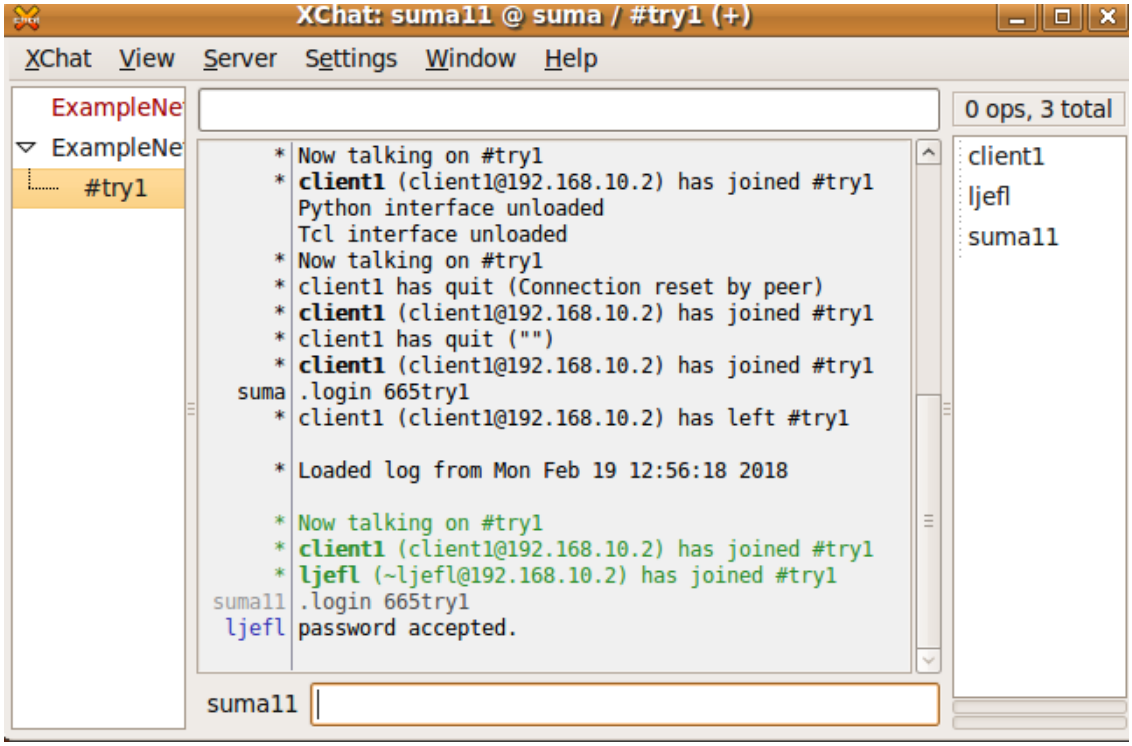
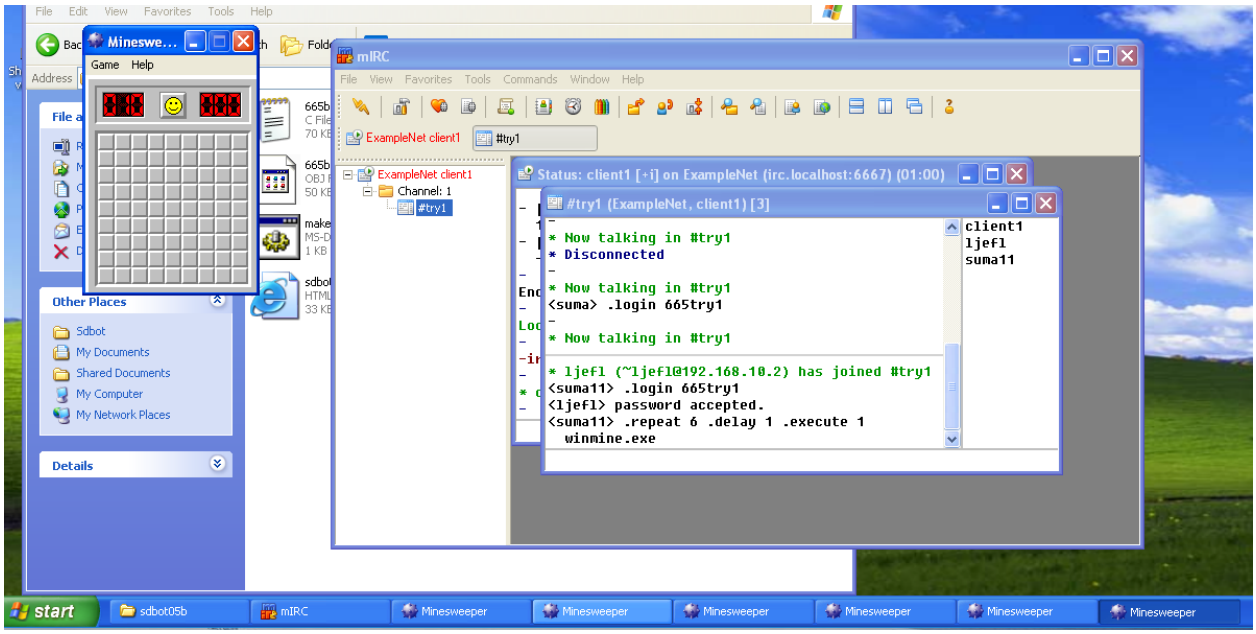
665 HW 2: Botnet Lab and Attack Trace analysis
GARUDA SUMA PRANAVI
UIN 926009146

Task One: Run SDBot

Screenshot One



Q1.1



Ans. The result of .repeat 6 .delay 1 .execute winmine.exe is that 6 minesweeper tabs are opened remotely on the Windows XP machine after executing a command on the Ubuntu XChat server. (each opened with a delay of 1ms-not noticeable)

Task Two: Attack using SDBot

Q 2.2. Which command did you use?

Ans. .udp 192.168.10.3 1000 4096 1 23 #for UDP flood, 1000 4096 byte packets to port 23 of the victim machine, using a 1ms delay

32	4.750138	192.168.10.3	192.168.10.2	ICMP	Destination unreachable (Port unreachable)
36	4.765409	192.168.10.2	192.168.10.3	UDP	Source port: ams Destination port: telnet
39	4.780030	192.168.10.2	192.168.10.3	UDP	Source port: ams Destination port: telnet
42	4.795916	192.168.10.2	192.168.10.3	UDP	Source port: ams Destination port: telnet
45	4.811672	192.168.10.2	192.168.10.3	UDP	Source port: ams Destination port: telnet
48	4.826905	192.168.10.2	192.168.10.3	UDP	Source port: ams Destination port: telnet
51	4.842287	192.168.10.2	192.168.10.3	UDP	Source port: ams Destination port: telnet
55	4.858363	192.168.10.2	192.168.10.3	UDP	Source port: ams Destination port: telnet

Q.2.3. What happens if you don't specify the port number to use for the UDP flood?

3299	17.142122	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3302	17.157767	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3305	17.173394	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3308	17.189076	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3311	17.204960	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3314	17.220504	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3317	17.237853	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3320	17.253907	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3323	17.269424	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux
3326	17.285269	192.168.10.2	192.168.10.3	UDP	Source port: sbl Destination port: tcpmux

If we don't specify the port number, udp port value might have a garbage value (i.e. udp port value might be less than 0) , then the udp port value is set to 1 and the packets are sent to the TCPMUX port (1).It is a multiplexing service accessed with a network protocol used to access a number of available TCP services of a host on a single, well known port number.

Q.2.3. How many bots would be needed to flood a 1Gbit link with UDP packets?

According to the Wireshark Summary, if each bot sends around 1000 packets (according to our command, but the packets received per second is just 64.849), then we would need around 1659 ((10^9)/(64.849*1161.470*8)) bots. However, if we assume that in general each bot sends out only one packet, then we would need around 107,623 bots.

Display

Display filter: ((ip.src == 192.168.10.2) && (ip.dst == 192.168.10.3) && udp)

Traffic	Captured	Displayed	Marked
Packets	3336	1123	0
Between first and last packet	17.532 sec	17.317 sec	
Avg. packets/sec	190.279	64.849	
Avg. packet size	1389.799 bytes	1161.470 bytes	
Bytes	4636369	1304331	
Avg. bytes/sec	264449.993	75320.658	
Avg. MBit/sec	2.116	0.603	

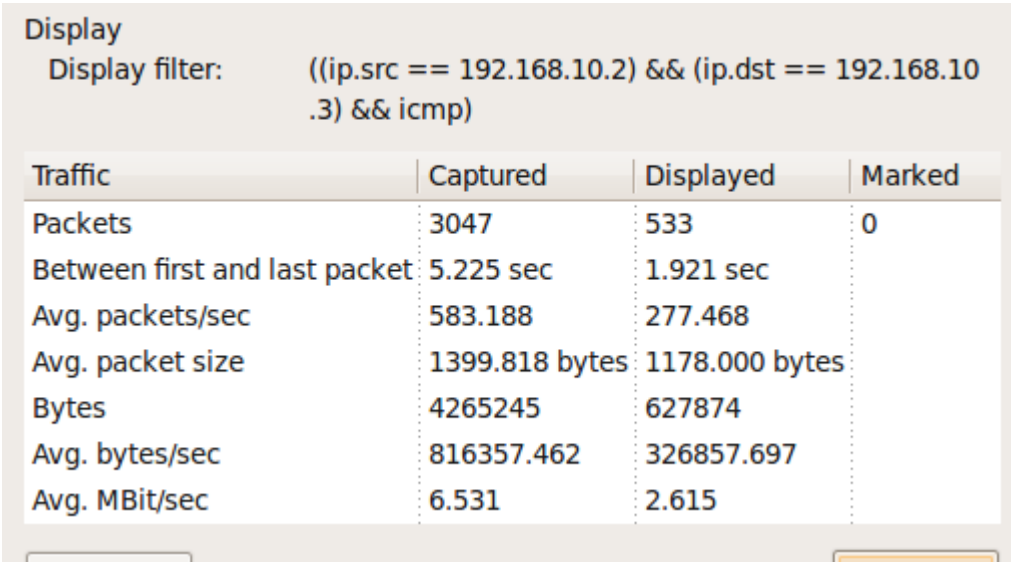
Q.2.4. How might this attack be prevented from the perspective of the flood target? From the perspective of the infected victim?

From the perspective of the flood target, it can limit the no. of UDP packets/sec or no. of UDP packets in total that can be received by an IP address, preventing UDP floods.

The infected victim can use a proxy service like CloudFlare- to proxy all web traffic through its networks and servers, which are heavily fortified to withstand DDoS attacks and also able to intercept common hack attempts. Legitimate traffic will then be forwarded to the web server while suspicious traffic is dropped upstream, leaving the target unaffected.

Q.2.5. What command did you use?
Ans. .ping 192.168.10.3 1000 4096 1 #sending 1000 pings to 192.168.10.3. Wait timeout (1ms)

Q.2.6. How many bots would be needed to flood a 1Gbit link with ICMP packets?



Display

Display filter: ((ip.src == 192.168.10.2) && (ip.dst == 192.168.10.3) && icmp)

Traffic	Captured	Displayed	Marked
Packets	3047	533	0
Between first and last packet	5.225 sec	1.921 sec	
Avg. packets/sec	583.188	277.468	
Avg. packet size	1399.818 bytes	1178.000 bytes	
Bytes	4265245	627874	
Avg. bytes/sec	816357.462	326857.697	
Avg. MBit/sec	6.531	2.615	

If each bot sends 533 packets (according to our command and wireshark summary), it will take 383 bots to congest the link ((10^9)/(277.468*1178*8)). However, if we assume that each bot sends only 1 packet for uniformity, the it will take 106,112 bots.

Q.2.7. From the results of the two floods, which one is more efficient: UDP or ICMP flood?

From the results of our experiment, since the ICMP flood takes less bots to congest the link, it is more efficient. (avg. no of packets per sec is much higher than UDP in ICMP)

Q2.8. Based on your answer to question 2.7, when would you not use the more efficient one?

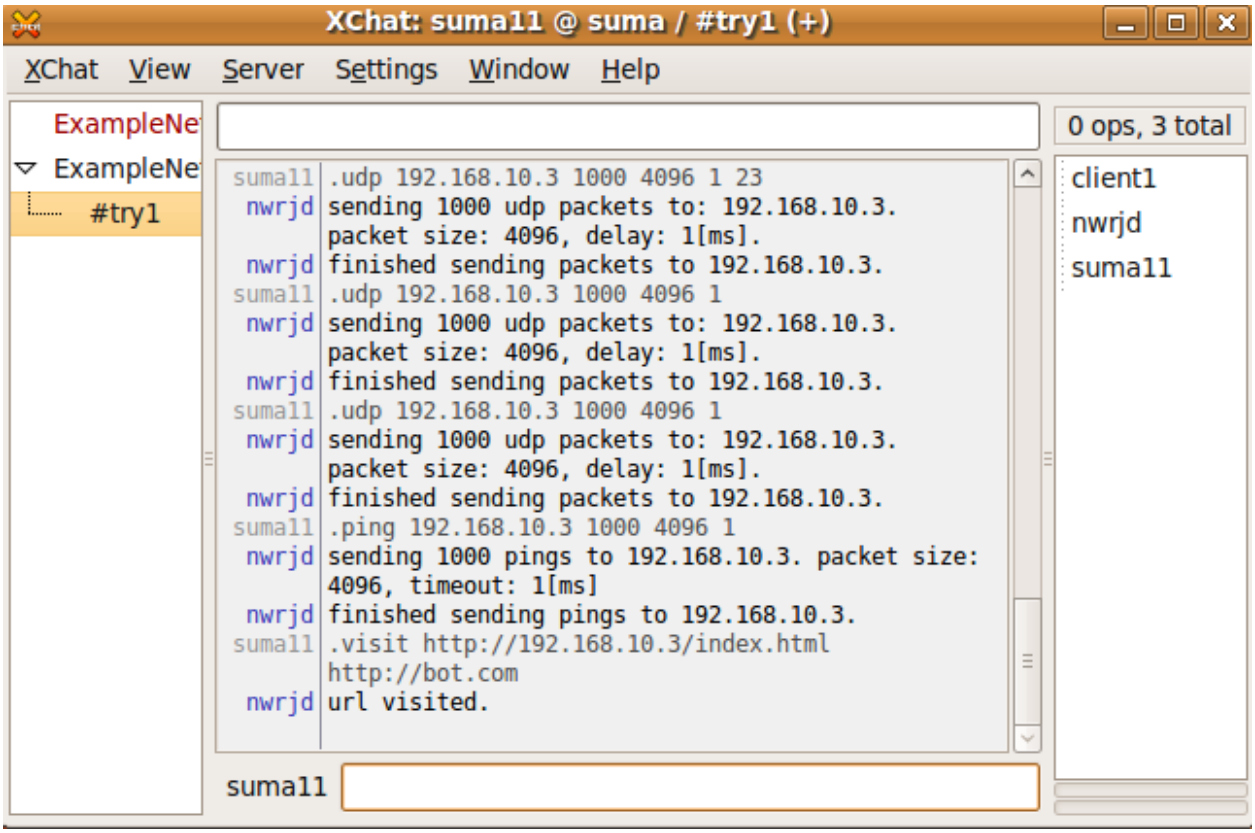
In some cases, we might even encounter that ICMP services/ have been closed, as the device does not want to be probed and hence ICMP floods won't even affect these devices. UDP may be more stealthy, as it is a form of delivering traffic (like video which require a lot of packets/sec) and thus it might not be that easy to differentiate network traffic and attack traffic. Moreover, in some cases, the UDP ports are randomized every 10 packets, hence the attacker will be gaining access/ utilizing resources for a larger surface area. One more possibility, could be if we had anomaly detecting packet rate limiting services on the end devices, there is a better possibility of detecting ICMP.

Fraudulent Pay-per-click count

Screenshot 2:

Source: http://192.168.10.3 /index.html

Referrer: http://bot.com



Task Three: Bot removal

Q3.1. Where are the registry entries? Why are the entries placed in these two locations?

Ans. Start -> Run -> Regedit
Registry entries:
1.HKEY_LOCAL_MACHINE/Software/Microsoft/Windows/CurrentVersion/Run/ConfigurationLoader.
Any program adding something to the startup will be added in run.
2.HKEY_LOCAL_MACHINE/Software/Microsoft/Windows/CurrentVersion/RunServices/ConfigurationLoader.
Applications running as services depend on human interaction.

Software stores information about the how the software will perform on the Windows PC and the default Windows settings. This malware hides in the registry to make it persistent. Moreover, it's not a file which can be scanned easily.

Q.3.2. How would a user know where in their registry the bot is located if the source code were not available for inspection?

If Windows is behaving unexpectedly, the user should know that he should have a look at their registry. All software installed in a system is kept track of in the registry, even if it doesn't have a file assigned to its name. The problems being caused are most probably because some random registry keys got edited or some new services were introduced.

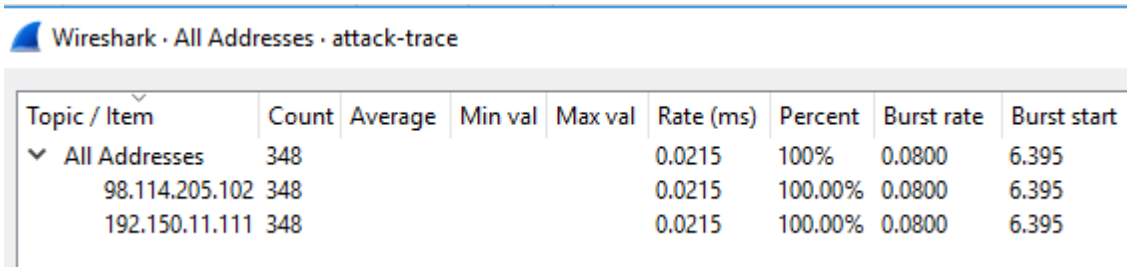
Open source tools such as Regshot which compare the registry values can be used. This tool allows us to take a snapshot of the entire registry when our computer is operating at peak efficiency. It can also take a snapshot of important directories, so that we can later compare to see if any changes were made there.

Task Four: Attack Trace Analysis

Q4.1. What IP addresses (and their roles) are involved?

Ans. Using Statistics -> IPv4 statistics -> All addresses (Wireshark)

98.114.205.102
192.150.11.111



Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
All Addresses	348				0.0215	100%	0.0800	6.395
98.114.205.102	348				0.0215	100.00%	0.0800	6.395
192.150.11.111	348				0.0215	100.00%	0.0800	6.395

We also notice that the first packet has been sent from 98.114.205.102 -> 192.150.11.111. This is a TCP connection establishment packet on port 445 (Microsoft-ds). It is a known vulnerable port and a vector for worm propagation. Therefore, we can conclusively say that 98.114.205.102 is the attacking host and 192.150.11.111 is the victim.

Q.4.2. Where is the attacker located?

Using Online Tools such as Maxmind GeoIP:

GeoIP2 City Results

IP Address	Country Code	Location	Postal Code	Approximate Coordinates*	Accuracy Radius	ISP	Organization	Do
98.114.205.102	US	Philadelphia, Pennsylvania, United States, North America	19154	40.0925, -74.9853	5	Verizon Fios	Verizon Fios	ve

The attacker is located in Philadelphia, Pennsylvania, United States, North America. The exact latitude and longitude is also mentioned.

Q4.3. How many TCP sessions are contained in the PCAP file?

Ans. tshark -r attack-trace.pcap -qnz conv,tcp

```
C:\Program Files\Wireshark>tshark -r attack-trace.pcap -qnz conv,tcp
The NPF driver isn't running.  You may have trouble capturing or
listing interfaces.
================================================================================
TCP Conversations
Filter:<No Filter>
                                                |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                                | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
98.114.205.102:2152       <-> 192.150.11.111:1080      112      6056     159    167332     271    173388     6.142326000       10.0719
98.114.205.102:1828       <-> 192.150.11.111:445        17      1828      14      4997      31      6825     0.134550000        4.9381
192.150.11.111:36296      <-> 98.114.205.102:8884       12      1018      15      1051      27      2069     5.082620000       11.1366
192.150.11.111:1957       <-> 98.114.205.102:1924        6       483       6       334      12       817     2.091833000        3.1000
98.114.205.102:1821       <-> 192.150.11.111:445         3       170       4       242       7       412     0.000000000        0.3543
================================================================================
```

We can see 5 TCP sessions in the capture file (ordered by the amount of bytes exchanged).

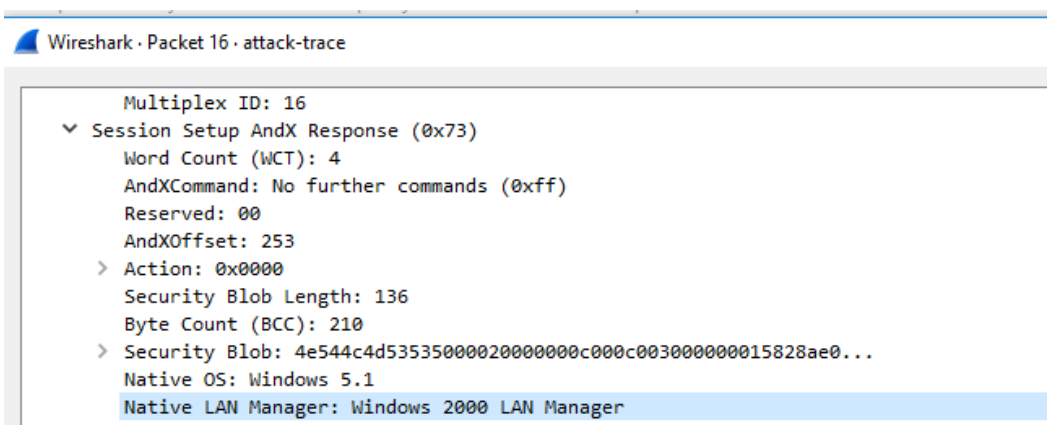
Q. 4.4. How long did the attack last?

Ans. We can see the time difference between the first and last packet through the GUI in Wireshark or use tshark: capinfos attack-trace.pcap -u

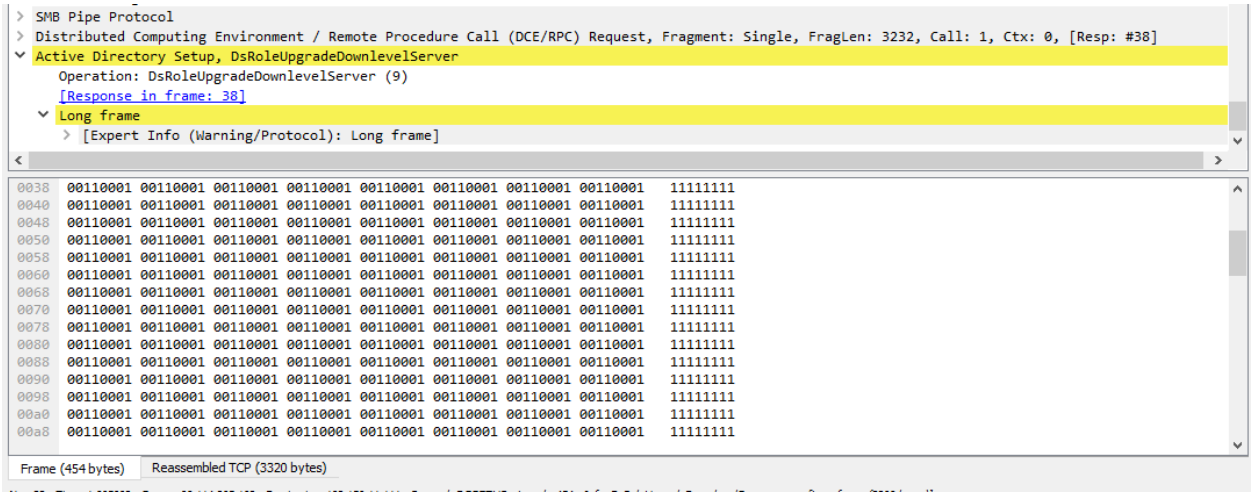
```
C:\Program Files\Wireshark>capinfos attack-trace.pcap -u
File name:           attack-trace.pcap
Capture duration:    16.219218 seconds
```

Q.4.5. Which operating system was targeted by the attack? And which service? Which vulnerability?

Ans. In the analysis we notice that the SMB (server message block) protocol is being used which is an interprocess communication mechanism. SMB sends some OS information in Session Setup and Response.



We can see that Windows 5.1 is being used (which is actually Windows XP) through the Header field.



Attacked Service:

The Active Directory over port 445 (SMB Pipe): LSASS (Local Security Authority Subsystem Service). We can see Wireshark reporting a “long frame” warning and a big list of 1’s.

Vulnerability:

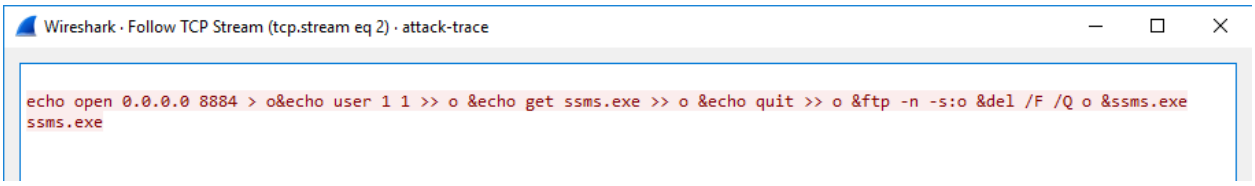
After some research, I found that this type of attack was listed in Common Vulnerabilities and Exposures. (CVE-2003-0533). It is a stack-based overflow in certain Active Directory service functions of LSASS in Windows NT 5.0 SP6a, 2000 SP2 through SP4, XP SP1, Server 2003, NetMeeting, Windows 98, and Windows ME. It allows remote attackers to execute arbitrary code via a packet that causes the DsRolerUpgardeDownlevelServer function (exploited by Sasser Worm).

Q.4.6. Can you sketch an overview of the general actions performed by the attacker?

1. TCP connection 1 is just for Network Reconnaissance (testing for potential vulnerabilities in a computer network). Here, the attacker does a port scan (finds 445 is an open port).

2. In the second TCP connection, SMB pipe protocol is established over 445. It calls the DsRoleUpgradeDownLevelServer() function and exploits the LSASS service through a stack buffer overflow.

3. Once the shellcode is executed through buffer overflow on the victim’s computer, it binds to port 1957 and obtains a shell. The attacker prepares and executes a FTP session form the victim’s computer to his own, with the command (Follow TCP Stream). And try to download ssms.exe.



4.These commands will connect to the FTP server (logging in via a FTP backdoor) and requests a binary to be downloaded and executed on the victim machine.

Q.4.7. What specific vulnerability was attacked?

Ans. The buffer overrun in LSASS (Local Security Authority Subsystem Service) allows remote code execution and once successfully exploited, a remote attacker is able to gain full control of the affected system. (MS04-011 exploit).

The Sasser worm in place here affects computers running vulnerable versions of the Microsoft OS and spreads by exploiting the system through a vulnerable port.

More details mentioned in Q.4.5.

Q.4.9. Was there malware involved? Can you find out the name of the malware?

Ans. Yes, a malware was involved. It is called: Net-Worm.Win32.Sasser.a (Kaspersky), W32/Sasser.worm.a (McAfee), W32.Sasser.gen (Symantec) etc.

Experience/Thought: I think that this was a great assignment, where I got to learn a lot about botnets, vulnerability assessments and questioned the things that I did know. However, there were a few things that I'm not particularly sure about. I did not fully understand why we could capture more than 1000 packets in UDP or less than 1000 in ICMP floods. I went through the source code of the bot, but the assignments had been done properly from the listed arguments. So, it raises a question as to why we have such different observations. It would also be great, if a document could be shared on how to extract shellcode from the wireshark capture and make sense of it. (I had a difficult time trying).