Assignment 4
Detection of Alerts and Analyzing Network Traffic using Snort IDS

A Snort analysis

You can perform offline Snort IDS by analyzing pcap files. To do so, you run Snort with a rules file and with a trace pcap file.

- Download **http://asecuritysite.com/log/newtrace.zip,** and extract **newtrace.pcap**
- Download https://asecuritysite.com/log/rules.zip, and extract **rules.rules**
- Put both files in the **bin** directory of Snort.
- In **rules.rules** file, keep only the initial content as shown in Appendix A. The rest of content will be used when needed.
- Run Snort and use the rules given in Appendix A or in https://asecuritysite.com/subjects/chapter15, to answer the below questions, and provide the screenshot evidences in a separate file (from command line execution or log files):

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:

For example, if the path installation of Snort is C:\Snort, run Snort with:

C:\Snort\bin>snort -c rules.rules -l C:\Snort\log -r newtrace.pcap

The alerts relevant to the rules will be generated in **alert.ids** in the **log** folder.
For each Snort run, you replace the new rule with the old one.
Optional: You can choose to delete the content of **alert.ids** before each run to only keep the logs of the newest run.

B Snort analysis

In the following, if it is required, you can extract your artefacts in an HTTP trace with:

File-> Export Objects -> HTTP, select the file and click on **Save.**

Answer the below questions using Appendix A, https://asecuritysite.com/subjects/chapter15, or https://asecuritysite.com/forensics/snort2

Use the following PCAP files and their corresponding rules to run Snort, and provide the screenshot evidences of successful detection (from command line execution or log files)

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.

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

Rules used to detect: ?

Objective: Detect SYN flood.

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

Rules used to detect: ?

Objective: Detect FIN flood.

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

- Rules used to detect: ?

Objective: Detect file attachments.

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

- Rules used to detect: ?

Objective: Detect credit card details and also email addresses.

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

- Rules used to detect: ?

Objective: Detect ping sweep.

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

- Rules used to detect: ?

Objective: Detect PDF files.

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

- Rules used to detect: ?
- Can your extract the pdf file?

Objective: Detect MP3 connections

Trace: 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

Trace: 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

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

- Rules used to detect: ?
- What are the names of the ZIP files ?
- Can your extract the files and access them?

Objective: Detect and extract GZip files

Trace: 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

Trace: 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

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

- Rules used to detect: ?

Appendix A

Bad FTP logins:

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

Successful FTP logins:

```
alert tcp any 21 -> any any (msg:"FTP Good login"; content:"230 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;)
```

Initial content of rules.rules:

```
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, \s
    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
```