ABSTRACT

IP spoofing is a method of attacking a network in order to gain unauthorized access. The attack is based on the fact that Internet communication between distant computers is routinely handled by routers which find the best route by examining the destination address, but generally ignore the origination address. The origination address is only used by the destination machine when it responds back to the source.

In a spoofing attack, the intruder sends messages to a computer indicating that the message has come from a trusted system. To be successful, the intruder must first determine the IP address of a trusted system, and then modify the packet headers to that it appears that the packets are coming from the trusted system.

In essence, the attacker is fooling (spoofing) the distant computer into believing that they are a legitimate member of the network. The goal of the attack is to establish a connection that will allow the attacker to gain root access to the host, allowing the creation of a backdoor entry path into the target system.

CONTENTS

1.	Introduction	1
2.	TCP/IP Protocol Suite	2
	2.1 Internet Protocol.	2
	2.2 Transmission Control Protocol	3
	2.3 Consequences of the TCP/IP Design	5
3.	Spoofing Attacks	6
	3.1 Non-Blind Spoofing	6
	3.2 Blind Spoofing	6
	3.3 Man In The Middle Attack	7
	3.4 Denial of Service Attack	11
4.	Mechanism of the Attack	12
5.	Methods to Prevent IP Spoofing	19
	5.1 Packet Filtering	19
	5.2 Filtering at the Router	21
	5.3 Encryption and Authentication	21
	5.4 Be Un-trusting and Un-trustworthy	22
	5.5 Cryptographic Methods	22
	5.6 Initial Sequence Numbering Randomizing	22
6.	Applications of IP Spoofing	23
	6.1 Asymmetric Routing (Splitting Routing)	23
	6.2 SAT DSL	24
	6.3 NAT	27
	6.4 IP Masquerade	28
	6.5 Services Vulnerable to IP Spoofing	30
	6.6 TCP and IP Spoofing Tools	30
7.	Conclusion	31
8.	References	

1. INTRODUCTION

Criminals have long employed the tactic of masking their true identity, from disguises to aliases to caller-id blocking. It should come as no surprise then, that criminals who conduct their nefarious activities on networks and computers should employ such techniques. IP spoofing is one of the most common forms of on-line camouflage. In IP spoofing, an attacker gains unauthorized access to a computer or a network by making it appear that a malicious message has come from a trusted machine by "spoofing" the IP address of that machine. In the subsequent pages of this report, we will examine the concepts of IP spoofing: why it is possible, how it works, what it is used for and how to defend against it.

Brief History of IP Spoofing

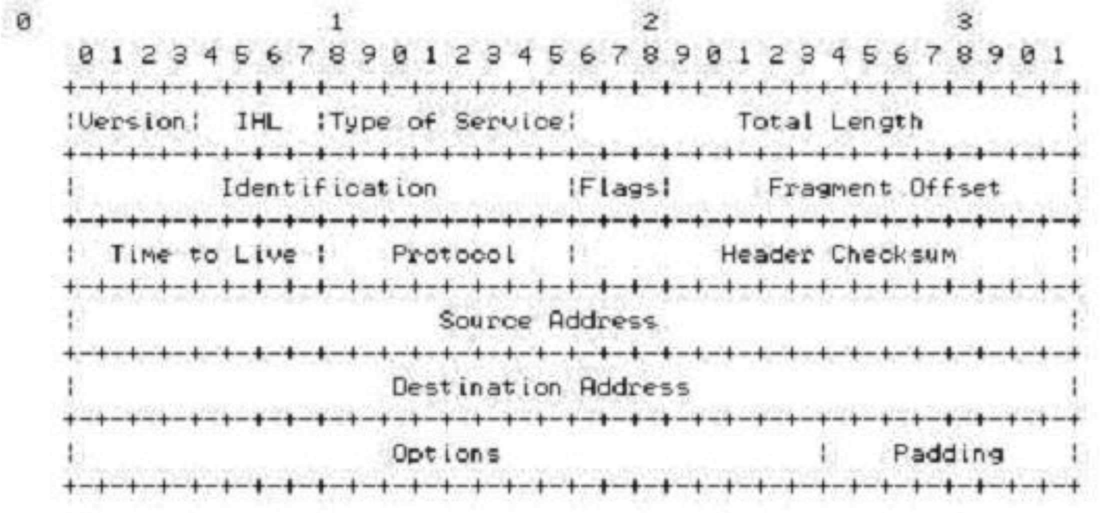
The concept of IP spoofing was initially discussed in academic circles in the 1980's. In the April 1989 article entitled: "Security Problems in the TCP/IP Protocol Suite", author S. M Bellovin of AT & T Bell labs was among the first to identify IP spoofing as a real risk to computer networks. Bellovin describes how Robert Morris, creator of the now infamous Internet Worm, figured out how TCP created sequence numbers and forged a TCP packet sequence. This TCP packet included the destination address of his "victim" and using an IP spoofing attack Morris was able to obtain root access to his targeted system without a User ID or password. Another infamous attack, Kevin Mitnick's Christmas Day crack of Tsutomu Shimomura's machine, employed the IP spoofing and TCP sequence prediction techniques. While the popularity of such cracks has decreased due to the demise of the services they exploited, spoofing can still be used and needs to be addressed by all security administrators. This is generally not true. Forging the source IP address causes the responses to be misdirected, meaning you cannot create a normal network connection. However, IP spoofing is an integral part of many network attacks that do not need to see responses (blind spoofing).

2. TCP/IP PROTOCOL SUITE

IP Spoofing exploits the flaws in TCP/IP protocol suite. In order to completely understand how these attacks can take place, one must examine the structure of the TCP/IP protocol suite. A basic understanding of these headers and network exchanges is crucial to the process.

2.1 Internet Protocol – IP

The Internet Protocol (or IP as it generally known), is the network layer of the Internet. IP provides a connection-less service. The job of IP is to route and send a packet to the packet's destination. IP provides no guarantee whatsoever, for the packets it tries to deliver. The IP packets are usually termed datagrams. The datagrams go through a series of routers before they reach the destination. At each node that the datagram passes through, the node determines the next hop for the datagram and routes it to the next hop. Since the network is dynamic, it is possible that two datagrams from the same source take different paths to make it to the destination. Since the network has variable delays, it is not guaranteed that the datagrams will be received in sequence. IP only tries for a best-effort delivery. It does not take care of lost packets; this is left to the higher layer protocols. There is no state maintained between two datagrams; in other words, IP is connection-less.

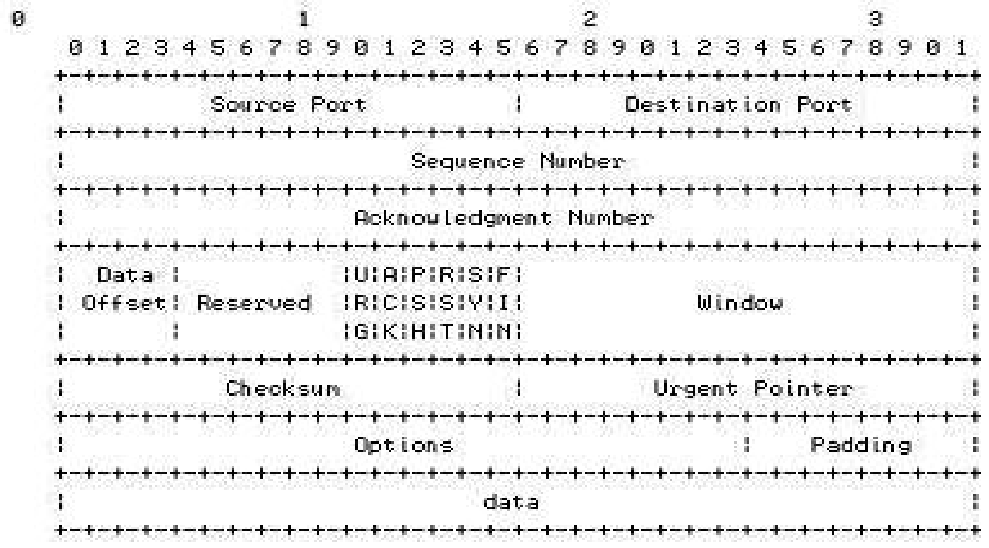


IP PACKET HEADER

The IP Header is shown above. The Version is currently set to 4. In order to distinguish it from the new version IPv6, IP is also referred to as IPv4. The source address and the destination address are 4-byte Internet addresses. The Options field contains various options such as source based routing, and record route. The source based routing allows the sender to specify the path the datagram should take to reach the destination. Record route allows the sender to record the route the datagram is taking. None of the IP fields are encrypted and there no authentication. It would be extremely easy to set an arbitrary destination address (or the source address), and IP would send the datagram. The destination has no way of ascertaining the fact that the datagram actually originated from an IP address other than the one in the source address field. It is easy to see why any authentication scheme based on IP-addresses would fail.

2.2 Transmission Control Protocol – TCP

IP can be thought of as a routing wrapper for layer 4 (transport), which contains the Transmission Control Protocol (TCP). Unlike IP, TCP uses a connection-oriented design. This means that the participants in a TCP session must first build a connection - via the 3-way handshake (SYN-SYN/ACK-ACK) - then update one another on progress - via sequences and acknowledgements. This "conversation", ensures data reliability, since the sender receives an OK from the recipient after each packet exchange.



```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Source Port          |       Destination Port        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        Sequence Number                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                    Acknowledgment Number                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Data |           |U|A|P|R|S|F|                               |
 | Offset| Reserved  |R|C|S|S|Y|I|            Window             |
 |       |           |G|K|H|T|N|N|                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           Checksum            |         Urgent Pointer        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                    Options                    |    Padding    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                             data                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP PACKET HEADER

As you can see above, a TCP header is very different from an IP header. We are concerned with the first 12 bytes of the TCP packet, which contain port and sequencing information. Much like an IP datagram, TCP packets can be manipulated using software. The source and destination ports normally depend on the network application in use (for example, HTTP via port 80). What's important for our understanding of spoofing are the sequence and acknowledgement numbers. The data contained in these fields ensures packet delivery by determining whether or not a packet needs to be resent. The sequence number is the number of the first byte in the current packet, which is relevant to the data stream. The acknowledgement number, in turn, contains the value of the next expected sequence number in the stream. This relationship confirms, on both ends, that the proper packets were received. It's quite different than IP, since transaction state is closely monitored.

2.3 Consequences of the TCP/IP Design

Now that we have an overview of the TCP/IP formats, let's examine the consequences. Obviously, it's very easy to mask a source address by manipulating an IP header. This technique is used for obvious reasons and is employed in several of the attacks discussed below. Another consequence, specific to TCP, is sequence number prediction, which can lead to session hijacking or host impersonating. This method builds on IP spoofing, since a session, albeit a false one, is built. We will examine the ramifications of this in the attacks discussed below.

3. SPOOFING ATTACKS

There are a few variations on the types of attacks that successfully employ IP spoofing. Although some are relatively dated, others are very pertinent to current security concerns. IP-spoofing consists of several steps, which I will briefly outline here, then explain in detail. First, the target host is chosen. Next, a pattern of trust is discovered, along with a trusted host. The trusted host is then disabled, and the target's TCP sequence numbers are sampled. The trusted host is impersonated, the sequence numbers guessed, and a connection attempt is made to a service that only requires address-based authentication. If successful, the attacker executes a simple command to leave a backdoor.

3.1 Non-Blind Spoofing

This type of attack takes place when the attacker is on the same subnet as the victim. The sequence and acknowledgement numbers can be sniffed, eliminating the potential difficulty of calculating them accurately. The biggest threat of spoofing in this instance would be session hijacking. This is accomplished by corrupting the data stream of an established connection, then re-establishing it based on correct sequence and acknowledgement numbers with the attack machine. Using this technique, an attacker could effectively bypass any authentication measures taken place to build the connection.

3.2 Blind Spoofing

This is a more sophisticated attack, because the sequence and acknowledgement numbers are unreachable. In order to circumvent this, several packets are sent to the target machine in order to sample sequence numbers. While not the case today, machines in the past used basic techniques for generating sequence numbers. It was relatively easy to discover the exact formula by studying packets and TCP sessions. Today, most OSes implement random sequence number generation, making it difficult to predict them accurately. If, however, the sequence number was compromised, data could be sent to the target. Several years ago, many machines used host-based authentication services (i.e. Rlogin). A properly crafted attack could add the requisite data to a system (i.e. a new user account), blindly, enabling full access for the attacker who was impersonating a trusted host.

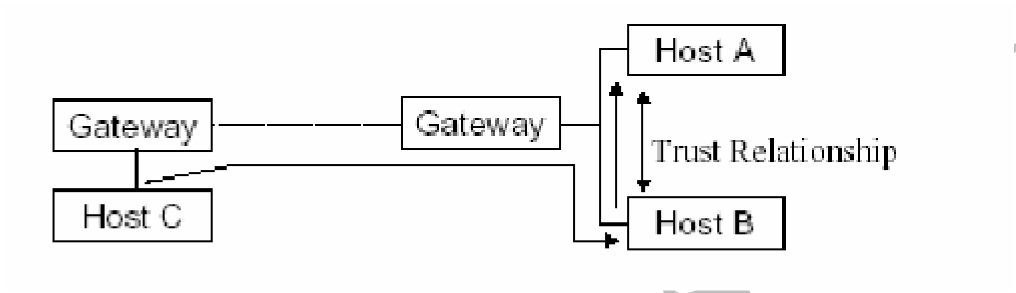


Fig. Blind Spoofing

Usually the attacker does not have access to the reply, and abuses trust relationship between hosts. For example:

Host C sends an IP datagram with the address of some other host (Host A) as the source address to Host B. Attacked host (B) replies to the legitimate host (A)

3.3 Man In The Middle Attack

Both types of spoofing are forms of a common security violation known as a man in the middle (MITM) attack. In these attacks, a malicious party intercepts a legitimate communication between two friendly parties. The malicious host then controls the flow of communication and can eliminate or alter the information sent by one of the original participants without the knowledge of either the original sender or the recipient. In this way, an attacker can fool a victim into disclosing confidential information by "spoofing" the identity of the original sender, who is presumably trusted by the recipient.

If an attacker controls a gateway that is in the delivery route, he can

- sniff the traffic
- intercept / block / delay traffic
- modify traffic

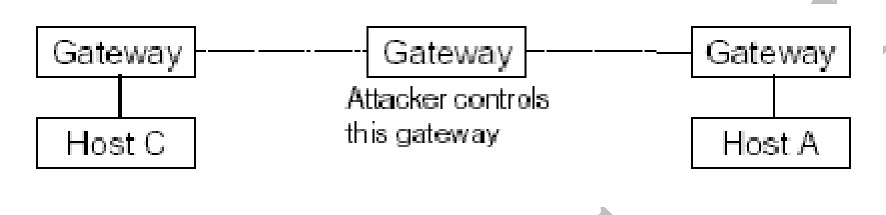


Fig. Man in the Middle Attack

This is not easy in the Internet because of hop-by-hop routing, unless you control one of the backbone hosts or source routing is used. This can also be done combined with IP source routing option. IP source routing is used to specify the route in the delivery of a packet, which is independent of the normal delivery mechanisms. If the traffic can be forced through specific routes (=specific hosts), and if the reverse route is used to reply traffic, a host on the route can easily impersonate another host. The attack procedure could be:

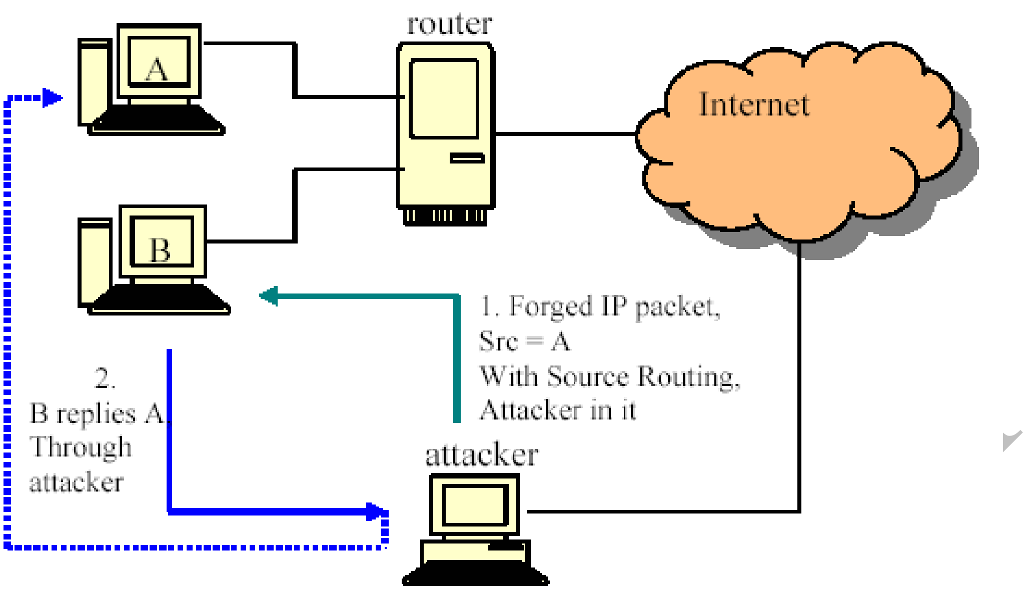


Fig. Source Routing Attacks

Connection hijacking exploits a "desynchronized state" in TCP communication. When the sequence number in a received packet is not the same as the expected sequence number, the connection is said to be "desynchronized." Depending on the actual value of the received sequence number, the TCP layer may either discard or buffer the packet. There is a choice, because TCP uses a sliding window protocol to allow efficient communication even in the presence of packet loss and high network latency. So, if the received packet is not the one expected, but is within the current window, the packet will be saved on the premise that it will be expected later (various TCP mechanisms ensure that the expected packet will eventually arrive). If the received packet is outside of the current window, it will be discarded.

Thus, when two hosts are desynchronized enough, they will discard (ignore) packets from each other. An attacker can then inject forged packets with the correct sequence numbers (and potentially modify or add commands to the communication). Obviously, this requires the attacker to be located on the communication path between the two hosts so that he may eavesdrop, in order to replicate packets being sent. The key to this attack is creating the desynchronized state. Joncheray describes two possible ways to do this: one is during the three-way handshake, and the other is in the middle of an established connection.

Note that "ignored" packets may actually generate ACKs, rather than being completely ignored. When the other end receives packets with incorrect sequence numbers, it replies with an ACK packet containing the sequence number it is expecting. But the receiver of these ACK discards them, as they have the wrong sequence numbers! The receiver then sends its own ACK to notify the sender... Thus, a large number of ACKs are generated in this attack. This "signature" of the attack could be used to detect connection hijacking.

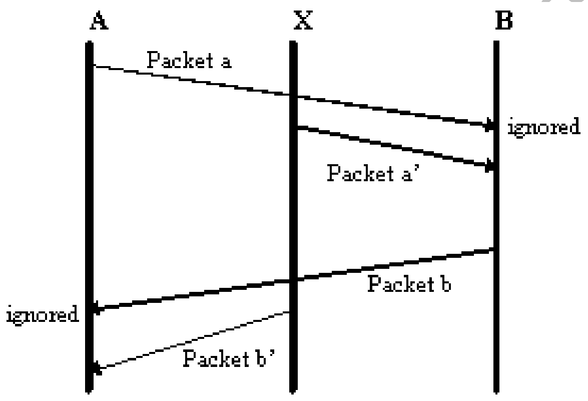


Fig. Connection Hijacking

Desynchronization during connection establishment

In this form of desynchronization, the attacker resets a connection during the three-way handshake. After host B sends the SYN&ACK packet to host A, the attacker forges new packets from B (to A) in which the connection is first closed via the RST bit, and then a new three-way handshake is initiated with A -- identical to the original, "real" handshake but with different sequence numbers. Host B now ignores messages from A (because A is using the attacker's new sequence numbers), and Host A ignores messages from B (because A is expecting messages with the attacker's sequence numbers).

The attacker then replicates new packets, with the correct sequence numbers, whenever A and B try to communicate. In doing so, the attacker may also modify the messages or inject his own.

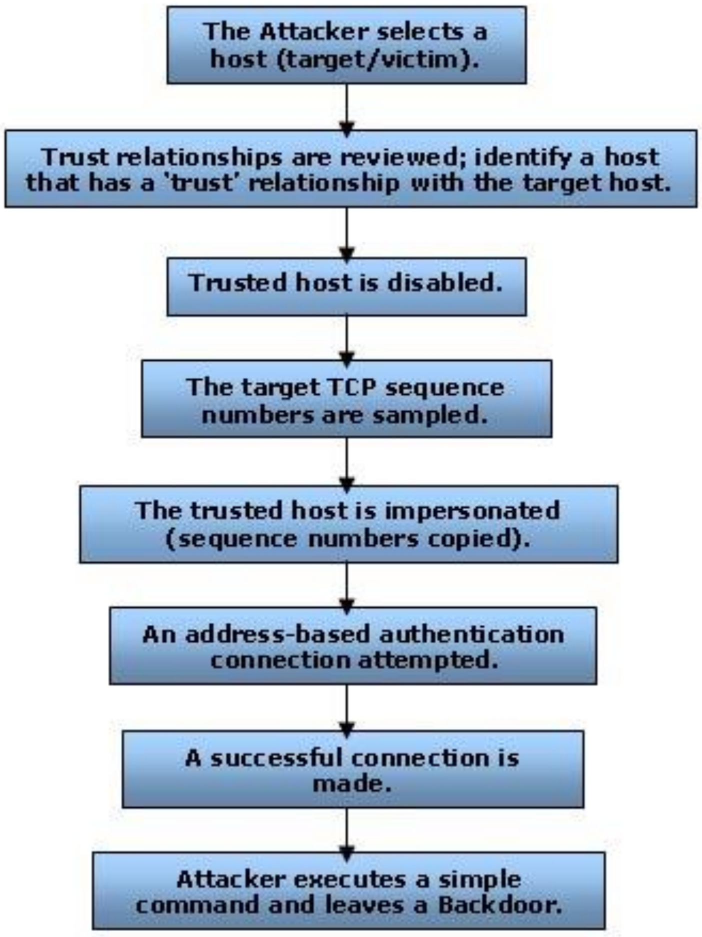
Desynchronization in the middle of a connection

The previous attack is limited to the initial connection. If a RST packet is sent in the middle of a connection, the connection is closed -- and the application/user is notified of this. To cause desynchronization in the middle of a connection, without closing the connection, only the sequence number counters should be altered. The Telnet protocol, in particular, provides an interesting mechanism to do this. Telnet allows special "NOP" commands to be sent. These commands do nothing, but the act of sending the bytes in the NOP command increments the expected sequence number counter on the receiver. By sending enough of these NOP commands, an attacker can cause the connection to become desynchronized. The attacker can then begin replicating new packets, with the correct sequence numbers, as before.

3.4 Denial of Service Attack

IP spoofing is almost always used in what is currently one of the most difficult attacks to defend against – denial of service attacks, or DoS. Since crackers are concerned only with consuming bandwidth and resources, they need not worry about properly completing handshakes and transactions. Rather, they wish to flood the victim with as many packets as possible in a short amount of time. In order to prolong the effectiveness of the attack, they spoof source IP addresses to make tracing and stopping the DoS as difficult as possible. When multiple compromised hosts are participating in the attack, all sending spoofed traffic, it is very challenging to quickly block traffic.

4. MECHANISM OF THE ATTACK

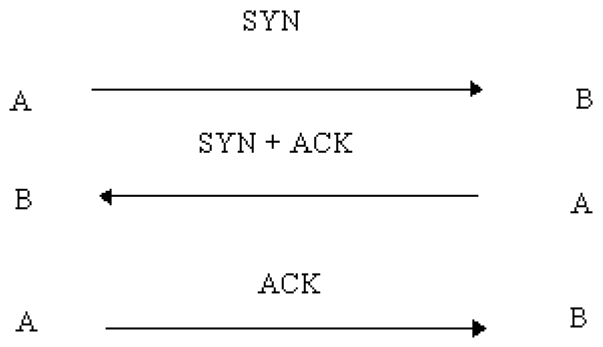

The Attacker selects a host (target/victim).
Trust relationships are reviewed; identify a host that has a 'trust' relationship with the target host.
Trusted host is disabled.
The target TCP sequence numbers are sampled.
The trusted host is impersonated (sequence numbers copied).
An address-based authentication connection attempted.
A successful connection is made.
Attacker executes a simple command and leaves a Backdoor.

Generally the attack is made from the root account on the attacking host against the root account on the target. If the attacker is going to all this trouble, it would be stupid not to go for root. (Since root access is needed to wage the attack, this should not be an issue.)

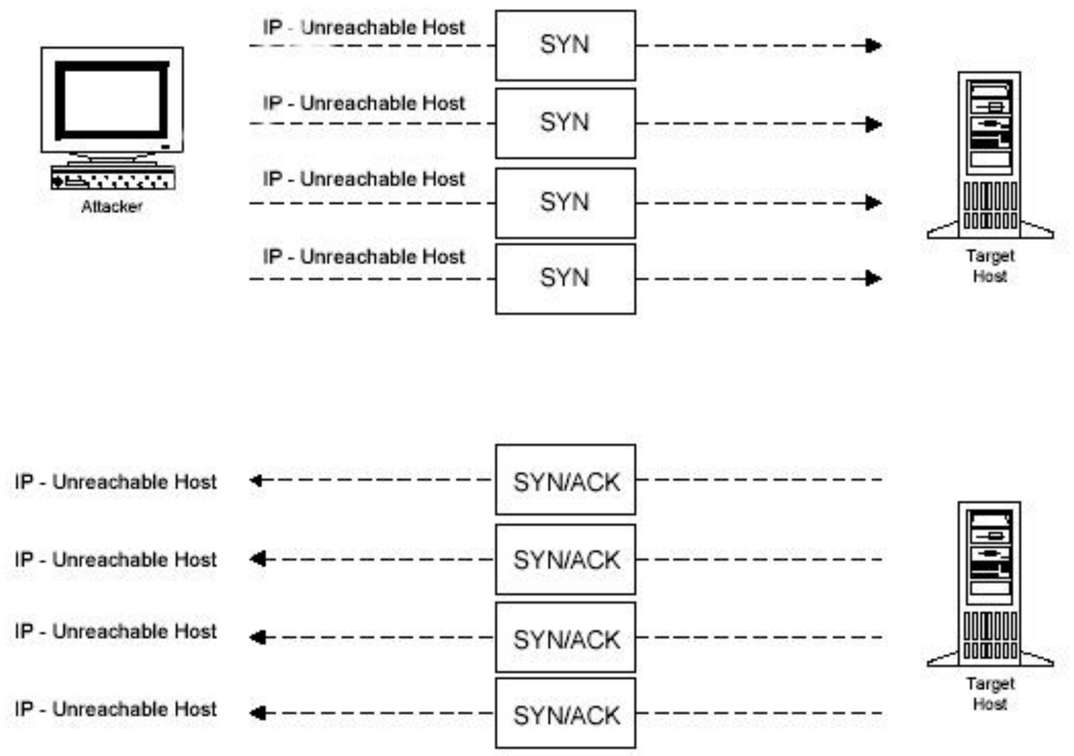
One often overlooked, but critical factor in IP-spoofing is the fact that the attack is blind. The attacker is going to be taking over the identity of a trusted host in order to subvert the security of the target host. The trusted host is disabled using the method described below. As far as the target knows, it is carrying on a conversation with a trusted pal. In reality, the attacker is sitting off in some dark corner of the Internet, forging packets purportedly from this trusted host while it is locked up in a denial of service battle. The IP datagrams sent with the forged IP-address reach the target fine (recall that IP is a connectionless-oriented protocol-- each datagram is sent without regard for the other end) but the datagrams the target sends back (destined for the trusted host) end up in the bit-bucket. The attacker never sees them. The intervening routers know where the datagrams are supposed to go. They are supposed to go the trusted host. As far as the network layer is concerned, this is where they originally came from, and this is where responses should go. Of course once the datagrams are routed there, and the information is demultiplexed up the protocol stack, and reaches TCP, it is discarded (the trusted host's TCP cannot respond-- see below). So the attacker has to be smart and *know* what was sent, and *know* what reponse the server is looking for. The attacker cannot see what the target host sends, but she can *predict* what it will send; that coupled with the knowledge of what it *will* send, allows the attacker to work around this blindness.

After a target is chosen the attacker must determine the patterns of trust (for the sake of argument, we are going to assume the target host *does* in fact trust somebody. If it didn't, the attack would end here). Figuring out who a host trusts may or may not be easy. A 'showmount -e' may show where file systems are exported, and rpcinfo can give out valuable information as well. If enough background information is known about the host, it should not be too difficult. If all else fails, trying neighboring IP addresses in a brute force effort may be a viable option.

Once the trusted host is found, it must be disabled. Since the attacker is going to impersonate it, she must make sure this host cannot receive any network traffic and foul things up. There are many ways of doing this, the one I am going to discuss is TCP SYN flooding.



A TCP connection is initiated with a client issuing a request to a server with the SYN flag on in the TCP header. Normally the server will issue a SYN/ACK back to the client identified by the 32-bit source address in the IP header. The client will then send an ACK to the server (as we saw in figure 1 above) and data transfer can commence. There is an upper limit of how many concurrent SYN requests TCP can process for a given socket, however. This limit is called the backlog, and it is the length of the queue where incoming (as yet incomplete) connections are kept. This queue limit applies to both the number of incomplete connections (the 3-way handshake is not complete) and the number of completed connections that have not been pulled from the queue by the application by way of the accept() system call. If this backlog limit is reached, TCP will silently discard all incoming SYN requests until the pending connections can be dealt with. Therein lies the attack.



The attacking host sends several SYN requests to the TCP port she desires disabled. The attacking host also must make sure that the source IP-address is spoofed to be that of another, currently unreachable host (the target TCP will be sending it's response to this address. (IP may inform TCP that the host is unreachable, but TCP considers these errors to be transient and leaves the resolution of them up to IP (reroute the packets, etc) effectively ignoring them.) The IP-address must be unreachable because the attacker does not want any host to receive the SYN/ACKs that will be coming from the target TCP (this would result in a RST being sent to the target TCP, which would foil our attack). The process is as follows:

```
1     Z(x)   ---SYN--->    B
      Z(x)   ---SYN--->    B
      Z(x)   ---SYN--->    B
      Z(x)   ---SYN--->    B
      Z(x)   ---SYN--->    B
          ...
```

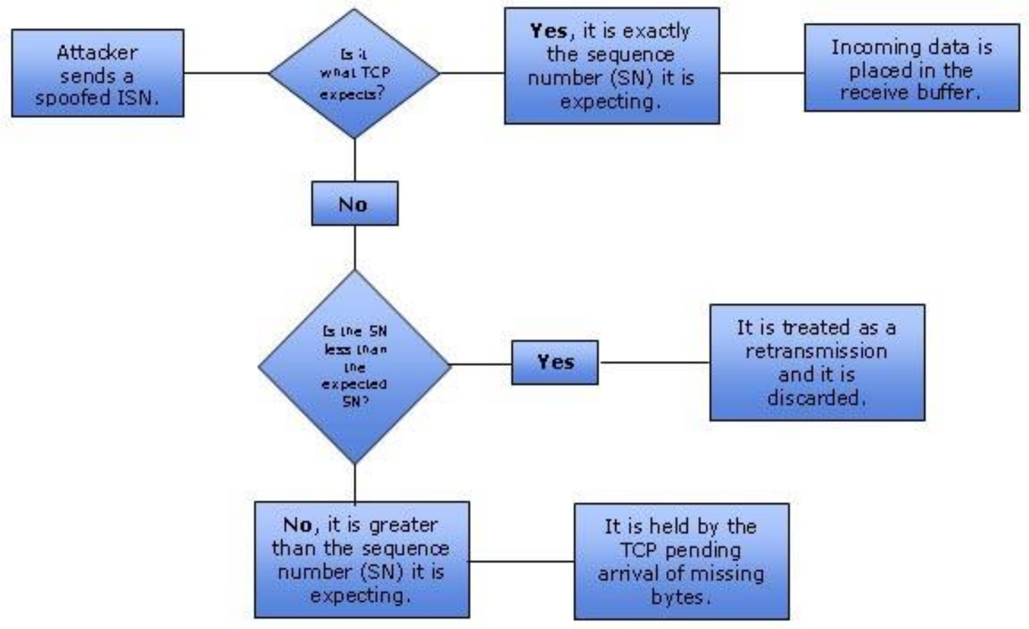
```
2      X    <---SYN/ACK---     B
       X    <---SYN/ACK---     B
                ...

3      X     <---RST---      B
```

At

(1) the attacking host sends a multitude of SYN requests to the target (remember the target in this phase of the attack is the trusted host) to fill its backlog queue with pending connections.

(2) The target responds with SYN/ACKs to what it believes is the source of the incoming SYNs. During this time all further requests to this TCP port will be ignored. Different TCP implementations have different backlog sizes. BSD generally has a backlog of 5 (Linux has a backlog of 6). There is also a 'grace' margin of 3/2. That is, TCP will allow up to backlog*3/2+1 connections. This will allow a socket one connection even if it calls listen with a backlog of 0.


Attacker sends a spoofed ISN.
Is it what TCP expects?
Yes, it is exactly the sequence number (SN) it is expecting.
Incoming data is placed in the receive buffer.
No
Is the SN less than the expected SN?
Yes
It is treated as a retransmission and it is discarded.
No, it is greater than the sequence number (SN) it is expecting.
It is held by the TCP pending arrival of missing bytes.

Now the attacker needs to get an idea of where in the 32-bit sequence number space the target's TCP is. The attacker connects to a TCP port on the target (SMTP is a good choice) just prior to launching the attack and completes the three-way handshake. In this process, the attacker will save the value of the ISN sent by the target host. Often times, this process is repeated several times and the final ISN sent is stored. The attacker needs to get an idea of what the RTT (round-trip time) from the target to her host is like. (The process can be repeated several times, and an average of the RTT's is calculated.) The RTT is necessary in being able to accurately predict the next ISN. The attacker has the baseline (the last ISN sent) and knows how the sequence numbers are incremented (128,000/second and 64,000 per connect) and now has a good idea of how long it will take an IP datagram to travel across the Internet to reach the target (approximately half the RTT, as most times the routes are symmetrical). After the attacker has this information, she immediately proceeds to the next phase of the attack (if another TCP connection were to arrive on any port of the target before the attacker was able to continue the attack, the ISN predicted by the attacker would be off by 64,000 of what was predicted). When the spoofed segment makes its way to the target, several different things may happen depending on the accuracy of the attacker's prediction:

- If the sequence number is EXACTLY where the receiving TCP expects it to be, the incoming data will be placed on the next available position in the receive buffer.
- If the sequence number is LESS than the expected value the data byte is considered a retransmission, and is discarded.
- If the sequence number is GREATER than the expected value but still within the bounds of the receive window, the data byte is considered to be a future byte, and is held by TCP, pending the arrival of the other missing bytes. If a segment arrives with a sequence number GREATER than the expected value and NOT within the bounds of the receive window the segment is dropped, and TCP will send a segment back with the *expected* sequence number.

Here is where the main thrust of the attack begins:

```
1       Z (b)    ---SYN--->      A
2       B     <---SYN/ACK---    A
3       Z (b)    ---ACK--->      A
```

```
4       Z (b)    ---PSH--->      A
```

The attacking host spoofs her IP address to be that of the trusted host (which should still be in the death-throes of the D.O.S. attack) and sends its connection request to port 513 on the target (1). At 2), the target responds to the spoofed connection request with a SYN/ACK, which will make its way to the trusted host (which, if it *could* process the incoming TCP segment, it would consider it an error, and immediately send a RST to the target). If everything goes according to plan, the SYN/ACK will be dropped by the gagged trusted host. After (1), the attacker must back off for a bit to give the target ample time to send the SYN/ACK (the attacker cannot see this segment). Then, at (3) the attacker sends an ACK to the target with the predicted sequence number (plus one, because we're ACKing it). If the attacker is correct in her prediction, the target will accept the ACK. The target is compromised and data transfer can commence (4).

Generally, after compromise, the attacker will insert a backdoor into the system that will allow a simpler way of intrusion. (Often a `cat + + >> ~/.rhosts` is done. This is a good idea for several reasons: it is quick, allows for simple re-entry, and is not interactive. Remember the attacker cannot see any traffic coming from the target, so any responses are sent off into oblivion.)

5. METHODS TO PREVENT IP SPOOFING ATTACK

5.1 Packet filtering

The router that connects a network to another network is known as a border router. One way to mitigate the threat of IP spoofing is by inspecting packets when they the leave and enter a network looking for invalid source IP addresses. If this type of filtering were performed on all border routers, IP address spoofing would be greatly reduced. Egress filtering checks the source IP address of packets to ensure they come from a valid IP address range within the internal network. When the router receives a packet that contains an invalid source address, the packet is simply discarded and does not leave the network boundary. Ingress filtering checks the source IP address of packets that enter the network to ensure they do not come from sources that are not permitted to access the network. At a minimum, all private, reserved, and internal IP addresses should be discarded by the router and not allowed to enter the network. In Linux, packet filtering can be enabled using:

```
echo 2 > /proc/sys/net/ipv4/conf/*/rp_filter
```

Limits of packet filtering

Packet filtering normally may not prevent a system from participating in an attack if the spoofed IP address used could fall within the valid internal address range. However it will simplify the process of tracing the packets, since the systems will have to use a source IP address within the valid IP range of the network.

We take a campus network as example:

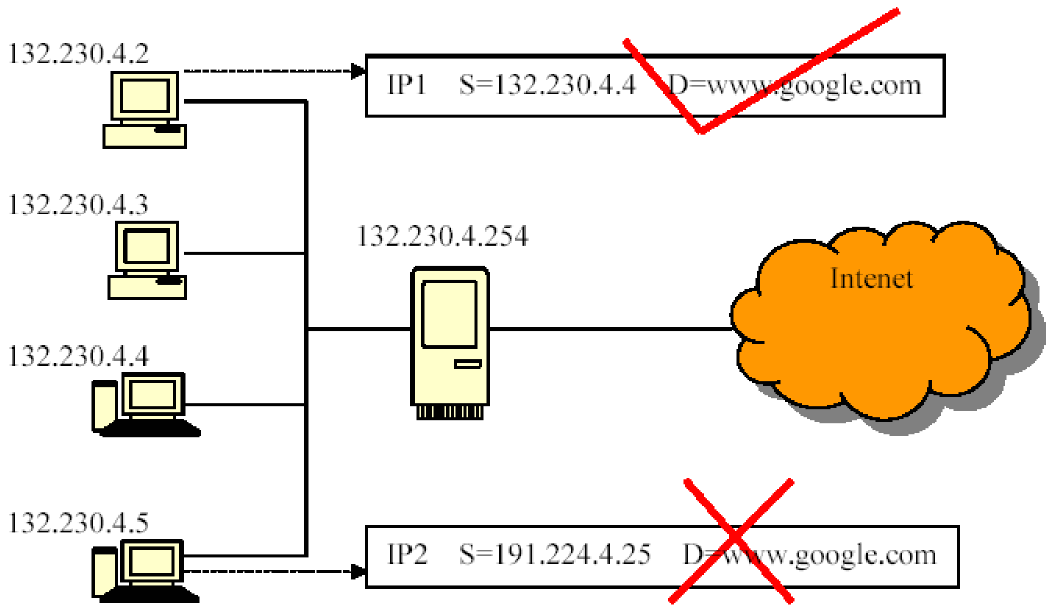

132.230.4.2
IP1 S=132.230.4.4 D=www.google.com
132.230.4.3
132.230.4.254
Intenet
132.230.4.4
132.230.4.5
IP2 S=191.224.4.25 D=www.google.com

fig. Campus Network

The network number is 132.230.0.0/16. The packet filtering of the router is enabled. For IP packet 1, host 132.230.4.1 forges a packet from 132.230.4.4, the source IP address is in the valid IP range, the router thinks it is valid packet and sends it out to internet. For IP packet 2, host 132.230.4.4 forges a packet from 191.224.4.25, the source IP address is not in the valid IP range, the router thinks it is invalid and discards it. Packet filtering can pose problems if you use splitting routing (packets from you to a host take a different path than packets from that host to you). If splitting routing is in use, enabling packet filtering facility will block all packets with spoofed source addresses. To turn rp_filter off, use:

```
echo 0 > /proc/sys/net/ipv4/conf/<device>/rp_filter
```

or

```
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
```

Instances where you *might* need to disable packet filtering include:

- If you want to do asymmetric routing (accepting returning packets inbound an interface other than the outbound interface).
- If the box has multiple interfaces up on the same network.
- If you are using special VPN interfaces to tunnel traffic (e.g. FreeS/WAN)

Another problem is that many ISPs do not have the technical ability to arrange packet filtering to block packets with spoofed source addresses. Also, packet filtering reduces equipment performance.

5.2 Filtering at the Router

If your site has a direct connection to the Internet, you can use your router to help you out. First make sure only hosts on your internal LAN can participate in trust-relationships (no internal host should trust a host outside the LAN). Then simply filter out *all* traffic from the outside (the Internet) that purports to come from the inside (the LAN).

Implementing ingress and egress filtering on your border routers is a great place to start your spoofing defense. You will need to implement an ACL (access control list) that blocks private IP addresses on your downstream interface. Additionally, this interface should not accept addresses with your internal range as the source, as this is a common spoofing technique used to circumvent firewalls. On the upstream interface, you should restrict source addresses outside of your valid range, which will prevent someone on your network from sending spoofed traffic to the Internet.

5.3 Encryption and Authentication

Implementing encryption and authentication will also reduce spoofing threats. Both of these features are included in Ipv6, which will eliminate current spoofing threats. Additionally, you should eliminate all host-based authentication measures, which are sometimes common for machines on the same subnet. Ensure that the proper authentication measures are in place and carried out over a secure (encrypted) channel.

5.4 Be Un-trusting and Un-trustworthy

One easy solution to prevent this attack is not to rely on address-based authentication. Disable all the r* commands, remove all .rhosts files and empty out the /etc/hosts.equiv file. This will force all users to use other means of remote access (telnet, ssh, skey, etc).

5.5 Cryptographic Methods

An obvious method to deter IP-spoofing is to require all network traffic to be encrypted and/or authenticated. While several solutions exist, it will be a while before such measures are deployed as defacto standards.

5.6 Initial Sequence Number Randomizing

Since the sequence numbers are not chosen randomly (or incremented randomly) this attack works. Bellovin describes a fix for TCP that involves partitioning the sequence number space. Each connection would have it's own separate sequence number space. The sequence numbers would still be incremented as before, however, there would be no obvious or implied relationship between the numbering in these spaces. Suggested is the following formula:

ISN=M+F(localhost,localport,remotehost,remoteport)Where M is the 4 microsecond timer and F is a cryptographic hash. F must not be computable from the outside or the attacker could still guess sequence numbers. Bellovin suggests F be a hash of the connection-id and a secret vector (a random number, or a host related secret combined with the machine's boot time).

6. APPLICATIONS OF IP SPOOFING

6.1 Asymmetric routing (Splitting routing)

Asymmetric routing means traffic goes over different interfaces for directions in and out. In other words, asymmetric routing is when the response to a packet follows a different path from one host to another than the original packet did. The more correct and more general answer is, for any source IP address 'A' and destination 'B', the path followed by any packet (request or response) from 'A' to 'B' is different than the path taken by a packet from 'B' to 'A'.

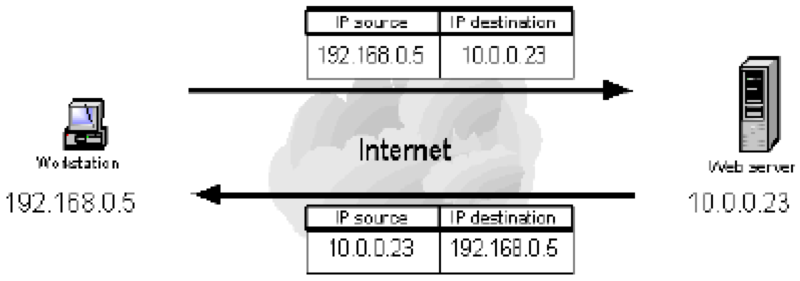


Fig. Valid Source IP Address

Implementation of asymmetric routing

Modern O.S. allows us to receive packets from an input interface, different from the output interface.

In Linux, we can implement asymmetric routing using iptables (linux 2.4):

```
iptables –A POSTROUTING –t nat –j SNAT –to 192.168.0.5 –o eth0
```

This means, for all the packets going out via eth0, their source IP address will be changed to 192.168.0.5. We also have to "disable" reverse path filtering

```
echo “0” > /proc/sys/net/ipv4/conf/all/rp_filter
```

6.2 SAT DSL

Satellite DSL (SAT DSL) makes use of asymmetric routing.

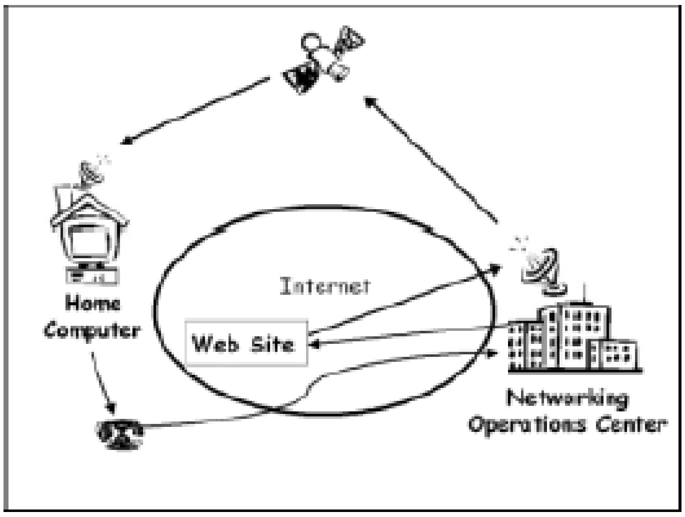


Fig. Satellite DSL

The advantage of a satellite network is to provide high bandwidth services independent of the users location over a wide geographical area. A satellite network consists of two types of stations: feeds and receivers. Every receiver has a satellite dish connected to a user station. The user station has an extra interface, DSL modem connected to the ISP, this is called return channel. All requests to Internet are sent via DSL connection, and responses from Internet should be routed by a feed on the satellite network. After the information is sent from the feed to a satellite, it will be broadcast to all the receivers that belong to the satellite coverage. Installing feeds in strategic positions over the Internet will create shorter paths and higher bandwidth provided by the satellite network. The user host has therefore two IP addresses, one for the satellite subnetwork and the other for the regular connection subnetwork (return channel).

The traffic path of satellite DSL is:

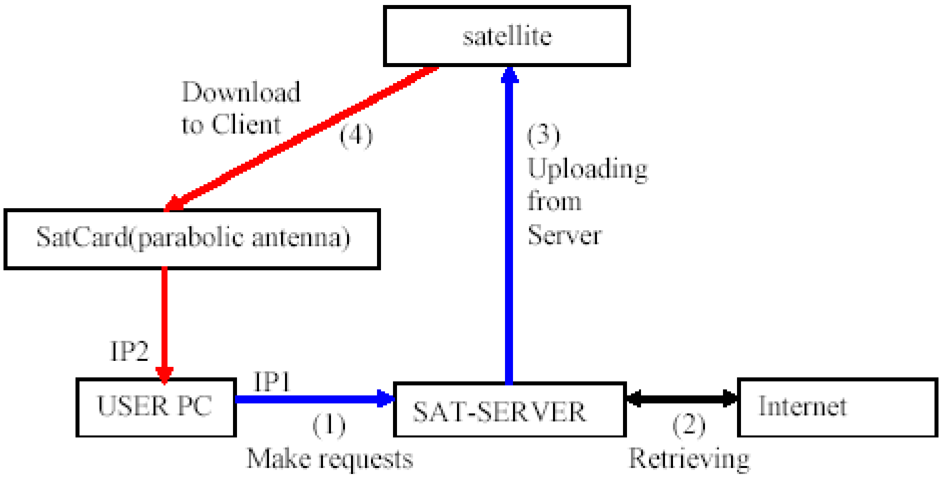


Fig. Traffic Path of Satellite DSL

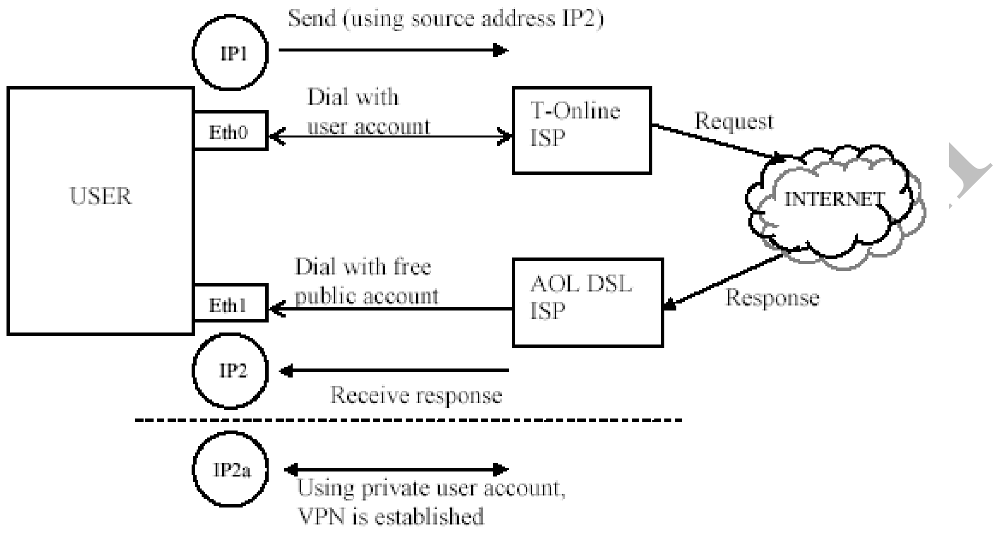
First we make the request (1) (using our Internet connection) to the Sat-Server, after it retrieves out info from Internet (2) it will send it to Satellite (3); in the end we would receive data from the satellite(4) to our home using a parabolic antenna and a Sat Card.

Probable problem with AOLs DSL connection setup

AOL DSL service implements a certain connection setup procedure in order to apply VPN (Virtual Private Network) for its users. When a user dials in to the AOL DSL ISP, these procedures are taken place:

1. User is connected to the ISP using a public account and so a network connection between user and the ISP is established. But user can only receive data using this connection, thus is not able to send any internet request.
2. On top of this connection, A VPN is established using user's private account. After the authentication succeeds, a user can send and receive data through this VPN connection. This certain procedures are AOL's attempt to create secure internet traffic over DNS connection. But as it usually is, one solution to a security problem may lead to another problem. And this applies also to AOL's

DSL connection setup. With certain setup and an **IP address spoofing** technique, a user can connect to AOL DSL ISP, and download as much data as he wants using this connection without paying any cent. This picture depicts such setup and how the attack works.



Send (using source address IP2)
IP1
Dial with user account
Eth0
T-Online ISP
Request
USER
INTERNET
Dial with free public account
Eth1
AOL DSL ISP
Response
IP2
Receive response
IP2a
Using private user account, VPN is established

Fig. Problem in AOL DSL

1. On first network interface, the user dials for a DSL connection to T-Online or other ISPs using his account. The user can send and receive data with this connection.
2. On second network interface, the user dials to AOL DSL ISP using a free public account to establish a DSL connection that goes one way from ISP to user.
3. Before the user sends packet through T-Online connection, he spoofs the source IP address of the packet into the IP address of the second network interface (which is connected to AOL DSL)
4. And so he sends requests through T-Online connection, and receives response through AOL DSL connection. This way the user only needs to pay for every bits he sends to T-Online, and get for free every bits he receives from AOL DSL, which would have cost a lot more than the cost for sending bits, because people usually spend more time downloading from the internet instead of sending data to the internet.

6.3 NAT

NAT is network address translation.

Normally, packets on a network travel from their source to their destination through many different links. None of these links really alter your packet, they just send it onward. If one of these links were to do NAT, then they would alter the source or destinations of the packet as it passes through. Usually the link doing NAT will remember how it mangled a packet, and when a reply packet passes through the other way, it will do the reverse mangling on that reply packet, so everything works.

NAT have several applications:

• **Modem Connections To The Internet**

Most ISPs give you a single IP address when you dial up to them. You can send out packets with any source address you want, but only replies to packets with this source IP address will return to you. If you want to use multiple different machines (such as a home network) to connect to the Internet through this one link, you'll need NAT.

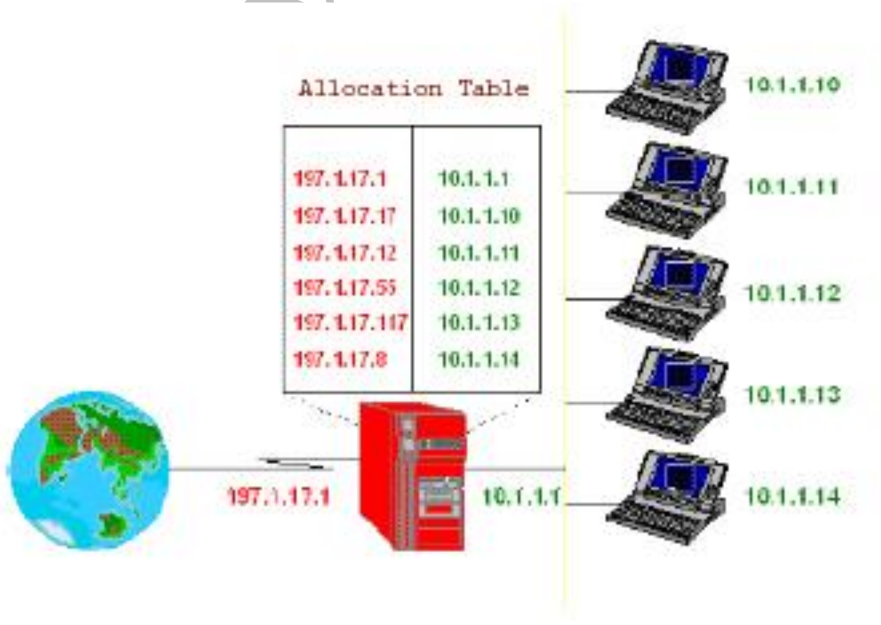
• **Multiple Servers**

Sometimes you want to change where packets heading into your network will go. Frequently this is because (as above) you have only one IP address, but you want people to be able to get into the boxes behind the one with the `real' IP address. If you rewrite the destination of incoming packets, you can manage this. This type of NAT was called port-forwarding. A common variation of this is load-sharing, where the mapping ranges over a set of machines, fanning packets out to them.

• **Transparent Proxying**

Sometimes you want to pretend that each packet which passes through your Linux box is destined for a program on the Linux box itself. This is used to make transparent proxies: a proxy is a program which stands between your network and the outside world, shuffling communication between the two. The transparent part is because your network won't even know it's talking to a proxy, unless of course, the proxy doesn't work. NAT has two different types: **Source NAT** (SNAT) and **Destination NAT** (DNAT). Source NAT is when you alter the source address of the first packet: i.e. you are changing where the connection is coming from. Source NAT is always done post-routing, just before the packet goes out onto the wire. Masquerading is a specialized form of SNAT.

Destination NAT is when you alter the destination address of the first packet: i.e. you are changing where the connection is going to. Destination NAT is always done before routing, when the packet first comes off the wire. Port forwarding, load sharing, and transparent proxying are all forms of DNAT.



197.1.17.1	10.1.1.1
197.1.17.17	10.1.1.10
197.1.17.12	10.1.1.11
197.1.17.55	10.1.1.12
197.1.17.117	10.1.1.13
197.1.17.8	10.1.1.14

Fig. NAT

6.4 IP Masquerade

IP Masquerade, is a specific form of Network Address Translation (NAT) which allows internally connected computers that do not have registered Internet IP addresses to communicate to the Internet via the Linux server's Internet IP address. IP masquerading lets you use a single Internet-connected computer running Linux with a real IP address as a gateway for non-connected machines with "fake" IP addresses. The Linux box with a real address handles mapping packets from your intranet out to the Internet, and when responses come back, it maps them back to your intranet. This lets you browse the web and use other Internet functions from multiple machines without having a special network setup from your ISP.

IP Masquerade is a networking function in Linux similar to the one-to-many (1:Many) NAT (Network Address Translation) servers found in many commercial firewalls and network routers. For example, if a Linux host is connected to the Internet via PPP, Ethernet, etc., the IP Masquerade feature allows other "internal" computers connected to this Linux box (via PPP, Ethernet, etc.) to also reach the Internet as well. Linux IP Masquerading allows for this functionality even though these internal machines don't have an officially assigned IP address. IP masquerading is different from NAT. While IP masquerading implements a specific many-to-one NAT, IP NAT allows complex many-to-many translations. For static real IP address we use NAT, while for dynamic real IP address (via PPP) we use IP masquerading.

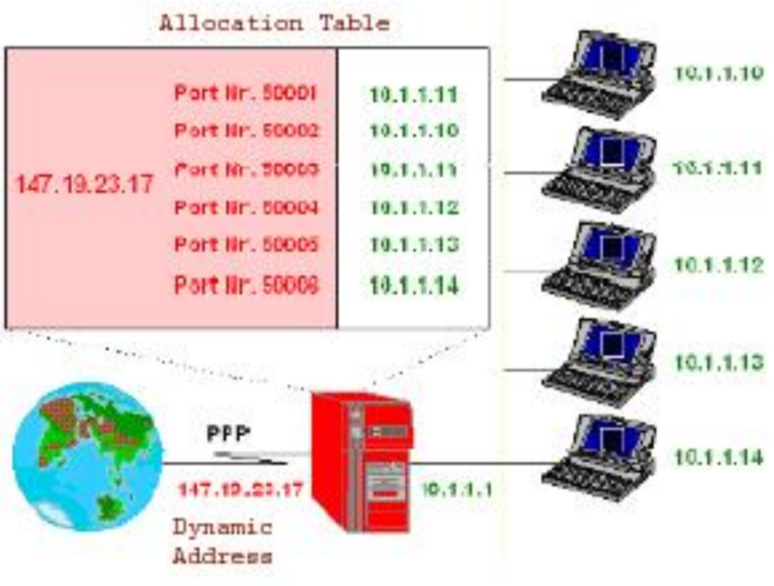


Fig. IP Masquerading

6.5 Services vulnerable to IP Spoofing

Configuration and services that are vulnerable to IP spoofing:

- RPC (Remote Procedure Call services)
- Any service that uses IP address authentication
- The X Window system
- The R services suite (rlogin, rsh, etc.)

6.6 TCP and IP spoofing Tools

1) Mendax for Linux

   Mendax is an easy-to-use tool for TCP sequence number prediction and rshd spoofing.

2) spoofit.h

   spoofit.h is a nicely commented library for including IP spoofing functionality into your programs.

3) ipspoof

   ipspoof is a TCP and IP spoofing utility.

4) hunt

   hunt is a sniffer which also offers many spoofing functions.

5) dsniff

   dsniff is a collection of tools for network auditing and penetration testing. dsniff, filesnarf, mailsnarf, msgsnarf, urlsnarf, and webspy passively monitor a network for interesting data (passwords, e-mail, files, etc.). arpspoof, dnsspoof, and macof facilitate the interception of network traffic.

7. CONCLUSION

IP spoofing is less of a threat today due to the patches to the Unix Operating system and the widespread use of random sequence numbering. Many security experts are predicting a shift from IP spoofing attacks to application-related spoofing in which hackers can exploit a weakness in a particular service to send and receive information under false identities. As Security professionals, we must remain current with the Operating Systems that we use in our day to day activities. A steady stream of changes and new challenges is assured as the hacker community continues to seek out vulnerabilities and weaknesses in our systems and our networks.

8. REFERENCES

- Following the Journey of a Spoofed Packet
  http://www.scs.carleton.ca/~dlwhyte/whytepapers/ipspoof.htm
- NAT and Networks
  http://www.suse.de/~mha/linux-ip-nat/diplom/node4.html
- Asymmetric routing - Jani Lakkakorpi
  http://keskus.hut.fi/tutkimus/ipana/paperit/QoSR/S130-QoSR-asymmetric.pdf
- TCP/IP protocol suite - Thomas Toth
  http://www.infosys.tuwien.ac.at/Teaching/Courses/InetSec/slides/slides2.pdf
- Security problems in the TCP/IP protocol suite, S.M. Bellovin, AT&T Bell Laboratories, Murray Hill, New Jersey 07974
  http://www.research.att.com/~smb/papers/ipext.pdf
- Linux 2.4 NAT HOWTO
  http://www.netfilter.org/unreliable-guides/NAT-HOWTO/
- Linux IP Masquerade HOWTO
  http://www.tldp.org/HOWTO/IP-Masquerade-HOWTO/index.html
- Linux 2.4 Advanced Routing HOWTO
  http://www.linuxdocs.org/HOWTOs/Adv-Routing-HOWTO.html
- Introduction To Network Address Translation (NAT)
  http://www.firewall.cx/nat-intro.php
- Network Address Translation (NAT/ PAT/ IP Masquerading)
  http://home.t-online.de/home/TschiTschi/ip_masquerading.htm
- Attacks over the internet
  http://zork.net/~phil/Cracking/Internet.html
- IP spoofing
  http://bear.cba.ufl.edu/teets/projects/ISM6222F102/perryna/index.html