#contentWrapper #fs, #sidebarContent #fs, #contentWrapper div [id * = 'myExtraContent'], #sidebarContent div [id * = 'myExtraContent'] {display: block;}

Kermith's workshop (https://translate.googleusercontent.cc depth=1&hl=en&prev=search&pto=aue&rurl=translate.goog

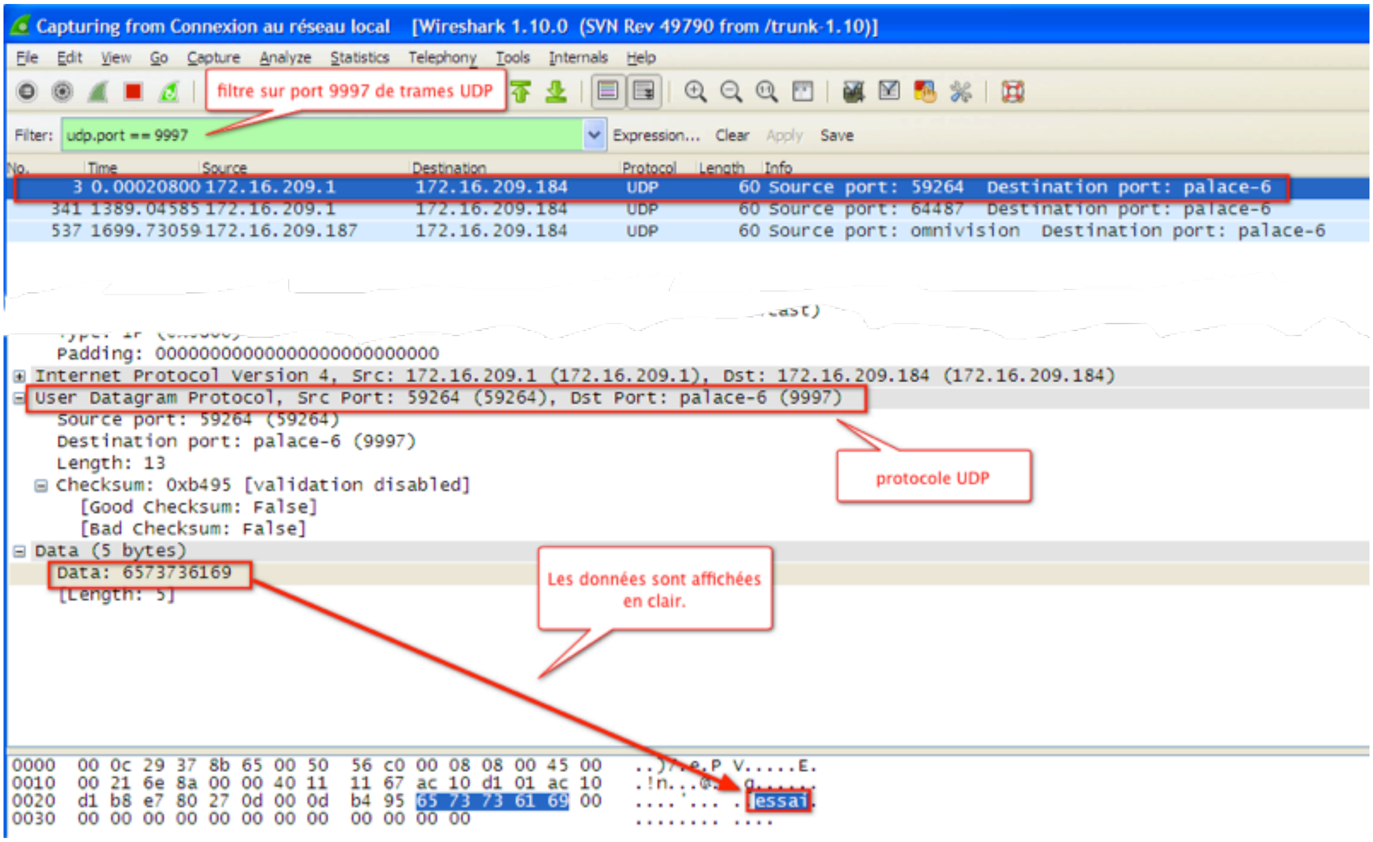
The other way to see supervision ...

Secure your flows

Sending data over the network is good. But when we start to take a closer look, we notice that the data appears in clear in the frames. Only one way to add a little confidentiality: encryption. We will use the same routine from the LiveCode-Clapi project article in the Preferences chapter.

Analysis of a UDP frame

To analyze a UDP frame from our previous program, I use the excellent Wireshark sniffer. The filter of the sniffer will be positioned on port 9997 of UDP frames.


Capturing from Connexion au réseau local [Wireshark 1.10.0 (SVN Rev 49790 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

filtre sur port 9997 de trames UDP

Filter: udp.port == 9997

```
No.   Time         Source          Destination     Protocol  Length  Info
    3 0.00020800   172.16.209.1    172.16.209.184  UDP       60 Source port: 59264  Destination port: palace-6
  341 1389.04585   172.16.209.1    172.16.209.184  UDP       60 Source port: 64487  Destination port: palace-6
  537 1699.73059   172.16.209.187  172.16.209.184  UDP       60 Source port: omnivision  Destination port: palace-6
```

```
                                                         ...ast)
  Padding: 0000000000000000000000000
⊞ Internet Protocol Version 4, Src: 172.16.209.1 (172.16.209.1), Dst: 172.16.209.184 (172.16.209.184)
⊟ User Datagram Protocol, Src Port: 59264 (59264), Dst Port: palace-6 (9997)
    Source port: 59264 (59264)
    Destination port: palace-6 (9997)
    Length: 13
  ⊟ Checksum: 0xb495 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
⊟ Data (5 bytes)
    Data: 6573736169
    [Length: 5]
```

protocole UDP

Les données sont affichées en clair.

```
0000  00 0c 29 37 8b 65 00 50  56 c0 00 08 08 00 45 00   ..)7.e.P V.....E.
0010  00 21 6e 8a 00 00 40 11  11 67 ac 10 d1 01 ac 10   .!n...@. .g......
0020  d1 b8 e7 80 27 0d 00 0d  b4 95 65 73 73 61 69 00   ....'... ..essai.
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

Analysis of a UDP frame

As we can see, the data travels in clear in the frame. If your application uses sensitive data, we need to find a solution to secure it. One solution, the encryption that we will discover below.

Data encryption

We will use the two functions ReturnCrypte and ReturnDecrypte described in the Preferences article of the LiveCode-Clapi Project. We will modify two lines of the client and server program. Drop these two functions into each stack.

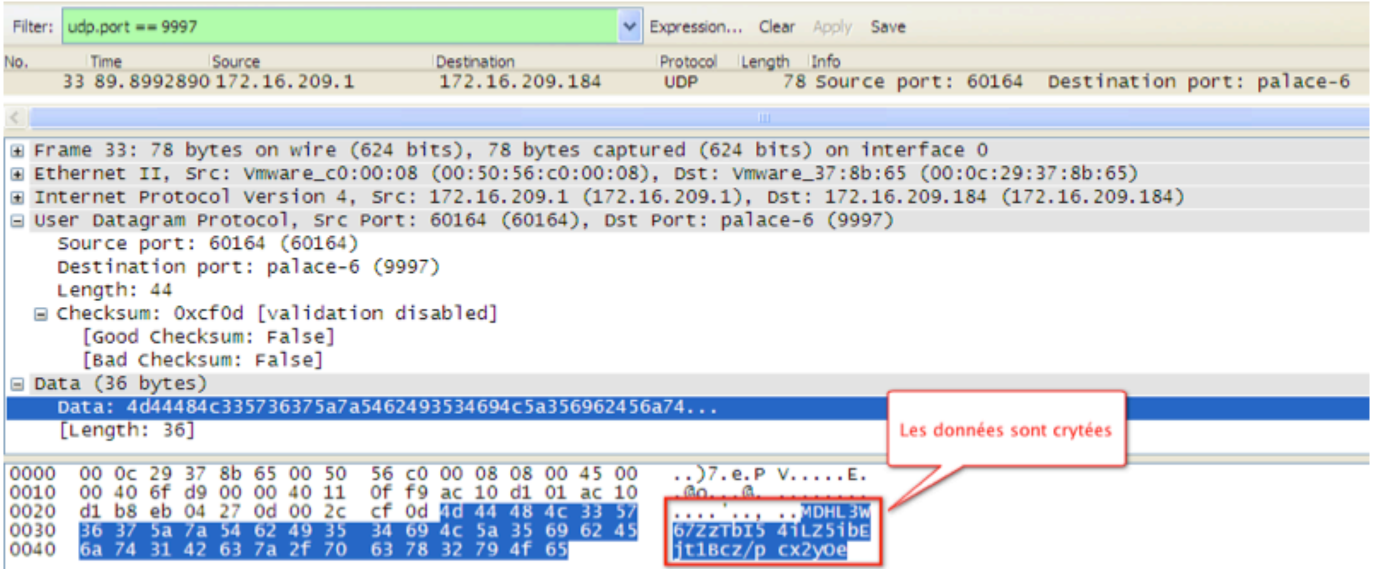
Customer program

We will use the encryption function to send the data. **– it sends the message without acknowledgment of receipt write** ReturnCrypte (eld "txtMessage") to socket ipDest

Server program

We will use the decryption function after recovering the data. **– display of UDP message dialog box received answer** pSocket & cr & ReturnDecrypte (pMsg) Attention: do not put an accented character in the key if you are using different OS.

Verification of data encryption.



```
No.  Time        Source        Destination     Protocol Length Info
  33 89.8992890 172.16.209.1  172.16.209.184  UDP          78 Source port: 60164  Destination port: palace-6
```

```
⊞ Frame 33: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
⊞ Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_37:8b:65 (00:0c:29:37:8b:65)
⊞ Internet Protocol Version 4, Src: 172.16.209.1 (172.16.209.1), Dst: 172.16.209.184 (172.16.209.184)
⊟ User Datagram Protocol, Src Port: 60164 (60164), Dst Port: palace-6 (9997)
    Source port: 60164 (60164)
    Destination port: palace-6 (9997)
    Length: 44
  ⊟ Checksum: 0xcf0d [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
⊟ Data (36 bytes)
    Data: 4d44484c335736375a7a5462493534694c5a356962456a74...
    [Length: 36]
```

```
0000  00 0c 29 37 8b 65 00 50  56 c0 00 08 08 00 45 00   ..)7.e.P V.....E.
0010  00 40 6f d9 00 00 40 11  0f f9 ac 10 d1 01 ac 10   .@o...@. ........
0020  d1 b8 eb 04 27 0d 00 2c  cf 0d 4d 44 48 4c 33 57   ....'.., ..MDHL3W
0030  36 37 5a 7a 54 62 49 35  34 69 4c 5a 35 69 62 45   67ZzTbI5 4iLZ5ibE
0040  6a 74 31 42 63 7a 2f 70  63 78 32 79 4f 65         jt1Bcz/p cx2yOe
```

Les données sont crytées

UDP data encryption

This solution can be easily implemented for TCP frames.

0 Commentaires Sugarbug Règles de confidentialité de Disqus 1 S'identifier

Recommander Tweet Partager Les meilleurs

Commencer la discussion…

S'IDENTIFIER AVEC

OU INSCRIVEZ-VOUS SUR DISQUS ?

Nom

Soyez le premier à commenter.

S'abonner Ajoutez Disqus à votre site web !Ajouter DisqusAjouter

Contact me (mailto:eric.coquard@sugarbug.fr) Last modified: 06/30/2020