1 What is a binary log file and how is one created ?

A binary log file is a result of the sniffing operations made to dump the content of the traffic packet, this option results in much faster operation of sniffing program since it doesn't have to spend time in the packet binary->text converters, the packets can be logged into a compact form for later analysis.

Logging in "binary mode" with snort will save the packets in "tcpdump format" to a single binary file in the logging directory (#./snort -l ./log -b).

2 What is MD5 and what value does it provide ?

MD5 was developed by Professor Ronald L. Rivest of MIT. As it was described in the "rfc 1321", the MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.

3 What is the attacker's IP address ?

According to the snort capture, the attack originated from 192.168.102.9.

4 What is the destination IP address ?

According to the snort capture, the attack is conducted to 192.168.102.99.

5 We scanned the honeypot using five different methods. Can you identify the five different scanning methods, and describe how each of the five works

5.1 Ping Sweep [Ref 3]

Definition : A ping sweep is a kind of network probe. In a ping sweep, the intruder sends a set of ICMP ECHO packets to a network of machines (usually specified as a range of IP addresses) and sees which ones respond. The whole point of this is to determine which machines are alive and which aren't. Once the intruder knows which machines are alive, he can focus on which machines to attack and work from there. Note that there are legitimate reasons for performing ping sweeps on a network, in fact, a network administrator may be trying to find out which machines are alive on a network for diagnostic reasons

example from the snort capture : packet number : 1, 2.

5.2 Half open Scan : TCP SYN Scanning [Ref 5]

Definition : This technique is often referred to as "half-open" scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and you wait for a response. A SYN|ACK indicates the port is listening. A RST is indicative of a non-listener. If a SYN|ACK is received, a RST is immediately sent to tear down the connection. Root privileges is needed to build these custom SYN packets.

Sequence number from the snort capture :

- open ports : example : packet N° : 18331, 18332, 18355
- closed ports : exmaple : packet N° : 5, 6

5.3 Stealth Scan : TCP NULL Scanning [Ref 4]

Definition : Clearly through it's endowed named, the NULL scan unsets ALL flags available in the TCP header. ACK, FIN, RST, SYN, URG, PSH all become unassigned. The reserved bits (RES1, RES2) actually do not effect the result of any scan, whether or not they are set clearly does not matter. On arrival of this packet to the server, BSD networking code informs the kernel to drop the incoming call if the port is open.

client -> NULL (no flags)
server -> -

Alternatively, an RST packet will be returned if a closed port has been reached (yes another inverse mapped scan).

client -> NULL (no flags) [Ref 1]
server -> RST

Sequence number from the snort capture :

- closed ports : example : packet N° : 148011, 148012
- open ports : example : packets N° : 148067

5.4 Stealth Scan : XMAS Scanning [Ref 4]

Contrastedly, a so called XMAS scan is the inverse of the NULL scan method. All the available flags in the TCP header are set (ACK, FIN, RST, SYN, URG, PSH). XMAS or "Christmas Tree" scanning is named rightly so after the decorative effect the scan has with the flagging implementation. The reserved bits do not effect the scan result, so setting or unsetting is of no importance. Once again, since this method is based on BSD networking code the technique will only work against UNIX hosts.

XMAS scanning works by initializing all the flags and transmitting this packet to the remote host. The kernel will drop the packet if an open port is at the receiving end. A returned RST flag will reflect a closed, NON-LISTENING port again this is an inverse mapped scan, so false positives is all a client has to detect an open/closed port.

- client -> XMAS (all flags)
- server -> -

This signature tells us that the port is in LISTENING state, or the packet was filtered by a firewall/router. Alternatively a closed port will produce the following reply :

- client -> XMAS (all flags)
- server -> RST

The RST would be sent to the client because the server is tricked into thinking that the client has a connection on that port without negotiating with the standard three-way handshake. Since TCP is stateful the kernel sends a reset bit (RST) back to the client to end transmission immediately.

Sequence number from the snort capture :

- closed ports : example : packets N° : 150761, 150762.
- open ports : example : packets N° : 151175.

5.5 Open Scan : TCP Connect [Ref 5]

Definition : This is the most basic form of TCP scanning. The connect() system call provided by your operating system is used to open a connection to every interesting port on the machine. If the port is listening, connect() will succeed, otherwise the port isn't reachable. One strong advantage to this technique is that you don't need any special privileges. Any user on most UNIX boxes is free to use this call. Another advantage is speed. While making a separate connect() call for every targeted port in a linear fashion would take ages over a slow connection, you can hasten the scan by using many sockets in parallel. Using non-blocking I/O allows you to set a low time-out period and watch all the sockets at once. This is the fastest scanning method supported by nmap, and is available with the -t (TCP) option. The big downside is that this sort of scan is easily detectable and filterable. The target hosts logs will show a bunch of connection and error messages for the services which take the connection and then have it immediately shutdown.

Sequence number from the snort capture :

- closed ports : example : packets N° : 154812, 154813
- open ports : example : packets N° : 154805, 154806, 154811

6 Which scanning tool was used to scan our honeypot ? How were you able to determine this ?

The scanning tool which is used to scan the honeypot is : nmap. We identified the nmap scan accrording to theses arguements :

- A common signature of NMAP is the high source ports. Normally, NMAP's source ports are above 20000 (this feature can be changed with the -p switch). The thought process behind setting the port so high is that some IDS and firewall programs will not flag these scans because of this. That thought process still holds true today in some cases, many times the high source ports alert an IDS analyst or firewall administrator that they are being scanned.

- NMAP's -O function identifies a probbable Operating System to the user. As you can see in Figure 3 NMAP is a combination of packets which include FIN | PUSH | URG, SIN | FIN | PSH, SYN packets and a few UDP packets.

7 What is the purpose of port scanning ?

The purpose of port scanning is to :
- Identify accessible TCP and UDP network services running on the target hosts.
- Access filtering systems between you and the target hosts.
- Guess the operating systems running by analysing IP responses
- Access the TCP sequence number predictability of the target hosts for TCP spoofing and sequence prediction attack potential.

8 What ports were found open on our honeypot ?

According to the snort binary log file, the list of open ports are : 80 (http), 53 (DNS), 443 (https), 111 (sunrpc), 22 (ssh), 32768

9 Bonus : Operating system used by the attacker

There is a wide variety of techniques to determine a host OS. For this case, we will use a passive fingerprinting method, which relies on Window, TTL, ToS, and DF values.

The definition of this terms is :

- **Window :** TCP Packet Window-size - the maximum amount of packets that can be sent out without receiving an acknowledgement.
- **TTL :** Time-To-Live - the maximum number of hops a packet can pass through before being discarded.
- **ToS :** Type of Service.
- **DF :** Don't Fragment bit.

These factors can be used in determining what kind of operating system a remote host is running. Depending on the combination of all of these flags, a match can be ran against a database of flags and an operating system guess can be made. We will use this method to determin not the remote host scanned but the attacker OS.

From the third packet of the binary log file, we have extracted this values :
TTL = 53
Win = 2048
TOS = 0
DF = n (0)

After we have gathered the values, we have to run them against the database of known fingerprints and see if a match can be made. The TTL is no constant since it relies on the number of hops the packet travels through to get from the source host to the destination host. Hence, we'll accept this match and leave the TTL matching over to the Host Path Projection check [Ref 1].

By projecting the path a packet traversed, we can determine a somewhat accurate TTL value and make a possible OS guess. The description of this method is : Take the TTL value of the database and let it lay between that and the preceding TTL value + 1.

TTL value	**TTL good match**
32	0-32
64	33-64
128	65-128
255	129-255

If we run our TTL value against the table above, we come up with the following :

The packet TTL value of 53 lies between the TTL good match value of 33 through 64, so we can assume that the TTL on the target box is probably 64.

From the list of fingerprints for passive fingerprint monitoring [Ref 2], we suggest that the system of the attacker should be one of this described in this table :

OS	**Version**	**Platform**	**TTL**	**Window**	**DF**	**TOS**
SCO	R5	Compaq	64	24820	n	0
FTP(UNIX)	3.3	STRATUS	64	32768	n	0
Unisys	x	Mainframe	64	32768	n	0

Ref 1 : Advanced Remote OS Detection Methods/Concepts using Perl (http ://cert.uni-stuttgart.de/archive/bugtraq/2001/02/msg00195.html)

Ref 2 : http ://project.honeypot.org/papers/finger/traces.txt

Ref 3 : http ://www.linuxjournal.com/article.php ?sid=4234

Ref 4 : http ://www.synnergy.net/downloads/papers/portscan.txt

Ref 5 : http ://www.insecure.org/nmap/nmap_doc.html