IP Spoofing: A Mammoth Description

By Ankit Fadia ankit@bol.net.in

IP spoofing is the most exciting topic you will hear wannabe hackers talking about. It is also a subject about which no one knows much. Before we continue I would like to tell you that IP Spoofing is quite difficult to understand and a lot of people have trouble understanding how it is done. The other downside it has is the fact that it can almost not be done using a Windows system and a system administrator can easily protect his system from IP spoofing

IP Spoofing is a trick played on servers to fool the target computer into thinking that it is receiving data from a source other than you. This in turn basically means to send data to a remote host so that it believes that the data is coming from a computer whose IP address is something other than yours. Let's take an example to make it clear:

Let your IP Address be: 203.45.98.01 (REAL)
Let the IP Address of the Victim computer be: 202.14.12.1 (VICTIM)
Let the IP Address of the system you want data to be sent from: 173.23.45.89 (FAKE)

Normally sitting on the computer whose IP is REAL, the datagrams you send to VICTIM will appear to have come from REAL. Now consider a situation in which you want to send data packets to VICTIM and make him believe that they came from a computer whose IP is FAKE i.e.173.23.45.89. This is when you perform IP Spoofing.

IP Spoofing thus, can be said to be the process by which you change or rather spoof your IP Address, so as to fool the target system into believing that your identity is not the one, which is actually yours, but make it believe that you actually are the computer having the spoofed address. Let us take a real life example to understand better. Say there are 3 people, A, B and C. You are person B and you wish to fool person A over the phone, into believing that you are person C, when actually you are person B. In order to do so, you might disguise your voice and make it sound more like that of person C, so that person A, might get fooled into believing that he is actually communicating with person C. If you now, replace the three persons in the above example, with a computer and change the term ‘voice’ to ‘IP Address’ then you would know what we actually mean by IP Spoofing.

Problems faced while performing IP Spoofing

One of the reasons as to why IP Spoofing is considered so much tough to perform is the fact that it is blind attack. What we mean by a blind attack is that, we do not get any messages or any feedback regarding our progress. When an attacker is trying to perform IP Spoofing, then, there is no mechanism which tells him, whether he has been successful or not, if yes then by what extent or if no then what when wrong. Thus, he is literally performing the attack blindly, assuming that things went right. Taking the assumptions made earlier, we can explain this problem in the following manner:

The main problem with IP Spoofing is that even if you (REAL) are able to send a spoofed datagram to the remote host (VICTIM), making it believe that the datagram came from FAKE, then the remote host (VICTIM) will reply to the spoofed IP Address (FAKE) and not your real IP Address (REAL), thus, as a result, REAL does not get any feedback whatsoever, regarding his progress. The following is the explanation of the blind nature of IP Spoofing, using the concept of the three-way handshake, which has to take place, each time a TCP/IP connection is established.

If REAL wants to establish a TCP/IP connection with VICTIM, without spoofing of any IP Address, then typically the three way handshake would take place as follows:

1. REAL sends a SYN packet to VICTIM.
2. VICTIM sends back a SYN/ACK packet to REAL.

3. REAL acknowledges this by replying with a SYN packet.

However, if REAL wants to spoof his IP Address and make it appear to be FAKE, then the following will take place:

1. REAL sends a SYN packet to VICTIM, but this time with the source address being FAKE.

2. VICTIM sends back a SYN/ACK packet to FAKE. There is no way that REAL can determine when and if VICTIM has actually replied with a SYN/ACK addressed to FAKE. This is the blind part and REAL just has to let some time pass (once it has sent a SYN packet to VICTIM) and assume that by then VICTIM must have sent a SYN/ACK to FAKE.

3. After some time has passed, REAL then has to send a SYN packet to VICTIM acknowledging that FAKE has received the SYN/ACK packet. (Assuming that it indeed has.)

If and only if all the above three steps go as planned, can a TCP/IP connection be established between VICTIM and FAKE, by REAL.

However, the third step gives rise to another problem. In the second step in the IP Spoofing handshake we see that VICTIM sends a SYN/ACK packet to FAKE. This gives rise to two scenarios:

1. Let us assume that a system with the IP Address FAKE does indeed exist, then, FAKE would actually receive the SYN/ACK packet, which VICTIM sends to it. However, since FAKE did not request a connection, it would not have any idea whatsoever, as to what this packet is meant for and thus it would reply, by sending a NACK (Non-Acknowledgement) message to VICTIM.

****
HACKING TRUTH: A NACK or a Non-Acknowledgement message is meant for ending the connection, resulting in no further communication between the two systems involved.
****

Thus, VICTIM on receiving this NACK message from FAKE would terminate the connection, which REAL is trying to establish between VICTIM and FAKE (without their knowledge). Thus, REAL's attempt to spoof his IP Address would get foiled.

IP Spoofing can be successful only if the computer with the FAKE IP does not reply to the victim and not interrupt the spoofed connection. Take the example of a telephone conversation; you (B) can call up a person ' A ' and pretend to be ' C ' as long as ' C ' does not interrupt the conversation and give the game away.

2. Let us assume that a system with the IP Address FAKE does not exist, then, VICTIM will not receive any ACK message from FAKE for the SYN/ACK that it sent to it. Thus, a 'timed out' would take place and VICTIM would end the connection with FAKE, thus foiling REAL's attempt to spoof his IP Address.

Thus, the above explanation brings us to the following conclusions:

1. IP Spoofing is a blind attack and we do not get any feedback regarding our progress and thus we have no idea as to whether we have been successful or not. The fake packets that we send to the target system, pretending to be a FAKE host have to be sent on certain assumptions and after correct intervals of time.

2. If REAL wants to spoof his IP Address and make it seem to be FAKE, then the following conditions must be true:

    a) FAKE must exist and must be connected to the Internet.

b) FAKE must not at any point respond to the SYN/ACK packet, which VICTIM sends to it. This is where a popular DOS Attack, SYN Flooding comes in. (Explained later)
c) If you are exploiting a trust relationship, then FAKE must be chosen such that VICTIM and FAKE have a trust relationship with each other.

Before we actually move onto a step-by-step guide to using IP Spoofing to exploit trust relationships, we need to understand certain concepts involved in IP Spoofing.

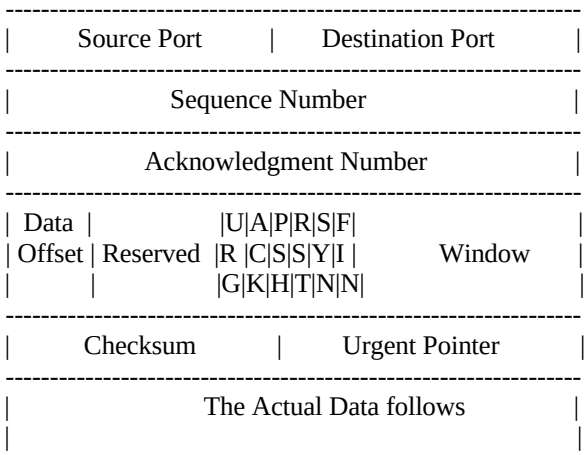
Networking Basics involved in IP Spoofing

Sequence Numbers are the reason why the destination system is able to put back the smaller chunks of data that it receives from the source system, into a larger chunk. All data being transferred on the net is broken down into smaller packets at the source and joined together at the destination. The data is broken down into packets in a particular sequence at the source. This means that, for example, the first byte has the first sequence number and the second byte the second sequence number and so on. Packets are free to travel independently on the net, thus sometimes, when the data packets reach the destination they arrive out of sequence. This means that sometimes the packet of data having the sequence number 4 arrives before the data packet having a sequence number of 3.

The sequence numbers of the packets help the destination system to put back the data packets received, in the correct order. The application or the layer running at the destination automatically builds up the large chunks of data by reassembling the smaller chunks in the correct order, with the help of sequence numbers.

Sequence numbers not only help in the reassembly of packets at the destination system, but also ensure that TCP remains a reliable protocol, which can deal with lost, duplicated and unordered packets.

The following is a typical TCP Header of a data packet being sent by the host to the client:



NOTE: In the following paragraphs we assume that the fields and values being talked about belong to a packet being sent by a host to a client.

The Sequence number field in the TCP header contains the sequence number of the data packet being carried currently. Each packet being sent across the Internet is sequenced. There are certain things about sequence numbers, which one must be well conversed with. They are all discussed in the succeeding paragraphs.

A sequence number is a 32-bit number and it can range from 1 to 4,294,967,295. How does a host decide as to what sequence number, should be assigned to a particular connection? This question introduces us to yet another new term i.e. The Initial Sequence Number or ISN.

The Initial Sequence Number or the ISN is the first (initial) sequence number given to a host at the time when the system is being bootstrapped. At bootstrapping time, the ISN is assigned a value of 1. Whenever a host establishes a connection, then the value of its ISN at that particular point of time is treated as the sequence number of the first fragment being sent to the client with whom the connection is being established. Once a system is assigned an ISN value of 1 at bootstrapping time, then this value keeps increasing automatically and quite predictably with the passage of time, transfer of data and establishment of connections.

This Initial Sequence Number gets incremented by 128,000 every second and with every connection being established, it gets incremented by 64, 000. Due to the fact that with the passage of every single second the ISN gets incremented by 128,000, its value gets wrapped every 9.32 hours. (Assuming that no connections are established.)

For Example,

If the ISN of a host were 1897737287, then after 3 connections and 2 seconds, its ISN would be equal to: 1898185287 = 1897737287+ (3*64 000) + (2*128 000)

To recapitulate, we can say that the value of the sequence number in the TCP header would be sequence number of the first byte of data in that particular fragment. One thing to note here is that the sequence number of the first byte of data being sent by the host to the client would be equal to the value of the ISN plus 1. This is because the SYN flag takes up 1 sequence number. This will be clearer after the sequence numbers and connection establishment section.

In the acknowledgement number field too there would be a 32-bit sequence number, however, this sequence number is not of the host but of the client. (Please note that we have assumed that this packet was sent to the client by the host) What we mean to say is that the sequence number in the acknowledgement number field actually represents the value of the next sequence number (i.e. value of the next sequence number of data) that the host expects the client to sent. This number also acknowledges that all data up to this number minus one has been received safely.

Sequence Numbers and Connection Establishment and Termination

For a TCP/IP connection to be established, a three-way handshake has to take place between the client and the host. The following are the 3-steps, which would take place for a complete and successful connection to take place between the host and the client-:

1. The client sends a SYN packet to the server, requesting for a connection to be established. This SYN packet would contain the client's ISN. Let us assume it to be 4894305. As the client still does not know the sequence number of the host and as it does not have to acknowledge any received data, it sets the Acknowledgement Number field to 0. This packet would also contain other information like the destination address, port number etc.

2. The host on receiving this packet would respond with a SYN/ACK packet. This packet would contain the server's ISN. Let us assume it to be 1896955367. It would also contain in the acknowledgement number, a sequence number, which signifies the next, expected sequence of data and acknowledges the receipt of the earlier data. The value of this acknowledgement number is always the client's ISN plus one. Thus, in our case its value would be 4894305+1=4894306.

3. The client then replies with an ACK packet. The ACK number field would now be, the server's ISN plus one, which is equal to 1896955368.

One can represent this diagrammatically as below:

Client----------SYN (4894305)----------→ Host

Host----------SYN (1896955367) and ACK (4894306)---------→ Client

Client----------ACK (1896955368)----------→ Host

Above we described the establishment of a TCP/IP connection with relation to sequence numbers, now after following the above steps, we suddenly decide to end the connection (continuing with the same values of SYN and ACK sequence numbers) then the following would take place:

Client----------FIN (4894306) and ACK (1896955368)----------→Host

Host----------ACK (4894307)----------→Client

Host----------FIN (1896955368) and ACK (4894307)-------→Client

Client----------ACK (1896955368)----------→Host

A deeper look into Sequence Numbers

To better understand the phenomenon of SYN and ACK values, let us perform a test in which first we telnet to port 23 of a remote system and then immediately disconnect from it using the Quit command. Headers of all packets being transferred would be captured using a Sniffer. (Please note that we would be referring to the remote system as Host and our own system as Client)

#telnet targetsystem.com 23

On typing the above command, the following transfer of packets takes place:

1. **Client ----------SYN (856779)----------→ Host**

   The captured frame of this data transfer is given below for your further study (The part in bold represents the Sequence number of the packet.):

   20 53 52 43 00 00 44 45 53 54 00 00 08 00 45 00 00 2C C3 00 40 00 20 06 10 0C CB 5E FD BA CB 5E F3 47 04 07 00 17 **00 0D 12 CB** 00 00 00 00 60 02 20 00 D9 70 00 00 02 04 05 B4 2D

   Here we see that the client is sending a packet with the SYN option on, which means that it is requesting for a TCP/IP connection to be established. SYN stands for Synchronize sequence numbers In this case the Sequence number of the data packet sent by the client is 856779

2. **Host----------SYN (758684758) and ACK (856780)------→ Client**

   The captured frame of this data transfer is given below for your further study (The parts in bold represents the Sequence number and ACK value of the packet.):

   44 45 53 54 00 00 20 53 52 43 00 00 08 00 45 00 00 2C 8C 05 40 00 39 06 2E 07 CB 5E F3 47 CB 5E FD BA 00 17 04 07 **2D 38 9C 56 00 0D 12 CC** 60 12 83 2C AC A4 00 00 02 04 05 B4

   Here we see that the host on receiving the request for establishing a TCP/IP connection responds to the request by sending a packet with both the SYN and ACK options turned on. The SYN option is still on, as the host still needs to send its own Sequence number to the client. In our case the Sequence number of the host is 758684758. The host turns the ACK option on and gives it a value, which is equal to the client's initial sequence number plus one. The ACK number acknowledges the data received till now and represents the next expected sequence number that the host expects to receive. Here the ACK number sent by the host is the client's ISN + 1= 856779 + 1= 856780

3. **Client----------SYN (856780) and ACK (758684759)----------→ Host**

   The captured frame of this data transfer is given below for your further study (The parts in bold represents the Sequence number and ACK value of the packet.):

   20 53 52 43 00 00 44 45 53 54 00 00 08 00 45 00 00 28 C4 00 40 00 20 06 0F 10 CB 5E FD BA CB 5E F3 47 04 07 00 17 **00 0D 12 CC 2D 38 9C 57** 50 10 22 38 25 56 00 00

   In the above captured frame we find that the client replies to the SYN/ACK sent by the host, with a ACK message, which acknowledges the receipt of the data till now and also mentions the sequence of data expected next from the host by the client. In this case the ACK value is equal to the host's ISN plus one. Thus, ACK value= Host's ISN + 1= 758684758 + 1= 758684759.

One thing to note here is that in step 3, since the client is sending only an ACK message to the host (and not a SYN message), thus the sequence number will not get incremented. The third packet just contains the ACK option turned on, while the SYN option is still turned off. As a result the sequence number does not get incremented. What this means is that the next packet sent by the client to the server too will have the same sequence number, as it did not get incremented.

Once the above three steps have been completed, a complete 3-way handshake is said to have taken place and a TCP/IP connection has thus been established between the client and the host. However, in our case, as soon as this is done, we use the Quit command to disconnect from the telnet daemon. On doing this, the following data transfer takes place:

1. **Client----------FIN (856780) and ACK (758684759)----------→ Host**

   The captured frame of this data transfer is given below for your further study (The parts in bold represents the Sequence number and ACK value of the packet.):

   20 53 52 43 00 00 44 45 53 54 00 00 08 00 45 00 00 28 C5 00 40 00 20 06 0E 10 CB 5E FD BA CB 5E F3 47 04 07 00 17 **00 0D 12 CC 2D 38 9C 57** 50 11 22 38 25 55 00 00

   Eagle Eyes readers, must have noticed that the hexadecimal values i.e. the values in bold in the above captured frame are same as that in the third step in the connection establishment process. However, the difference lies in the fact that in this case, the packet has the FIN (Finish) and ACK options turned on, while in the third step of the connection establishment process, the packet had the SYN and ACK options turned on. Thus, in this step the client sends a FIN\ACK packet to host. The FIN option tells the host that client wants to terminate the connection that has been established between them. The sequence number of this packet does not change from the third step, as the third step carried only an ACK message, which does not consume any sequence number. Although this particular packet is indeed carrying a FIN message, which does in fact consume a sequence number, however a sequence number of a fragment represents a value, which is applicable at the start of that particular fragment and not the end. Thus, the sequence number consumed by the FIN option would increase the sequence number of the next packet and does not affect the sequence number of this packet. The ACK value of this packet is 758684759, which represents the sequence of the packet expected by client from the host. The ACK value of this message does not change from that of the third step, as no data was received by the client from the host and the client still expects host to send the data sequence 758684759.

2. **Host----------ACK (856781)----------→ Client**

   The captured frame of this data transfer is given below for your further study (The parts in bold represents the Sequence number and ACK value of the packet.):

   44 45 53 54 00 00 20 53 52 43 00 00 08 00 45 00 00 28 8F BE 40 00 39 06 2A 52 CB 5E F3 47 CB 5E FD BA 00 17 04 07 **2D 38 9C 57 00 0D 12 CD** 50 10 83 2C C4 60 00 00

Here the host sends a packet with the ACK option turned on, which confirms the receipt of the data sent by the client to host till now. Here the ACK value is 856781. Also note that the sequence number of this particular is treated as 758684759, which is the same as the next message sent by the host to the client, as this particular message is carrying only an ACK option, which does not consume any sequence number.

3. Host----------FIN (758684759) and ACK (856781)----------→ Client

The captured frame of this data transfer is given below for your further study (The parts in bold represents the Sequence number and ACK value of the packet.):

44 45 53 54 00 00 20 53 52 43 00 00 08 00 45 00 00 28 8F E0 40 00 39 06 2A 30 CB 5E F3 47 CB 5E FD BA 00 17 04 07 **2D 38 9C 57 00 0D 12 CD** 50 11 83 2C C4 5F 00 00

In this captured frame, the host sends a FIN\ACK packet to the client, with the sequence number equal to 758684759, which is same as that of the earlier step. Even the ACK number remains the same, as Host still received no data, sent by the client.

4. Client----------ACK (758684760)----------→ Host

The captured frame of this data transfer is given below for your further study (The parts in bold represents the Sequence number and ACK value of the packet.):

20 53 52 43 00 00 44 45 53 54 00 00 08 00 45 00 00 28 C6 00 40 00 20 06 0D 10 CB 5E FD BA CB 5E F3 47 04 07 00 17 **00 0D 12 CD 2D 38 9C 58** 50 10 22 38 25 54 00 00

Here the client replies to the host's FIN/ACK packet by sending an ACK packet, which acknowledges all data received till now and thus the connection gets terminated. Please note that the sequence number of this packet is treated as 856781.

Thus, from the above sections we can conclude that the sequence number gets incremented in the following cases and manners:

Cases	**Increment**
Transfer of FIN Packet	1
Transfer of SYN Packet	1
Transfer of ACK Packet	0
Transfer of SYN/ACK Packet	1
Transfer of FIN/ACK Packet	1
Passage of 1 second	128,000
Establishment of 1 connection	64,000

If one is able to learn the art of prediction of sequence numbers, then the following could easily be done:

1. Hijacking TCP Connections and diverting data.
2. Exploiting Trust Relationships

Now that we truly understand the increment of sequence numbers and ACK numbers, let us now learn as to how IP Spoofing can be used to exploit trust relationships.

Trust Relationships

Everyone is encountered with some form of authentication process or the other. The Username-Password pair is the most commonly used form of authentication, with which we are very much familiar. What happens in the Username-Password form of authentication is that the remote host to which the client is connected to, challenges the client by asking the User to type in the Username and Password. Thus, in this form of authentication, the User needs to intervene and the remote host challenges the user to enter the Username and Password, which act as a form of authentication.

Besides the Password-Username form of authentication, there is yet another form of authentication most users do not know of. This is the Trust relationship form of authentication, in which the IP Address of the client system acts as a proof of authenticity. In this form of authentication, what happens is that the remote host gets or finds out the IP address of the client and compares it with a predefined list of IP's who are allowed access. If the IP Address of the client who is trying to establish a connection with the remote host is found in the list of allowed IP's maintained by the host, then it allows the client access to the shell 'without a password' as the identity of the client has already been authenticated. (Using the trust relationship concept.)

Such kind of trust relationships are common in Unix Systems which have certain 'R services' like rsh, rlogin, rcp etc which have certain security problems and should be avoided. Despite the threat involved most ISP's still keep the ports of the R services open to be exploited by Hackers. Once can establish a rlogin connection with a remote system by simply using the following Unix shell command:

$>rlogin IP address

****
HACKING TRUTH: There is definitely a cooler way of establishing a trust relationship with a remote host, i.e. using Telnet. The default port numbers at which the R services run are 512, 513,514
****

Thus, this means that if there exists a trust relationship between two systems (i.e. the client and the server), then if you are able to change your IP Address and pretend to be the client, then you could actually authenticate yourself making use of this new, spoofed identity and establish a proper connection with the server. Such a connection, would probably entitle you to all commands and all parts of the server. Thus, we can conclude by saying that if one is able to spoof ones IP Address and use this spoofed identity to exploit trust relationships, then it could have some very serious consequences.

Spoofing your IP Address to exploit trust relationships

Before we move on to the various steps that we have to perform, in order to spoof our IP Address and in turn exploit trust relationships, it is important to understand that IP Spoofing is extremely difficult to perform and it is also very complex. Hardened networking experts too find it fairly mind-boggling. What really makes it so tough to execute, is the fact that it is a blind attack i.e. one does not get any feedback regarding ones progress and also it is based on a lot of assumptions, which have to be made on the part of the attacker. Thus, one might fail to successfully spoof ones IP Address several times, however, practice makes a man perfect, so keep trying. IP spoofing is definitely very difficult to perform, however, it is not impossible to perform and has certainly been carried out before.

The various steps involved in successfully spoofing ones IP Address and in turn exploiting a trust relationship, are as follows:

NOTE: Please note that for the following steps, we would be using three systems:

a.) The Victim system, which will be referred to as VICTIM or the target system.
b.) The Trusted system, which is capable of establishing a trust relationship with VICTIM. It would be referred to as TRUSTED.
c.) The attacker's system, which would be referred to as the ATTACKER.

1. Detecting a Trusted System

The first step in exploiting trust relationships is to actually find out as to with which system does the target system enjoy a trust relationship. In this step what is done is nothing but finding out as to which computer does the target system trust and usually establish a trust relationship based on authentication by IP Address, with. In effect, we would want to find out the trusted system, the system that is allowed by the target system to establish a remote trust connection with itself i.e. the system named TRUSTED.

The trusted system establishes a trust connection with the target system using what are known as the r services. We can find out as to with system the target system establishes a trust relationship, by using any of the following methods:

a.) By using various useful commands like: (This is the most likely method to work.)

rpcinfo -p
showmount -e

b.) Social Engineering: One should try and dig out as much information on the target system and its network as possible.

c.) Method of Brute force, in which all systems in the same network are checked for whether or not they are capable of establishing a trust relationship with the target system. However, this method is extremely tedious and slow.

Once you have found out the trusted system with which the target system is capable of establishing a trust connection, what one has to do is: DOS attack this trusted computer, so as to render it useless, thus ensuring that it does not interrupt or end our spoofed connection with the target system.

2. Blocking the Trusted System

This step is a very important step. As soon as you find the trusted system, your next step is to block it or disable it. If you remember, we had discussed the problems that one faces while trying to perform IP Spoofing. After a long description, we had concluded some necessities for IP Spoofing to take place successfully. Amongst them, one was as follows:

"…FAKE must not at any point respond to the SYN/ACK packet, which VICTIM sends to it…"

Thus, in order to ensure that the above takes place, one has to make sure that the trusted system is blocked i.e. one must make sure that all the memory of the trusted system is used up, so that it is not able to respond to the SYN/ACK packets addressed to it (sent by the VICTIM system.) One very easy method of doing so is to perform a SYN Flooding Denial of Services attack on it. For a detailed description regarding SYN Flooding, read the DOS Attacked!!! Section.

Once we have a long queue of SYN Floods on the target system and hog up all the memory available to it, then we can probably be sure that it will not respond to any SYN/ACK packets sent to it by the victim system. Thus, once the blocked system is feeling the affects of the SYN Flood attack, then, we can be assured of being freed of one of the numerous problems faced during the process of performing IP Spoofing. (The problem of the trusted system intervening and ending the spoofed connection.)

3. Getting the Final Sequence Number and Predicting the Succeeding Ones

Once the trusted system has been disabled, the attacker then must get the sequence number of the target system. This is the step, in which the attacker has to make a lot of quick calculations and assumptions. In order to get the target system's sequence number, the attacker connects to Port 23 or Port 25 just immediately before launching the attack and records the Sequence number of the

last packet sent by the target system. It is advisable to repeat this step several times and then record and note down the last sequence number of the target system. The attacker must also deduce the Round Trip Time or the RTT using a utility called icmptime. (For more information read the ICMP section.) To get an accurate value of the RTT, one should repeat the process of sending a packet and recording the time, several rimes.

****

HACKING TRUTH: RTT or Round Trip Time is actually the time taken by a packet to travel from the source to the destination and then back. Thus, the time interval taken by the packet to reach from the source to the destination can be found out by RTT/2.

****

Once the attacker knows the last sequence number of the target system, then on the basis of the above calculated values of RTT, the amount of time passed (between the recording of the last sequence number and the actual execution of the attack) and the other cases when there can be an increment of the sequence number, he could actually predict the next sequence number.

To effectively perform this step, the following should be done:

1. Record the RTT or the Route Trip Time and in turn the time taken for a packet to travel from the target system to your system, beforehand itself. So that time is saved during the actual execution of the attack.

2. Once you have logged the last sequence number of the server, then do not waste anytime whatsoever. Calculate the next sequence number as quickly as possible and quickly move onto the actual execution of the attack, because more the time you waste, the greater is the chance of another system on the Internet establishing a connection with the target system, hence increasing its sequence number to a value, which is 64, 000 more than what you might have successfully predicted.

3. Learn the Case-Increment table, given earlier by heart and keep a calculator nearby, before you actually sit down to perform IP Spoofing.

4. Then close your eyes and pray and keep your fingers crossed, hoping that your gray cells work fast enough and well enough!!!

When you are predicting the next sequence number, then ideally your predicted value, should be same as what the target system's next sequence number really is. However, even if the sequence number that you predicted is not that much greater than the actual next sequence number, then, the target system would just queue it up, treating it as a packet for future use.

Let us assume that the attacker is actually able to predict the correct next sequence number and move onto the next step that he would have to follow.

4. **The Actual Attack**

Once you have been able to predict the next sequence number, then comes the time of performing the actual attack as follows:

1. The ATTACKER sends a SYN packet with the spoofed IP Address, (which is actually the IP Address of the trust system) to the VICTIM system. This SYN packet is addressed to the rlogin port (513) and requests for a trust connection to be established between VICTIM and the trusted system.

2. The VICTIM responds to this SYN packet by replying with a SYN/ACK packet addressed to the trusted system. If all the memory of the trusted system had not been hogged up by SYN

flooding it, then it would have replied to this SYN/ACK packet with a NACK and the attacker's attempt to spoof his IP, would get foiled. However, as in our case the trusted host has been disabled, thus it is not able to respond to this SYN/ACK packet sent by VICTIM. As a result this SYN/ACK packet sent by VICTIM gets discarded.

3. After a certain time (very small amount of time) has passed i.e. when the attacker is sure that VICTIM must have sent a SYN/ACK packet to the trusted system, he sends an ACK to VICTIM. This ACK message is designed such that it has a spoofed address, which actually makes it seem to have come from the trusted system. It is important to make sure that this ACK packet has an acknowledgement number which is the predicted sequence number plus one. Also the sequence number of this packet should be the sequence number of the packet sent by the attacker in Step 1 plus one.

If everything goes as planned and if the attacker guesses the sequence number correctly, then the target system would accept the connection and a trust relationship would be established.

5. Putting the Trusted system out of the spell of the DOS Attack

Finally, after the entire attack has been carried out, one has to end the SYN Flood on the trusted system, which can be done by sending packets with the FIN option turned on to the trusted system.

Fadia's Hot Pick for Sending Custom Designed Packets to a remote system

1. **Utility Name:** Libnet

   Features: Allows the user to send custom made packets. It also gives a very powerful control over various options of the custom made packets. A brilliant tool overall.

   Download URL: http://www.packetfactory.net/libnet

Countermeasures

1. One should try and avoid making use of trust relationships, which rely on IP Address of the client, rather than on a Username-Password pair, as an authentication tool. Thus, try and avoid making use of trust connections, as far as possible. Using TCP Wrappers to allow access only from certain systems has been known to be a good countermeasure.

2. One of the easiest methods to combat IP Spoofing, is by installing a firewall or a filtering rule, which filters out all packets coming from the outside of a network, but having an IP Address belonging to a system within the internal network structure. Also, one should filter outgoing packets that have a source address different from the internal network. This prevents a source IP spoofing attack originating from your network. One could prevent incoming packets, coming from outside of the network, but having a source IP, belong to within the network, by adding the following router ACL:

   access-list 101 deny ip 201.94.0.0 0.0.255.255 0.0.0.0 255.255.255.255

   This particular ACL is applicable to a network having an internal source address of 201.94.xx.xx. If you are looking at providing your network protection from only source IP Spoofing attacks, which might originate from your network, then the following ACL will do the trick:

access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255

3. Using Encrypted and Secure Protocols like IPSec

4. By the Introduction of the use of Random Initial Sequence Numbers, to do away with the predictable nature (increment) of the sequence numbers. One such solution is the use of pseudo-random-number generators (PRNGs). However, PRNG's too sometimes are not up to the mark, when it comes to randomness of sequence numbers selected.

Reference Material: IP Spoofing Demystified By daemon9/route/infinity.

That is all for now, please do let me know as to what you think of this manual. I would really appreciate any emails or praise or criticism.

Ankit Fadia
ankit@bol.net.in

http://hackingtruths.box.sk

To receive tutorials written by Ankit Fadia on everything you ever dreamt of in your Inbox, join his mailing list by sending a blank email to: programmingforhackers-subscribe@yahoogroups.com