DEMYSTIFYING REMOTE HOST (Part 1)

By Abhisek Datta
abhisek@gamebox.net
http://hackersclub.up.to

This manual will present a concise overview on the methods used to gather necessary information about a remote host targeted for hacking or for some its cracking. Starting from basic newbie friendly methods like Banner Grabbing, TTL value detection, this manual will cover advanced method of detecting the Operating System running by the remote host like ICMP Error Messaging Quenching, ICMP Error Message Quoting, Initial Sequence Number Sampling, Sniffing Packets etc.
Note: Basic knowledge about TCP/IP implementation is suggested.

This is the part one of this manual. In here I'll explain methods of accurate Operating System detection.

So you want to hack a system. I am sure most of the newbies out there who don't know what to do will try to get the IP of the remote host even if they don't know what to do with an IP, then probably they will try to play with powerfull tools like Telnet, FTP etc and not to mention all those script kiddies will set themselves with tools like port scanners, vulnerability scanners, exploits, client-server applications etc even if they don't have any idea about the above mentioned terms. But peoples who are at least close to the term hacker, who know what they are doing will start of with a logical plan. The plan includes whats the need of hacking, he will ask himself why I want to hack the system. Mostly its because of curiosity and the urge to know the world behind the strong powerful walls. Then, he will began his actual task by gathering information about the target system. The more information you will have about your victim, the more easier it will be for you to penetrate into the system. Hacking is not a blind game. You should know what you are doing. You need to step forward according to the response of you target. So for all this the first and foremost step is to make an accurate detection of the Operating System running by the target system.
In this manual I'll talk about the various methods of detecting the remote operating system.

REMOTE OPERATING SYSTEM DETECTION

(a). TTL DETECTION

Though this method is will not present an accurate picture of the Operating System running by the remote host but using this method you can at least differentiate between different platforms like Windows or Unix.

So what exactly is a TTL value ?

TTL stands for Time To Live. These values are particular numeric codes induced in a data packet by its source operating system.
When you send a data across the internet, it is not send at once, but it is broken down into small fragments called data packets at the source system and is reformed into the original data from these small data packets using its Sequence Numbers and Offset fields. Now you don't need to worry about what sequence numbers and offset fields are cause I'll explain it nicely later in this article.

For example if you want to send a file of 1000 bytes across the internet, this file will not be send at once but will be broken down into small fragments of data called data packets. Lets say the data is broken down into 10 small packets each carrying 10 bytes. Now these 10 data packets will be transported to the target system from the source system. At the target system these data packets will be reformed to produce the original file of 1000 bytes in size. A typical TCP/IP data packet consists of various headers. I'll produce a schematic diagram of a typical TCP/IP data packet later in this article. Now when these data packets are transferred across the internet using the TCP/IP stack each and every data packet contains a TTL filed in its header part. The TTL value stored in these field may sometimes give us a clear picture of its source Operating System cause TTL values are particular to an Operating System. For example data packets originating from Windows will have a TTL of 128 whereas a data packet originating from Linux will have a TTL of 245. Thus from the TTL value of a data packet you can have a conception about the Operating System running by the target system.

You can get the TTL value of a data packet originating from a target system just by simply pinging the target system.

Use the command line :

C:\windows>ping 203.197.102.1

Reply from 203.197.102.1: bytes=32 time<1ms TTL=128

Reply from 203.197.102.1: bytes=32 time<1ms TTL=128

Reply from 203.197.102.1: bytes=32 time<1ms TTL=128

Reply from 203.197.102.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum =5ms, Maximum =12ms,Average = 6 ms

Thus we see that on sending ping (ICMP Echo requests) to 203.197.102.1, the system replies with data packets of TTL 128, now its easy to say that the system is running windows Operating System .

The Following chart gives you the TTL values of some platforms

OS	VERSION	PLATFORM	TTL
Windows	9x/NT	Intel	32
Windows	9x/NT	Intel	128
Windows	2000	Intel	128
DigitalUnix	4.0	Alpha	60
Unisys	x	Mainframe	64
Linux	2.2.x	Intel	64
FTX(UNIX)	3.3	STRATUS	64
SCO	R5	Compaq	64

Netware	4.11	Intel	128
AIX	4.3.x	IBM/RS6000	60
AIX	4.2.x	IBM/RS6000	60
Cisco	11.2	7507	60
Cisco	12.0	2514	255
IRIX	6.x	SGI	60

Note: An important miscellaneous information in this regard is that the TTL values may be reduced by 1 at each router. For example if you have 10 routers in between you and your target system then if the TTL value originally generated by the target system is 255 then you may receive it is 255 – (10*1) = 245. In order to find out the number of routers in between you and your target system you can use the tracert command in DOS. Tracert performs a data path check between you and your target system, that is, it sends data packets from your system to the target system and outputs the path followed by the data packet in reaching its destination, i.e it will give the IP address or names of the routers in between you and you target.

For more information about Remote Operating System detection using ping method read my article "Remote Operating System Detection" at http://hackersclub.up.to

Disadvantages of this method:

Though this method is quite easy and newbie friendly but this method is not applicable in practice cause of the following disadvantages.

1. TTL values can be faked easily.
2. TTL values does not give us the correct version of the target Operating System . We can only differentiate between different platforms using this method, we cannot have any info about the version of the target system which is very important cause obviously the vulnerabilities of Linux Red-Hat 6.1 is not the same as Linux Red-Hat 7.2.
3. The ping method is not applicable against professionals cause obviously they will use firewalls which can easily protect the system from echoing back the ping requests. In that case you wont be able to know the TTL value of the target Operating System. So you need to capture a data packet originating from the target system using any sniffer program then study it thoroughly to determine its correct TTL. This method is ofcourse not newbie friendly.

(b). Banner Grabbing

Just because experienced security experts says that *nix (Unix Variants) is much more secure than windows, most of you must be arguing with other peoples about Windows. You think you can hack into a windows system and by using red-hat Linux (or any other) you will prevents hackers to break into your system. Think again pal. There is nothing so insecure like factory default of Linux (I am taking Linux as an example.. cause among Unix Varints I am more familiar with it). By default Linux installs a lot of open services like Telnet Server, FTP server, RSH Server, Finger Server. So anybody even an windows user can use this information if he has the minimum knowledge to break into your system. An intermediate hacker will probably somehow (the method is beyond the scope of this manual) create a .rhosts file in the home directory of your Linux box with the attackers IP in it. Then when he uses the rlogin command to your Linux box and he will be logged in with current user permissions without the need of any authentication.

Note: Hey *nix guys please don't come flaming to me cause of me saying Linux insecure. I do agree Linux is much more secure than windows. I am just saying that the factory default is not up to the mark. I believe even the best and the most secure operating systems like Open BSD or Free BSD is useless in the hands of an incompetent system administrator.

Newbie Tip: By factory default Linux Box, I mean a system running Linux Operating System with its default configurations which are created as a part of the installation process. By *nix I mean all the Unix variants like Linux (all flavours), BSD, Solaris etc..

In this method of Operating System detection I am going to highlight one flaw of a factory default systems (both Windows & *nix). Say when you install a webserver or a SMTP server in your home Linux box with its default configurations then by default these services will show a banner indicating its version and operating system version when somebody uses the service. For example take a look at the following ftp session:

Microsoft Telnet> telnet ftp.target.com 21
Trying 207.200.74.26...
Connected to ftp.target.com
Escape character is '^]'.
220 **ftp29 FTP server (UNIX(r) System V Release 4.0) ready**.
SYST
215 **UNIX Type: L8 Version: SUNOS**

in here I have used telnet to use the resources of the ftp server of netscape. Just on connection the ftp server responds with a banner indicating that it is running **UNIX(r) System V Release 4.0** as the core**.** Then on using the **SYST** command the server responds with its Operating System version.

Note: the SYST command is used primitively. Its no longer available in most of the servers.

Thus we see by grabbing the banner of a particular service running on the remote host we can easily find out the Operating System of the remote host along with its version information.

I'll produce another example. Take a look at the console capture below.

Microsoft Telnet> echo **'GET / HTTP/1.0\n' | nc hotbot.com 80 | egrep '^Server:'**
Server: Microsoft-IIS/4.0
Microsoft Telnet>

Thus you see by executing the command in bold above in the telnet prompt the HTTP service running on port 80 of hotbot.com responds with its actual web server info. Thus anybody now can use known exploits particular to that web server to launch a DoS attack or penetrating into the system with maximum privileges.

Similarly by connecting to the SMTP servers using telnet you can obtain similar information as described above.
Thus we can conclude that banner grabbing is an effective method of detecting the Operating System running by the remote host.

Web Server Detection

Though the method of banner grabbing has lots of disadvantages mentioned later but recently I have came across a method of detecting the web server running on the remote host through banner grabbing. This method is quite successful and a very effective method of detecting the web server running on the remote host.

Description

Suppose you want to detect the web server running on http://loginnet.passport.com (hotmail's authentication server).
Telnet to port 80 of http://loginnet.passport.com using the command line :

Microsoft Telnet> telnet loginnet.passport.com 80
After you get connected you will see a completely empty terminal, that is, there will be no response from the server. Now type anything and press enter. Repeat the process quite a few times you until you get disconnected. After you get disconnected, you will see that the server has responded with some HTML coding and also the web server information.

Newbie Tip: Guys who are not at all acquainted with telnet.. read my article "Telnet Explained [1]" at http://hackersclub.up.to

Web Server of some websites detected using this method

1. http://loginnet.passport.com → Microsoft IIS 5.00 # No Comments !!
2. http://www.happyhacker.org → thttpd 2.16
3. http://www.aol.com → AOLServer3.4.2 # oh.. open source web server.. so confident.. lol. !!
4. http://www.kevinmitnick.com → Apache 1.2.23
5. http://www.pakistan.org → Apache 1.2.3
6. http://www.blacksun.box.sk → Apache 1.2.36

How it works:

Almost all the web servers of present are such that when a 400 or 402 error message is generated in raw mode (i.e in Telnet Session) then the server replies with that particular error message along with its version.
But this can be configured to prevent such information exposure. But almost 80 of 100 web servers are not configured to protect exposure of such information. I have tested this method on various security based websites but I must say the security of sites like http://www.insecure.org (One of my Fav.. I admire Fyodor a lot..) are very good and are protected from this simple but quite effective method of web server detection.

Disadvantages of Operating System detection though Banner Grabbing

Though it is an easy to implement newbie friendly method buts its not quite in practice at present. This method is only applicable against lamers. A good system administrator will definitely remove all the banners which provide any trace of information about his system from the banners of his servers. Its not a tough job to customize or remove these system banners which are a part of the default installation process.

TCP/IP Features

(Study requires for Operating System fingerprinting explained later in this whitepaper)

Before I pass on to Operating System detection through finger printing methodology, I need to give you a basic overview about TCP/IP data packets and the information carried by a data packet and overcoming various problems which may come in the mind of an average user regarding data transfer across the internet.

Take a look at the schematic diagram below which represents a typical TCP/IP data packet with all its headers.

IP Data Packet

Version	IHL	TOS	Total Length
Identification	Flags (SYN, ACK, PSH, FIN, RST etc.)		Fragment Offset
TTL	Protocol		Header Checksum
Source Address			
Destination Address			
Options			Padding

TCP Data Packet

Source Port			Destination Port					
Sequence Number								
ACK Number								
Data Offset	Reserved	u r y	a 6 k	P 6 h	r s t	s s n	f y n	Window Size
Checksum				Urgent Pointer				
Options				Padding				
Data								

Some Important facts about TCP/IP

(TCP = Transmission Control Protocol (connection oriented) || IP = Internet Protocol (connectionless))

TCP is regarded as connection oriented protocol. This is because it ensures the reliability of proper transport of data packets across the internet. It also provides the advantage of error correction and overcome certain shortcomings of other protocols in data exchange. It introduces the concept of sequence numbers. For data exchange between two computers, both the computers must be using TCP as its primary suite of protocol.
IP is regarded as connection less protocol. This is because, IP doesn't provide the reliability of proper data exchange. Its function is just to transport data across the internet. It contains the source and destination IP. It acts as the transport protocol for TCP.

Operating System FINGERPRINTING

This is the mostly used and widely accurate method of detecting the remote Operating System. By Operating System finger printing I mean to say detecting the Operating System running on the remote host by some of its exhibited character induced in many of its inherits like the data packets originating from

the system, the time interval of getting a particular error message from a system. These methods are very critical and need to be implemented with proper understanding and using your own intelligence and experience. There are various types of Operating System finger printing but in this manual I'll mainly deal with the following :

1. Features of TCP Implementation
2. ICMP Error Message Quenching
3. Sniffing data packet
4. Initial Sequence Number Sampling

These methods are quite accurate and intelligent Operating System detection can be implemented using these methods. But I must say these methods are not at all newbie friendly, but I'll try my best to pen it according to the understandings of an average user.

(a). Features of TCP implementation

There are various errors existing in the TCP implementation of various Operating System which enables to make an accurate Operating System detection.

If we send a TCP FIN (or any packet without any SYN or ACK flag) packet to a remote system, then according to the correct RFC 793 behavior the system should not respond. But many broken implementations like MS Windows, BSDI, CISCO, HP/UX, MVS, etc. sends a reset back. Thus we can use this property of those Operating System for its detection remotely.

Another method is SYN Flooding (it's a kind of DoS attack.. to know more about it read my article "Deadly DoS Attack") your target host. Basically prior Operating Systems used to have a small back log size (Number of incomplete SYN request stored). Most of the Operating System have a back log size of 8. that means it can only handle a total of 8 incomplete SYN requests before crashing (rarely) or banning SYN request from your IP.

(b). ICMP Error Message Quenching

Description of ICMP (Internet Control Message Protocol)

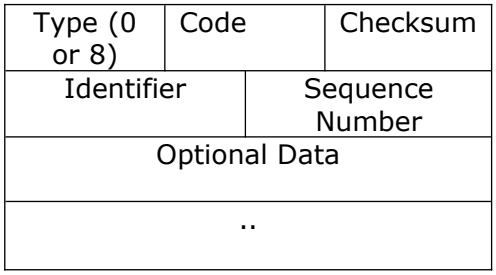
- **ICMP is:**
  - A required part of the IP protocol which must always be included.
  - Provides communication between IP software on two different machines (not just source and destination machines).
  - Not restricted to gateways. It provides a single mechanism that is used for ALL control and information messages.
  - Reports errors but does not correct them. The source node must take action to correct problems.
- The source node may not be able to handle all the problems and must trust that remote operators can fix problems of that type:
  - Gateways routing to incorrect locations
  - Gateways using corrupted routing tables.
- ICMP is not considered a high-level protocol.

Most of the Operating System of today implements their network protocols according to RFC 1812 which suggests limiting the rate at which various error messages are generated by a particular Operating System. Now when you send an ICMP (Internet Control Message Protocol) ping request to a system and the system responds by echoing back a part of your ICMP request. This is a simple

ping. This ping method is used to determine network connectivity between two systems and to detect weather the system is responding or not. According to RFC 1812 if you send an ICMP data packet directed to some high UDP (User Datagram Protocol) port to a system then the port unreachable error message that is almost certain to be generated by the Operating System running on the target system will be generated after a certain interval. RFC 1812 suggest that each and every ICMP error messages must be generated after a certain interval. Now this is the basis. But different Operating System implements this principal differently. Different Operating System have different intervals of generating ICMP error messages. So if you can find out the interval at which ICMP error messages are generated by the target system then by comparing that interval with some standards we can find out which Operating System is running on the target system.

A typical ICMP echo requesting packet is give below in schematic view.. when you ping a system from console this kind of ICMP data packets are send :

ICMP Echo Request/Reply Format



Now in case of an ICMP reply the code field in the ICMP packet header indicates what kind of reply it is. The different type of error messages have different codes. Following is a list of codes along with their respective error messages :

Code Value	Meaning
0	Network Unreachable
1	Host Unreachable
2	Protocol Unreachable
3	Port Unreachable
5	Source Route Failed
6	Destination Network Unknown
7	Destination Host Unknown
12	Host Unreachable for Type of Service

Now comes the practical execution of ICMP error message quenching..
First you need to ping normally to the target system and calculate the RTT (Round Trip Time). You must be wondering what RTT is. Well it's the total time required for an ICMP data packet to complete a trip from your system to the target system and vice versa. After pinging the target system we find out the average RTT.

Newbie Tip : Don't know what is ping and how to ping a system. Just open DOS and type ping and press enter. You will get a complete description of the ping utility.

Now we need to send some ICMP packets to some high UDP ports to the target Operating System. In response the system will generate port unreachable error

message. Now we need to calculate the RTT again. This time evidently the average RTT will be greater than the previous time when we get valid ICMP replies from the target system, but this time we are getting port unreachable error message and as I have mentioned above that according to RFC 1812 error messages are generated by Operating System after a certain interval. Now by finding out the difference in the RTT we will get the average time interval at which port unreachable error message is generated by the Operating System running in the target system.

(c). Sniffing Data Packets

This methodology lies in the class of Passive Stack Fingerprinting. In here what we do is capture a data packets originating from the target system using a good sniffer program which offers viewing of all packet headers. From the schematic diagram of a typical TCP/IP data packet produced before in this article it is obvious that a whole lot of information about the target system's Operating System is easily available from the packet headers.

What are sniffers ?

Well sniffers are programs which captures data packets in a raw mode directly from your network adapter. I guess this statement doesn't make any sense to most of you peoples. Gonna explain it nicely.
When you are surfing across the net, your modem acts as a modulator-demodulator which means it converts digital data into analog format for transportation across your telephone line. You cannot see any information about a particular data packet. Yeah users with little knowledge can find out the TTL values of data packets originating from a system by pinging it. But the other fields like Flags, Windows Size, TOS , IHL etc are not easily available. Here comes the need of a sniffer program. You need to install a sniffer computer at your end.. that is in your computer and it will capture all data packets exchanged by your system in accordance with its configuration. You can configure it to capture particular headers or all headers of a data packet.

Now you need to sniff (catching with a sniffer) a data packet originating from your target whose Operating System you want to determine through a sniffer program (example : Win Sniffer). Now since all the headers are available you can find out a whole lot of information about its source operating system due to the fact the data packets originating from a computer depends and inherits the property of the Operating System running by the system. For example if you capture a data packet of TTL 128 you can say that it is generated by a Windows system.

(d). Initial Sequence Number Sampling

What is Sequence Numbers ?

The brief definition of sequence numbers says that it is a 32 bit number ranging between 0 to 4,294,967,295 used in TCP header for data sequencing. As I have explained earlier that when data exchange is performed over across the internet through data packets and the short coming of overlapping is these data packets is overcome with the concept of sequence numbers. By looking the schematic diagram of you will see that there is a separate field for sequence numbers. Each and every packet of data send across the internet is sequenced. This sequence number is not a static number. It is constantly randomized. The TCP sequence number gets incremented by 128000 each second and for every successful connection it is incremented by 64000. So calculating in this light the 32 bit

sequence number is bootstrapped at every 9.32 hour considering that no connections are made so no increment of 64000. The calculation is like this :

(9.32*60*60) * 128000 + (64000 * 0) = 4,294,967,295 (Approximately.. not exact cause I have taken the time as 9.32 which is an approximate constant)

Though in Initial Sequence Number sampling detailed description of sequence numbers are not required but I will present a brief over view about sequence number and TCP/IP 3 way handshake in the light of sequence number in order to provide you with clear conception.. (I am trying my best.. don't know how far I am successful.. lol ☺)

Now the Initial Sequence Number at the time of boot strapping (it means when the Sequence Number counter starting from reset state.. i.e it is at 0) is chosen to be 1.

- TCP 3 way Handshake in light of TCP Sequence Numbers

Though I must say that the readers of this manual must have the minimum knowledge about 3 way handshake but for the sake of the moment I am going to explain it once again. The very basis of connection between 2 computers (client & server) across a network in accordance with TCP/IP suit is 3 way handshake. It is the method by which connection is established.
At first the client sends a data packet flagged as SYN to the server.
The server replies with a SYN/ACK packet.
The client acknowledges back with an ACK packet to establish the connection.

A FIN packet is send from any of the end to terminate the connection. RST is used to reset the connection.

```
              SYN
Client ----------------→
              SYN/ACK
Server ----------------→ Client
              ACK
Client ----------------→ Server
```

This is the very basic overview of TCP/IP 3 way handshake.
Now this method is quite complicated when we bring in the concept of sequence numbers. When the client sends the SYN packet to the server requesting for the connection, the packet consists of a Initial Sequence Number of the client. The ACK field is 0. Then the server replies with a data packet with its Initial Sequence Number and at the ACK field it is Client's Initial Sequence Number + 1. Now after receiving this data packet the client replies with a data packet with ACK field server's Initial Sequence Number + 1 to establish the connection.

Let the Initial Sequence Number of Client = 55555555
Let Server's Initial Sequence Number be = 66666666

Client ---------- (SYN [Initial Sequence Number = 55555555, ACK = 0]) -------→ Server

Server -------→ (SYN [Initial Sequence Number = 66666666, ACK = 55555555 + 1]) ----→ Client

Client -------→(SYN[Initial Sequence Number = Random Number depending on Time,ACK=666666+1]) → Server

Now since you have the basic concept about sequence numbers we can move forward to detecting Operating System through it.

Now the very basis of the Initial Sequence Number sampling method Operating System Operating System detection is to find patterns in the Initial Sequence Number chosen by the server on request to a connection from the client. I mean to say that you need to note down the Initial Sequence Number chosen by the server in replying with a SYN/ACK packet to the SYN packet send from the client. Now on receiving the Initial Sequence Number of the server you need to do quite a few research on some standards as what kind of Initial Sequence Number has different Operating System have. Now matching with these standards you can reach the Operating System.

Thus in actual implementation of this method you need few tools like Libnet (for sending custom data packets) and a packet sniffer (for reading the headers of data packets received from the server.)

Newbie Tip: Don't know what a packet sniffer is ? Well these are small tools which are able to capture each and every data packet on the installed network and enables the user to read its headers. That is you will be able to see all the fields of a packet header actually which you can see in a schematic diagram in this manual.

Now send a SYN packet to the server with any Initial Sequence Number from your end and wait for the response. The server will respond with its own Initial Sequence Number and an ACK field which is equal to your Initial Sequence Number + 1. Note down the Initial Sequence Number received from the server. Now send a FIN packet to terminate the session.
Wait for some time (note the time) and then again repeat the above process. In this way you will get quite a few Initial Sequence Number from the server. Now calculate with respect to time and try to get an approximate constant Initial Sequence Number. This Initial Sequence Number chosen by the server is particular to its Operating System. Thus on reaching the Initial Sequence Number we can easily have a clear conception about the Operating System running on the remote server.

Aint it simple ?? guys ?? well.. but when I learnt it.. it's a hell complex and hard to understand for me.. don't know how much I have succeded in making myself clear to you. I know this method is quite hard to understand for newbies. So any queries at abhisek@gamebox.net regarding the implementation of this method will be gladly accepted..

Finally I have come to the end of another article. For any queries mail me at abhisek@gamebox.net
Gathering information about the remote host is not only based on Operating System detection but you need to gather a whole lot of information about your target for any sort of cracking or sometimes elite activities. I don't want to make this manual too big so I have decided to write "Demystifying Remote Host" in parts. This is the first. I'll start writing the second part very soon which will cover advanced port scanning techniques along with more kewl methodologies.

Abhisek Datta
http://hackersclub.up.to

abhisek@gamebox.net