ORNL/TM-2001/114

Computer Science and Mathemtics Division

Network Research Group

BACKTRACKING SPOOFED PACKETS

Tom Dunigan

Network Research Group
Oak Ridge National Laboratory
P.O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367
thd@ornl.gov

Date Published: June 2001

Research was supported by the Office of Counter Intelligence, U.S. Department of Energy.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
managed by
UT-Battelle LLC
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-96OR22464

Contents

BACKTRACKING SPOOFED PACKETS

Tom Dunigan

Abstract

This report describes techniques for backtracking internet packets with spoofed source addresses to their point of origin. Spoofed packets are often used in mounting denial of service attacks on the Internet. Algorithms for determining if routers pass spoofed addresses are also described.

1. Introduction

Each internet packet on the Internet carries a destination address and a return address. The destination address is used by the network routers in delivering the packets to its destination. The source address (or the return address) is only needed by the final destination for constructing a reply. Like US postal return addresses, the source address is ignored and can be forged or spoofed. Normal network applications provide legitimate source addresses, but a hacker can persuade an operating system (you need "root" on a UNIX system) to use spoofed addresses for the source address.

Spoofed addresses can be used to hide one's network identity or to direct return packets to another host/target. Spoofed addresses can also be used to masquerade as another host that is possibly trusted by the target host. Spoofed addresses have been used in the following Internet attacks:

- DoS –smurf attacks (ICMP broadcast)
- DoS –fraggle attacks (e.g., UDP broadcast to port 7)
- DoS –teardrop attack (IP fragmentation bugs)
- DoS –land attack (IP source = destination address)
- DoS –SYN flooding
- distributed denial of service attacks
- providing bogus packet flows to hide real attack source
- masquerading (TCP sequence number guessing)
- stealth/idle scanning

The goal of this research project was to identify current techniques that can be used to prevent or limit packets with spoofed source addresses and techniques for backtracking a spoofed packet to its point of origin. In the next section, we describe how spoofed source addresses are used in denial-of-service attacks on the Internet. In section 3, we describe administrative procedures and software for preventing spoofed packets. In section 4, we describe various methods for backtracking spoofed packets to their point (or points) of origin.

2. Spoofing source addresses

The Internet Protocol (IP) is used to carry data over the Internet and IP-based intranets. Every data packet carries at least an IP header, and often additional protocol headers in addition to the user data (Figure 2.1). IP provides a connectionless service based on datagrams, or packets. IP packets may be lost, duplicated, altered, and may not arrive in the order sent.

IP header	TCP/UDP header	application header	user data

Figure 2.1: *IP packet layout.*

Each IP packet, including fragments, carries the IP header in the first 20 bytes (or more if options are present). Figure 2.2 shows the layout of the IP header. Each packet carries the 32-bit destination address and a 32-bit source address. Each packet is routed independently using only the destination address. For more details on the function of each field refer to Stevens [10], Comer [2], or RFC791 [7].

<table>
<tr><td>version/lth</td><td>TOS</td><td colspan="2">length</td></tr>
<tr><td colspan="2">ID</td><td>flags</td><td>offset</td></tr>
<tr><td>TTL</td><td>proto</td><td colspan="2">checksum</td></tr>
<tr><td colspan="4">source address</td></tr>
<tr><td colspan="4">destination address</td></tr>
<tr><td colspan="4">options (if any)</td></tr>
</table>

Figure 2.2: *IP header.*

Internet addresses are assigned to an organization and are unique within the Internet. Host names can be associated with internet (IP) addresses, but the routing of packets uses only the 32-bit destination address. The destination and source address of an IP packet are normally filled by the operating system, but software exists for most operating systems for the hacker to insert any address into the source or destination address fields, or, for that matter, to modify any of the IP header fields. For operating systems like UNIX, such raw packet modification requires "root" privileges. The source address is used by the destination host to reply to the message, and some network services use the source address to grant or deny access to network resources (a dubious feature, since addresses can be easily spoofed).

As noted in the introduction, spoofed source addresses are used to hide one's network identity, to direct return packets to another host/target, or to masquerade as another host that is possibly trusted by the target host. Spoofed source addresses are commonly used in flooding a target host to deny service to legitimate users or in sending "poison" packets that crash or hang the target host.

Denial of service attacks prevent the targeted site from providing network services by either flooding the site with packets (smurf or fraggle attacks), consuming finite network resources (SYN flooding), or causing the site to crash (teardrop or land attacks). None of these attacks require that the source address of the attack packet be valid. By design, the attack flow often looks like legitimate traffic, so it is very difficult to filter out the attacking flow at the targeted site. Since the source addresses are spoofed, it is impossible to know what the real source of the packets is without backtracking an active flow.

In February of 2000, more sophisticated distributed denial of service attacks were launched against major Internet sites. Tools for mounting distributed denial of service attacks began appearing in the fall of 1999, and CERT and others were aware of attack preparations. The attackers broke into a number of Internet computers (usually attached to high speed nets) and installed attack agents. The attackers used these distributed agents to mount simultaneous denial of service attacks against a targeted site.

3. Preventing spoofed packets

In an ideal world, each router would be configured with ingress filters that would drop packets arriving from "internal" networks whose source address was not a member of the set of network addresses that this router serves. This is what is recommended in RFC 2267 [3]. The majority of routers could be so configured. Backbone routers and edge routers for complex topologies probably could not be configured with such filters. These ingress filters should be required as part of a "good neighbor policy." Ingress filters would not totally eliminate denial of service attacks but could greatly reduce such attacks. An attacker could still spoof an address within a local subnet, but that would permit backtracking the packets to the source subnet.

At ORNL as part of this research, we developed prototype client-server software for testing ISP router filters. The user can download this spoof-tester client (client software would be needed for each OS). The spoof-tester contacts a server (with TCP) and obtains a spoofed address for testing. The spoof-tester then transmits a series of spoofed packets (TCP, UDP, ICMP) from the user's machine to the server, some with the Record-Route option. The server then notifies the spoof-tester if the spoofed packets are detected. (The actual IP address of the user's machine is embedded in the spoofed packets.) If the spoofed packets are detected, the user or testing service could then notify the ISP. (The spoofed packets transport checksums are wrong so there are no packets reflected to the spoofed address.) The prototype used Linux, though it should port to most UNIX systems. We evaluated several ISPs with the tool and found it effective. A version for Windows requires software for transmitting raw packets. A prototype web page *http://www.epm.ornl.gov/~dunigan/oci/spoof.html* for downloading the client was also developed for the spoof-tester.

ICSA developed and deployed a similar tool based on single bootable floppy image (thus avoiding Windows/UNIX issues and avoiding downloading privileged code into the user's OS). The ICSA tool, *netlitmus* [5], tests if the user's ISP permits spoofed IP addresses (including the "internal only" IP addresses, eg. 10.*.*.*). We tested the tool as part of this research and it was effective in identifying malconfigured ISP routers. The bootable floppy must be able to identify and support the user's network interface.

Most routers are (should be) configured with egress filters that prevent spoofed internal addresses from being passed from an external interface. These egress filters reduce the risk of internal hosts using an IP address as the basis of trust decisions. SANS has some hints for testing your router configuration (see *http://www.sans.org/dosstep/index.htm*).

4. Backtracking spoofed packets

Back tracking spoofed packets is easy in principle. Assuming the attack is ongoing, one determines which router at the target site the attack flow is coming from, call it router Z. One logs into router Z, and determines from which interface/router the attack is coming from, call it router W. Then one logs into router W, and so forth. In practice, backtracking is complicated by the following:

- it is a slow manual process
- the flow must stay active during the traceback, much like tracing a phone call
- the flow may come from multiple sources and have varying signatures
- one or more routers along the path may not have the facility for identifying the upstream source
- one may need physical access to some routers to enable backtracking
- router personnel may be unavailable to assist

- eventually, one starts to cross into other administrative domains or countries and encounter legal/political obstacles

4.1. DosTracker

In 1996, MCI developed some Perl scripts for Cisco routers to follow a packet flow back through the upstream routers. The user provides the address of a starting router, the victim address and optionally a source address to dostrack. The program logs into the starting router, deploys an access control list (ACL) and places it in debug mode. Any packet that arrives with the victim subnet will be picked up by the debug statement. Dostrack analyzes the debug output to find the upstream router, spawns a process to login to the upstream router and re-initiates the same ACl/debug procedure. Thus dostrack tracks the flow upstream toward the source. The script requires password access to the Cisco routers. For this project, using ORNL Cisco routers, we verified that dostrack still works.

4.2. CenterTrack

Robert Stone's CenterTrack uses an overlay network of IP tunnels to selectively reroute suspicious datagrams from edge routers to special tracking routers [11]. The tracking router can determine the ingress edge router by noting the tunnel on which the packet arrived. The tracking router can inspect the suspicious datagram and then either drop it or forward it. The scheme permits rerouting flooding packets and can determine the ingress point on the enterprise network.

4.3. Tracer daemons

At ORNL, we developed some software agents for backtracking a spoofed IP network flow. We deployed the agents on a small testbed consisting of several layers of routers (R), tracer agents (T), target machines, and attack machines (X) (Figure 4.1).

The target machine or its attending IDS would start a trace back when an attack was detected. The trace back would proceed in parallel upstream through the routers back toward the source or sources of the attack. If the tracer agents could also control the intervening routers, then filters were applied to block the attack stream as the trace moved toward the source. The design proved effective against both single source and distributed denial of service attacks from multiple sources, all possibly using spoofed source addresses.

The demonstration system consisted of the following components:

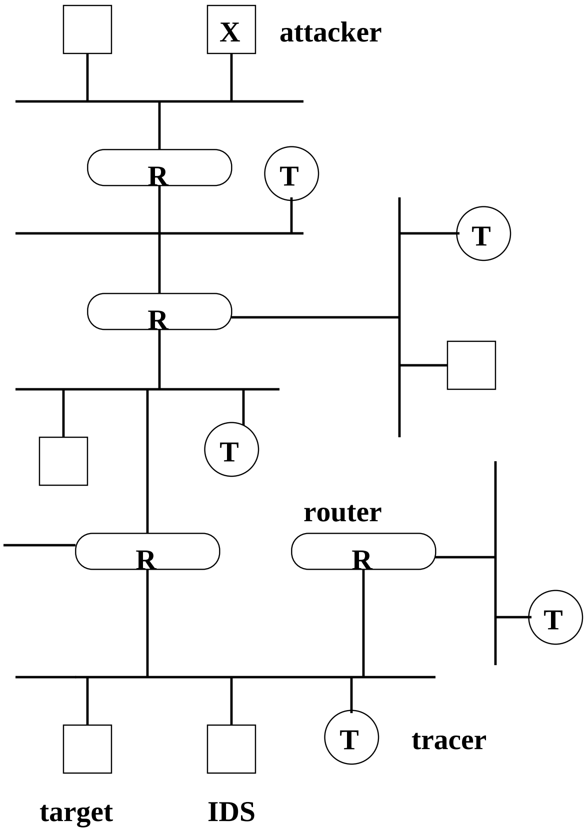

Figure 4.1: *ORNL tracer daemon test bed.*

- One or more intrusion detection systems at or near the target (attacked) site.
- master tracer process that interacts with person or IDS systems to provide detector pattern to first layer tracers
- routers – some controllable by tracers
- tracer processes on outside of routers
- detector processes spawned by tracers with filter to detect attack
- one or more target processors and one or more attack processors

Each tracer agent knows the network topology (at least to one hop) so it can control the downstream router and spawn tracer agents on the segments of the upstream router(s). Tracer processes are spawned with *rsh/ssh* from the target site or from downstream tracer processes. The tracer processes have a timer for detecting the attack pattern and will die off after the timer expires.

The detector agents are small intrusion detection processes based on the UNIX *libpcap* facility [6]. They are forked off by the tracer process with the attack filter as an argument. If the detector detects the attack signature it notifies the tracer process. The tracer process can optionally try to establish a filter on the downstream router and then spawn upstream tracer processes.

In this proof-of-principle test system, the master tracer process can request that it be notified of success/failure by each tracer process, or it can operate in "stealth mode" where the tracer processes work silently. The master tracer can also specify whether the tracer processes are to attempt to set filters on the routers when the attack pattern is detected. The testbed used Linux and the footprint of the tracer and detector processes was only 64 KB. Routing was done with Linux boxes and *ipchains/ipfwadm* was used for setting router filters. The system was tested with single and multiple attack sources using various attack methods – port scans, fraggle, smurf.

Ideally, one would like these tracer and detector processes to be integrated with the router. Or at a minimum, the routers should have some easy way of backtracking spoofed packets. Since the routers in an attack path may cross enterprise or national borders, cooperation is also required to backtrack spoofed packets. For an enterprise network, it might be possible to have router segments where these tracer/detector agents could be deployed. Other extensions to be considered.

- dynamic topology discovery by tracers
- scripts for controlling other routers (Cisco, Bay, 3com, etc)
- start with enterprise edge routers
- remove filters from routers that are covered by upstream filters
- filter removal when attack has ceased

4.4. IP packet marking

Savage et al. [9] describes current techniques to block or trace spoofed IP packets. They propose having routers randomly mark IP packets with an edge identifier. The identifier would be the two endpoints of one hop in the path of the packet. Given enough packets from an attack flow, the victim can reconstruct the attack path from the marked packets. They propose encoding the information in the IP ID field of an IP packet. Routers would need to be modified to support such a marking scheme.

4.5. ICMP traceback messages

In the summer of 2000, the Internet Engineering Task Force (IETF) launched a working group to develop ICMP Traceback Messages, which would let network managers discover the path that packets take through the Internet [1]. Routers would be modified to randomly (for example, one traceback message for every 20,000 packets) generate a traceback message about a packet and send it to the packet's destination. Each traceback message would provide authenticated information about the packet being traced, what time it was sent, where it came from, where it went. With enough traceback messages from enough routers, a network manager could find the source of a spoofed flow. Of course, this would require that the Internet routers would have to be modified to support the new ICMP traceback. The IETF hopes to have a draft standard by January, 2001.

4.6. Deciduous

The DECIDUOUS technology ([8]) is based on emerging IPSec standards and uses IPSec security associations (SA's) built with Authentication Header (AH) protection. The premise of the approach is that if an attack packet has been correctly authenticated by a certain router, the attack packet must have transited that router. Therefore, iteratively building security associations with routers at increasing distances from the victim will allow one to perform a secure traceroute that will trace the attack packet to the router closest to it, even if the attacker used spoofed IP addresses or if the router itself was compromised.

4.7. Active networks

Though mainly of academic interest at the present, active networks allow their users to inject customized programs/packets into the routers/nodes of the active network. Several university research papers describe using active networks to respond to spoofed source address flows and denial of service attacks. Halbig [4] describes an active network implementation that injects capsules into the network when a target detects an attack. The capsule instructs the adjacent routers to look for the flow and to then modify the upstream router of the flow to do the same. The result is that the flow can be traced back to its point of origin in the active network. The capsule can also instruct the routers to drop the offending flow as it works its way upstream to the point of origin. Van [12] injects filters into the active network to block attack streams. Van's filter also "pushes" itself upstream toward the source one hop at a time.

5. Summary

The threat from spoofed source address packets can be reduced if network administrators would configure ingress and egress filters in their routers. There are tools available from ICSA (and our prototype spoof-tester) that an end user can use to verify that proper filters are deployed.

Backtracking spoofed packets can be done within an enterprise, but manual backtracking is slow and people-intensive. Automated backtracking tools (DosTracker, CenterTrack, tracer daemons) are available but are vendor or enterprise specific. Tracking packets across administrative domains with these tools is still problematic. Extensions to router software to facilitate backtracking (like ICMP traceback messages) would eliminate most of the technical and administrative hurdles, but modifying all of the Internet's routers will not happen quickly. IPsec authenticated-packets would reduce the threat of spoofed packets penetrating past the network layer, but authenticating a flood of spoofed packets could still slow down a targeted host or router.

6. References

[1] S. Bellovin. "ICMP Traceback Messages". *IETF draft*, March 2000. URL: http://www.ietf.org/internet-drafts/draft-bellovin-itrace-00.txt.

[2] Douglas Comer. *"Internetworking with TCP/IP"* . Prentic Hall, 1988.

[3] P. Ferguson and D. Senie. Network Ingress Filtering. *RFC 2267*, January 1998.

[4] Gregory Halbig. "An Active Network Approach to defending and tracking denial-of-service attacks". UoF Master's Thesis, 1999. URL: http://www-pub.cise.ufl.edu/ ghalbig/proposal.html.

[5] ICSA. "NetLitmus", 2000. URL: http://www.icsa.net/html/communities/ddos/alliance/checker/index.shtml.

[6] S. McCanne and V. Jacobson. The BSD Packet Filter. *USENIX*, January, 1993.

[7] J. Postel. Internet Protocol. *RFC 791*, September 1981.

[8] Chandru Sargor. "Decentralized Source Identification for Network-Based Intrusions ", 1999. URL: http://www.anr.mcnc.org/projects/Deciduous/Deciduous.html.

[9] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. *SIGCOMM*, 2000.

[10] Richard Stevens. *TCP/Illustrated, Volume 1* . Addison Wesley, 1994.

[11] Robert Stone. "CenterTrack", 1999. URL: http://www.nanog.org/mtg-9910/robert.html.

[12] Van Van. "A Defense Against Address Spoofing Using Active Networks". MIT Master's Thesis, 1997. URL: http://www.sds.lcs.mit.edu/publications/van97.html.