Example of Router Stamping

Consider the network in Figure 2. Packets are traveling from S to T via routers R_1, R_2 or R_3, and R_4. For the first packet R_1 writes in slot 1 that it received the packet on interface 0. R_2 does not write anything, and R_4 records in slot 0 that it received the packet on interface 0. Packet 2 travels via the same route. This packet is stamped by R_2 in slot 0. R_4 records that it received the packet from R_2 in slot 0 as well, overwriting the previous stamp. A third packet goes through R_3. It is stamped by R_1 in slot 0 and by R_4 in slot 1. Section 5 describes how the information recorded in the headers can be used to identify the source of a *DoS* attack.

3.2 Choosing the value of p

Let $r_{\max}$ be an estimate of the maximum number of routers any packet encounters. Let s be the number of slots in the IP header where stamps are stored. Let p be the probability that a router stamps a given packet. Consider a packet traveling through a router R, and suppose that the packet travels through $r-1$ subsequent routers before arriving at its destination. The probability that the packet has R's stamp when the packet reaches its destination, $P(E_1)$ is

$$P(E_1) = sp(1-p)^{r-1} \tag{1}$$

We assign $p := 1/r_{\max}$. Suppose $r_{\max} \geq 2$. Then $1 - 1/r_{\max} \geq \exp(-1/(r_{\max}-1))$, where $\exp(\cdot)$ denotes exponentiation with base e. Hence the probability (1) is at least

$$(s/r_{\max}) \exp(-(r-1)/(r_{\max}-1)) \tag{2}$$

which, since $r \leq r_{\max}$, is at least $s/er_{\max}$.

For example, let $r_{\max}$ be 30 [6] for the purpose of analysis. Let s be 4, which is considerably smaller than the number of slots used by the *Record Route* option of the IP header. Consider a given packet and the path it takes, and let R be the router immediately following the last non-compliant router. The probability is at least .048 that R's stamp remains when the packet reaches its destination. Thus about 5% of the packets from a given source will contain the IP

Cloudflare, Inc. Exhibit 1007 Page 2

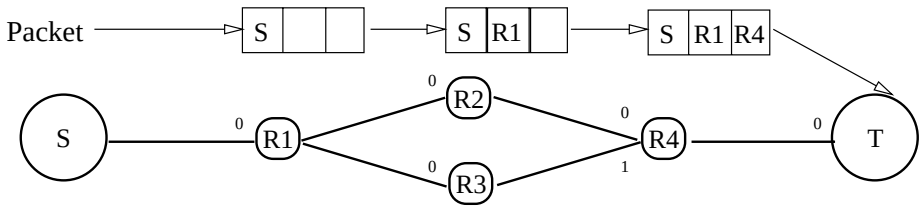

Figure 1: The packet travels from S to T via routers R_1, R_2 and R_4. Each router records the IP address of its predecessor before it forwards the packet.

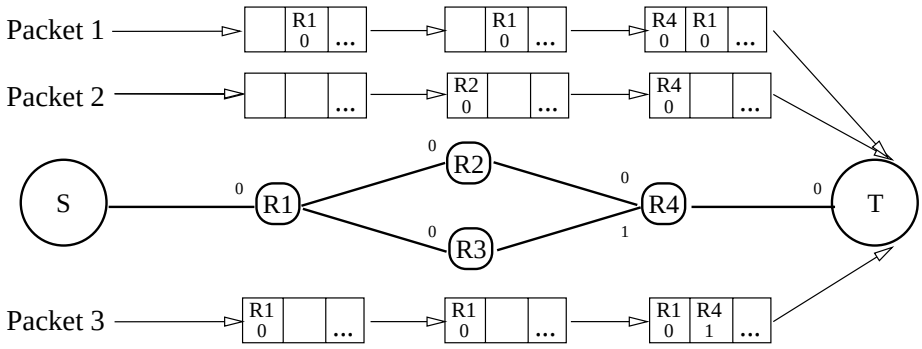

Figure 2: Packets 1, 2, 3 travel from S to T. They are processed by R_1, R_2, R_3 and R_4 which implement *Router Stamping*.

address of the last non-compliant router. In other words, in expectation only about 20 packets from a source are required to obtain such a stamp.

4. COSTS OF ROUTER STAMPING

In order to implement *Router Stamping*, it is necessary to increase the header size of IP packets. This translates into a larger per-packet overhead, requiring a longer amount of time to send the same amount of data. In addition, large headers will adversely affect some routers that have been optimized to handle a 40-byte combined TCP/IP header. The tradeoff between the space requirements and algorithm performance is discussed in Section 4.1.

Furthermore, the algorithm itself requires a slightly greater per-packet processing time. Although we do not believe that *Router Stamping* will add a significant overhead to packet processing, we discuss these costs and methods to minimize them in Section 4.2.

4.1 IP Header Costs

It is necessary for $P(E_1)$ to be large enough to handle *DDoS* attacks that employ few packets per computer. Since $P(E_1)$ depends on the value of s, it is important that enough space is allocated in the packet header to store the IP addresses. Space for eight slots adds 40 bytes to the IP header, since each slot contains a 4-byte IP address and 1-byte interface number. If a reasonable-sized packet of 1500 bytes is sent over the network, 40 bytes create an additional overhead of only 2.7%.

However, this overhead may affect some high-throughput, real-time traffic that uses small packets. Some compromise between security and speed could be reached, if necessary. Table 1 shows the tradeoff between the number of bytes allocated to router stamps and the number of packets needed to identify the source correctly.

4.2 Computational Load on Routers

In addition to the costs associated with a larger header, there is a slight increased cost for every router for the processing of each packet. In the worst case the load on a router is not greater than is currently placed by the Record Route option in IP packets. In fact the load is lower, since if a packet travels through 30 routers and each has the probability of $sp = (8)(.033) = .26$ of placing its stamp, only eight in 30 routers should modify the header. To further lower the cost of stamping, we suggest that the s slots not be included in the computation of the TCP and IP checksums. This would eliminate an additional expensive and unnecessary computation at the routers.

5. DETERMINING THE SOURCE OF AN ATTACK

In order to implement a successful defense against a *DoS* attack, it is necessary to identify its source. This is not always an easy task, as it may be difficult to separate malignant packets from desirable ones. In order to combat this kind of attack, sophisticated intrusion detection tools that use behavior-based approaches should be employed in combination with the algorithm presented in this section. For many types of attacks, though, it is easy to identify and isolate problem-causing packets, even though it might be difficult to defend against them using current methods. Due to space limitations we concentrate on the *TCP SYN Flooding Attack* as our example [4].

5.1 TCP SYN Flooding Attack

The SYN Flooding Attack consumes scarce resources on the server and denies service to legitimate users by exploiting

Cloudflare, Inc. Exhibit 1007 Page 3

Space s	$P(E_1)$	Expected number of packets needed to obtain at least one occurrence of E_1	Number of packets needed to obtain at least one occurrence of E_1 with 99% probability
1	0.0121	83	380
2	0.0241	41	189
3	0.0362	28	125
4	0.0482	21	93
5	0.0603	17	74
6	0.0723	14	61
7	0.0844	12	52
8	0.0964	10	45
9	0.1085	9	40

Table 1: Tradeoff between space allocated for routers' stamps and the performance of the algorithm. The table is computed for a route of length 30, so $p = .033$.

a weakness in the TCP protocol. An attacker bombards the server with SYN packets. The server follows the TCP protocol and responds to each with a SYN-ACK packet, allocates resources for a connection, and waits for an ACK from the client. The connection times out and the resources are freed only after 3 minutes, so a large number of malicious packets quickly consume all resources available on the server.

What makes it particularly difficult to defend against this attack is that each malicious SYN packet arrives bearing a different forged IP address. Hence, the server cannot tell whether any given packet is a valid request for a connection, and trustingly allocates resources for each one. However, it is reasonable to assume that if many half-open connections have been waiting for an ACK from the client for a long time, then the system is under a *TCP SYN Flooding Attack*. In all likelihood the majority of the half-open connections have been caused by the attack.

If *Router Stamping* is used, the system administrator can log the packets that caused the half-open connections. Once enough packets have been processed, their source router can be identified using the stamps collected by *Router Stamping*. Of course, it is possible that some of the stamps contained in the packet have been forged by the attacker. Some of the fakes might even form routes to innocent networks. For the purpose of this algorithm, however, we will treat all stamps as valid. A method for coping with forged stamps is presented in Section 6.1.

For the purpose of the algorithm it is necessary for the attacked system administrator to possess a "map" of the Internet that would show the which routers neighbor other routers. This map is easy to obtain for a small network if OSPF [10] is used. For information further from the source, BGP tables [11] may be used to obtain this information. However, currently it is not always possible to obtain all the necessary table entries to build a map of the Internet. To fill in the gaps in knowledge a system administrator can build his own map of the Internet using traceroute [9].

5.2 A Simple Source Identification Algorithm

The following simple algorithm attempts to determine the likely sources of the attack using the headers of malicious packets. The algorithm accomplishes this task by looking for routers that had little incoming traffic, as evidenced by the absence of their stamps, while their neighbors recorded a large amount of traffic from this node.

1. For each R_i
2. $\quad x \leftarrow$ the number of packets with R_i's stamps
3. $\quad y \leftarrow 0$
4. $\quad$ For each neighbor of R_i, R_j
5. $\quad\quad y \leftarrow y +$ the number of packets routed to R_j from R_i that bear the stamps of R_j
6. $\quad$ If $y \gg x$
7. $\quad\quad R_i$ is probably a source of malicious packets

5.3 Example of Source Identification

Suppose the network T mentioned in Section 3.1 is flooded with a large number of SYN packets that cause half-open connections. Suspecting a *TCP SYN Flooding Attack*, he identifies a set of 1000 connections that could be the result of the attack. Suppose that analysis of the packet headers shows that 260 of them were stamped by R_4, 160 of which were received from R_2 and 100 from R_3. R_2 stamped 130 packets all of which were received from R_1, while R_3 stamped 95 packets. Finally R_1 stamped 200 packets all of which were received from S.

Applying the algorithm presented above, the system administrator notices that R_2 and R_3 stamped as many packets as were marked by R_4, implying that they are not a source of the attack. S, however, did not stamp any packets, while its neighbor R_1 stamped 200. Hence, it is reasonable to assume that S is the source of the attack. The administrator can now implement the defensive measures described in Section 6.

6. FILTERING UNDESIRABLE PACKETS

In order to defend against a *DoS* attack, it is necessary to prevent the malicious packets from reaching their target. Current techniques attempt to filter these packets close to the target. They must either rely on the validity of the source IP address, or deny service to a large portion of the Internet.

Router Stamping relies on similar filters. Although it cannot identify the origin of each individual packet, it offers a rapid way to identify the possible sources of the attack. Hence, a filter can be made more selective by placing it as

Cloudflare, Inc. Exhibit 1007 Page 4

close to the source of the attack as possible. Of course, our approach might affect innocent networks as well, but it should deny service to a smaller portion of the Internet community than would be affected by current defensive measures.

We suggest the following algorithm for placing the filters correctly:

1. While T is under attack
2. Administrator of T identifies R_x as a possible source of the malicious packets
2. Administrator of T uses the method described in Section 6.1 to verify that the stamps identifying R_x are not being forged
3. Administrator of T places a filter on all routers connected to R_x. The routers drop all packets whose destination is T and were forwarded to them by R_x

6.1 Coping with Forged Stamps

Before blocking traffic from a given source it is important to verify that the trail from it is valid. An attacker could have put fake stamps into the packet. A smart attacker might have made these fakes create new trails to different, innocent networks. Blocking these networks is undesirable and could lead to a new kind of *DoS* attack.

In order to ensure that the stamps of some router R_i are in fact valid, the administrator of T needs to contact the router and ask it to change its stamp for all packets routed to T. Of course, the attacker should not be able to predict this change and modify his packets accordingly. Then, when the packets are received at T, the modified stamps of R_i can be identified and the packets that bear R_i's usual stamp are identified as fakes.

6.2 Example of Filtering Undesirable Packets

Suppose the attacker from Section 3.1 has added a stamp claiming that R_0 received this packet on interface 0 from some system F (Figure 3). The source identification algorithm from Section 5.2 would identify F as a possible source of the attack. The administrator of T needs to tell R_0 to modify its stamps. He tells R_0 and write Q_0 instead. When packets arrive claiming to come from F stamped with R_0, the administrator identifies them as fakes.

Now that the system administrator has identified system S as responsible for the stream of malicious packets, he requests that a filter be set up on R_1. The filter blocks all traffic from S to T, as shown in Figure 4.

Suppose, now, R_1 turns out to be noncompliant and a filter cannot be placed there. The attacks on the system continue and the administrator must, once again, identify their source. Analysis shows that it is still S. Since the filter at R_1 was ineffective, the new filters are placed downstream from R_1, at R_2 and R_3.

6.3 Security

There currently exists no automated method for placing filters on routers outside of one's network. The system administrator of T must contact the system administrator of R_x directly and ask him to set up this filter. A protocol that allows the administrator of T to set up a filter on R_x without additional human intervention is needed. One of the requirements of this protocol is that the administrator of T must authenticate himself to the router. Of course, this scheme must be designed carefully, lest it be used by hackers as a different sort of *DoS* attack.

However, an attacker's ability to abuse this system is limited. Even if the attacker can authenticate himself as an administrator of T, he is limited to placing filters that deny traffic only to T. He cannot place filters that prevent packets from reaching other systems.

Rather than using filters as designed, an attacker could attempt a *DoS* attack by placing a lot of filters on busy routers. However, we do not believe that the cost imposed by filtering would be significant. Hardware solutions that minimize the delay due to filtering already exist [7]. Even without specialized hardware, costs due to filtering should be minimal; an additional field in the routing table entry is sufficient to signify a filter. Since the routing table is already consulted for every packet, an extra comparison should not incur a great delay.

7. MOTIVATION FOR ADOPTION

When *Router Stamping* is used in conjunction with filtering, it is desirable to set up the filters as close to the source of the malicious packets as possible. The fewer routers that implement *Router Stamping*, the further from the source the filter is located, and the more innocent users that suffer. Conversely, the more routers that implement *Router Stamping*, the closer to the source one can place a filter. Since the goal of the defense is to minimize collateral damage, ISPs that prioritize customer satisfaction would be motivated to implement *Router Stamping* on their systems.

In addition to the basic algorithm provided, ISPs can choose to implement a variety of modifications that are best suited for their networks. These modifications can improve the precision of source identification or give system administrators greater flexibility in selecting the tools for combating a *DoS* attack. It is up to each individual service provider to decide whether implementation of these extensions is feasible and cost-efficient.

7.1 Leaf Networks and Router Stamping

It may be advantageous for routers of leaf networks to stamp outgoing packets with a probability of 1 rather than p. This increases the chance that a packet contains the stamp of that router, but does not affect later stamps. Thus, we permit faster and more precise identification of the source of an attack by reducing the number of packets needed to identify the leaf network.

However, this approach should be implemented only at leaf routers. If p is increased for intermediate routers, it increases the probability that an earlier stamp is overwritten and reduces the overall performance of the algorithm.

7.2 Additional Filters

It may also be desirable to implement more selective and more powerful filters that address certain specific types of attacks. For instance, one could implement a filter that drops only SYN packets that come from the compromised area. This filter would be used to defend against the SYN attack but would not affect UDP-based services for the network. However, our defense must protect against a variety of attacks, and the basic filtering algorithm presented in Section 6 still must be available.

Cloudflare, Inc. Exhibit 1007 Page 5

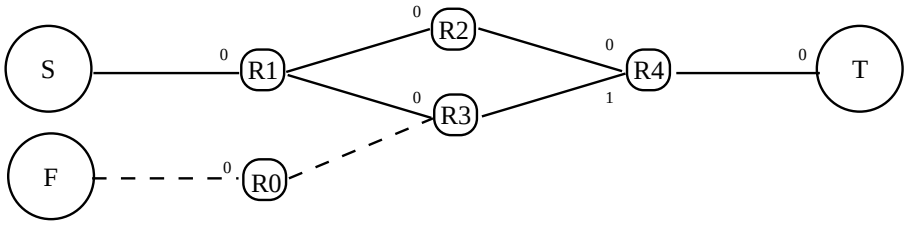

Figure 3: R_0's stamp is forged by the attacker. Before the administrator of T contacts R_0, this stamp is indistinguishable from the real stamp. The administrator tells R_0 to modify its stamp and the fake can now be identified.

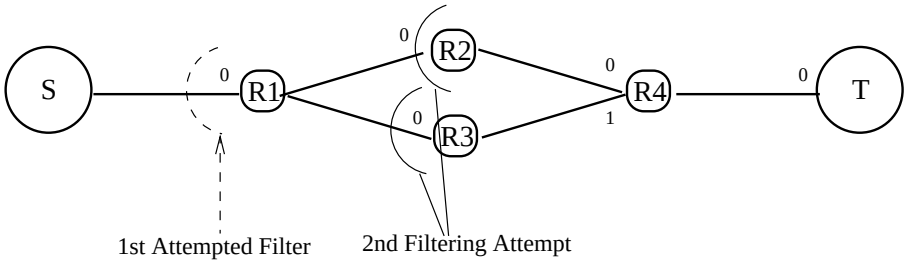

Figure 4: The administrator of T attempts to block traffic from T at R_1. When that fails, he or she attempts to set up filters at R_2 and R_3.

7.3 Using Current Defenses

Finally, *Router Stamping* can be used in conjunction with other, more expensive defense measures. Often these algorithms are too expensive to use when the majority of the traffic is benign. However, they can be very efficient in identifying and stopping certain kinds of attacks once they are underway. *Router Stamping* can be used to identify a network that is being used as a source of the attack. The appropriate countermeasures can then be started on that particular network.

For instance, say that ISP X finds that it is too expensive to implement and maintain *Ingress Routing* on its routers. A hacker connected to X uses IP Spoofing and launches a *DoS* attack on T. The administrator of T uses *Router Stamping* to identify X as a source of the attack. Rather than blocking all traffic from X to T, X turns on *Ingress Routing* [2]. This blocks all packets with spoofed IP addresses while still allowing normal users to connect to T.

8. CONCLUSION

Router Stamping offers a viable solution to help combat IP spoofing. Although it cannot stop the attacks before they occur, and can be used only once some damage has been done to the targeted system, it is inexpensive to implement and maintain. In addition it can be used effectively even if there exist some poorly maintained, noncompliant, or compromised systems. Furthermore, more advanced and more expensive algorithms can be used in conjunction with *Router Stamping* to help combat many types of attacks. We propose that *Router Stamping* should be studied in greater detail, for use as an effective tool against *DoS* attacks that employ IP spoofing.

9. REFERENCES

[1] "Internet Protocol" *DARPA Internet Program Protocol Specification*, RFC791, September 1981.

[2] P. Ferguson, "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing," RFC2267, January 1998.

[3] F. Baker, "Requirements for IP Version 4 Routers," RFC 1812, June 1995.

[4] "TCP SYN Flooding and IP Spoofing Attacks," CERT Advisory CA-96.21, September 19, 1996.

[5] I. Sager, N. Gross and J. Carey, "Locking Out the Hackers," *BusinessWeek Online*, February 28, 2000; see: www.businessweek.com/2000/00_09/b3670104.htm.

[6] C. Liu, Research on Internet Measurements; see: www2.hawaii.edu/~chunliu/ics699/699.htm.

[7] Juniper Networks, "Internet Processor II: Performance Without Compromise"; see: www.juniper.net/products/brochures/150006.html.

[8] R. Reddy, "Towards a Dependable Self Healing Internet," Input to March 8, 2000 Congressional Testimony; see www.rr.cs.cmu.edu/shn.htm.

[9] G. Malkin, "Traceroute Using an IP Option," RFC1393, January 1993.

[10] J. Moy, "OSPF Version 2," RFC2178, April 1998.

[11] Y. Rekhter, "A Border Gateway Protocol 4(BGP-4)," RFC1771, March 1995.

[12] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," ACM SIGCOM 2000, September 2000.

[13] S. M. Bellovin, "ICMP Traceback Messages," Network Working Group Internet Draft, March 2000.

[14] R. Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," UUNET, October 1999.

Cloudflare, Inc. Exhibit 1007 Page 6