

Challenge SSTIC 2026 - Step 0 : linenoise

nebucca

Hello analyst,

Following an alert from the ABSSI (*Agence Bretonne de la Sécurité des Systèmes d'Information*), we are suspecting a compromise of one of our contractors. An extract, containing a **dubious network traffic**, has been supplied. Could you please have a look at it? As usual, we are looking for an analysis of the exchanges, and any IOCs if relevant.

Please be careful with contained data, if any. The, maybe, compromised contractor is working on a highly sensitive infrastructure, all information related to this system **MUST BE** reported at the earliest opportunity.

Thanks for your assistance and your discretion,

-Incident Dispatcher - Investigation des Moyens et Plateformes Sous-traitées

1 Analyse

Direction *WireShark* pour l'analyse des traces réseaux. Il s'agit de communication entre deux machines en utilisant le protocole QUIC. Les données sont chiffrées.

Mais chose curieuse, le début de chaque paquet semble systématiquement "compréhensible" :

The screenshot displays a Wireshark capture of network traffic. The top pane shows a list of 30 QUIC packets. The middle pane shows the details of the selected packet (No. 115), which is a QUIC packet. The details pane shows the structure of the packet, including the short header and the protected payload. The bottom pane shows the raw data of the selected packet, with some fields highlighted in green.

Packet 115 details:

- Frame 8: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface unknown, id 0
- Linux cooked capture v2
- Internet Protocol Version 4, Src: 203.0.2.95, Dst: 203.0.17.102
- User Datagram Protocol, Src Port: 443, Dst Port: 36933
- QUIC (QTF)
 - QUIC Connection Information
 - [Expert Info (Note/Protocol): Unknown QUIC connection. Missing Initial Packet or migrated connection?]
 - [Unknown QUIC connection. Missing Initial Packet or migrated connection?]
 - [Severity level: Note]
 - [Group: Protocol]
 - [Packet Length: 67]
 - QUIC Short Header
 - 0... .. = Header Form: Short Header (0)
 - = Fixed Bit: True
 - ..0... .. = Spin Bit: False
 - Remaining Payload: 096e09745f63727915e6f2a37727e5a6a566ca374291378228c0cfa4d55e508e0712ed31bc99ac42f17c1c78dd1564368dec

Packet bytes (hex):

```
0000 00 00 00 00 00 00 02 00 01 00 06 00 00 27 00 .....E.....  
0010 21 f2 00 00 45 00 00 5f 0c 03 40 00 40 11 84 c5 1...E...@...  
0020 cb 00 02 5f c0 00 11 66 01 b0 00 45 00 40 55 1c .....T...E...  
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
0040 a6 a5 66 ca 37 42 91 37 02 20 c0 e4 a4 05 5e 05 .....f-78-7.....  
0050 00 e0 71 2e d1 1b c9 9a c4 2f 17 c1 1c 78 dd 15 .....q.....  
0060 a4 36 08 ac 7b fa 1c 71 d8 c1 4b 03 21 01 03 15 .....m-freq-c315..  
0070 16 07 05 .....
```

Du coup l'idée est d'extraire ces parties des différents échanges. Pour ce faire, on va adapter un script ¹ afin de ne garder que la partie qui nous intéresse :

```
local strfld = tostring(ftvbr:string(ENC_UTF_8))
local strxt = string.sub(strfld, 2, 9)
tree:add(exfield_string, strxt)
```

Il n'y a plus qu'à lancer le script pour extraire de ce qui nous intéresse :

```
.\tshark.exe -r .\client_capture.pcapng -X lua_script:extract.lua -X
lua_script1:udp.payload -T fields -e extract.string > extract_full.
txt
```

On se rend compte assez vite que de nombreuses lignes sont dupliquées (des rejeux?).

Petit travail de nettoyage et de mise en forme de ce qui s'avère être un malware en python.

On trouve là, le premier flag.

1. <https://wiki.wireshark.org/Lua/Examples#extract-field-values>