Network Traffic Analysis

Pandora (Hard)

This challenge evaluates the contestant’s ability to understand a packet capture containing network traffic using a non-standard protocol. It was suggested to use the Wireshark program to solve the challenge. The following description of the protocol is provided:

Overview

The communication between the client and server will contain three types of messages: Initialization, Hash Request, and Hash Response. A connection is started with the client sending an Initialization message, which contains the number of Hash Requests that the client wishes to make. Then, the server will send the length of its response. Then, the client sends their Hash Requests to the server. After all of the Hash Requests have been received, the server will finish sending a single Hash Response which contains hashes of all of the data that was sent by the client.

Initialization (Client -> Server)

1. N - A 4-byte integer in network byte order that represents the number of Hash Requests that will be sent.

Hash Request (Client -> Server)

1. Check - A fixed 2-byte integer in network byte order that verifies the integrity of the message.
2. Len - A 4-byte integer in network byte order that represents the length of the data in bytes.
3. Data - The data that will be hashed.

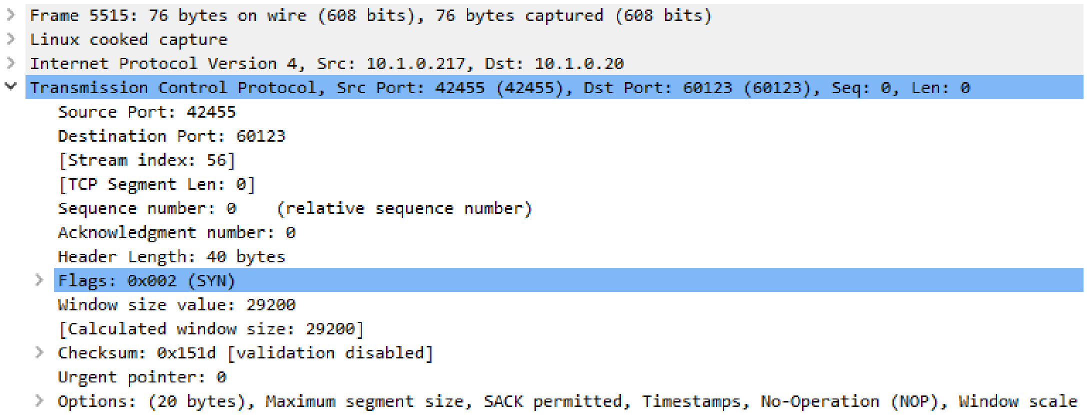
Hash Response (Server -> Client)

1. Count - The length of the data, in bytes, that follows.
2. Hashes - The hashes requested by the client. Each hash is in the form of a fixed-length chunk. These hashes are in the same order that the requests were made.

Questions 1 – 3 can be solved by filtering down the packet capture to just the custom protocol. The packet capture contains both SSH and HTTP traffic. The filter below will help remove noise:

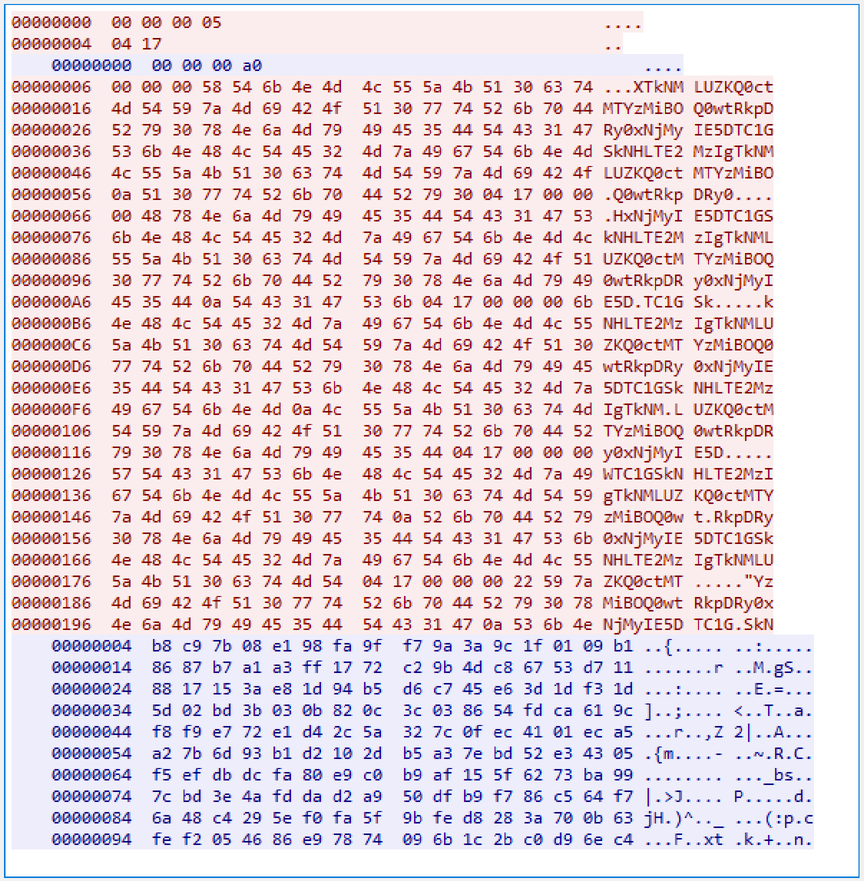
```
tcp && !(tcp.port == 22)  && !(tcp.port == 80)
```

From there, the first packet will show the client establishing a connection with the server.



```
> Frame 5515: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
> Linux cooked capture
> Internet Protocol Version 4, Src: 10.1.0.217, Dst: 10.1.0.20
v Transmission Control Protocol, Src Port: 42455 (42455), Dst Port: 60123 (60123), Seq: 0, Len: 0
    Source Port: 42455
    Destination Port: 60123
    [Stream index: 56]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 0
    Header Length: 40 bytes
  > Flags: 0x002 (SYN)
    Window size value: 29200
    [Calculated window size: 29200]
  > Checksum: 0x151d [validation disabled]
    Urgent pointer: 0
  > Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
```

Questions 4 – 11 can be solved by following the TCP stream for the first filtered packet and viewing the data as a hex dump. As per the protocol specification, the first 4 bytes represent the number of requests (5) and the next two bytes are the 2-byte magic number check. Once the 2-byte check has been sent, the length of the request follows. For the first request, this length is 0x58 or 88 in decimal. For the second request, which you can identity by the second instance of 0x0417, the length is 0x48 or 72 in decimal. The hash-length can be determine by taking the total number of requests (5) and dividing it by the length of the response as advertised by the server 0xa0 or 160 decimal, yielding a result of 32 bytes. After determining that each hash is 32 bytes, the first hash is then known to be the first 32 bytes with the second hash being the next 32 bytes. In this packet capture, the hidden flag is sent over by the client. Any of the hash requests can be base64 decoded to reveal the hidden flag.



```
00000000  00 00 00 05                                        ....
00000004  04 17                                              ..
    00000000  00 00 00 a0                                        ....
00000006  00 00 00 58 54 6b 4e 4d  4c 55 5a 4b 51 30 63 74 ...XTkNM LUZKQ0ct
00000016  4d 54 59 7a 4d 69 42 4f  51 30 77 74 52 6b 70 44 MTYzMiBO Q0wtRkpD
00000026  52 79 30 78 4e 6a 4d 79  49 45 35 44 54 43 31 47 Ry0xNjMy IE5DTC1G
00000036  53 6b 4e 48 4c 54 45 32  4d 7a 49 67 54 6b 4e 4d SkNHLTE2 MzIgTkNM
00000046  4c 55 5a 4b 51 30 63 74  4d 54 59 7a 4d 69 42 4f LUZKQ0ct MTYzMiBO
00000056  0a 51 30 77 74 52 6b 70  44 52 79 30 04 17 00 00 .Q0wtRkp DRy0....
00000066  00 48 78 4e 6a 4d 79 49  45 35 44 54 43 31 47 53 .HxNjMyI E5DTC1GS
00000076  6b 4e 48 4c 54 45 32 4d  7a 49 67 54 6b 4e 4d 4c kNHLTE2M zIgTkNML
00000086  55 5a 4b 51 30 63 74 4d  54 59 7a 4d 69 42 4f 51 UZKQ0ctM TYzMiBOQ
00000096  30 77 74 52 6b 70 44 52  79 30 78 4e 6a 4d 79 49 0wtRkpDR y0xNjMyI
000000A6  45 35 44 0a 54 43 31 47  53 6b 04 17 00 00 00 6b E5D.TC1G Sk.....k
000000B6  4e 48 4c 54 45 32 4d 7a  49 67 54 6b 4e 4d 4c 55 NHLTE2Mz IgTkNMLU
000000C6  5a 4b 51 30 63 74 4d 54  59 7a 4d 69 42 4f 51 30 ZKQ0ctMT YzMiBOQ0
000000D6  77 74 52 6b 70 44 52 79  30 78 4e 6a 4d 79 49 45 wtRkpDRy 0xNjMyIE
000000E6  35 44 54 43 31 47 53 6b  4e 48 4c 54 45 32 4d 7a 5DTC1GSk NHLTE2Mz
000000F6  49 67 54 6b 4e 4d 0a 4c  55 5a 4b 51 30 63 74 4d IgTkNM.L UZKQ0ctM
00000106  54 59 7a 4d 69 42 4f 51  30 77 74 52 6b 70 44 52 TYzMiBOQ 0wtRkpDR
00000116  79 30 78 4e 6a 4d 79 49  45 35 44 04 17 00 00 00 y0xNjMyI E5D.....
00000126  57 54 43 31 47 53 6b 4e  48 4c 54 45 32 4d 7a 49 WTC1GSkN HLTE2MzI
00000136  67 54 6b 4e 4d 4c 55 5a  4b 51 30 63 74 4d 54 59 gTkNMLUZ KQ0ctMTY
00000146  7a 4d 69 42 4f 51 30 77  74 0a 52 6b 70 44 52 79 zMiBOQ0w t.RkpDRy
00000156  30 78 4e 6a 4d 79 49 45  35 44 54 43 31 47 53 6b 0xNjMyIE 5DTC1GSk
00000166  4e 48 4c 54 45 32 4d 7a  49 67 54 6b 4e 4d 4c 55 NHLTE2Mz IgTkNMLU
00000176  5a 4b 51 30 63 74 4d 54  04 17 00 00 00 22 59 7a ZKQ0ctMT ....."Yz
00000186  4d 69 42 4f 51 30 77 74  52 6b 70 44 52 79 30 78 MiBOQ0wt RkpDRy0x
00000196  4e 6a 4d 79 49 45 35 44  54 43 31 47 0a 53 6b 4e NjMyIE5D TC1G.SkN
    00000004  b8 c9 7b 08 e1 98 fa 9f  f7 9a 3a 9c 1f 01 09 b1 ..{..... ..:.....
    00000014  86 87 b7 a1 a3 ff 17 72  c2 9b 4d c8 67 53 d7 11 .......r ..M.gS..
    00000024  88 17 15 3a e8 1d 94 b5  d6 c7 45 e6 3d 1d f3 1d ...:.... ..E.=...
    00000034  5d 02 bd 3b 03 0b 82 0c  3c 03 86 54 fd ca 61 9c ]..;.... <..T..a.
    00000044  f8 f9 e7 72 e1 d4 2c 5a  32 7c 0f ec 41 01 ec a5 ...r..,Z 2|..A...
    00000054  a2 7b 6d 93 b1 d2 10 2d  b5 a3 7e bd 52 e3 43 05 .{m....- ..~.R.C.
    00000064  f5 ef db dc fa 80 e9 c0  b9 af 15 5f 62 73 ba 99 ........ ..._bs..
    00000074  7c bd 3e 4a fd da d2 a9  50 df b9 f7 86 c5 64 f7 |.>J.... P.....d.
    00000084  6a 48 c4 29 5e f0 fa 5f  9b fe d8 28 3a 70 0b 63 jH.)^.._ ...(:p.c
    00000094  fe f2 05 46 86 e9 78 74  09 6b 1c 2b c0 d9 6e c4 ...F..xt .k.+..n.
```

Question	Answer
What is the IP address of the server?	10.1.0.20
What is the IP address of the client?	10.1.0.217
What port is the server listening on?	60123
What is the magic 2-byte integer check in decimal?	1047
How many encrypt requests were made by the client?	5
What is the length of the first encrypt request?	88
What is the length of the second encrypt request?	72
How large is an individual encrypt hash in bytes?	32
What was the encrypt response (in the form 0xFFFF) for the first request?	0xb8c97b08e198fa9ff79a3a9c1f0109b18687b7a1a3ff1772c29b4dc86753d711
What was the encrypt response (in the form 0xFFFF) for the second request?	0x8817153ae81d94b5d6c745e63d1df31d5d02bd3b030b820c3c038654fdca619c
What is the hidden flag being sent over the protocol?	NCL-FJCG-1632