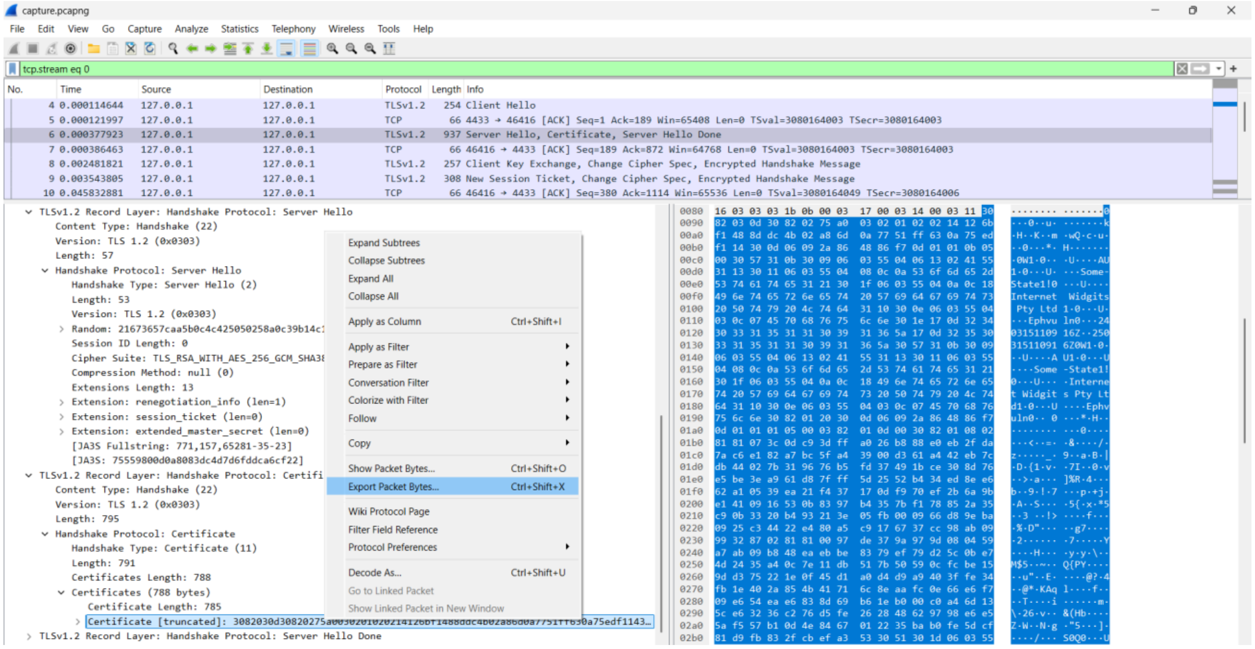
So we got a Network-Cryptography chall that gives us a pcap with TLS 1.2…. what can we do…. Wait did you say TLS 1.2??

Unlike TLS 1.3, TLS 1.2 can be broken given the cipher suite is weak enough, which in this case it is. First, we need to extract the certificate:



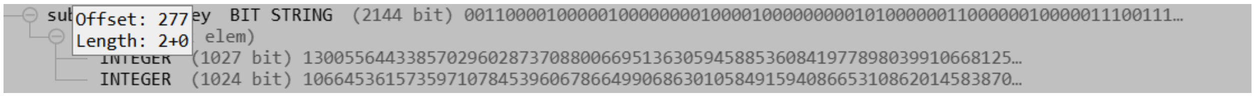
```
tcp.stream eq 0

No. Time        Source     Destination Protocol Length Info
  4 0.000114644 127.0.0.1  127.0.0.1   TLSv1.2  254 Client Hello
  5 0.000121997 127.0.0.1  127.0.0.1   TCP       66 4433 → 46416 [ACK] Seq=1 Ack=189 Win=65408 Len=0 TSval=3080164003 TSecr=3080164003
  6 0.000377923 127.0.0.1  127.0.0.1   TLSv1.2  937 Server Hello, Certificate, Server Hello Done
  7 0.000386463 127.0.0.1  127.0.0.1   TCP       66 46416 → 4433 [ACK] Seq=189 Ack=872 Win=64768 Len=0 TSval=3080164003 TSecr=3080164003
  8 0.002481821 127.0.0.1  127.0.0.1   TLSv1.2  257 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
  9 0.003543805 127.0.0.1  127.0.0.1   TLSv1.2  308 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
 10 0.045832881 127.0.0.1  127.0.0.1   TCP       66 46416 → 4433 [ACK] Seq=380 Ack=1114 Win=65536 Len=0 TSval=3080164049 TSecr=3080164006
```

```
v TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 57
  v Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 53
      Version: TLS 1.2 (0x0303)
    > Random: 21673657caa5b0c4c425050258a0c39b14c1
      Session ID Length: 0
      Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA38
      Compression Method: null (0)
      Extensions Length: 13
    > Extension: renegotiation_info (len=1)
    > Extension: session_ticket (len=0)
    > Extension: extended_master_secret (len=0)
      [JA3S Fullstring: 771,157,65281-35-23]
      [JA3S: 75559800d0a8083dc4d7d6fddca6cf22]
v TLSv1.2 Record Layer: Handshake Protocol: Certifi
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 795
  v Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 791
      Certificates Length: 788
    v Certificates (788 bytes)
        Certificate Length: 785
      > Certificate [truncated]: 3082030d30820275a003020102021412 6bf1488ddc4b02a86d0a7751ff630a75edf1143…
> TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
```

```
0080  16 03 03 03 1b 0b 00 03  17 00 03 14 00 03 11 30   ........ .......0
0090  82 03 0d 30 82 02 75 a0  03 02 01 02 02 14 12 6b   ...0..u. .......k
00a0  f1 48 8d dc 4b 02 a8 6d  0a 77 51 ff 63 0a 75 ed   .H..K..m .wQ.c.u.
00b0  f1 14 30 0d 06 09 2a 86  48 86 f7 0d 01 01 0b 05   ..0...*. H.......
00c0  00 30 57 31 0b 30 09 06  03 55 04 06 13 02 41 55   .0W1.0.. .U....AU
00d0  31 13 30 11 06 03 55 04  08 0c 0a 53 6f 6d 65 2d   1.0...U. ...Some-
00e0  53 74 61 74 65 31 21 30  1f 06 03 55 04 0a 0c 18   State1!0 ...U....
00f0  49 6e 74 65 72 6e 65 74  20 57 69 64 67 69 74 73   Internet  Widgits
0100  20 50 74 79 20 4c 74 64  31 10 30 0e 06 03 55 04    Pty Ltd 1.0...U.
0110  03 0c 07 45 70 68 76 75  6c 6e 30 1e 17 0d 32 34   ...Ephvu ln0...24
0120  30 33 31 35 31 31 30 39  31 36 5a 17 0d 32 35 30   03151109 16Z..250
0130  33 31 35 31 31 30 39 31  36 5a 30 57 31 0b 30 09   31511091 6Z0W1.0.
0140  06 03 55 04 06 13 02 41  55 31 13 30 11 06 03 55   ..U....A U1.0...U
0150  04 08 0c 0a 53 6f 6d 65  2d 53 74 61 74 65 31 21   ....Some -State1!
0160  30 1f 06 03 55 04 0a 0c  18 49 6e 74 65 72 6e 65   0...U... .Interne
0170  74 20 57 69 64 67 69 74  73 20 50 74 79 20 4c 74   t Widgit s Pty Lt
0180  64 31 10 30 0e 06 03 55  04 03 0c 07 45 70 68 76   d1.0...U ....Ephv
0190  75 6c 6e 30 82 01 20 30  0d 06 09 2a 86 48 86 f7   uln0.. 0 ...*.H..
01a0  0d 01 01 01 05 00 03 82  01 0d 00 30 82 01 08 02   ........ ...0....
01b0  81 81 07 3c 0d c9 3d ff  a0 26 b8 88 e0 eb 2f da   ...<..=. .&..../.
01c0  7a c6 e1 82 a7 bc 5f a4  39 00 d3 61 a4 42 eb 7c   z....._. 9..a.B.|
01d0  db 44 02 7b 31 96 76 b5  fd 37 49 1b ce 30 8d 76   .D.{1.v. .7I..0.v
01e0  e5 be 3e a9 61 d8 7f ff  5d 25 52 b4 34 ed 8e e6   ..>.a... ]%R.4...
01f0  62 a1 05 39 ea 21 f4 37  17 0d f9 70 ef 2b 6a 9b   b..9.!.7 ...p.+j.
0200  e1 41 09 16 53 0b 83 97  b4 35 7b f1 78 85 2a 35   .A..S... .5{.x.*5
0210  c9 0b 33 20 b4 93 21 3e  05 fb 00 09 66 d8 9e ba   ..3 ..!> ....f...
0220  09 25 c3 44 22 e4 80 a5  c9 17 67 37 cc 98 ab 09   .%.D"... ..g7....
0230  99 32 87 02 81 81 00 97  de 37 9a 97 9d 08 04 59   .2...... .7.....Y
0240  a7 ab 09 b8 48 ea eb be  83 79 ef 79 d2 5c 0b e7   ....H... .y.y.\..
0250  4d 24 35 a4 0c 7e 11 db  51 7b 50 59 0c fc be 15   M$5..~.. Q{PY....
0260  9d d3 75 22 1e 0f 45 d1  a0 d4 d9 a9 40 3f fe 34   ..u"..E. ....@?.4
0270  fb 1e 40 2a 85 4b 41 71  6c 8e aa fc 0e 66 e6 f7   ..@*.KAq l....f..
0280  09 e6 54 ea e6 83 8d 69  b6 1e b0 00 c0 a4 6d 13   ..T....i ......m.
0290  5c e6 32 36 c2 76 d5 fe  26 28 48 62 97 98 e6 e5   \.26.v.. &(Hb....
02a0  5a f5 57 b1 0d 4e 84 67  01 22 35 ba b0 fe 5d cf   Z.W..N.g ."5...].
02b0  81 d9 fb 83 2f cb ef a3  53 30 51 30 1d 06 03 55   ..../... S0Q0...U
```

Next up, we need to decode the der certificate. For that I used https://lapo.it/asn1js/, however openssl and other websites can be used as well. What we need are the public key values n and e (since this is RSA)

For me, said values can be found here:



```
sub…ey  BIT STRING  (2144 bit) 0011000010000010000000010000100000000010100000011000000100000111001111…
         elem)
    INTEGER  (1027 bit) 130055644338570296028737088006695136305945885360841977898039910668125…
    INTEGER  (1024 bit) 106645361573597107845396067866499068630105849159408665310862014583870…
```

Next up, we use dcode.fr to factorize n and figure out the other values here: https://www.dcode.fr/rsa-cipher

Next, we need to build the private key. Using python and the values we got from dcode we can do this easily.

```
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization

n =
1300556443385702960287370880066951363059458853608419778980399106681258797515744867649068169234104827392051556571142405277681682030539978956790151962476764197038895389990349060267828555439357834339975969219121466183673392888401910948635901667567503975718401986776689082275450777219576137405254696522822929086 79
```

```
e =
1066453615735971078453960678664990686301058491594086653108620145838700620617046622
3075428483238789692042720975323686254880074666239860921268837361318697910297030841
7884832531601035544107102590028211579550508699494971288803583755640940424098301
4258957388989092224259103397313291213626350508108474899121181685 59
d = 18337576115307010477422604901329942564585132748760458572280480612907518779399
p =
25620367680132110555879075164092814080992627707664402686174667544938777823227941 7
467429462291409906746865989861534939632608465093981738544910139327703796 99
q =
50762598711424764578309835161028536505017062779535728598193739021422991384075097 6
19364837920473739477192741993467746530937273178568830967465304495915565021
dp = d % (p - 1)
dq = d % (q - 1)
qi = pow(q, -1, p)
private_numbers = rsa.RSAPrivateNumbers(
    p=p, q=q, d=d, dmp1=dp, dmq1=dq, iqmp=qi,
    public_numbers=rsa.RSAPublicNumbers(e, n)
)
private_key = private_numbers.private_key()
with open("private_key.pem", "wb") as f:
    f.write(
        private_key.private_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PrivateFormat.PKCS8,
            encryption_algorithm=serialization.NoEncryption(),
        )
    )

print("Private key saved as private_key.pem")
```

After doing this we got our private key. All that's left is to import it (Edit -> Preferences -> Protocols -> TLS -> RSA keys, select the file we got and save everything). Now the pcap should have refreshed. To get the flag we simply need to right click a TLS packet and Follow -> TLS stream.

Made with love by: AndreiCat