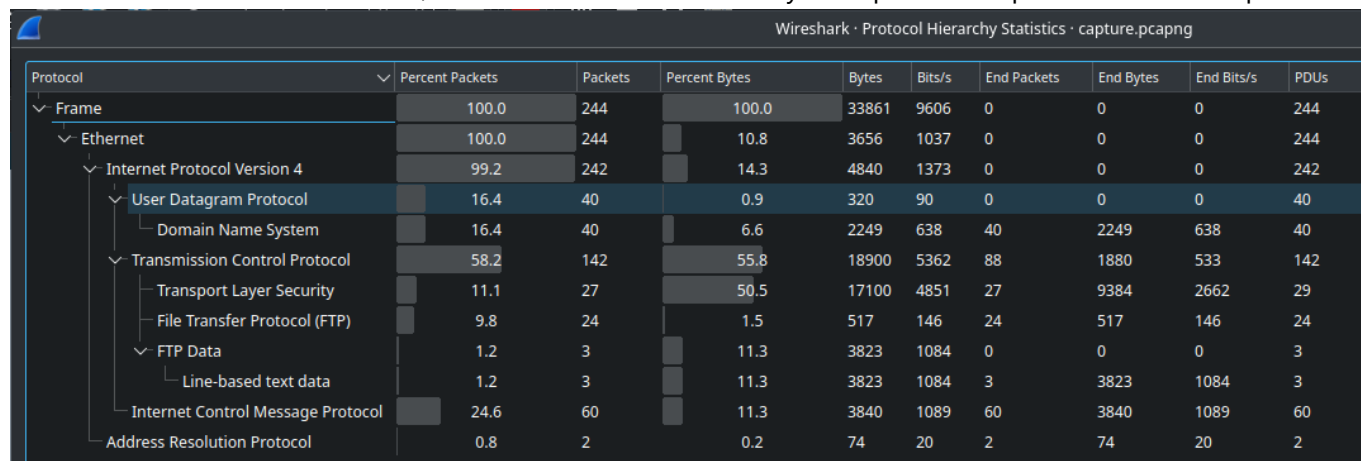


Applicatives Are The Way

Auteur: bWlrYQ

1. Recon

L'énoncé ne nous donne pas vraiment d'information quant à la teneur de la capture ou bien ce qui est à faire. Il va falloir fouiller de nous même, on commence avec une analyse de protocoles présents dans la capture

A screenshot of the Wireshark Protocol Hierarchy Statistics window. The window title is 'Wireshark · Protocol Hierarchy Statistics · capture.pcapng'. It displays a tree view of protocols on the left and a corresponding table of statistics on the right. The table has columns: Protocol, Percent Packets, Packets, Percent Bytes, Bytes, Bits/s, End Packets, End Bytes, End Bits/s, and PDUs. The data is as follows:

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	244	100.0	33861	9606	0	0	0	244
Ethernet	100.0	244	10.8	3656	1037	0	0	0	244
Internet Protocol Version 4	99.2	242	14.3	4840	1373	0	0	0	242
User Datagram Protocol	16.4	40	0.9	320	90	0	0	0	40
Domain Name System	16.4	40	6.6	2249	638	40	2249	638	40
Transmission Control Protocol	58.2	142	55.8	18900	5362	88	1880	533	142
Transport Layer Security	11.1	27	50.5	17100	4851	27	9384	2662	29
File Transfer Protocol (FTP)	9.8	24	1.5	517	146	24	517	146	24
FTP Data	1.2	3	11.3	3823	1084	0	0	0	3
Line-based text data	1.2	3	11.3	3823	1084	3	3823	1084	3
Internet Control Message Protocol	24.6	60	11.3	3840	1089	60	3840	1089	60
Address Resolution Protocol	0.8	2	0.2	74	20	2	74	20	2

Les protocoles ne sont pas nombreux et plutôt classiques d'après ce que l'on voit. FTP, ICMP, DNS, ARP, TLS (sûrement HTTP derrière) et bien évidemment du TCP et un peu d'ARP. Les paquets ARP sont négligeables.

2 FTP

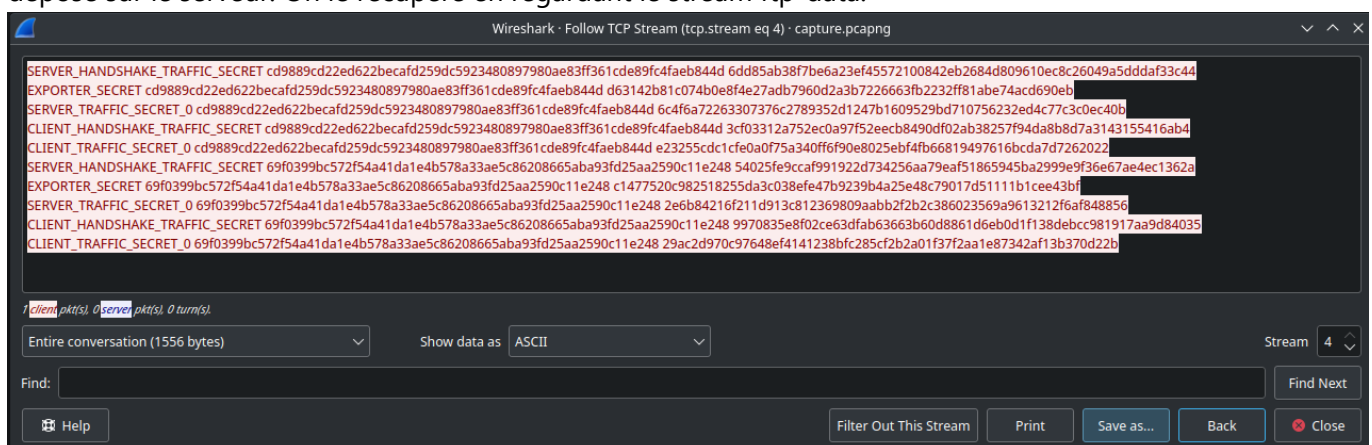
L'une des premières choses que l'on remarque ce sont les paquets FTP qui est un protocole non chiffré, on peut donc commencer à chercher de ce côté en analysant l'échange (on peut suivre le flux FTP)

```

220 (vsFTPD 3.0.2)
USER mika
331 Please specify the password.
PASS mika
230 Login successful.
SYST
215 UNIX Type: L8
FEAT
211-Features:
EPRT
EPSV
MDTM
PASV
REST STREAM
SIZE
TVFS
UTF8
211 End
EPSV
229 Entering Extended Passive Mode (||47234|)
LIST -la
150 Here comes the directory listing.
226 Directory send OK.
TYPE I
200 Switching to Binary mode.
EPSV
229 Entering Extended Passive Mode (||47551|)
STOR ssl.log
150 Ok to send data.
226 Transfer complete.
QUIT
221 Goodbye.

```

On peut donc noter que l'utilisateur et le mot de passe sont mika:mika. On peut possiblement le garder pour plus tard, on remarque aussi qu'un fichier ssl.log a été déposé sur le serveur. On le récupère en regardant le stream ftp-data.



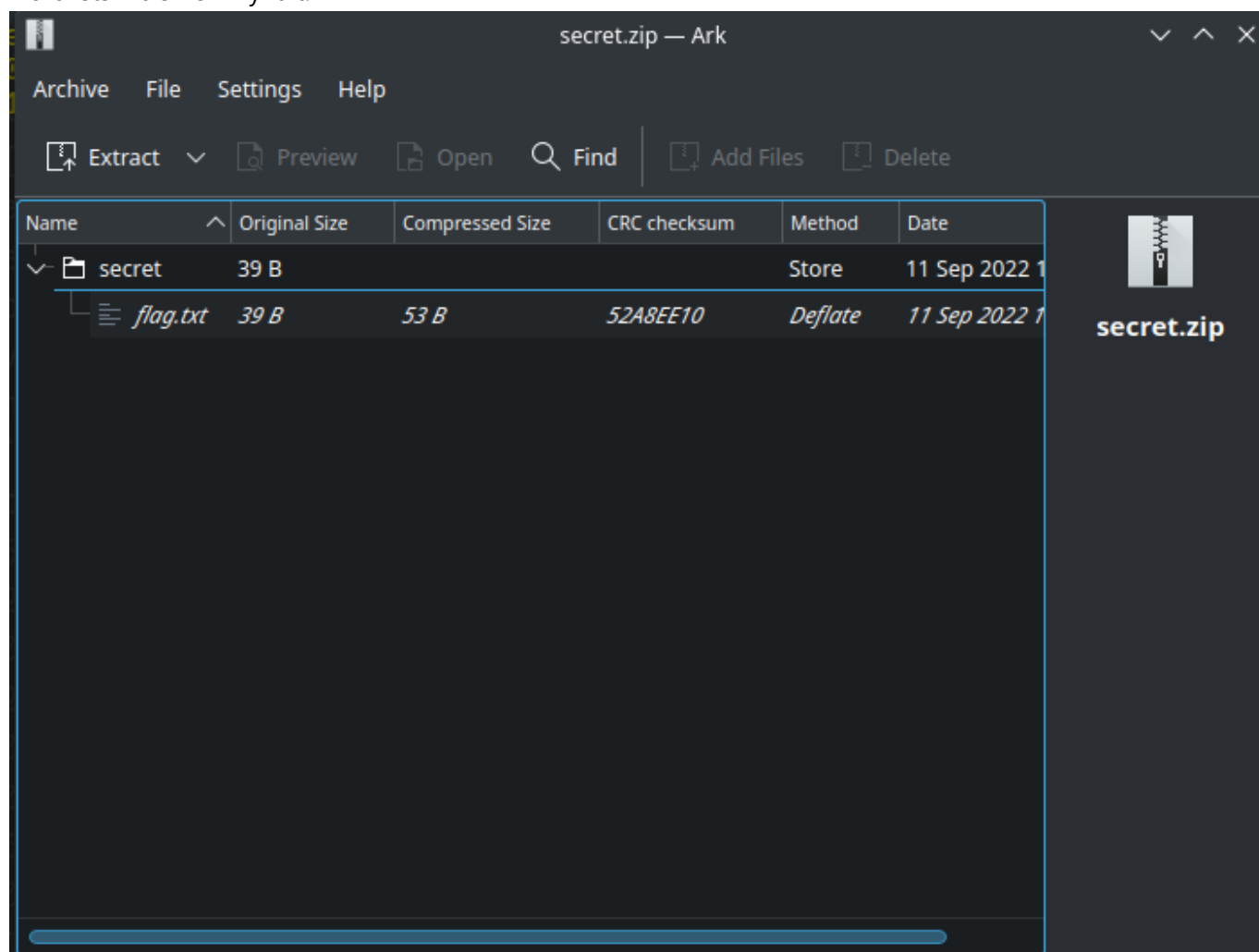
On peut sauvegarder le fichier, on remarque que ce sont sûrement des ciphers qui servent à du chiffrement, peut-être que l'on peut déchiffrer les paquets TLS avec. Essayons.

3. TLS/HTTP

The image shows the 'Wireshark · Preferences' dialog box, specifically the 'Transport Layer Security' section. On the left is a list of protocols: TDS, TeamSpeak2, TECMP, TELNET, Teredo, TETRA, TFP, TFTP, Thread, Thrift, Tibia, TIME, TIPC, TiVoConnect, TLS (highlighted), TNS, Token-Ring, TPCP, TPKT, and TPLINK-SMAF. The main area for TLS contains the following settings: 'RSA keys list' with an 'Edit...' button; 'TLS debug file' with a text field containing '/home/mika/Desktop/debug.txt' and a 'Browse...' button; three checkboxes: 'Reassemble TLS records spanning multiple TCP segments' (checked), 'Reassemble TLS Application Data spanning multiple TLS records' (checked), and 'Message Authentication Code (MAC), ignore "mac failed"' (unchecked); 'Pre-Shared Key' with an empty text field; and '(Pre)-Master-Secret log filename' with a text field containing '/home/mika/Desktop/ssl.log' and a 'Browse...' button. At the bottom are 'Help', 'OK', and 'Cancel' buttons.

```
echo
"504b03040a0000000000139b2b550000000000000000000000000070000007365637265742f504b030
4140009000800f68d2b5510eea85235000000270000000f0000007365637265742f666c61672e74787
4e00867410a41f95c8a98839488ed9cfa0e0e5e2898ccf12c5e16ceae0f0531283f07156e9d248e663
fd7e6af5ec35aa56926778b58504b070810eea8523500000027000000504b01021f000a00000000001
39b2b550000000000000000000000000007002400000000000001000000000000007365637265742f0
a0020000000000001001800193d7a5f03c6d801193d7a5f03c6d80179a0c04303c6d801504b01021f0
0140009000800f68d2b5510eea85235000000270000000f0024000000000000020000000250000007
365637265742f666c61672e7478740a002000000000000100180000d014d5f5c5d801b8fd805f03c6d
80100d014d5f5c5d801504b05060000000002000200ba000000970000000000" > secret && xxd -
r -p secret secret.zip
```

On peut ensuite ouvrir notre fichier, malheureusement il est chiffré donc on ne peut rien en extraire pour le moment. On essaie le mot de passe du ftp qui ne passe pas, on peut aussi essayer de le casser avec des wordlists mais rien n'y fait.



Il rest encore des protocoles que nous n'avons pas analysé, penchons nous dessus.

4. DNS

On remarque des requêtes dont le format est particulier (avec des = dans les noms de domaines), qui ne donnent que des erreurs, le domaine est `exfiltrate.net`, on peut filtrer les requêtes vers ce domaine en utilisant le filtre `dns.qry.name contains "exfiltrate.net"`.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	114	Standard query 0x0dab A bXlQYXNzd29yZA==.exfiltrate.net OPT
2	0.001385332	192.168.1.1	10.0.2.15	DNS	102	Standard query response 0x0dab No such name A bXlQYXNzd29yZA==.exfiltrate.net OPT
3	0.018417859	10.0.2.15	192.168.1.1	DNS	102	Standard query 0xd108 A aVM=.exfiltrate.net OPT
4	0.019562826	192.168.1.1	10.0.2.15	DNS	90	Standard query response 0xd108 No such name A aVM=.exfiltrate.net OPT
5	1.034218798	10.0.2.15	192.168.1.1	DNS	102	Standard query 0x5862 A czA=.exfiltrate.net OPT
6	1.035534188	192.168.1.1	10.0.2.15	DNS	90	Standard query response 0x5862 No such name A czA=.exfiltrate.net OPT
7	1.564556226	10.0.2.15	192.168.1.1	DNS	114	Standard query 0xc6db A Y29tcGxpY2F0ZWQ=.exfiltrate.net OPT
8	1.565750670	192.168.1.1	10.0.2.15	DNS	102	Standard query response 0xc6db No such name A Y29tcGxpY2F0ZWQ=.exfiltrate.net OPT
9	2.090067980	10.0.2.15	192.168.1.1	DNS	106	Standard query 0x2ed4 A dGhhdA=.exfiltrate.net OPT
10	2.091336361	192.168.1.1	10.0.2.15	DNS	94	Standard query response 0x2ed4 No such name A dGhhdA=.exfiltrate.net OPT
11	2.618410634	10.0.2.15	192.168.1.1	DNS	102	Standard query 0xfbe2 A SQ=.exfiltrate.net OPT
12	2.619804690	192.168.1.1	10.0.2.15	DNS	90	Standard query response 0xfbe2 No such name A SQ=.exfiltrate.net OPT
13	3.143526496	10.0.2.15	192.168.1.1	DNS	106	Standard query 0x6ea8 A aGFZQZ=.exfiltrate.net OPT
14	3.145116102	192.168.1.1	10.0.2.15	DNS	94	Standard query response 0x6ea8 No such name A aGFZQZ=.exfiltrate.net OPT
15	3.675892029	10.0.2.15	192.168.1.1	DNS	110	Standard query 0xb14c A Zm9yZ290dGVu.exfiltrate.net OPT
16	3.678318719	192.168.1.1	10.0.2.15	DNS	98	Standard query response 0xb14c No such name A Zm9yZ290dGVu.exfiltrate.net OPT
17	4.202788059	10.0.2.15	192.168.1.1	DNS	102	Standard query 0x258e A SXQ=.exfiltrate.net OPT
18	4.204200172	192.168.1.1	10.0.2.15	DNS	90	Standard query response 0x258e No such name A SXQ=.exfiltrate.net OPT
19	4.727541804	10.0.2.15	192.168.1.1	DNS	102	Standard query 0x4e04 A aW0=.exfiltrate.net OPT
20	4.728961171	192.168.1.1	10.0.2.15	DNS	90	Standard query response 0x4e04 No such name A aW0=.exfiltrate.net OPT
23	5.257324645	10.0.2.15	192.168.1.1	DNS	102	Standard query 0x05ea C c28=.exfiltrate.net OPT
24	5.258639213	192.168.1.1	10.0.2.15	DNS	90	Standard query response 0x05ea No such name A c28=.exfiltrate.net OPT

Les noms de machines sont du base64, décodons-le. On va extraire ça de la capture avec un one-liner basé sur tshark

```
tshark -r capture.pcapng -Y '(ip.src==10.0.2.15 && dns.qry.name contains "exfiltrate.net")' -T fields -e dns.qry.name | tr -d 'exfiltrate.net'
```

On n'obtient rien du décodage, soit les strings sont chiffrés soit ils ne veulent rien dire.

5. ICMP

Dernier protocole en vue, ICMP. On remarque directement que le buffer n'est pas celui par défaut. De plus chaque fin de buffer est ponctuée d'un \x00. Cela peut vouloir dire que chaque string envoyé doit être utilisé indépendamment de l'autre. Peut-être une wordlist ?

On peut faire un `strings capture.pcapng`, on voit que l'on retrouve le contenu de notre data dans des strings de 40 lignes avec un schéma qui se répète dans les strings, peut-être que la même donnée est écrite plusieurs fois.

Par défaut, la taille du buffer sur un ping est de 56 octets, mais si la taille de la donnée envoyée est inférieure à 56 octets et que la taille du buffer n'a pas été changée, alors la commande ping va remplir le buffer jusqu'à 56 octets en répétant la donnée jusqu'à ce qu'il soit plein.

En regardant un peu la donnée brute, on remarque que notre data est répétée environ 2.5 fois. Il faut donc trouver le pattern qui se répète exactement deux fois pour avoir la donnée avant que ping ne la modifie pour correspondre au buffer.

On scripte tout ça

```
import os
os.system("strings -n 40 capture.pcapng | grep -x '.\{40\}' >
```

```

/home/mika/Desktop/pass.txt")
with open("/home/mika/Desktop/pass.txt", "r") as pass_list:
    passwords = []
    a = 0
    for line in pass_list:
        if a%2==0:
            passwords.append(line.replace("\n", "")[26:40])
            a+=1
    pass_list.close()

with open("/home/mika/Desktop/wordlist.txt", "w") as wordlist:
    for i in range(len(passwords)):
        wordlist.write(passwords[i]+"\n")

```

6. Dictionnaire & Flag

Maintenant qu'on a notre wordlist et un .zip, on va essayer de bruteforce ça avec john.

```

mika@bwlryq ~/Desktop ./rw
$ zip2john secret.zip > hash.txt
secret.zip/secret/ is not encrypted!
ver 1.0 secret.zip/secret/ is not encrypted, or stored with non-handled
compression type
ver 2.0 secret.zip/secret/flag.txt PKZIP Encr: cmplen=53, decmplen=39,
crc=52A8EE10
mika@bwlryq ~/Desktop ./rw
$ john hash.txt --wordlist=./wordlist.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
v}WFPV5E*8z7Bd (secret.zip/secret/flag.txt)
1g 0:00:00:00 DONE (2022-10-29 20:28) 3.703g/s 111.1p/s 111.1c/s 111.1C/s
X^THh5fqYekf^S..JCup@2]6]D>t/L
Use the "--show" option to display all of the cracked passwords reliably
Session completed

```

Le mot de passe est donc v}WFPV5E*8z7Bd. On peut déchiffrer le .zip.

```

mika@bwlryq ~/Desktop ./rw
$ unzip secret.zip
Archive: secret.zip
  creating: secret/
[secret.zip] secret/flag.txt password:
  inflating: secret/flag.txt

```

Et on obtient notre flag : FMCTF{Y0u_Kn0w_W1R3Sh@rK_R3477Y_wEIL:})