Author : Praveen Kumar Pendyala
Email : m@praveen.xyz

Problem 4.1

Task 4.1.1

Using a fully content based filter would require to makes assumptions about the contents of the *index.html* as well. So, I combined the content based filter with URL based filter to avoid false alarms with *index.html*

```
alert tcp any any -> $HTTP_SERVERS $HTTP_PORTS (\
                msg:"NetSec Exam Page Access Requested"; \
                sid:100002; rev:1; \
                pcre:!"/index.html/i"; \
                flowbits:set, epage_access; \
                flowbits: noalert; )
alert tcp any any <> $HTTP_SERVERS $HTTP_PORTS (\
                msg:"NetSec Exam Page Access"; \
                sid:100003; rev:1; \
                file_data; pcre:"/Network Security Exam \d{4}/i"; \
                flowbits:isset, epage_access; \
                priority:3; )
```

Task 4.1.2

Exam file Download rule :

```
alert tcp any any -> $HTTP_SERVERS $HTTP_PORTS (\
                msg:"NetSec Exam Download Requested"; \
                sid:100004; rev:1; \
                pcre:"/netsec\/exam\d{4}.pdf/i"; \
                flowbits:set, exam_download; \
                flowbits:noalert; )
alert tcp any any <> $HTTP_SERVERS $HTTP_PORTS (\
                msg:"NetSec Exam Download"; \
                sid:100005; rev:1; \
                file_data; pcre:"/pdf/i"; \
                flowbits:isset, exam_download; \
                priority:2; )
```

Task 4.1.3
Solution Access Rule :

```
alert tcp any any -> $HTTP_SERVERS $HTTP_PORTS (\
                msg:"NetSec Solution Access Requested"; \
                sid:100006; rev:1; \
                pcre:"/netsec\/exam\d{4}_solution-v\d{3}.jpg/i"; \
                flowbits:set, solution_access; \
                flowbits:noalert;)
alert tcp any any <> $HTTP_SERVERS $HTTP_PORTS (\
                msg:"NetSec Solution Access"; \
                sid:100007; rev:1; \
                content:"200"; http_stat_code; \
                flowbits:isset, solution_access; \
                priority:1; )
```

Problem 4.2

Task 4.2.1

a) OS : Linux 2.6.X
   Reasons for the guess :
   - `sudo nmap -Pn 130.83.194.149 -v -O` guesses it to be Linux
   - Apache version says "`Apache httpd 2.2.22 ((Debian))`"
   - OpenSSH version reads "`OpenSSH 6.0p1 Debian 4+deb7u2`"

   All of them hint in favour of a Linux based Operating system

b) Yes.
   - SSH daemon is running on port 22222
   - Public key : b8:62:e6:08:86:09:55:2d:10:cd:8e:12:f3:8b:a6:d9

c) Total ports found on port scan : 16 (11 - Open, 5 - filtered)

   Port stats :
   Open : 80, 1337, 8080, 8443, 22222, 27017, 28017, 30000, **54430, 57269**, 64738
   Filtered : 135, 139, 445, 1433, 1434
   Note : 54430, 57269 could be ephemeral ports. Their availability was not consistent across different scans.

   Unusual ports : 22222, 27017, 28017, 30000, 54430, 57269, 64738

d) Services and their ports

Port	Service	Comments / Explanation
80	http	common port for http service
1337	postgresql	Normal postgresql ports : 5432 / 5433. Port 1337 is normally used by neo4j-shell
8080	ftp	Normal FTP port : 21 8080 used commonly by HTTP alternative for Web proxy / Caching
8443	http	8443 is also used by Apache Tomcat SSL
22222	ssh	Unusual port for ssh ssh normally runs on port 22
27017	mongod, mongos	The default port for mongod and mongos instances
28017	mongodb	Default port for Mongo web status page
30000	http	Unusual port for http http normally runs on ports 80
64738	murmur	Default port used by murmur service

Justification :

- Simply by running **“`nmap -Pn -sV 130.83.194.149 -v -p 1-65535`”**
- Checking ports 27017, 28017 for the common service that uses them, MongoDB. Which is indeed the case.

e) Services and Applications used to offer them

Service (port)	Application used	Version
http (80)	Apache	Apache httpd 2.2.22 ((Debian))
postgresql (1337)	PostgreSQL	PostgreSQL 9.1.14
ftp (8080)	ProFTPD	ProFTPD 1.3.4a
http (8443)	Apache	Apache httpd 2.2.22
ssh (22222)	OpenSSH	OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
mongodb (27017, 28017)	MongoDB	MongoDB v2.4.9
http (30000)	MiniServ	MiniServ 1.740 (Webmin httpd)
murmur (64738)	Murmur	Murmur 1.2.4

Task 4.2.2

Token collection:

1. Service : HTTP 80
   Token : Netsec15-B1-Xc0538fd7f566e7cddbb9153b4db218e6
   Method :
   - Accessing *http://130.83.194.149/* from browser

2. Service : Postgresql 1337
   Token : Netsec15-E1-N3t$ec-RUL3Z
   Method :
   - Connected to postgresql remote server using given credentials
   - Checked available info using :

```
SELECT table_schema,table_name
FROM information_schema.tables
ORDER BY table_schema,table_name;
```

   - Below query to get token :

```
SELECT * FROM token;
```

3. Service : FTP 8080
   Token : Netsec15-A2-7b7b2fce18392d7c4898a39bcdf8313a
   Method :
   - Accessing welcome message on FTP server

4. Service : HTTP 8443
   Token : Netsec15-X1-AllYour$3rver$bel0ngt0u$
   Method :
   - Accessing *https://netsec-recon-a.net.hrz.tu-darmstadt.de:8443/login.php* from browser

5. Service : HTTP 30000
   Token : Netsec15-Y1-Winni3PuhI$Gr33tingU
   Method :
   - Accessing https://netsec-recon-a.net.hrz.tu-darmstadt.de:30000/ from browser

6. Service : Murmur
   Token : Netsec15-A1-1337Haxx0r
   Method :
   - Used Mumble and connected to Murmur using given credentials
   - Token visible after connection

7. Service : MongoDB
   Token : Netsec15-D1-3xpl0it$
   Method :
   - Connected to MongoDB remote server on port 27017
   - “show collections” to see list of collections
   - “db.token.find()” to check the token

8. Service : HTTP 8443
   Token : TOKEN-Netsec15-X2-2c6a53959a059eda4497b8e234d81bf1
   Username : netsec
   Password : 177f2428e8
   Method :
   - Exploited the SSL Heartbleed bug using https://github.com/sensepost/heartbleed-poc
   - The credentials were exposed in the heartbeat response from server
   - It did take a couple of failed attempts since the leaked credentials were not always the valid set.

Messages left in services:

All hashes are SHA1

1. Service : Murmur
   Hash : d3a4a17a005686c349aeae1c66d68df23bcc7914
   Plain value : praveendath92

2. Service : Postgresql
   Secrethash : d3a4a17a005686c349aeae1c66d68df23bcc7914
   id : 526
   Plain value : praveendath92

3. Service : FTP
   File name : praveen
   Contents : praveendath92

4. Service : MongoDB
   ObjectId : 55750c5da57d72046445f0ff
   Secrethash : d3a4a17a005686c349aeae1c66d68df23bcc7914
   Plain value : praveendath92