Detecting Spoofed Packets

Steven J. Templeton, Karl E. Levitt
Department of Computer Science
U.C. Davis
{templets,levitt}@cs.ucdavis.edu

Abstract

Packets sent using the IP protocol include the IP address of the sending host. The recipient directs replies to the sender using this source address. However, the correctness of this address is not verified by the protocol. The IP protocol specifies no method for validating the authenticity of the packet's source. This implies that an attacker can forge the source address to be any desired. This is almost exclusively done for malicious or at least inappropriate purposes. Given that attackers can exploit this weakness for many attacks, it would be beneficial to know if network traffic has spoofed source addresses. This knowledge can be particularly useful as an adjunct to reduce false positive from intrusion detection systems. This paper discusses attacks using spoofed packets and a wide variety of methods for detecting spoofed packets. These include both active and passive host-based methods as well as the more commonly discussed routing-based methods. Additionally, we present the results of experiments to verify the effectiveness of passive methods.

1. Introduction

Packets sent using the IP protocol [22] include the IP address of the sending host. The recipient directs replies to the sender using this source address. However, the correctness of this address is not verified by the protocol. The IP protocol specifies no method for validating the authenticity of the packet's source. This implies that an attacker could forge the source address to be any he desires. This is a well-known problem and has been well described [5][10][12]. In all but a few rare cases, sending spoofed packets is done for illegitimate purposes.

Sending IP packets with forged source addresses is known as *packet spoofing* and is used by attackers for several purposes. These include obscuring the true source of the attack, implicating another site as the attack origin, pretending to be a trusted host, hijacking or intercepting network traffic, or causing replies to target another system. Because none of these are desirable, it is useful to determine if a packet has a spoofed source address. In cases where an ongoing attack is occurring it is beneficial to determine if the attack is from a particular location. In many cases we are able to determine when packets are spoofed, and generally from where they originate.

Spoofing of network traffic can occur at many layers. Examples include network layer spoofing (e.g. Ethernet MAC spoofing), non-IP transport layer spoofing (e.g. IPX, NetBEUI), as well as session and application layer spoofing (e.g. email spoofing). All of these have significant security concerns. However, for the purposes of this paper we will focus only IP packet spoofing.

A related issue is attacks that cause packets to be routed to a different host than the sender intends. These are attacks on routing [9][31] and the DNS system [4]. Packet spoofing is restricted to false source addresses in the IP packet header.

This paper discusses a variety of methods that can help determine if received packets have spoofed source addresses. Routing-based methods rely on routers and other network devices to identify traffic with unexpected source addresses or can aid spoofed packet detection. Non-routing methods include both active and passive techniques a host can use to determine if a received packet is spoofed. Active methods involve either probes, such that the response will corroborate the authenticity of a received packet, or methods that cause changes in network behavior such that the observed change (or lack of change) can corroborate the authenticity of the packets. Passive methods involve observing packet data that would be anomalous in legitimate packets.

These methods are not intended to function in isolation, rather to provide supplemental information to other IDS components or to help assess the significance of far to common nuisance alerts generated by commercial IDSs. Spoofed packet detection is an example of techniques to provide supplemental information to corroborate other IDS reports when needed.

The paper is organized as follows. Section 1 provides an overview of the various detection methods. Section 2 discusses routing methods. Section 3 discusses active methods. Section 4 discusses passive methods. Section 5 discusses work related to detecting spoofed packets. Section 6 discusses our experiment in passive detection. The last section is a discussion of the experimental results and further issues in detecting spoofed packets.

2. Packet Spoofing Attacks

Because packet spoofing can be part of many different types of attacks, it is important to have an understanding of how they are used. A key factor in all packet-spoofing attacks is that it is not necessary for the attacker to directly receive packet replies from the target. Replies are either unimportant, their contents can be inferred, or the packets can be observed in transit. This section describes several such attacks and discusses their security implications.

2.1. SYN-flood

Perhaps the archetypal denial-of-service attack, SYN-flooding demonstrates an attack where replies are irrelevant. SYN-flooding requires the attacker to continually sends a large number of TCP SYN packets to the target. For each SYN-packet received, the target host sends an acknowledgement packet to the supposed sender. The target then waits for a reply to the acknowledgment. The attacker never sends the reply, causing the target to continue to wait. This ties up a buffer on the target host When all buffers are used no further network connections to the target are possible. Eventually the waiting connection will time-out, the kernel will free the buffer and make it available for new connections. Because of the high volume of packets sent during a SYN-flood, an attacker's packets is more likely to use the buffer than a packet from a legitimate connection.

In this attack, return packets are irrelevant to the attacker. However, for this attack to be successful, the attacker must spoof source addresses from hosts that are non-existent or inactive. If the source address is of an active host, because this host did not send the initial SYN-packet, when it receives the acknowledgement packet from the target, it will reply with a reset and thus release the waiting slot.

2.2. Smurf

In the Smurf attack [8], spoofed ICMP echo request (ping) packets are sent to a subnet broadcast address. This will cause each active host to send an echo reply to the source. In this attack the source address is set to the address of the *target*. This causes a large number of replies to be sent to the target causing degradation of service on its network. The Smurf attack exploits the concepts of packet amplification and address spoofing to overwhelm the target network.

Again, seeing return packets is not important to the attacker; in fact, it is generally not desired. For this attack to be successful, the attacker must have access to a broadcast address that responds to ICMP echo requests. Unfortunately, these are widely available.

2.3. TCP Connection Spoofing

This attack requires coordination of several attacks; primarily denial-of-service of a trusted host, and packet spoofing of the attack target. The DoS component can be anything that prevents the trusted host from sending reset packets to the target. One such means would be a SYN-flood. The other component requires sending packets spoofed to be from the trusted host to the target. Because of the DoS attack, the trusted host cannot reply to packets received from the target, and the attacker can cause the target to believe the packets are from the trusted host. This will allow the attacker to use the target as if it were the trusted host.

This attack is made difficult because TCP requires reply packets to include the sequence number of the preceding packet. If the attacker cannot directly observe the packets, it must guess the sequence numbers. RFC 1948[6] provides recommendations for increasing the difficulty of predicting sequence numbers. Theoretically sequence numbers could be made unguessable. However, while more difficult than in the past, it is still possible and not as difficult as is widely believed [32].

2.4. Bounce Scan

A difficulty in scanning computer sites is that the attacker must see the replies. This makes it difficult to used spoofed addresses. The simplest way to do this is to spoof the address of another computer on your network segment and monitor network traffic for replies to the spoofed address. However, movement to switched Ethernet environments or away broadcast networks altogether makes this less feasible. A clever alternative to is to use spoofed packets and to indirectly observe the target's replies. This is illustrated by the bounce scan attack [1].

This attack takes advantage of the regular nature of the IP header "identification number" field. In most implementations, this number is increased by one with each packet sent. The bounce attack uses this by sending spoofed SYN packets to a port on the target host. If the port is closed, the target replies with a reset. The spoofed host takes no action on receipt of a reset [31]. If the port is open, the target replies to the spoofed source with an

acknowledgment. Because the spoofed host did not initiate the SYN, it sends a reset to the target, and increments its IP id number. The attack requires three steps: (i) probe the spoofed host to find its current id number; (ii) send the spoofed scan packet to the target; (iii) recheck the id number on the spoofed host. From this the attacker can learn if the target host's port was open or not: if the id number went up by one, the port was closed, if it went up by two it was open.

To ensure that other packets are not sent to the spoofed host during the scan, the attacker should select a host to spoof with little or no network traffic (e.g. a networked printer, late at night). Alternatively, or if the spoofed host does not increment id numbers by one, the attacker can uses multiple probes to each port and infer its state by profiling the observed changes in id numbers.

2.5. Zombie Control

Distributed denial-of-service attacks such as Trinoo and Stacheldraht [19], send one-way control messages to their "zombies". This allows the attacker to control the zombies without observing a reply, and consequently to be able to spoof the origin of the control messages. Also, because these are one-way communications, any protocol can be used. This makes it easier to send control messages through firewalls that may block some traffic.

Similar to the bounce scan, control messages can be sent indirectly. By sending a packet to any host on the network, but setting the TTL to be shorter than the required number of hops, when the TTL reaches zero, the router will send a TTL expired packet to the expired packets source. By spoofing this to be the address of the zombie and encoding control information in the original IP header, this method can effectively send indirect spoofed control methods to the zombies.

The above illustrates some of the many ways spoofed packets can be used in attacks. For some of these it is useful to know if the packets are spoofed or not. This can help rule out potential attack sources, prevent false attribution of the attack, estimate the level of sophistication of the attacker, the stealth of the attack, and possibly lead to discovery the true attack source. Next we describe how spoofed packets can be detected.

3. Spoofed Packets Detection Methods

Detection methods can be classified as those requiring router support, active host-based methods, passive host-based methods, and administrative methods. Administrative methods are the most commonly used methods today. When an attack is observed, security personnel at the attacked site contact the security personnel at the supposed attack site and ask for corroboration. This is extremely inefficient and generally fruitless. An automated method of determining the whether packets are likely to have been spoofed is clearly needed. This section describes a number of such methods.

3.1. Routing methods

Because routers (or IP level switches) can know which IP addresses originate with which network interface, it is possible for them to identify packets that should not have been received by a particular interface. For example, a border router or gateway will know whether addresses are internal to the network or external. If the router receives IP packets with external IP addresses on an internal interface, or it receives IP packets with an internal IP address on an external interface, the packet source is most likely spoofed.

In the wake of recent denial-of-service attacks involving spoofed attack packets, ISPs and other network operators have been urged to filter packets using the above-described method. Filtering inbound packets, known as ingress filtering, protects the organization from outside attacks. Similarly, filtering outbound packets prevents internal computers from being involved in spoofing attacks. Such filtering is known as egress filtering. It is interesting to note that if all routers were configured to use ingress and/or egress filtering, attacks would be limited to those staged within an organization or require an attacker to subvert a router.

Internal routers with a strong notion of inside/outside can also detect spoofed packets. However, certain network topologies may contain redundant routes making this distinction unclear. In these cases, host based methods (discussed in section 4.2) can be used at the router.

A number of IP addresses are reserved by the IANA for special purposes. These are listed in table 1. The addresses in the first group are private addresses and should not be routed beyond a local network. Seeing these on an outside interface may indicate spoofed packets. Depending on the particular site, seeing these on an internal address would also be suspicious. The other addresses in table 1 are special purpose, local only addresses and should never be seen on an outer interface.

Many firewalls look for the packets described in this section. Typically they are dropped when received. Because firewalls have been a popular security product, research into routing methods has been active. Most all research has been in this area.

Routers can also take a more active role in detecting spoofed packets. A number of advanced router projects have dealt with this and spoofed packet traceback [27][28]. These are discussed in section 6. We have proposed a

Table #1: Special IP Addresses

Private Networks (RFC 1918) --	
10.0.0.0/8	
172.16.0.0/12	
192.168.0.0/16	
Special / IANA Reserved --	
0.0.0.0/8	- Historical Broadcast
127.0.0.0/8	- Loopback
169.254.0.0/16	- Link Local Networks
192.0.2.0/24	- TEST-NET
240.0.0.0/5	- Class E Reserved
248.0.0.0/5	- Unallocated
255.255.255.255/32	- Broadcast

number of proactive methods that can be used to detect and prevent spoofed packets.

One limitation of routing methods is that they are effective only when packets pass through them. An attacker on the same subnet as the target could still spoof packets. When the attacker is on the same Ethernet subnet as the target, both the source IP address and the Ethernet MAC would be spoofed. If the spoofed source address was an external address, the MAC would be that of the router. This implies that other techniques are required.

3.2. Non-routing methods

Computers receiving a packet can determine if the packet is spoofed by a number of active and passive ways. We use the term active to mean the host must perform some network action to verify that the packet was sent from the claimed source. Passive methods require no such action, however an active method may be used to validate cases where the passive method indicates the packet was spoofed.

3.2.1. Active Methods

Active methods either make queries to determine the true source of the packet (reactive), or affect protocol specific commands for the sender to act upon (proactive). These methods have an advantage over routing methods in that they do not require cooperation between ISPs and can be effective even when the attacker is on the same subnet as the target.

Active methods require a response from the claimed source. Only if the spoofed host is active (i.e. connected to the network and receiving and processing packets) can it be probed. A host that is heavy firewalled and cannot respond to probes is effectively inactive. Because inactive hosts are commonly used as source addresses in spoofed packets, if these packets are seen in an attack, it is likely they are spoofed. When hosts will not respond to any probes, passive methods will be required for corroboration.

TTL methods. As IP packets are routed across the Internet, the time-to-live (TTL) field is decremented. This field in the IP packet header is used to prevent packets from being routed endlessly when the destination host can not be located in a fixed number of hops[26]. It is also used by some networked devices to prevent packets from being sent beyond a host's network subnet.

The TTL is a useful value for detecting spoofed packets. Its use is based on several assumptions, which, from our network observations, appear to be true.

- When a packet is sent between two hosts, as long as the same route is taken, the number of hops will be the same. This means that the initial TTL will be decremented by the same amount.
- Packets sent near in time to each other will take the same route to the destination.
- Routes change infrequently.
- When routes change, they do not result in a significant change in the number of hops.

If these assumptions do not hold, the described methods may result in false positives, that is, valid packets may appear to be spoofed. However, repeated checks should not consistently violate these assumptions. In general, they will hold. This allows improved accuracy by repeating the detection method to corroborate the results. Also, other non-TTL methods may be used for further corroboration.

Direct TTL probes. By sending a packet to the claimed host that will cause a reply we can check to see if the TTL in the reply is the same as the packet being checked. If they are of the same protocol, they generally have the same TTL. Because different protocols use different initial TTLs, when the probe packet is of a different protocol, we must infer the actual hop count. Only a few initial TTL values are commonly used. For TCP/UDP, 64 and 128 are most commonly seen. ICMP commonly uses 128 and 255 as the initial value. By subtracting the observed TTL from the supposed initial value we can estimate the number of hops. For example, for an ICMP packet with an observed TTL of 241, we get 255-241 or 14 as the estimated number of hops. If we are checking a TCP packet with an observed TTL of 50, we get 128-50=78 and 64-50=14. Because 14 is the expected value, we can assume the packet was not spoofed. If we knew the actual initial TTL for the host this would be more certain. Using information about a particular host is discussed in the section on passive methods, but it should be noted here that combined methods are feasible and will result in better detection.

If the attacker happened to be the same number of hops from the target as the spoofed source, this method would result in a false negative. Similarly, if the attacker knew the number of hops between the spoofed host and target, it

may be possible to spoof the TTL field as well. Issues related to this are discussed in section 9.

IP Identification Number

As discussed in the section on Bounce Scanning, the sending host increments the Identification Number (ID) in the IP header with each packet sent. Because this is a value that is easily probed and changes in its value are predictable, we can use it to determine if a packet is spoofed. Unlike TTL values, IP ID numbers can be used to detect spoofed packets even when the attacker and the target are on the same subnet.

If we send probe packets to the claimed source and we receive a reply, the ID values should be near the value of questionable packets recently received from the host. Also, the ID values observed in the probe should be greater than the ID values in the questionable packets. If not the packets were likely not sent by the claimed source. If the host associated with the claimed source is very active, the ID values may change rapidly. To be effective, the probes must be done very close in time to receipt of the questionable packets.

Some systems change initial ID values using more sophisticated method than increment by or some other constant value. To avoid violating RFC 79I [22], for fragmented packet assembly, ID numbers only need be sequential for the fragments of a particular datagram. This allows for more complicated ID number usage. Two common alternatives are to use a separate counter for each packet stream, or to use pseudo-random values. The implementation challenge is to prevent overlapping existing IP data streams.

In cases where sophisticated ID number assignments are implemented, using ID numbers to detect spoofed packets may be problematic. However, if the attacker's computer does not use the same ID number creation method, probes to classify the ID numbering method used would readily show a difference. Also, some OSs exhibit quirky ID number assignment for certain protocols or services. For example, the Linux (kernel 2.4.0-2.4.4) ICMP echo request/reply packets always set the ID to zero. This defeats the simplest ID number probes, but it does facilitate more sophisticated probes.

OS Fingerprinting

The above techniques illustrate aspects of the more general task of OS fingerprinting where a series of various probes are used to identify the operating system of a particular host. Active fingerprinting refers to direct probing of a computer, while passive fingerprinting refers to monitoring traffic and comparing it to expected norms for different OSs. We can perform a limited passive fingerprint as we observe network traffic from a particular host, then by comparing this to an active OS fingerprint, we can determine if the two are likely to be the same OS. If not we can infer the packets are spoofed.

TCP Specific Methods. Spoofed TCP packets can be checked using a number of methods in addition to the IP packet methods discussed above. TCP assures reliable packet transmission. To implement this, communication between both sides of the connection must occur. This allows us to detect spoofed packets by taking advantage of the fact that the sender's spoofed packets will not respond to any packets from the receiver. TCP's primary control messages are acknowledgement packets (ACK-packets). We will discuss two methods detection methods using ACK-packets. One causes the sender to pause sending packets; the other causes the sender to retransmit a packet.

Flow Control. The TCP header includes a *window size* field. This is used to communicate the maximum amount of data the recipient can currently receive. This can also be interpreted as the maximum amount of data the sender can transmit without an acknowledgement from the recipient. This is the TCP flow control method. If the window size is set to zero, the sender should not send more data.

If the packets we are receiving are spoofed, then the sender will never see the recipient's ACK-packets. This implies that the sender will not respond to flow control. If the recipient does not send any ACK-packets, the sender should stop after the initial window size is exhausted. If it does not, it is likely the packets are spoofed. One way of implementing this check is to always send an initial window size that is extremely small. If packets received exceed this threshold, we can infer the packets are spoofed.

Because spoofing replies with the correct sequence number to multiple TCP packets may be challenging, most spoofed TCP connections do not progress past the first ACK-packet. This implies that the best chance to detect spoofed packets requires it be done in the handshake. Fortunately the TCP handshake requires the host sending the initial SYN wait for the returned SYN-ACK prior to sending its first ACK packet. By setting the window size in the SYN-ACK to zero, we can we can determine if the sender is receiving (and responding to) our packets. If the sender sends an ACK-packet with any data, we know the true source is not responding to our packets, and was likely a spoofed packet.

Packet Retransmission. TCP uses sequence numbers to determine which packets have been acknowledged. An ACK-packet communicates to the recipient that all packets it has sent, up to and including the packet with the sequence number in the packet have been successfully received. When a packet is received with an ACK-number that is less than the minimum expected, or greater than the

max expected, the packet is dropped and as a way to resynchronize the connection, a reply with the minimum expected ACK-number is sent. We can exploit these replies to probe for spoofed packets. By sending a probe packet, spoofed to be from the internal host, with an ACK-number greater than the minimum expected, we can induce a resynchronization ACK from the host being probed. If the probe receives a RST in reply, we can infer the connection was spoofed. A concern with this method is that it may lead to an ACK-storm as both sides attempt to resynchronize [17]. This method is best performed on a firewall where the probe reply could be captured. This will prevent the internal host from seeing the reply, and will prevent an ACK-storm.

Traceroute

Traceroute [26] is a widely used network tool to discover the route from the site traceroute is executed on to another. When used to detect spoofed packets, it may tell you the number of hops to the true source. Unfortunately it is very slow and generally fails when the site being checked is behind a firewall. If the firewall blocks the probing UDP packets (or the ICMP replies), the traceroute program will know only the number of hops to the firewall. However, when the firewall is more hops away from the monitored site than the true site, traceroute will return a hop count greater than expected of the questionable packet. In this case, traceroute can be useful as a detector.

Because of its performance, traceroute is a poor general technique for spoofed packet detection. However, in cases where the attacker is nearer the target than the true source site's firewalls, and the firewall will not allow probes to succeed, traceroute or similar techniques should be considered.

The issues with traceroute introduce a different method of spoofed packet detection base only on previously observed packets. Because the TTL and ID fields are set by the true source, we can learn the expected values for a particular host. Such passive methods are discussed in the next section.

3.2.2. Passive Methods

Passive methods are a logical extension of the reactive methods discussed earlier. Where observed data will have a predictable value, not relative to some prior packet, we can learn what values are to be expected and consider packets with unexpected values suspicious. Because TTL values are a function of a host's OS, the packet's protocol, and the network topology, all which are reasonably static, TTLs can be used as a basis for passive detection. Conversely, IP ID numbers, which generally have a strong relation to prior packets, do not make good candidates for the basis of a passive system. The next section describes several different passive methods and how they could be used to detect spoofed packets.

Passive TTL Methods

As discussed in section 4.2.1.1, TTL values are an indication of how many network hops exist between a packet's source and destination. By recording, over a period of time, the TTL values of distinct source IP address/protocols we can learn which values are expected from particular hosts. We believe that these are reliable, predictable values of a given IP address/protocol. (See section 7 for experimental validation of this.) This will give us a reasonable basis for identifying suspicious packets from previously observed hosts.

Our implementation of this compares observed packets to the expected TTL values for that packet. If the values were anomalous, the packet would be flagged as suspicious.

In many cases, we will receive packets from hosts not previously encountered. These will have no entry in the table. Without further information we will not be able to know if the packet's TTL values are suspicious. How to flag such packets should be left up to the particular application.

However, by taking advantage of the fact that similar IP addresses are commonly the same number of hops away from a monitoring point, we can expand the above method to predict values for previously unseen packets. In addition to learning IP address/protocol to TTL relations we can also learn IP subnet to TTL relations. The predictability based on subnets is not expected to be as high as specific IP address/protocols, but will provide additional information.

Rather than use passive methods alone, by using them in combination with reactive methods we can construct an efficient spoofed packet detection system. The reactive method can be initiated only when the packet seems suspicious. This minimizes the amount of probing required, and allows us to test packets using a number of methods. The specifics or our implementation are described in sections 5 and 7.

One of the strengths of passive TTL methods is that they are resistant to network routing attacks. These occur when packets intended for a particular host are routed to another host posing as the first. Such an attack is not strictly packet spoofing because the packets are coming from the effective IP address of the sender. However, if the network distance between the two hosts has changed, we will identify these packets as spoofed. This allows passive spoofed packet detection to also act as a routing change detector.

OS Idiosyncrasies. We have identified a number of other features that can be used to find suspicious (possibly

spoofed) packets. These include the expected source port for a TCP or UDP communication, expected ID values for certain packets, and type of service (ToS) or differential service code point (DSCP) values. The TCP window size has also been observed to be highly predictable given the source [33]. Other useful features are likely. Basically, any that is specific to a particular host, OS, NIC, etc. is a potential identifier for that host.

How useful a particular feature is depends on how predictable a particular feature is and how likely another computer will generate the same value as the claimed source. Features with values common to many computers will tend to generate false negatives while those that vary significantly will tend to generate false positives. This illustrates why these methods are best used in combination. The more useful features we use the more likely our assessment will be correct.

4. Use in Intrusion Detection Systems

Spoofed packet detection can be implemented as either an IDS sensor or as a firewall process. As a sensor, packets believed to have spoofed source addresses will generate alerts for use by the IDS. Used in a firewall, the packets can be dropped or passed but flagged as possibly spoofed. Security monitoring systems could use this in detecting attacks.

A robust and efficient spoofed packet detection process should use a combination of methods to make its determinations. The system we are constructing first determines if the packet is suspicious using passive techniques then active probes to corroborate the passive detectors prediction. A number of different probes are used to determine if the packet was spoofed. If the probes indicate the packet is not spoofed, the system will update the static classifier to include the new values. Although updating the detector could incorporate router attacks as valid packet sources, it also allows us to learn more of the existing network relations.

While it is possible to check all packets, for efficiency we see these methods being most useful as an on-demand adjunct to a primary IDS.

We are finalizing an IDS module for detecting spoofed packets. This will incorporate many of the techniques discussed above. Details are described in section 9, Current and Future Work.

5. Related Work

We found no published work discussing detection of spoofed packets. However there are a number of papers on other spoofing attacks.

ARP spoofing [30] involves sending packets with Ethernet MAC of a different host than the IP address in the packet. This will cause hosts on the local network segment to direct packets to the wrong interface on the network. Work to detect ARP spoofing has been discussed in [2].

Because SMTP does not have the provision to relate email addresses to actual users it is possible to send email that appears to come from any user desired. Email spoofing is discussed in detail in [11].

Another type of spoofing is when a filename is created which is a link to some other file. This is a common tactic in many “race condition” attacks.

Efforts to detect the source of a spoofed packet are discussed in [15] and [24].

Some firewalls use “SYN-cookies”[3] as a means of minimizing the effects of SYN-flood type denial-of-service attacks. A SYN-cookie is a cryptographically chosen initial TCP sequence number. These are based on time, source IP address, port, etc. When a SYN packet is received, rather than open a buffer for the connection, the server sends the SYN-ACK packet with the SYN-cookie initial sequence number. No information about the connection is saved. This creates a stateless handshake. When an ACK packet is received for a socket that is not active, the returned sequence number is checked to see if it is valid for SYN packets sent from that host in recent time. If the sequence number is valid, a buffer is allocated and the connection begins. If the sequence number is not valid, the packet is dropped. While SYN-cookies do not detect spoofed packets per se, they do serve to mitigate SYN-flood attacks that utilize spoofed packets.

6. Experiments

This section describes the experiments we performed to support the assumptions made earlier and to evaluate the effectiveness of various techniques at detecting spoofed packets.

6.1. TTL Predictability

We collected packet data on our research lab network for 2 weeks. During this time approximately 23,000,000 IP packets were observed. These included packets from 23,461 unique IP addresses. Of these, 110 were hosts on the local network segment. The purpose of collecting this data was to assess the predictability of TTLs and evaluate them as a means of detecting spoofed packets.

The measure of predictability we used was conditional entropy. This measures the amount of information that knowing the source IP address (and IP protocol) provides us in estimating the true expected TTL of the host.

Conditional entropy

$$H(Y \mid X) = -\sum_{x,y} P(x,y) \log P(x \mid y)$$

is a measure of unpredictability; values closer to zero are highly predictable, those numbers closer to one are highly unpredictable. Here, x is a particular IP address/Protocol, while y ranges over all possible TTL values (0..255).

We calculated the conditional entropy for each host and protocol seen and determined the mean C.E. and variance for all IP addresses, only source addresses outside our subnet, and only source addresses inside our subnet. The results are summarized in Tables 2-8. Only four IP protocols were observed: ICMP, IGMP, TCP, and UDP. Conditional entropies were very low. To eliminate the effect of hosts who connected only once, or a few times, we recalculated the C.E.s with only IP addresses included with a minimum number of observed packets: 10 packets, 50 packets, 250 packets and 500 packets. All had higher C.E.s than when IP addresses with single connections were included, however the C.E. values remained very low indicating that the TTLs were highly predictable given an IP address and protocol.

Plots of the data consistently show low values for all protocols. An interesting feature is that as the number of UDP packets received from an IP address increased, the more TTL values we observed. Examining the raw data showed that the TTL values generally clustered about one value.

We see several explanations for this. The majority of UDP packets observed appears to be streaming media. It appears that routers are forwarding these UDP packets over a different path than TCP or ICMP packet. Also, some of the IP addresses appear to be executing traceroute [26] against destination IP addresses that are behind firewalls on our subnet. Traceroute seeks to discover the network path between the originating host and the target by using the fact that a packet sent with a TTL too small to reach its target will cause the router at which the TTL becomes zero to send a "ICMP TTL expired" error message to the claimed originator. Traceroute begins by sending a UDP packet to the target with the TTL set to one. If traceroute receives a time-expired message in reply, it increments the TTL and retries. If the UDP packet successfully reaches the target host, this host will return a port-not-found message (assuming the chosen port is not active or otherwise blocked by a firewall). This probe manifests itself as a series of UDP packets with an increasing TTL value.

A few IP addresses on the UCD network, but not on the monitored subnet, had over 100 values. These packets, the result of by network experimentation, caused a small increase in C.E. values.

In any case, TTL values appear to be highly predicable and can be the basis for spoofed packet detection.

Table 2. All packets

Protocol	H mean	H-var.	# Addresses	# Packets
All	0.055759	0.029728	23461	22999999
ICMP	0.027458	0.023726	801	223341
IGMP	0	0	23	297
TCP	0.046149	0.023114	15891	20925893
UDP	0.065164	0.040655	7397	1850468

Table 3. External addresses only

Protocol	H mean	H variance	Number Addresses	Number Packets
All	0.055505	0.029731	23351	9229608
ICMP	0.026159	0.023271	780	88371
IGMP	0	0	3	26
TCP	0.046324	0.023201	15825	8857983
UDP	0.065537	0.041015	7306	283228

Table 4. Internal Addresses Only

Protocol	H mean	H variance	Number Addresses	Number Packets
All	0.109633	0.026097	110	13770391
ICMP	0.075714	0.03822	21	134970
IGMP	0	0	20	271
TCP	0.004189	0.000321	66	12067910
UDP	0.035207	0.010859	91	1567240

Table 5. Only Addresses with more than 10 packets

Protocol	H mean	H variance	Number Addresses	Number Packets
All	0.073805	0.036386	12846	22970416
ICMP	0.098512	0.121612	77	221392
IGMP	0	0	10	258
TCP	0.056556	0.027002	11371	20912732
UDP	0.155195	0.087024	1509	1834509

Table 6. Only Addresses with more than 50 packets

Protocol	H mean	H variance	Number Addresses	Number Packets
All	0.064924	0.034027	7712	22833787
ICMP	0.049445	0.024978	42	220729
IGMP	0	0	1	69
TCP	0.056169	0.028653	7284	20798541
UDP	0.152776	0.102203	441	1812386

Table 7. Only Addresses with more than 250 packets

Protocol	H mean	H variance	Number Addresses	Number Packets
All	0.060041	0.035521	2876	22338795
ICMP	0.035778	0.020212	33	219605
IGMP	0	0	1	0
TCP	0.051132	0.027288	2713	20332940
UDP	0.165818	0.175238	148	1779896

Table 8. Only Addresses with more than 500 packets

Protocol	H mean	H variance	Number Addresses	Number Packets
All	0.050635	0.031506	2306	22140140
ICMP	0.022401	0.014516	30	218560
IGMP	0	0	1	0
TCP	0.042716	0.022273	2190	20150197
UDP	0.164326	0.209436	104	1764716

6.2. Active Methods

We constructed a tool to check the TTL and IP ID for a sampling of packets received over a 2-week period. Each packet received had a 1:10 chance of being probed. This resulted in approximately 230,000 probes. These were performed immediately after observation of the questioned packet. 71 packets or 0.031% showed a difference in TTL. Difference in ID had a mean of 4.2, variance 5.8. Further analysis of active detection methods is currently under study.

7. Discussion

All the above methods have limitations however, the methods discussed significantly increase the level of difficulty required to spoof packets. They have shown themselves to be effective and in general relatively easy to implement. Only strong end-to-end authentication can prevent packet spoofing. When this is done at or above the transport layer [6][25] concern about spoofed packets affecting applications is minimal. However, this does nothing to mitigate the effects of existing denial-of-service attacks that exploit vulnerabilities in the IP stack.

In many cases the presumption is that the attack is from a spoofed source. If we always assume this, then there is little risk to the attacker of using his/her own computer. Similarly, if an attacker sought to hide his true activity amongst many spoofed packets, these methods could help isolate the true attack source.

While it is useful to detect and prevent spoofed packets, there are legitimate reasons for them. Typically these would be to test security. Example include running NESSUS [13], detect sniffers [20], or running the spoofed packet detection system described in this document. These should not lessen the value for detecting and preventing spoofed packets, as they are very rare situations.

8. Limitations

While the methods describe appear to be effective at detecting spoofed packets, they are not perfect. The intricacies of the modern computer networks can create situations that complicated detecting spoofed packets. Also, an attacker who knows that a system is being monitored for spoofed packets may craft more sophisticated packets to defeat the spoofed packet detector. This section discuses some of these problems and how we may work around them.

8.1. Asymmetric routes

Routing on the Internet is known to not be symmetric [21]. That is, packets from one host to another may be routed differently than packets in the reverse direction. Because we are comparing the reply packet to the original, we do not care if our probe was routed differently than the replies.

8.2. Redundant routes

In many cases there will be more than one routing path between the claimed source and the target. This was observed during our monitoring period. However, the variations seen were generally only 1 or 2 hops. When the number was greater (as in external UDP packets) they

were clustered about one or two values. Much greater differences are possible and might occur if significant network outages were occurring. This however would cause widespread changes in routing and would result in many packets showing anomalous TTLs. However, active probes would likely result in the same TTLs as packets recently sent.

8.3. DHCP

Computers connecting via dial-up modem, wireless networks, and non-static DSL commonly use DHCP [14] to assign their IP addresses. This results in computers having different addresses at different times. In most cases, the number of hops from such a computer to the monitored site will be the same, however IP ID number and other OS specific data would not change.

8.4. Network Address Translation

Some routers provide network address translation (NAT) as a means of obscuring the internal network and allowing multiple hosts to share a common external IP address. Unless this is done via a proxy, only the source address and checksum would be altered. The rest of the IP header would pass through unchanged. Because of this, active detection methods would work against NATed claimed sources. When multiple hosts share the same external address, anomaly detection methods may not be able to distinguish one internal host from another. Also, if the observed TTLs, IP IDs, etc. were extremely varied across the hosts being NATed, the resulting distribution of values would reduce the effectiveness of static detection.

8.5. Proxies

In cases where the claimed site is behind a firewall that proxies the questionable packets, we will be unable to distinguish any packets from sites behind the firewall. That is, an attacker behind the firewall could spoof packets from any internal host. However, the firewall could use anti-spoofing techniques and prevent this as an additional form of egress filtering.

8.6. Forged TTLs and IP IDs

Because an attacker can send a packet with any initial TTL desired, using observed TTLs as a means to determine spoofed packets might seem futile. However, for a spoofed packet to arrive at the target with the expected TTL, the attacker must (1) know the expected TTL, (2) know the number of hops from the spoofed source to the target, and (3) be able to set the initial TTL such that it will arrive with the expected value. Any other initial TTL will result in a packet that is suspect. Determining the first two may not be easy, the third requirement may be impossible.

Traceroute has been suggested as a trivial way to determine the number of hops between the spoofed source and the target. However, this will work only if the source of the traceroute and the target are on the same network path as the attacker and the target.

Depending on the initial TTL value that is expected from the target, if the attacker is further away from the target than the spoofed host, given that each hop decrements the TTL by one, the attacker may be required to start with a value greater than 255. This is not possible given the IP header format. For this reason it may be beneficial to use 255 as the initial TTL for all packets; it will make it more difficult for an attacker to provide the expected value.

Similar to forging TTLs, an attacker could also forge IP identification numbers. An attacker could first probe the claimed sources ID numbers, and then forge packets with the ID field set to values in this range.

Alternatively, by arbitrarily modifying the TTL and IP fields it would be easy for an attacker to make non-spoofed packets appear to be spoofed. These packets may be included in a large number of truly spoofed packets to conceal the attacker's true location. This would be a logical action when the attacker must receive the return packets but wishes to remain unknown. We know of no certain way to defeat this.

9. Current and Future Work

We are in the process of analyzing the effectiveness of the active and static methods discussed above as well as determining estimations of the false-positive and false-negative rates. We will present the predictability of the various methods in terms of conditional entropy, as well as a report of how many potentially spoofed packets were observed during subsequent monitoring. This report also includes the results of trial packet spoofing attacks against our research network from several different locations.

Additionally, we are constructing an IDS component integrating anomaly detection and active probes. This module monitors the network for packets that are anomalous relative to learned expected TTL values. When an anomaly occurs the active component seeks to corroborate the anomaly detection system. Questionable packets are reported as "anomalous packets" or "believed spoofed packet" depending on the results of the active component.

While the static method described here has shown to be effective, when implemented as a simple lookup table, it is useful only for detecting spoofed packets from hosts we have seen before. Currently we are working on a system that generalizes beyond previously observed packets,

making inferences about packets from previously unseen IP addresses. We assume that in general there is a relation between subnet and number of hops. This assumption is used in our classifier design to minimize error and can be used to explain the relations learned. Because we would like to know how likely the classifier's expected TTL is to be correct, the classifier will also output a certainty factor. This is based on number of supporting examples, observed entropy, and distance from known values.

As a means to empirically test the claims of this work, we are looking for sites to send spoofed packet from, and to run the data collection/spoofed packet detector from. Although our results look very good, we are collecting on a single network segment and have sent spoofed packets to the monitored segment from only a few sites. While we expect the work to be validated by broader testing, we feel such testing should be done.

It may also be possible to combine static classifiers between sites. This would allow better detection by incorporating local knowledge about network topology.

Detecting spoofed packets is only half of the solution: we need to be able to localize the true source of the packets. A number of projects have looked at this, but either required specially instrumented routers [27][15], or changes in the underlying network protocol [18]. While these are possible solutions, we feel that methods that do not have these requirements are more attractive. We believe that for some spoofing attacks, it is possible to use search techniques built upon some of the active detection methods described in this paper to accomplish this.

10. Conclusion

The original motivation for this research was our work in model based intrusion detection [29]. At issue was a lack of sensors to provide needed information to support correlation of events. Generally, these sensors required inferring something not directly in an observed packet. Some examples include, are attack packets are from the same attacker, was an attack successful, is a sniffer present, and if a packet was spoofed. We quickly found that such sensors are possible and could be used to support an IDS.

Furthermore, while investigating commercial IDSs, we observed that many of the alerts generated were false-positives and could be eliminated if corroborating information were available. The ability to know if the packets that generated the alerts were spoofed is just one example of supplemental information that would help in filtering out those alerts of low significance.

The utility of detecting spoofed packets extends beyond simple detection and assessment. When used at a firewall to detect and block spoofed packets, the discussed techniques can be used to prevent spoofed packet attacks. As spoofed packets are a common component of many attacks, detecting and possibly preventing them is an important aid to improving network security. For TCP connections, stateless methods such as SYN-cookies are invaluable. The alternative to automated methods is to ignore identification of spoofed packets or to have security staff at the attacked site contact security staff at the supposed source. In practice this is inefficient and generally fruitless.

We have shown that a wide variety of alternative techniques is available to detect spoofed packets. These can be used alone or in combination to improve detection effectiveness. They are easy to implement and have reasonable resource requirements. We acknowledge that these methods are not complete and there are cases where an attacker can still send spoofed packets undetected. No current intrusion detection method is 100% correct. That does not lessen their utility; an incremental improvement is better than doing nothing. These techniques are such an example. They are not a total solution, however they can greatly increase the ability to identify spoofed packet attacks.

11. Acknowledgement

This work is supported in part by the DARPA IA&S program under contract #F30602-00-C-0201 as part of the NAI/U.C. Davis SHIM project, and by NSA contract MDA904-01-C-1007 as part of the U.C. Davis "Model Based Scenario Intrusion Correlation" project. The authors would like to thank the anonymous reviewers for their valuable comments.

12. References

[1] antirez. New tcp scan method. BugTraq, http://www.securityfocus.com/archive/1/11581, February 2001.

[2] ARPwatch. Lawrence Berkely National Labs Network Research Group, http://ftp.ee.lbl.gov.

[3] T. Aura and P. Nikander. Stateless connections. Proc. International Conference on Information and Communications Security (ICICS'97), Beijing, China, 1997.

[4] S. Bellovin. Using the Domain Name System for System Break-ins. Proc. of the 5th UNIX Security Symposium, pp.199-208, June 1995.

[5] S. Bellovin. Security Problems in the TCP/IP Protocol Suite. Computer Communications Review, vol. 19, no. 2, pp. 32-48, April 1989.

[6] S. Bellovin. RFC 1948: Defending Against Sequence Number Attacks. http://www.ietf.org/rfc/rfc1948.txt, May 1996.

[7] H. Chang, R. Narayan, S. Wu, B. Vetter, X. Wang, M. Brown, J. Yuill, C. Sargor, F. Jou, and F. Gong. DECIDUOUS: decentralized source identification for network-based intrusions. Proc. of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, May 1999.
[8] CERT Coordination Center. Smurf IP denial-of-service attacks. CERT Advisory CA-1998-01, Computer Emergency Response Team, http://www.cert.org/advisories/CA-1998-01.html, January 1998.
[9] H. Chang, S. Wu and Y. Jou. "Real-Time Protocol Analysis for Detecting Link-State Routing Protocol Attacks". ACM Transaction on Information and System Security (TISSEC), Feb. 2001.
[10] Computer Incident Advisory Committee (CIAC) . Advisory Notice F-08 Internet Spoofing and Hijacked Session Attacks. 1995.
[11] CERT Coordination Center. Spoofed/Forged Email. http://www.cert.org/tech_tips/email_spoofing.html, April 1999.
[12] Daemon9. IP Spoofing Demystified. Phrack Magazine Review, Vol 7, No. 48, 48-14, June 1996.
[13] R. Deraison. Nessus Security Scanner. http://www.nessus.org, 1998.
[14] R. Droms. RFC 2131: Dynamic Host Configuration Protocol. http://www.ietf.org/rfc/rfc2131, March 1997.
[15] T. Dunigan. Backtracking spoofed packets. http://www.epm.ornl.gov/~dunigan/oci/back.ps, 2000.
[16] L. T. Heberlein and M. Bishop. Attack Class: Address Spoofing. Proc. of the 19th National Information Systems Security Conference, pages 371-377, October 1996.
[17] L. Joncheray. A Simple Active Attack Against TCP. Proc. Fifth Usenix UNIX Security Symposium, 1995.
[18] S. Kent and R. Atkinson. Security architecture for the Internet protocol. RFC 2401: Internet Engineering Task Force, November 1998.
[19] F. Lau, S. H. Rubin, M. H. Smith, and Lj. Trajkovic. Distributed denial of service attacks. Proc. 2000 IEEE Int. Conf. on Systems, Man, and Cybernetics, Nashville, TN, pp. 2275-2280, October 2000.
[20] L0pht Heavy Industries. Antisniff Technical Documentation. http://www.l0pht.com/antisniff/tech-aper.html, October 2000.
[21] V. Paxson. End-to-end Routing Behavior in the Internet. to appear in Proc. SIGCOMM '96, August 1996.
[22] J. Postel. RFC 791: DARPA Internet Program Protocol Specification. http://www.ietf.org/rfc/rfc791, September 1981.
[23] J. Postel. Transmission Control Protocol. RFC 793. http://www.ietf.org/rfc/rfc793.txt, September 1981.
[24] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. Proc. of the 2000 ACM SIGCOMM Conference, August 2000.
[25] C. Schuba and E. Spafford. Countering abuse of name-based authentication. Proc. 22nd Annual Telecommunications Policy Research Conference, 1996.
[26] W. Richard Stevens. TCP/IP Illustrated. Volume I – The Protocols. Addison-Wesley. 1st edition. December 1994.
[27] D. Schnackenberg, K. Djahandari., and D. Sterne. Infrastructure for Intrusion Detection and Response. Proc. of the DARPA Information Survivability Conference and Exposition (DISCEX '00), 2000.
[28] S. Staniford-Chen and L. T. Heberlein. Holding Intruders Accountable on the Internet. Proc. of the 1995 IEEE Symposium on Security and Privacy, Oakland, CA, pages 39-49, May 1995.
[29] S. Templeton and K. Levitt. A Requires/Provides Model for Computer Attacks. Proc. of the New Security Paradigms Workshop 2000, Cork Ireland, September 2000.
[30] S. Whalen. An Introduction to ARP Spoofing. http://packetstorm.securify.com/papers/protocols/intro_to_arp_spoofing.pdf, June 2001.
[31] S. Wu, H. Chang, et al. JiNao: Design and Implementation of a Scalable Intrusion Detection System for the OSPF Routing Protocol. Journal of Computer Networks and ISDN Systems, 1999.
[32] M. Zalewski. Strange Attractors and TCP/IP Sequence Number Analysis. http://razor.bindview.com/publish/papers/tcpseq.html, 2001.
[33] S. Staniford, J. Hoagland, and J. McAlerney. Practical Automated Detection of Stealthy Portscans. To be published, Journal of Computer Security. http://www.silicondefense.com/pptntext/Spice-JCS.pdf