

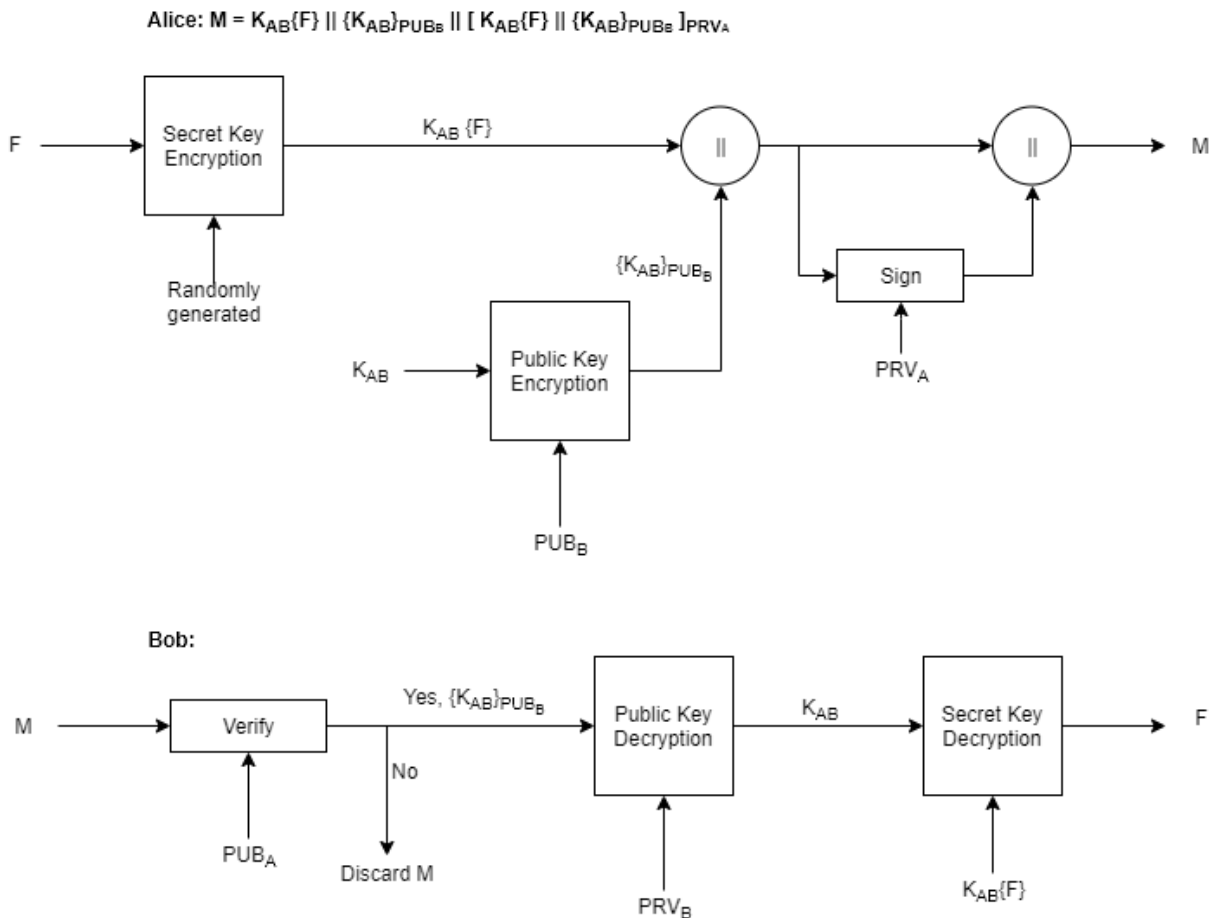
Samuel Steinberg

November 4<sup>th</sup>, 2019

### Assignment 3

#### Part I

##### Problem 1:



##### Problem 2:

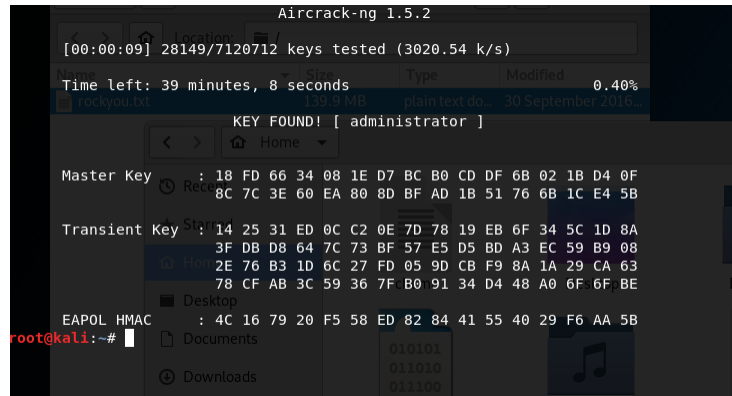
If random numbers  $R_c$  and  $R_s$  not used in the TLS protocol then both the client and server would not be able to derive encryption and Master Secret from the PreMaster Secret.  $R_c$  is a nonce created by the client, and  $R_s$  is a nonce created by the server. If there was an attack, the messages would not be encrypted and would be plain text readable for all attackers. The attack would be successful because without randomness in the nonce it would allow the server to use the same signature for many protocol sessions, instead of creating a new one for each. Additionally, the usage of only server nonces in signatures but not client ones would make the protocol even more vulnerable to attacks and further increase the efficiency and practicality of the successful attack.

## Part II

### Task 1:

#### Q1:

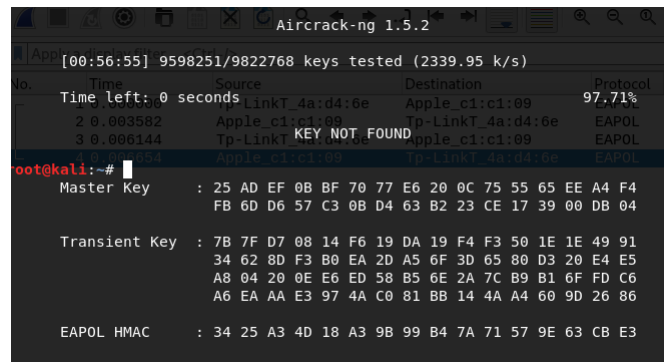
The password used to create the WPA cracking Demo-01.cap was **administrator**. This password was found in the dictionary **rockyou.txt**. Below is a screenshot of the success:



```
Aircrack-ng 1.5.2
[00:00:09] 28149/7120712 keys tested (3020.54 k/s)
Time left: 39 minutes, 8 seconds
Type: Modified: 0.40%
KEY FOUND! [ administrator ]
Master Key : 18 FD 66 34 08 1E D7 BC B0 CD DF 6B 02 1B D4 0F
8C 7C 3E 60 EA 80 8D BF AD 1B 51 76 6B 1C E4 5B
Transient Key : 14 25 31 ED 0C C2 0E 7D 78 19 EB 6F 34 5C 1D 8A
3F DB D8 64 7C 73 BF 57 E5 D5 B0 A3 EC 59 B9 08
2E 76 B3 1D 6C 27 FD 05 9D CB F9 8A 1A 29 CA 63
78 CF AB 3C 59 36 7F B0 91 34 D4 48 A0 6F 6F BE
EAPOL HMAC : 4C 16 79 20 F5 58 ED 82 84 41 55 40 29 F6 AA 5B
root@kali:~#
```

#### Q2:

The password to create the WPA cracking Demo-02.cap could not be found, as neither dictionary contained the password. Even the larger rockyou.txt dictionary file did not contain it, even after 56 minutes of searching:



```
Aircrack-ng 1.5.2
[00:56:55] 9598251/9822768 keys tested (2339.95 k/s)
Time left: 0 seconds
Source: Destination: Protocol: Length: Info:
2 0.003582 Apple_c1:c1:09 Tp-LinkT_4a:d4:6e EAPOL 155 Key (Message 2 of 4)
3 0.006144 Tp-LinkT_4a:d4:6e Apple_c1:c1:09 EAPOL 237 Key (Message 3 of 4)
KEY NOT FOUND
root@kali:~#
Master Key : 25 AD EF 0B BF 70 77 E6 20 0C 75 55 65 EE A4 F4
FB 6D D6 57 C3 0B D4 63 B2 23 CE 17 39 00 DB 04
Transient Key : 7B 7F D7 08 14 F6 19 DA 19 F4 F3 50 1E 1E 49 91
34 62 8D F3 B0 EA 2D A5 6F 3D 65 80 D3 20 E4 E5
A8 04 20 0E E6 ED 58 B5 6E 2A 7C B9 B1 6F FD C6
A6 EA AA E3 97 4A C0 81 BB 14 4A A4 60 9D 26 86
EAPOL HMAC : 34 25 A3 4D 18 A3 9B 99 B4 7A 71 57 9E 63 CB E3
```

#### Q3:

There are four total messages displayed in the following format:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Tp-LinkT_4a:d4:6e	Apple_c1:c1:09	EAPOL	133	Key (Message 1 of 4)
2	0.003582	Apple_c1:c1:09	Tp-LinkT_4a:d4:6e	EAPOL	155	Key (Message 2 of 4)
3	0.006144	Tp-LinkT_4a:d4:6e	Apple_c1:c1:09	EAPOL	237	Key (Message 3 of 4)
4	0.006654	Apple_c1:c1:09	Tp-LinkT_4a:d4:6e	EAPOL	133	Key (Message 4 of 4)

These are messages between the host, Apple, and the AP which is the TPLink. Each message has return data from the intercepted packers that expand when selected.

## Q4:

- The Anonce value is displayed below:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Tp-LinkT_4a:d4:6e	Apple_c1:c1:09	EAPOL	133	Key (Message 1 of 4)
2	0.003582	Apple_c1:c1:09	Tp-LinkT_4a:d4:6e	EAPOL	155	Key (Message 2 of 4)
3	0.006144	Tp-LinkT_4a:d4:6e	Apple_c1:c1:09	EAPOL	237	Key (Message 3 of 4)
4	0.006654	Apple_c1:c1:09	Tp-LinkT_4a:d4:6e	EAPOL	133	Key (Message 4 of 4)

```
Frame 3: 237 bytes on wire (1896 bits), 237 bytes captured (1896 bits) on 0
IEEE 802.11 QoS Data, Flags: ....F.
Logical-Link Control
802.1X Authentication
  Version: 802.1X-2004 (2)
  Type: Key (3)
  Length: 199
  Key Descriptor Type: EAPOL RSN Key (2)
  [Message number: 3]
  Key Information: 0x13ca
  Key Length: 16
  Replay Counter: 2
  WPA Key Nonce: 38ca952b886c20515757f16b9b5792f139bd3bfb636405a...
  Key IV: 00000000000000000000000000000000
  WPA Key RSC: c200000000000000
  WPA Key ID: 0000000000000000
  WPA Key MIC: 4ed3b6b48799d8d6e489e947d385b583
  WPA Key Data Length: 104
  WPA Key Data: 6dbc07a0d926b459a3ecceb344d9896983fb6b96555fdcf9...
```

```
0000 88 02 3a 01 70 56 81 c1 c1 09 60 e3 27 4a d4 6e  :.:pV...:.'J.n
0010 60 e3 27 4a d4 6e 10 00 06 00 aa aa 03 00 00 00  :.'J.n.....
0020 88 8e 02 03 00 c7 02 13 ca 00 10 00 00 00 00 00  :...8...+ 1 Qwwqk.
0030 00 00 02 88 ca 95 2b 88 6c 20 51 57 57 71 6b 9b  :...9...;..6@Z...1
0040 57 92 f1 39 bd 3b fb e6 36 40 5a 1e e2 8d b8 09  :.\.....
0050 ea 5c c9 00 00 00 00 00 00 00 00 00 00 00 00  :.....
0060 00 00 00 c2 00 00 00 00 00 00 00 00 00 00 00  :.....
0070 00 00 00 4e d3 b6 b4 87 99 d8 d6 e4 89 e9 47 d3  :...N....:G.
0080 85 b5 83 00 68 6d bc 07 a0 d9 26 b4 59 a3 ec ce  :...hm...:&.Y...
0090 b3 44 d9 89 69 83 fb 6b 96 55 5f df c9 1b d4 94  :.D..i..k..U_....
00a0 a4 51 44 3e b6 71 e8 e1 a5 06 1c ee 50 ad d7 82  :.QD>..q...:P...
00b0 27 e6 06 a9 51 3f 4e f4 05 ca 8c e1 53 11 0f 3d  :'....Q?N...:S...=
00c0 f8 71 78 56 91 bd 84 7f 08 42 64 d3 90 85 20 af  :.qxV....:Bd...
00d0 14 9e 33 cb d1 0d d6 1d 58 b9 3a 38 b6 72 f4 79  :.3....:X.:8.r.y
00e0 3a e7 90 00 3f d8 1b 51 29 2d a4 5e 90  :...?..Q )-..^.
```

The length is 64 Hex digits, equivalent to 256 bits.

- The Snonce value is displayed below:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Tp-LinkT_4a:d4:6e	Apple_c1:c1:09	EAPOL	133	Key (Message 1 of 4)
2	0.003582	Apple_c1:c1:09	Tp-LinkT_4a:d4:6e	EAPOL	155	Key (Message 2 of 4)
3	0.006144	Tp-LinkT_4a:d4:6e	Apple_c1:c1:09	EAPOL	237	Key (Message 3 of 4)
4	0.006654	Apple_c1:c1:09	Tp-LinkT_4a:d4:6e	EAPOL	133	Key (Message 4 of 4)

```

> Frame 2: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits)
> IEEE 802.11 QoS Data, Flags: .....T
> Logical-Link Control
> 802.1X Authentication
  Version: 802.1X-2001 (1)
  Type: Key (3)
  Length: 117
  Key Descriptor Type: EAPOL RSN Key (2)
  [Message number: 2]
  > Key Information: 0x010a
    Key Length: 0
    Replay Counter: 1
  WPA Key Nonce: 51274e0945c2bde61dc44c1cb64cc470242ce311d521622c...
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: 0000000000000000
    WPA Key ID: 0000000000000000
    WPA Key MIC: 732731b0f1e21d7c13212ecf9151fbd9
    WPA Key Data Length: 22
  > WPA Key Data: 30140100000fac020100000fac040100000fac020000

```

```

0000 88 01 3a 01 60 e3 27 4a d4 6e 70 56 81 c1 c1 09  ...:.'J'npV...
0010 60 e3 27 4a d4 6e 00 00 00 00 aa aa 03 00 00 00  ...:.'J'n.....
0020 88 8e 01 03 00 75 02 01 0a 00 00 00 00 00 00 00  ...:..u.....
0030 00 00 01 51 27 4e 09 45 c2 bd e6 1d c4 4c 1c b6  ...:.'Q'N'E.....L...
0040 4c c4 70 24 2c e3 11 d5 21 62 2c 02 83 bb b4 3c  L.p$,...!b,...<
0050 03 84 05 00 00 00 00 00 00 00 00 00 00 00 00 00  ...:.....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0070 00 00 00 73 27 31 b0 f1 e2 1d 7c 13 21 2e cf 91  ...:.'s'1... ..|..!...
0080 51 fb d9 00 16 30 14 01 00 00 0f ac 02 01 00 00  Q....0... ..
0090 0f ac 04 01 00 00 0f ac 02 00 00  .....

```

The length is 64 Hex digits, equivalent to 256 bits.

- There are three different non-zero MIC's in the messages. Each is 32 Hex digits in length, equivalent to 128 bits.
- A nonce is used in this scenario (and should only be used once with a given key) because it is important to ensure old keys/communications cannot be used in repeat attacks and is issued in an authentication protocol. A nonce random number used only once that acts as a placeholder for security strategies by making the authentication or hash unique. Message Authentication Codes (MACs) are important here because it provides an integrity check to prevent the adversary

from altering a message without being detected. It cannot, however, stop an adversary from modification or recover from the modification. It requires a secret key. Verification requires an input message, key, and tag and will output either valid or invalid. Additionally, verification will fail if an adversary tampers with a message or tag. In the situation of password cracking, the nonces and MACs are sniffed from the four-way handshake. This allows the PTK to be guessed, leading to the MICs, etc. which then leads to a verification from the MAC when the password is guessed.

**Q5:**

A. Always

**Q6:**

B. Only if a weak key/passphrase is chosen.

**Task 2:**

**Q7:**

There are six packets printed out. Below is a screenshot of these from the tcpdump:

```
root@kali:~# tcpdump -nn -r forensics.cap "tcp port 34404"
reading from file forensics.cap, link-type EN10MB (Ethernet)
18:28:31.126638 IP 206.159.116.111.34404 > 12.33.247.4.80: Flags [S], seq 729060851, win 31592, options [mss 1436,sackOK,TS val 294508259 ecr 0,nop,wscale 0], length 0
18:28:31.126680 IP 12.33.247.4.80 > 206.159.116.111.34404: Flags [S.], seq 931635884, ack 729060852, win 5792, options [mss 1460,sackOK,TS val 1031143416 ecr 294508259,nop,wscale 0], length 0
18:28:31.209835 IP 206.159.116.111.34404 > 12.33.247.4.80: Flags [.], ack 1, win 31592, options [nop,nop,TS val 294508267 ecr 1031143416], length 0
18:28:31.210006 IP 206.159.116.111.34404 > 12.33.247.4.80: Flags [F.], seq 1, ack 1, win 31592, options [nop,nop,TS val 294508267 ecr 1031143416], length 0
18:28:31.210082 IP 12.33.247.4.80 > 206.159.116.111.34404: Flags [F.], seq 1, ack 2, win 5792, options [nop,nop,TS val 1031143424 ecr 294508267], length 0
18:28:31.292876 IP 206.159.116.111.34404 > 12.33.247.4.80: Flags [.], ack 2, win 31592, options [nop,nop,TS val 294508275 ecr 1031143424], length 0
root@kali:~#
```

There following packet flags are in the output:

- [S] Flag: Short for the SYN flag, it means the packet is a syn packet. This is a session establishment request.
- [S.] Flags: The SYN and ACK flags. The ‘.’ in the flags signifies a placeholder for ACK, which is the acknowledgement flag for the sender’s data.
- [F.] Flags: Short for the FIN flag along with the ACK placeholder, it is an indication of termination (FIN flag) and acknowledgement of the sender’s data.
- [.] Flag: Placeholder for ACK flag. This acknowledges a sender’s data.

### Q8:

With the modified filter, only packets with the S flag are being filtered for. In other words, only session establishment packets are being requested (SYN flag).

```
root@kali:~# tcpdump -nn -r forensics.cap "tcp port 34404 and tcp[13]=2"
reading from file forensics.cap, link-type EN10MB (Ethernet)
18:28:31.126638 IP 206.159.116.111.34404 > 12.33.247.4.80: Flags [S], seq 729060851, win 31592, options [mss 1436,sackOK,TS val 294508259 ecr 0,nop,wscale 0], length 0
root@kali:~#
```

As expected from the Q7 output, since there was only one packet with an S flag, there was only one returned with the filter applied.

### Q9:

With the modified filter, only packets with the SYN/ACK flags (S.) will be returned.

```
root@kali:~# tcpdump -nn -r forensics.cap "tcp port 34404 and tcp[13]=18"
reading from file forensics.cap, link-type EN10MB (Ethernet)
18:28:31.126680 IP 12.33.247.4.80 > 206.159.116.111.34404: Flags [S.], seq 931635884, ack 729060852, win 5792, options [mss 1460,sackOK,TS val 1031143416 ecr 294508259,nop,wscale 0], length 0
root@kali:~#
```

As expected from the Q7 output, there is only one packet printed. This is because there was only one packet with the SYN/ACK flags on.

### Q10:

The modified filter is only searching for packets that contain the SYN flag.

```
root@kali:~# tcpdump -nn -r forensics.cap "tcp port 34404 and tcp[13]&2!=0"
reading from file forensics.cap, link-type EN10MB (Ethernet)
18:28:31.126638 IP 206.159.116.111.34404 > 12.33.247.4.80: Flags [S], seq 729060851, win 31592, options [mss 1436,sackOK,TS val 294508259 ecr 0,nop,wscale 0], length 0
18:28:31.126680 IP 12.33.247.4.80 > 206.159.116.111.34404: Flags [S.], seq 931635884, ack 729060852, win 5792, options [mss 1460,sackOK,TS val 1031143416 ecr 294508259,nop,wscale 0], length 0
root@kali:~#
```

As such, there are two packets printed because there are two packets that contain the SYN flag.

### Q11:

The results for the last three filter are different due to the desired output. When “tcp[13]=2” is added, this is telling the search to only return packets whose tcp filter value is 2 in tcpdump. This would be the SYN flag only in the output, for which there is only one. The next filter was “tcp[13]=18”, which is telling the filter to only search for packages with SYN and ACK flags, since their tcpdump flag is 18. Again, there would only be one such of these packets since there was only one packet with SYN/ACK flags on. Lastly, the filter “tcp[13]&2!=0” is

telling the filter to look for packets for a tcpdump of key 2 in their flags, which is the SYN flag. As such, there are two packets returned (one with just a SYN flag, another that is SYN/ACK).

### Q12:

An alternative filter would be: **“tcp port 34404 and tcp[13]&(2 & 18) != 0”**

This filter is only finding those that match up with SYN (tcpdump key is 2) and SYN/ACK (tcpdump key is 18), which is the same functionality as in the previous filter. Below is a screenshot of the success:

```
root@kali:~# tcpdump -nn -r forensics.cap "tcp port 34404 and tcp[13]&(2 & 18)!=0"
reading from file forensics.cap, link-type EN10MB (Ethernet)
18:28:31.126638 IP 206.159.116.111.34404 > 12.33.247.4.80: Flags [S], seq 729060851, win 31592, options [mss 1436,sackOK,TS val 294508259 ecr 0,nop,wscale 0], length 0
18:28:31.126680 IP 12.33.247.4.80 > 206.159.116.111.34404: Flags [S.], seq 931635884, ack 729060852, win 5792, options [mss 1460,sackOK,TS val 1031143416 ecr 294508259,nop,wscale 0], length 0
root@kali:~#
```

### Q13:

The TCP options being set in the packet are the following:

- Maximum segment size (mss): Parameter that specifies the largest amount of data (in bytes) that a device can receive in a single TCP segment.
- Selective acknowledgement permitted (sackOK): Allows a receiver to acknowledge non-consecutive data so that the sender can retransmit only what is missing at the receiver's end.
- Timestamp value (TS val): Current timestamp clock value of the TCP sending the option.
- Echo-reply (ecr): Timestamp value calculated based on the TS val in SYN packet.
- No operation (nop): Provides padding around other options, useful for acknowledging packet receipts without a forced resend.
- Window scale (wscale): Used for recording the bytes of buffer space the host has for receiving data.

```
root@kali:~# tcpdump -X -nn -r forensics.cap src host 12.33.247.3 and tcp src port 2545 and tcp[13]=2
reading from file forensics.cap, link-type EN10MB (Ethernet)
18:20:01.264269 IP 12.33.247.3.2545 > 12.33.247.4.1984: Flags [S], seq 203458792, win 32120, options [mss 1460,sackOK,TS val 484956704 ecr 0,nop,wscale 0], length 0
 0x0000: 4500 003c fef7 4000 4006 357a 0c21 f703  E..<.@.5z!..
 0x0010: 0c21 f704 09f1 07c0 0c20 88e8 0000 0000  .!.....
 0x0020: a002 7d78 2683 0000 0204 05b4 0402 080a  ..}x&.....
 0x0030: 1ce7 da20 0000 0000 0103 0300 0001 0000  .....
root@kali:~#
```

#### Q14:

The options with their Hex values in the decode dump are as follows:

- Located in segment 0x0020:
  - mss: 0x05b4 **matches option decimal value of 1460**
  - sackOK: 0x0402080a **matches the right and left edges of the SACK**
- Located in segment 0x0030:
  - TS val: 0x1ce7da20 **matches option decimal value of 484956704**
  - erc: 0x0000 **matches decimal option of 0**
  - nop: 0x0001 **matches the option of a single (1) nop in the options**
  - wscale: 0x0000 **matches decimal option of 0**

#### Q15:

The method I used for finding the SMTP, HTTP, and DNS server IP's was by searching for all network IP's in the range of 12.33.0.0/16 with a tcp port of 25, 80, and 53 respectively. This will return the traffic and reveal the IP's for the servers. For example, the command used for finding SMTP traffic was the following:

**# tcpdump -nr forensics.cap src net 12.33.0.0/16 and tcp port 25**

This command worked for each server, resulting in the following IP's found:

- HTTP
  - 12.33.247.2
  - 12.33.247.4
  - 12.33.247.10
- SMTP
  - 12.33.247.3
  - 12.33.247.30
- DNS
  - 12.33.247.3

#### Q16:

A consistent method to identify a port scanner is with netstat methods. Netstat is a command line utility that can display network connections for TCP. With this function there does not need to be a scanning interval and results are consistent. Additionally, the result was confirmed with the nmap command, an open source network scanner used to discover is used to discover hosts and services on a network. This is accomplished by sending probe packets viewing the responses. The port scanning us turned out to be port 111 serviced by rpcbind.

The following command was used to identify the port with netstat:

**# netstat -an find "listening"**



The -a option displays all active TCP/UDP ports on which the computer is listening while the -n option displays active TCP connections. The following output is given:

```
root@kali:~# netstat -an | grep "listening"
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*              LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
udp        0      0 0.0.0.0:68             0.0.0.0:*
udp        0      0 0.0.0.0:111            0.0.0.0:*
udp6       0      0 :::111                  :::*
raw6       0      0 :::58                   :::*                    7
Active UNIX domain sockets (servers and established)
```

As shown, the port of the TCP scanner is 111. This was confirmed with the nmap command:

**# sudo nmap -sT -O localhost**

This -sT option is the TCP connect scan and uses the system call of the same name to scan machines rather than probing packets. The -O option enables OS protection. Below is the result of the scan:

```
root@kali:~# sudo nmap -sT -O localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2019-11-06 20:12 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00015s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 999 closed ports
PORT      STATE SERVICE
111/tcp   open  rpcbind
```