Lab 9: Network Packet Analysis

Video demo: https://youtu.be/Ku7GqmPFBY8 Video demo: https://youtu.be/OSbZQG5AH2A

A Find content

1. Using the following files, perform searches and find the required content:

Obj	PCap file	Search filter	File name found
Find PNG	http://asecuritysite.com/log/with_png.zip	http contains "\x89\x50\x4E\x47"	
Find PDF	http://asecuritysite.com/log/with_pdf.zip	http contains "%PDF"	
Find GIF	http://asecuritysite.com/log/with_gif.zip	http contains "GIF89a"	
Find ZIP	http://asecuritysite.com/log/with_zip.zip	http contains "\x50\x4B\x03\x04"	
Find JPEG	http://asecuritysite.com/log/with_jpg.zip	http contains "\xff\xd8"	
Find MP3	http://asecuritysite.com/log/with_mp3.zip	http contains "\x49\x44\x33"	
Find RAR	http://asecuritysite.com/log/with_rar.zip	http contains "\x52\x61\x72\x21\x1A\x07\x00"	
Find AVI	http://asecuritysite.com/log/with_avi.zip	http contains "\x52\x49\x46\x46"	

2. Investigate the following files and display filters:

Obj	PCap file	Search filter	String that matches
Find Email Addresses	http://asecuritysite.com/log /email_cc2.zip	smtp matches "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9._%+-]"	

Find and IP address	http://asecuritysite.com/log/webpage.zip	http matches "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}.[0-9]{1,3}"	
Find Credit card details	http://asecuritysite.com/log/email_cc2.zip	smtp matches "5\\d{3}(\\s|-)?\\d{4}(\\s|-)?\\d{4}(\\s|-)?\\d{4}"	

3. The following file contains an FTP Hydra attack. Use a filter of: `ftp contains "530 User"` to investigate the following trace:

http://asecuritysite.com/log/hydra_ftp.zip

Outline some of usernames and passwords which have been tried: Can you determine the username and password which has been successful:

4. The following file contains an Telnet Hydra attack. Use a filter of: `ftp contains "530 User"` to investigate the following trace:

http://asecuritysite.com/log/hydra_telnet.zip

Outline some of usernames and passwords which have been tried: Can you determine the username and password which has been successful:

5. The following file contains a Telnet Hydra attack. Use a filter of: `ftp contains "530 User"` to investigate the following trace:

http://asecuritysite.com/log/ hping_syn.zip

Which is the IP address of the computer that is being attack: Which is the IP address of the computer attacking:

B Snort analysis 1

Either use your Windows 2003 instance or Linux Kali. We can also use Snort to analyse network traces, by using an off-line filtering system. Download the following file:

http://asecuritysite.com/log/**newtrace.zip**

For this you can run Snort with a rules file and with a trace:

snort -c 1.rules -l log -r newtrace.pcap

You can then look in the log filter for the log file and alert.ids.

Some rules you can use are given in Appendix A.

Now test Snort to see if it can detect the same content that you found before: Number of Bad FTP logins: Number of Successful FTP logins:

Number of GIF files in the trace:

Number of PNG files in the trace:

Can you detect the port scan on a host:

C Snort analysis 2

Use the following PCAP files, and detect the activity:

Objective: Detect bad FTP login.

Trace: http://asecuritysite.com/log/hydra_ftp.zip.

Rules used to detect:

Objective: Detect Telnet login.

Trace: http://asecuritysite.com/log/hydra_telnet.zip.

Rules used to detect:

Objective: Detect port scan.

http://asecuritysite.com/log/nmap.zip.

Rules used to detect:

Objective: Detect SYN flood.

http://asecuritysite.com/log/hping_syn.zip.

Rules used to detect:

Objective: Detect FIN flood.

http://asecuritysite.com/log/hping_fin.zip.

Rules used to detect:

Objective: Detect file attachments.

http://asecuritysite.com/log/email_two_attachments.zip.

Rules used to detect:

Objective: Detect credit card details and also email addresses.

http://asecuritysite.com/log/email_cc2.zip.

Rules used to detect:

Objective: Detect ping sweep.

http://asecuritysite.com/log/ping_sweep.zip.

Rules used to detect:

Can your extract the file and access it?

Objective: Detect PDF files.

http://asecuritysite.com/log/with_pdf.zip.

Rules used to detect:

Can your extract the file and access it?

Objective: Detect SNMP connections

http://asecuritysite.com/log/snmp.zip.

Rules used to detect:

Can your extract the file and access it?

Objective: Detect MP3 connections

http://asecuritysite.com/log/with_mp3.zip

Rules used to detect:

Can your extract the files and access them?

What is the sound file and what are the graphics?

Objective: Detect and extract RAR files

http://asecuritysite.com/log/with_rar.zip

Rules used to detect:

What is the name of the RAR file:

Can your extract the file and access it?

What are the contents of the file?

Objective: Detect and extract Zip files

http://asecuritysite.com/log/with_zip.zip

Rules used to detect:

What is the name of the ZIP file:

Can your extract the file and access it?

Objective: Detect and extract GZip files

http://asecuritysite.com/log/with_gzip.zip

Rules used to detect:

What is the name of the GZip file:

Can your extract the file and access it?

Objective: Detect and extract AVI files

http://asecuritysite.com/log/with_avi.zip

Rules used to detect:

What is the name of the AVI file:

Can your extract the file and access it?

Objective: Detect BitTorrent

http://asecuritysite.com/log/bit.zip

Rules used to detect:

Appendix

Bad logins:

```
alert tcp any 21 -> any any (msg:"FTP Bad login"; content:"530 User "; nocase; flow:from_server,established; sid:491;
rev:5;)
```

Detecting email addresses:

```
alert tcp any any <> any 25 (pcre:"/[a-zA-Z0-9._%+-]+@[a-zA-Z0-9._%+-]/"; \
msg:"Email in message";sid:9000000;rev:1;)
```

Detect DNS:

```
alert udp any any -> any 53 (msg: "DNS"; sid:10000;)
```

File types:

```
alert tcp any any -> any any (content:"GIF89a"; msg:"GIF";sid:10000)
alert tcp any any -> any any (content:"%PDF"; msg:"PDF";sid:10001)
alert tcp any any -> any any (content:"|89 50 4E 47|"; msg:"PNG";sid:10002)
alert tcp any any -> any any (content:"|50 4B 03 04|"; msg:"ZIP";sid:10003)
```

Telnet login:

```
alert tcp any any <> any 23 (flags:S; msg:"Telnet Login";sid:9000005;rev:1;)
```

Port scan:

```
preprocessor sfportscan:\
        proto { all } \
        scan_type { all } \
        sense_level { high } \
        logfile { portscan.log }
```

DoS on Web server:

```
alert tcp any any -> any 80 (msg:"DOS flood denial of service attempt";flow:to_server; \
detection_filter:track by_dst,  count 60, seconds 60; \
sid:25101; rev:1;)
```

Stealth scans:

```
alert tcp any any -> any any (msg:"SYN FIN Scan"; flags: SF;sid:9000000;)
alert tcp any any -> any any (msg:"FIN Scan"; flags: F;sid:9000001;)
alert tcp any any -> any any (msg:"NULL Scan"; flags: 0;sid:9000002;)
alert tcp any any -> any any (msg:"XMAS Scan"; flags: FPU;sid:9000003;)
alert tcp any any -> any any (msg:"Full XMAS Scan"; flags: SRAFPU;sid:9000004;)
alert tcp any any -> any any (msg:"URG Scan"; flags: U;sid:9000005;)
alert tcp any any -> any any (msg:"URG FIN Scan"; flags: FU;sid:9000006;)
alert tcp any any -> any any (msg:"PUSH FIN Scan"; flags: FP;sid:9000007;)
alert tcp any any -> any any (msg:"URG PUSH Scan"; flags: PU;sid:9000008;)
alert tcp any any -> any any (flags: A; ack: 0; msg:"NMAP TCP ping!";sid:9000009;)
```

ping sweep:

```
alert icmp any any -> any any (msg:"ICMP Packet found";sid:9000000;)
alert icmp any any -> any any (itype: 0; msg: "ICMP Echo Reply";sid:9000001;)
alert icmp any any -> any any (itype: 3; msg: "ICMP Destination Unreachable";sid:9000002;)
alert icmp any any -> any any (itype: 4; msg: "ICMP Source Quench Message received";sid:9000003;)
alert icmp any any -> any any (itype: 5; msg: "ICMP Redirect message";sid:9000004;)
alert icmp any any -> any any (itype: 8; msg: "ICMP Echo Request";sid:9000005;)
alert icmp any any -> any any (itype: 11; msg: "ICMP Time Exceeded";sid:9000006;)
```

Note you may have to add the following for the stream analysis:

```
preprocessor stream5_global: track_tcp yes, \
```

```
   track_udp yes, \
   track_icmp no, \
   max_tcp 262144, \
   max_udp 131072, \
   max_active_responses 2, \
   min_response_seconds 5
preprocessor stream5_tcp: policy windows, detect_anomalies, require_3whs 180, \
   overlap_limit 10, small_segments 3 bytes 150, timeout 180, \
    ports client 21 22 23 25 42 53 70 79 109 110 111 113 119 135 136 137 139 143 \
        161 445 513 514 587 593 691 1433 1521 1741 2100 3306 6070 6665 6666 6667 6668 6669 \
        7000 8181 32770 32771 32772 32773 32774 32775 32776 32777 32778 32779, \
    ports both 80 81 82 83 84 85 86 87 88 89 90 110 311 383 443 465 563 591 593 631 636 901 989 992 993 994 995 1220 1414 1830 2301 2381 2809 \
       3037 3057 3128 3443 3702 4343 4848 5250 6080 6988 7907 7000 7001 7144 7145 7510 7802 7777 7779 \
       7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910 7911 7912 7913 7914 7915 7916 \
        7917 7918 7919 7920 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180 8222 8243 8280 8300 8500 8800 8888 8899 9000 9060 9080 9090 \
        9091 9443 9999 10000 11371 34443 34444 41080 50000 50002 55555
preprocessor stream5_udp: timeout 180
```

D. TCP Forensics

Details

Aim: To provide a foundation in analysing TCP packets

Activities

1. .NET provides an excellent interface to capturing and reading back data packets. For this lab download the solution from:

↳http://www.dcs.napier.ac.uk/~bill/tcpForensics.zip

It has a Windows interface, such as:

Figure 1: Inteface

2. For the **Open** button add the following code:

```
PcapDevice device=null;
Packet packet=null;

openFileDialog1.ShowDialog();

try
{
   device = SharpPcap.GetPcapOfflineDevice(openFileDialog1.FileName);
   device.PcapOpen();

}
catch (Exception e1)
{
   MessageBox.Show("Error: " + e1.Message);
```

```
        return;
    }

    while( (packet=device.PcapGetNextPacket()) != null )
    {
        if (packet is TCPPacket)
        {
            TCPPacket tcp = (TCPPacket)packet;
            string srcIp = tcp.SourceAddress;
            string dstIp = tcp.DestinationAddress;
            int srcPort = tcp.SourcePort;
            int dstPort = tcp.DestinationPort;

            DateTime time = packet.PcapHeader.Date;
            int len = packet.PcapHeader.PacketLength;

            this.lbOutput.Items.Add(showFlags(tcp)+" Time: " +time.Hour+":"
                + time.Minute+ ":"+time.Second+
                " IP Src: " + srcIp+ " TCP Src " + srcPort+
                " IP Dest: " + dstIp+ " TCP Dest " + dstPort);

            ASCIIEncoding utf = new System.Text.ASCIIEncoding();
            string s = utf.GetString(tcp.Data);

            this.lbOutput.Items.Add("  Content: " + s);

        }
    }
```

3. Now download the file:

↳http://www.dcs.napier.ac.uk/~bill/capture2.zip

Read the file in, and determine the start of each conversation with the server, and complete Table 1 (note that the first entry has already been added).

Note: Identify a connection with the SYN, SYN/ACK and ACK flag sequence.

What is the domain name of the remote server?

What is the application protocol used?

For the first connection what is the HTTP request send (note look for commands such as GET, Accept: and so on)?

For the first connection what is the format of the HTTP reply (note look for a request such as HTTP/1.1 200)?

Table 1:

Connection	Src IP	Src Port	Dst IP	Dst Port
1	*192.168.1.102*	*1386*	*66.102.9.147*	*80*
2				
3				
4				
5				
6				
7				
8				

4. Now download the file:

↳**http://www.dcs.napier.ac.uk/~bill/capture2.zip**

Read the file in, and determine the start of each conversation with the server, and complete Table 1 (note that the first entry has already been added).

Note: Identify a connection with the SYN, SYN/ACK and ACK flag sequence.

What is the domain name of the remote server?

What is the trace of the traffic to and from the client to the server:

Which TCP ports are used on the server:

Table 1:

Connection	Src IP	Src Port	Dst IP	Dst Port
1	*192.168.1.102*	*1433*	*198.175.98.64*	*21*
2				
3				
4				
5				
6				
7				
8				