Hackerman's Hacking Tutorials

The knowledge of anything, since all things have causes, is not acquired or complete unless it is known by its causes. - Avicenna

JUL 30, 2018 - 5 MINUTE READ - COMMENTS - REVERSE ENGINEERING DVTA WRITEUP

DVTA - Part 3 - Network Recon

In this part, we will focus on network traffic. More often than not, thick client applications have some sort of network connectivity. They talk to some server(s) to do things.

Who am I?

I am Parsia, a security engineer at Electronic Arts.

I write about application security, reverse engineering, Go, cryptography, and (obviously) videogames.

Click on About Me! to know more.

Collections

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Previous parts are:

- DVTA - Part 1 - Setup
- DVTA - Part 2 - Cert Pinning and Login Button

Discovering the Endpoints

In part 1 we did some network discovering with Procmon. Now we will do more using both Wireshark and Procmon. IRL use whatever tool you are comfortable with.

We do this because we need to figure out where the application talks to and using what protocol. At your day job, this step is probably the best bang for your buck in terms of the number of vulnerabilities found. Thick client applications are notorious for having inadequate server-side controls and trusting the client too much.

Capturing Loopback Traffic on Windows with Wireshark

Since we have deployed our FTP and MSSQL servers locally, we need to be able to capture local traffic. Windows does not have a real loopback adapter so WinPcap driver (used by Wireshark) cannot do it. The fix is using the npcap driver instead. For more information read https://wiki.wireshark.org/CaptureSetup/Loopback.

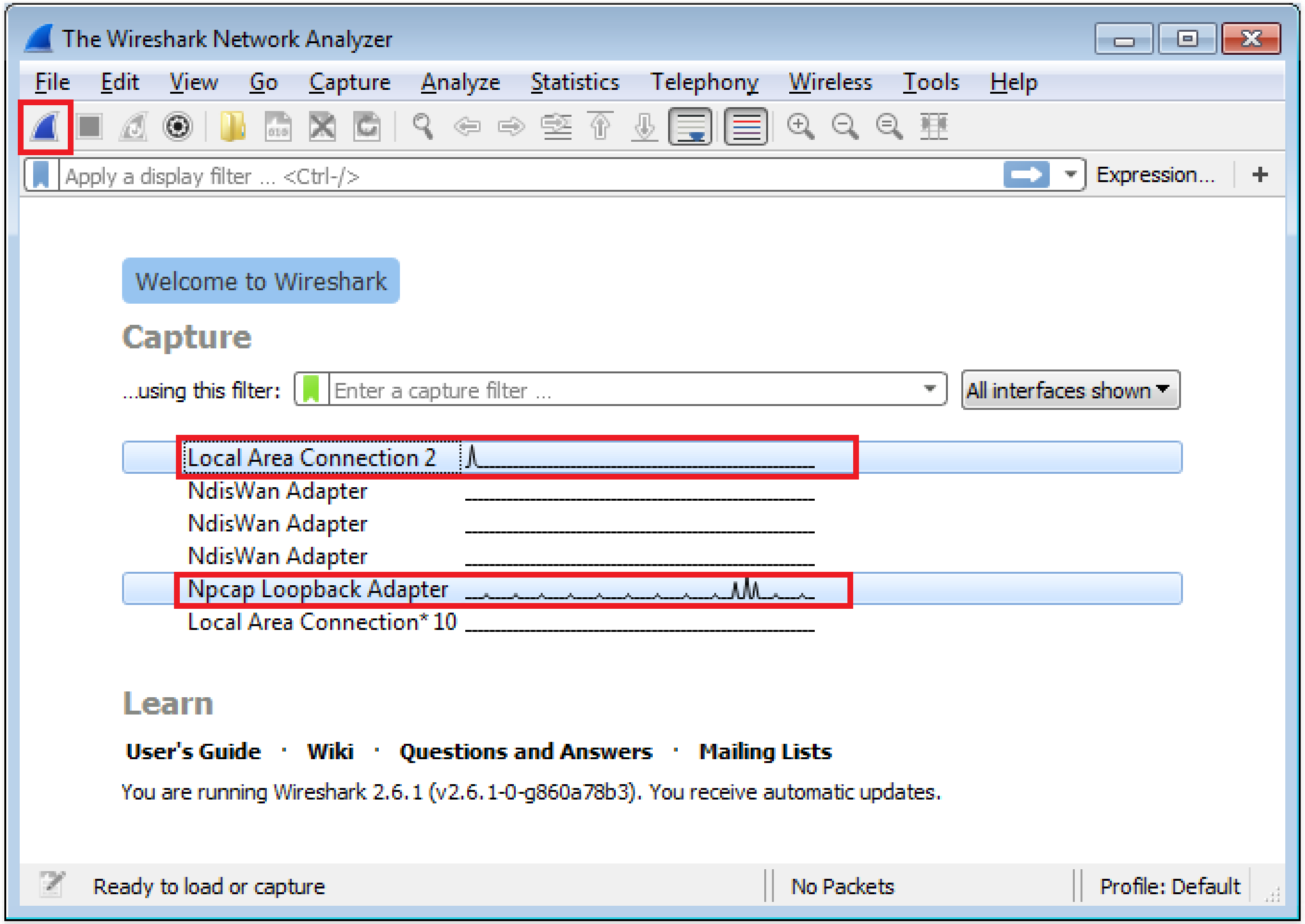
Download and install npcap from https://github.com/nmap/npcap/releases and then install Wireshark.

Recon with Wireshark

Run Wireshark, choose `Npcap Loopback Adapter`, and the VM's LAN. Then start capturing traffic.

Thick Client Proxying
Go/Golang
Blockchain/Distributed Ledgers
Automation
Reverse Engineering
Crypto(graphy)
CTFs/Writeups
WinAppDbg
AWSome.pw - S3 bucket squatting - my very legit branded vulnerability

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



You are running Wireshark 2.6.1 (v2.6.1-0-g860a78b3). You receive automatic updates.

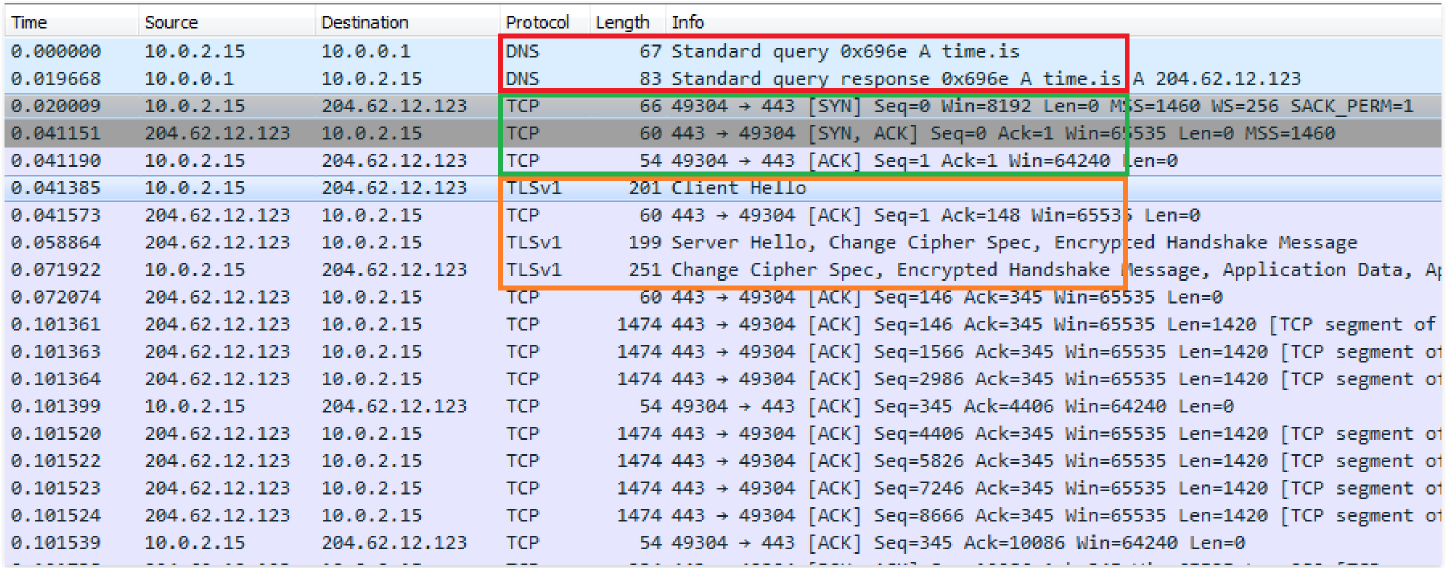
Setting up Wireshark to capture traffic

Run the patched application from the previous post but don't do anything.

Fetch Login Token

Click on the `Fetch Login Token` button. We already know where it goes, but let's inspect it with Wireshark.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



```
Time        Source         Destination    Protocol  Length  Info
0.000000    10.0.2.15      10.0.0.1       DNS          67 Standard query 0x696e A time.is
0.019668    10.0.0.1       10.0.2.15      DNS          83 Standard query response 0x696e A time.is A 204.62.12.123
0.020009    10.0.2.15      204.62.12.123  TCP          66 49304 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
0.041151    204.62.12.123  10.0.2.15      TCP          60 443 → 49304 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
0.041190    10.0.2.15      204.62.12.123  TCP          54 49304 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
0.041385    10.0.2.15      204.62.12.123  TLSv1       201 Client Hello
0.041573    204.62.12.123  10.0.2.15      TCP          60 443 → 49304 [ACK] Seq=1 Ack=148 Win=65535 Len=0
0.058864    204.62.12.123  10.0.2.15      TLSv1       199 Server Hello, Change Cipher Spec, Encrypted Handshake Message
0.071922    10.0.2.15      204.62.12.123  TLSv1       251 Change Cipher Spec, Encrypted Handshake Message, Application Data, Ap
0.072074    204.62.12.123  10.0.2.15      TCP          60 443 → 49304 [ACK] Seq=146 Ack=345 Win=65535 Len=0
0.101361    204.62.12.123  10.0.2.15      TCP        1474 443 → 49304 [ACK] Seq=146 Ack=345 Win=65535 Len=1420 [TCP segment of
0.101363    204.62.12.123  10.0.2.15      TCP        1474 443 → 49304 [ACK] Seq=1566 Ack=345 Win=65535 Len=1420 [TCP segment of
0.101364    204.62.12.123  10.0.2.15      TCP        1474 443 → 49304 [ACK] Seq=2986 Ack=345 Win=65535 Len=1420 [TCP segment of
0.101399    10.0.2.15      204.62.12.123  TCP          54 49304 → 443 [ACK] Seq=345 Ack=4406 Win=64240 Len=0
0.101520    204.62.12.123  10.0.2.15      TCP        1474 443 → 49304 [ACK] Seq=4406 Ack=345 Win=65535 Len=1420 [TCP segment of
0.101522    204.62.12.123  10.0.2.15      TCP        1474 443 → 49304 [ACK] Seq=5826 Ack=345 Win=65535 Len=1420 [TCP segment of
0.101523    204.62.12.123  10.0.2.15      TCP        1474 443 → 49304 [ACK] Seq=7246 Ack=345 Win=65535 Len=1420 [TCP segment of
0.101524    204.62.12.123  10.0.2.15      TCP        1474 443 → 49304 [ACK] Seq=8666 Ack=345 Win=65535 Len=1420 [TCP segment of
0.101539    10.0.2.15      204.62.12.123  TCP          54 49304 → 443 [ACK] Seq=345 Ack=10086 Win=64240 Len=0
```

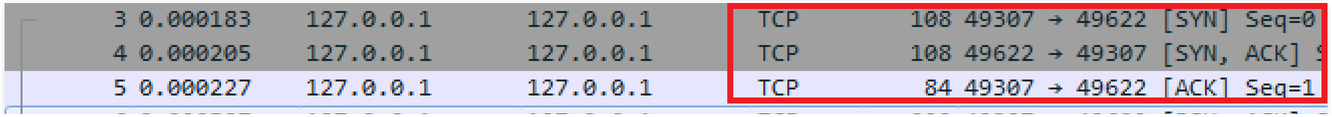
Captured traffic to time.is in Wireshark

Looking at the capture, it's clear what the application is doing.

- Red: DNS lookup for `time.is`
- Green: TCP connection to `time.is` (`204.62.12.123`). We can see the handshake `SYN-SYNACK-ACK`.
- Orange: TLS handshake with `time.is`. `ClientHello`, `ServerHello`, and the rest.

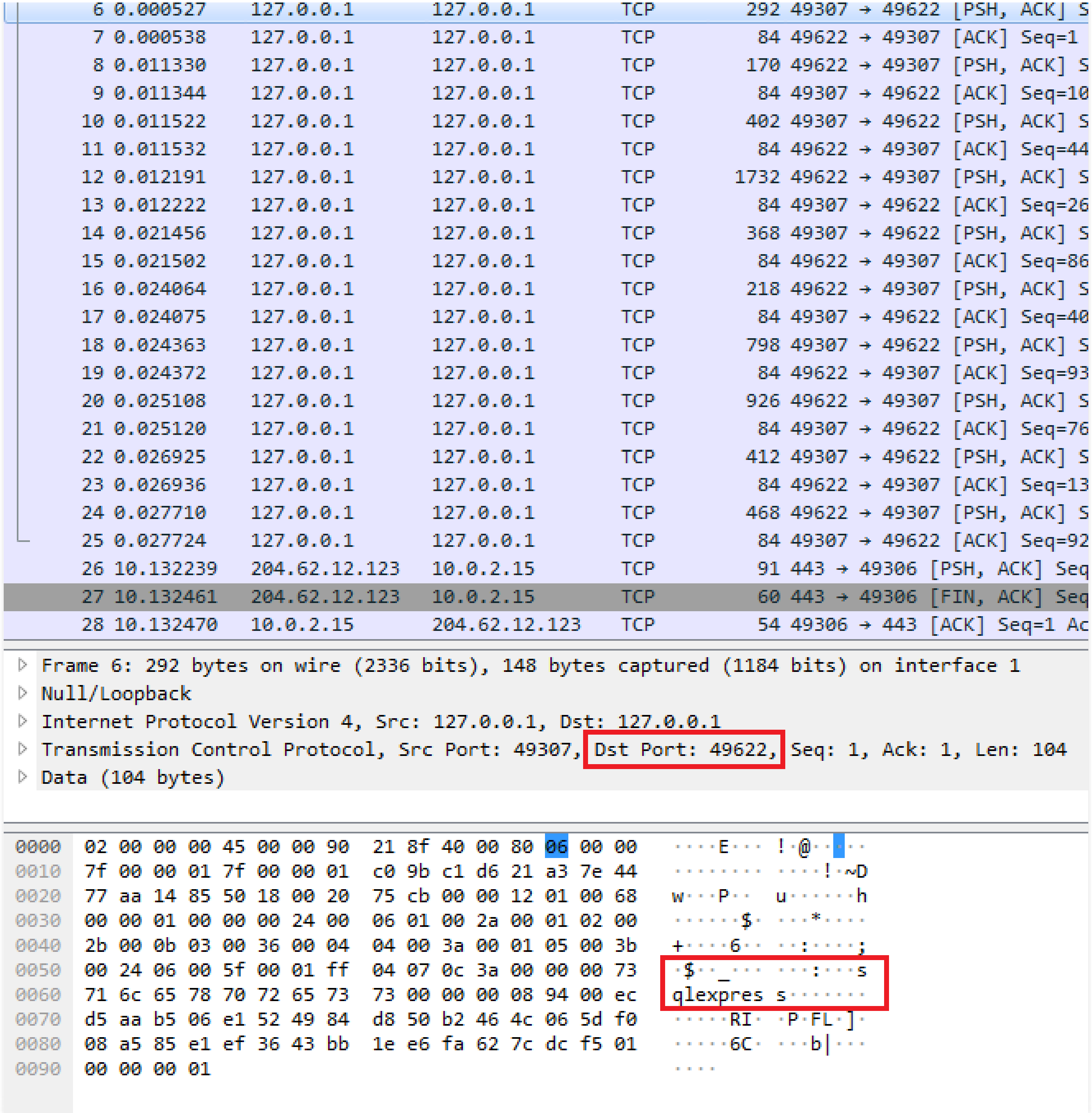
Normal User Login

Clear the capture and this time login with a valid set of non-admin credentials (e.g. `rebecca:rebecca`).



```
3 0.000183    127.0.0.1      127.0.0.1      TCP      108 49307 → 49622 [SYN] Seq=0
4 0.000205    127.0.0.1      127.0.0.1      TCP      108 49622 → 49307 [SYN, ACK] S
5 0.000227    127.0.0.1      127.0.0.1      TCP       84 49307 → 49622 [ACK] Seq=1
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

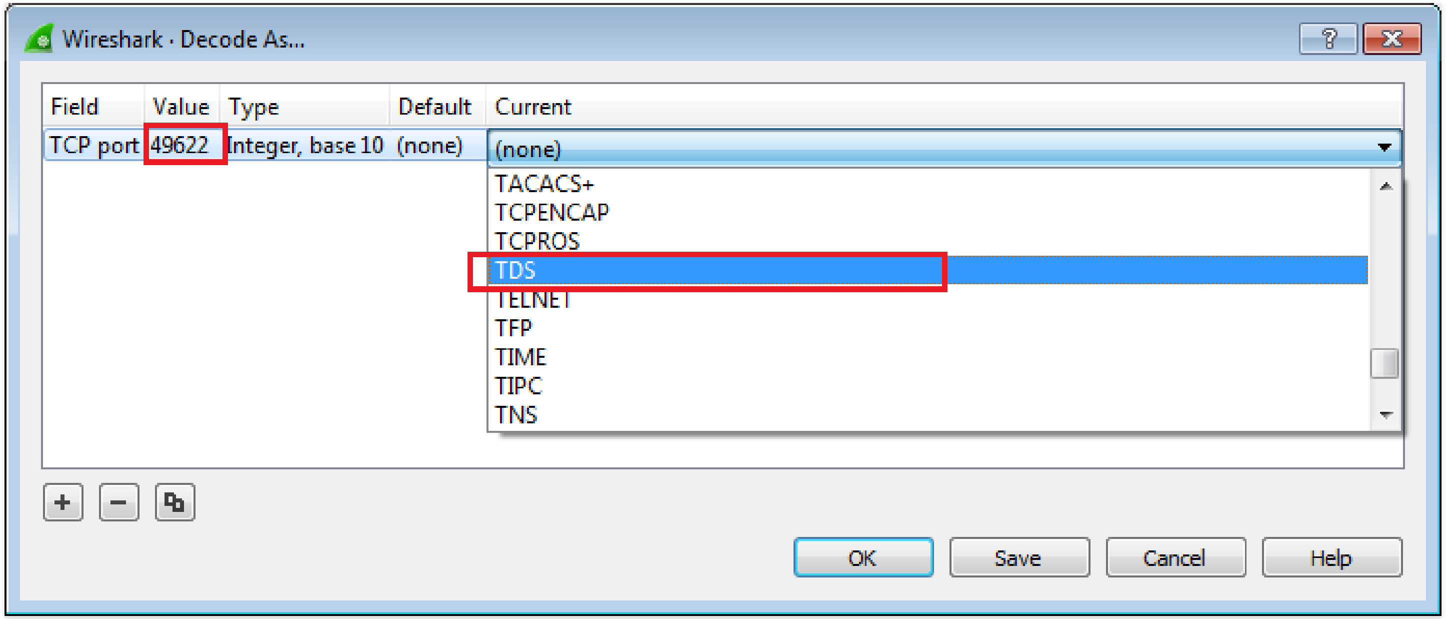


```
 6 0.000527   127.0.0.1     127.0.0.1     TCP   292 49307 → 49622 [PSH, ACK] S
 7 0.000538   127.0.0.1     127.0.0.1     TCP    84 49622 → 49307 [ACK] Seq=1
 8 0.011330   127.0.0.1     127.0.0.1     TCP   170 49622 → 49307 [PSH, ACK] S
 9 0.011344   127.0.0.1     127.0.0.1     TCP    84 49307 → 49622 [ACK] Seq=10
10 0.011522   127.0.0.1     127.0.0.1     TCP   402 49307 → 49622 [PSH, ACK] S
11 0.011532   127.0.0.1     127.0.0.1     TCP    84 49622 → 49307 [ACK] Seq=44
12 0.012191   127.0.0.1     127.0.0.1     TCP  1732 49622 → 49307 [PSH, ACK] S
13 0.012222   127.0.0.1     127.0.0.1     TCP    84 49307 → 49622 [ACK] Seq=26
14 0.021456   127.0.0.1     127.0.0.1     TCP   368 49307 → 49622 [PSH, ACK] S
15 0.021502   127.0.0.1     127.0.0.1     TCP    84 49622 → 49307 [ACK] Seq=86
16 0.024064   127.0.0.1     127.0.0.1     TCP   218 49622 → 49307 [PSH, ACK] S
17 0.024075   127.0.0.1     127.0.0.1     TCP    84 49307 → 49622 [ACK] Seq=40
18 0.024363   127.0.0.1     127.0.0.1     TCP   798 49307 → 49622 [PSH, ACK] S
19 0.024372   127.0.0.1     127.0.0.1     TCP    84 49622 → 49307 [ACK] Seq=93
20 0.025108   127.0.0.1     127.0.0.1     TCP   926 49622 → 49307 [PSH, ACK] S
21 0.025120   127.0.0.1     127.0.0.1     TCP    84 49307 → 49622 [ACK] Seq=76
22 0.026925   127.0.0.1     127.0.0.1     TCP   412 49307 → 49622 [PSH, ACK] S
23 0.026936   127.0.0.1     127.0.0.1     TCP    84 49622 → 49307 [ACK] Seq=13
24 0.027710   127.0.0.1     127.0.0.1     TCP   468 49622 → 49307 [PSH, ACK] S
25 0.027724   127.0.0.1     127.0.0.1     TCP    84 49307 → 49622 [ACK] Seq=92
26 10.132239  204.62.12.123 10.0.2.15     TCP    91 443 → 49306 [PSH, ACK] Seq
27 10.132461  204.62.12.123 10.0.2.15     TCP    60 443 → 49306 [FIN, ACK] Seq
28 10.132470  10.0.2.15     204.62.12.123 TCP    54 49306 → 443 [ACK] Seq=1 Ac

▷ Frame 6: 292 bytes on wire (2336 bits), 148 bytes captured (1184 bits) on interface 1
▷ Null/Loopback
▷ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▷ Transmission Control Protocol, Src Port: 49307, Dst Port: 49622, Seq: 1, Ack: 1, Len: 104
▷ Data (104 bytes)

0000  02 00 00 00 45 00 00 90  21 8f 40 00 80 06 00 00   ····E··· !·@·····
0010  7f 00 00 01 7f 00 00 01  c0 9b c1 d6 21 a3 7e 44   ········ ····!·~D
0020  77 aa 14 85 50 18 00 20  75 cb 00 00 12 01 00 68   w···P··  u······h
0030  00 00 01 00 00 00 24 00  06 01 00 2a 00 01 02 00   ······$· ···*····
0040  2b 00 0b 03 00 36 00 04  04 00 3a 00 01 05 00 3b   +····6·· ··:····;
0050  00 24 06 00 5f 00 01 ff  04 07 0c 3a 00 00 00 73   ·$··_··· ···:···s
0060  71 6c 65 78 70 72 65 73  73 00 00 00 08 94 00 ec   qlexpres s·······
0070  d5 aa b5 06 e1 52 49 84  d8 50 b2 46 4c 06 5d f0   ·····RI· ·P·FL·]·
0080  08 a5 85 e1 ef 36 43 bb  1e e6 fa 62 7c dc f5 01   ·····6C· ···b|···
0090  00 00 00 01                                        ····
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

MSSQL traffic captured in Wireshark

First, we see the TCP connection and then the login traffic to port `49622`. To decode the traffic with Wireshark, right-click on any outgoing packet and select `Decode As...`. Then select `TDS` for the combo box under `Current`. This tells Wireshark to decode all traffic to that port using the `TDS` dissector.

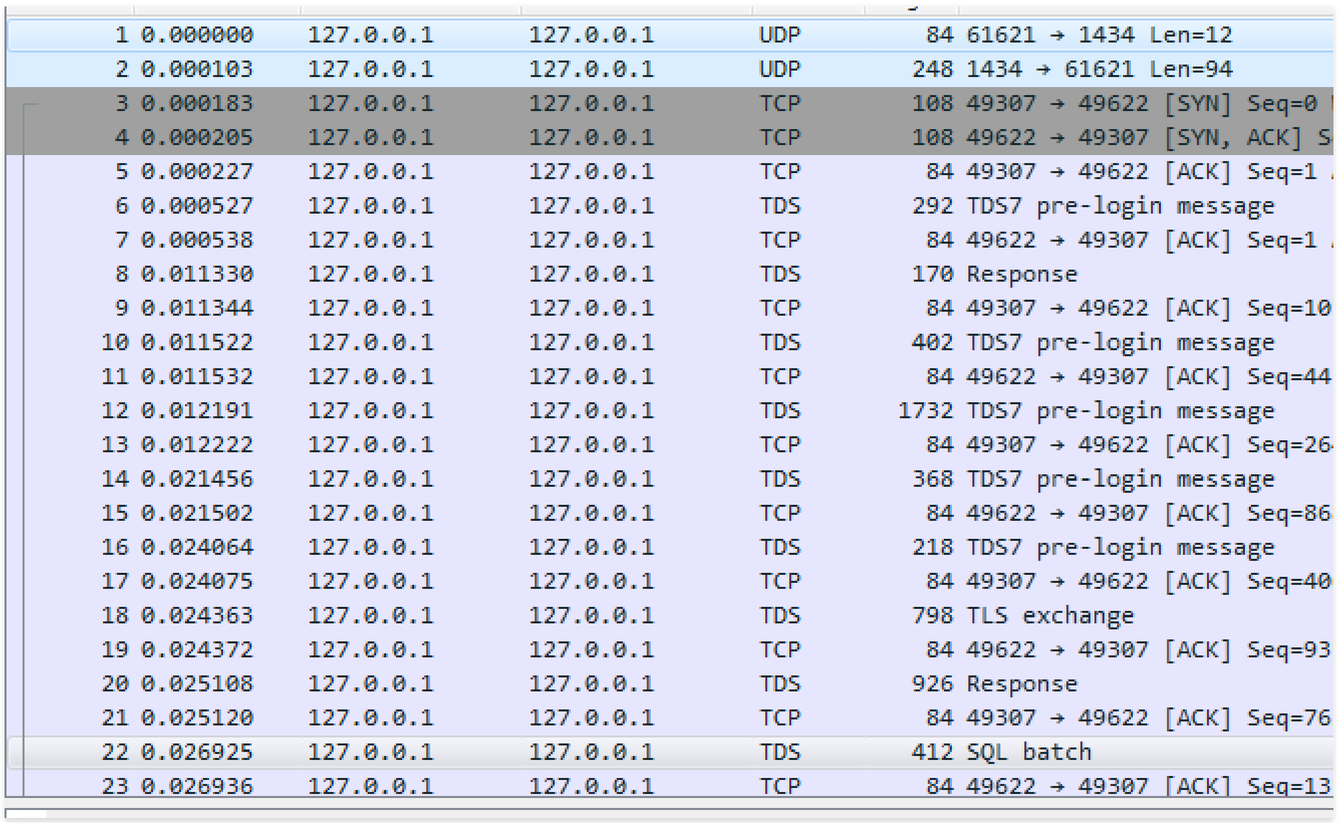


Field	Value	Type	Default	Current
TCP port	49622	Integer, base 10	(none)	(none)

Choosing the TDS dissector for MSSQL traffic

And now packets are annotated.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



```
 1 0.000000   127.0.0.1   127.0.0.1   UDP     84 61621 → 1434 Len=12
 2 0.000103   127.0.0.1   127.0.0.1   UDP    248 1434 → 61621 Len=94
 3 0.000183   127.0.0.1   127.0.0.1   TCP    108 49307 → 49622 [SYN] Seq=0
 4 0.000205   127.0.0.1   127.0.0.1   TCP    108 49622 → 49307 [SYN, ACK] S
 5 0.000227   127.0.0.1   127.0.0.1   TCP     84 49307 → 49622 [ACK] Seq=1
 6 0.000527   127.0.0.1   127.0.0.1   TDS    292 TDS7 pre-login message
 7 0.000538   127.0.0.1   127.0.0.1   TCP     84 49622 → 49307 [ACK] Seq=1
 8 0.011330   127.0.0.1   127.0.0.1   TDS    170 Response
 9 0.011344   127.0.0.1   127.0.0.1   TCP     84 49307 → 49622 [ACK] Seq=10
10 0.011522   127.0.0.1   127.0.0.1   TDS    402 TDS7 pre-login message
11 0.011532   127.0.0.1   127.0.0.1   TCP     84 49622 → 49307 [ACK] Seq=44
12 0.012191   127.0.0.1   127.0.0.1   TDS   1732 TDS7 pre-login message
13 0.012222   127.0.0.1   127.0.0.1   TCP     84 49307 → 49622 [ACK] Seq=26
14 0.021456   127.0.0.1   127.0.0.1   TDS    368 TDS7 pre-login message
15 0.021502   127.0.0.1   127.0.0.1   TCP     84 49622 → 49307 [ACK] Seq=86
16 0.024064   127.0.0.1   127.0.0.1   TDS    218 TDS7 pre-login message
17 0.024075   127.0.0.1   127.0.0.1   TCP     84 49307 → 49622 [ACK] Seq=40
18 0.024363   127.0.0.1   127.0.0.1   TDS    798 TLS exchange
19 0.024372   127.0.0.1   127.0.0.1   TCP     84 49622 → 49307 [ACK] Seq=93
20 0.025108   127.0.0.1   127.0.0.1   TDS    926 Response
21 0.025120   127.0.0.1   127.0.0.1   TCP     84 49307 → 49622 [ACK] Seq=76
22 0.026925   127.0.0.1   127.0.0.1   TDS    412 SQL batch
23 0.026936   127.0.0.1   127.0.0.1   TCP     84 49622 → 49307 [ACK] Seq=13
```

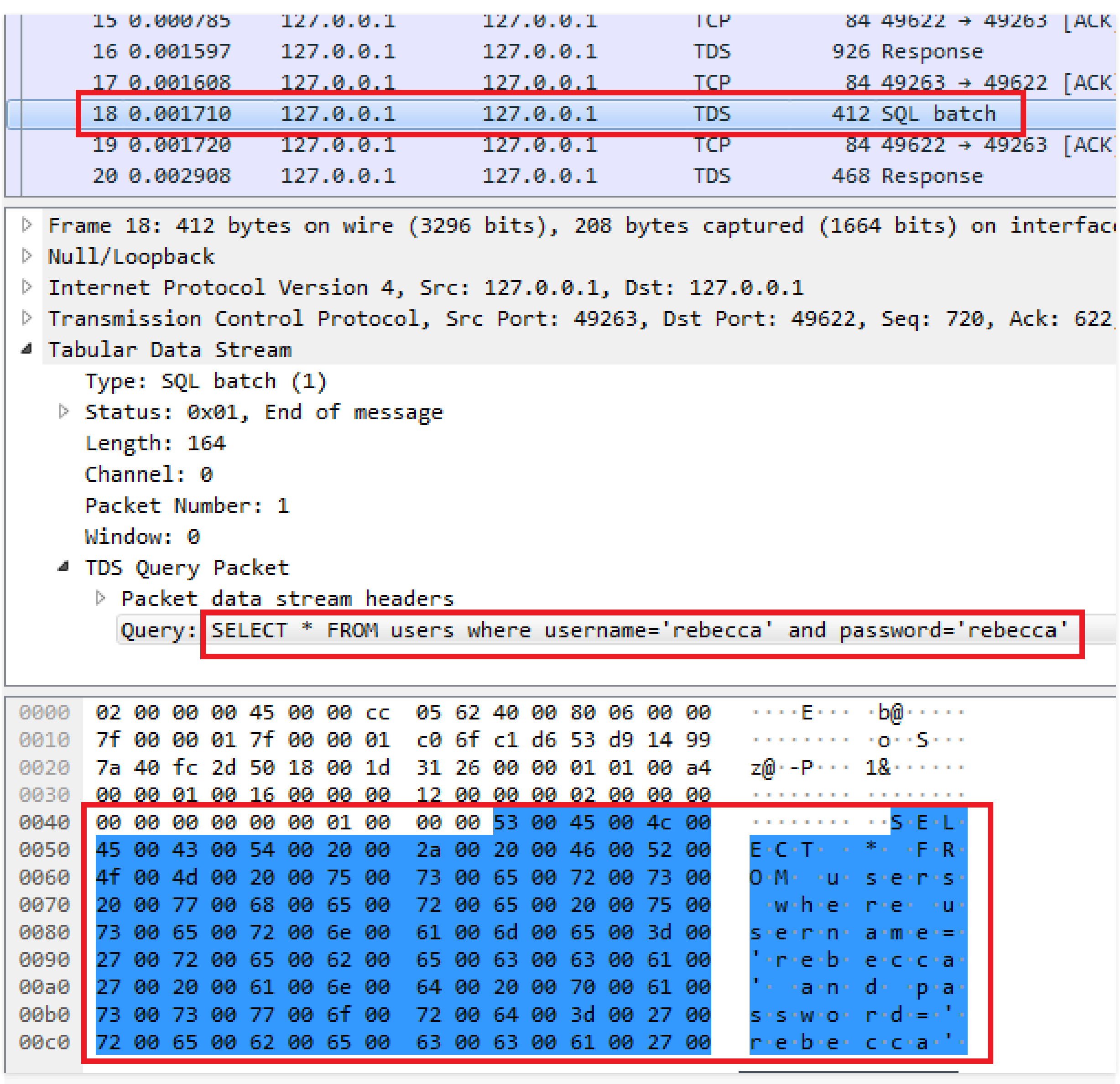
Annotated MSSQL traffic in Wireshark

Some observations:

1. TLS is not enabled. That's bad.
2. SQL queries are created on the client and sent outside. This is ripe for exploitation.

Going through the packets, select the one that says `SQL batch` and see the SQL query is created client-side and sent out. Any time you see client-side queries, you should be concerned.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



```
15 0.000785    127.0.0.1    127.0.0.1    TCP     84 49622 → 49263 [ACK
16 0.001597    127.0.0.1    127.0.0.1    TDS    926 Response
17 0.001608    127.0.0.1    127.0.0.1    TCP     84 49263 → 49622 [ACK
18 0.001710    127.0.0.1    127.0.0.1    TDS    412 SQL batch
19 0.001720    127.0.0.1    127.0.0.1    TCP     84 49622 → 49263 [ACK
20 0.002908    127.0.0.1    127.0.0.1    TDS    468 Response
```

```
▷ Frame 18: 412 bytes on wire (3296 bits), 208 bytes captured (1664 bits) on interfac
▷ Null/Loopback
▷ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▷ Transmission Control Protocol, Src Port: 49263, Dst Port: 49622, Seq: 720, Ack: 622
◢ Tabular Data Stream
    Type: SQL batch (1)
  ▷ Status: 0x01, End of message
    Length: 164
    Channel: 0
    Packet Number: 1
    Window: 0
  ◢ TDS Query Packet
    ▷ Packet data stream headers
      Query: SELECT * FROM users where username='rebecca' and password='rebecca'
```

```
0000  02 00 00 00 45 00 00 cc  05 62 40 00 80 06 00 00   ····E··· ·b@·····
0010  7f 00 00 01 7f 00 00 01  c0 6f c1 d6 53 d9 14 99   ········ ·o··S···
0020  7a 40 fc 2d 50 18 00 1d  31 26 00 00 01 01 00 a4   z@·-P··· 1&······
0030  00 00 01 00 16 00 00 00  12 00 00 00 02 00 00 00   ········ ········
0040  00 00 00 00 00 00 01 00  00 00 53 00 45 00 4c 00   ········ ··S·E·L·
0050  45 00 43 00 54 00 20 00  2a 00 20 00 46 00 52 00   E·C·T· · *· ·F·R·
0060  4f 00 4d 00 20 00 75 00  73 00 65 00 72 00 73 00   O·M· ·u· s·e·r·s·
0070  20 00 77 00 68 00 65 00  72 00 65 00 20 00 75 00    ·w·h·e· r·e· ·u·
0080  73 00 65 00 72 00 6e 00  61 00 6d 00 65 00 3d 00   s·e·r·n· a·m·e·=·
0090  27 00 72 00 65 00 62 00  65 00 63 00 63 00 61 00   '·r·e·b· e·c·c·a·
00a0  27 00 20 00 61 00 6e 00  64 00 20 00 70 00 61 00   '· ·a·n· d· ·p·a·
00b0  73 00 73 00 77 00 6f 00  72 00 64 00 3d 00 27 00   s·s·w·o· r·d·=·'·
00c0  72 00 65 00 62 00 65 00  63 00 63 00 61 00 27 00   r·e·b·e· c·c·a·'·
```

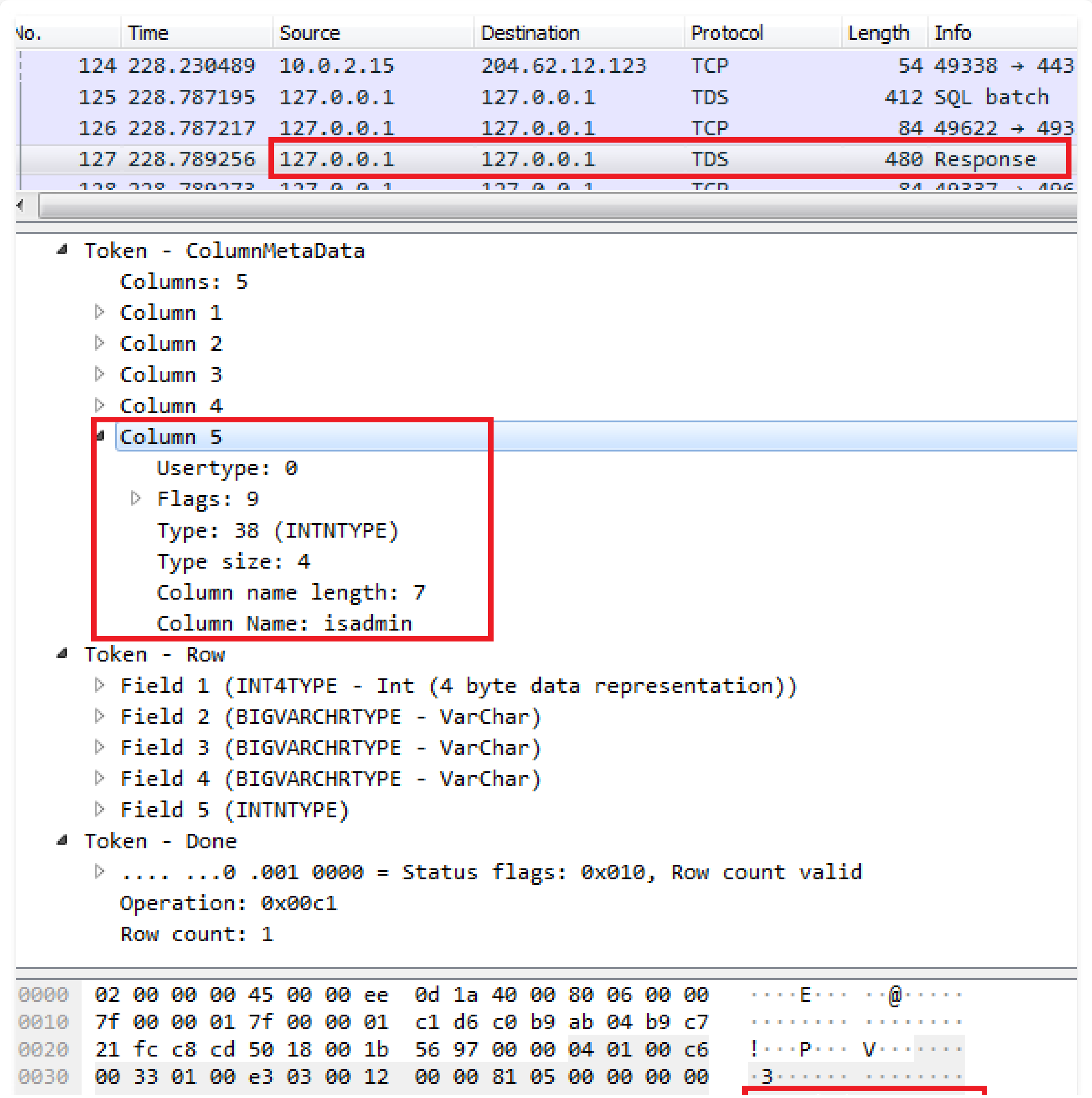
Client querying MSSQL server

The following query is executed (later we will come back and play with this):

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- `SELECT * FROM users where username='rebecca' and password='rebecca`

The response contains the query results which leaks the structure of the `users` table:

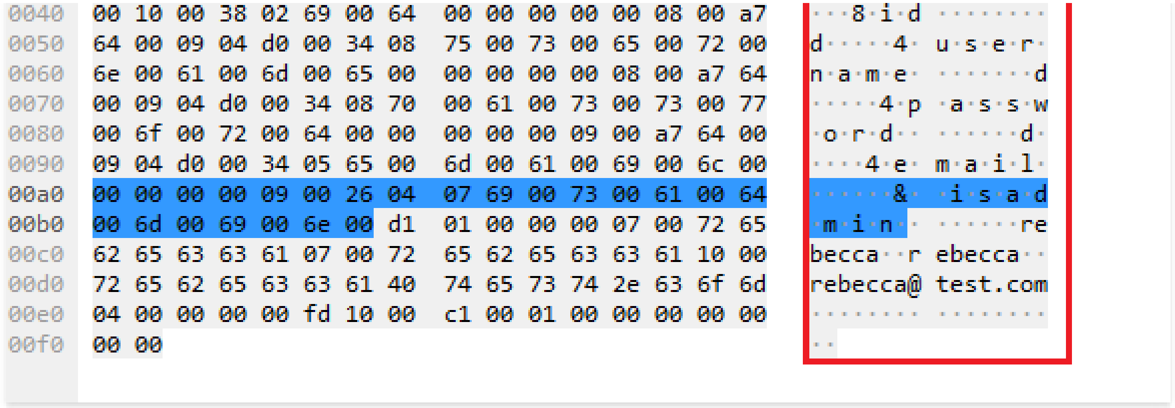


```
No.   Time        Source      Destination    Protocol  Length Info
  124 228.230489  10.0.2.15   204.62.12.123  TCP           54 49338 → 443
  125 228.787195  127.0.0.1   127.0.0.1      TDS          412 SQL batch
  126 228.787217  127.0.0.1   127.0.0.1      TCP           84 49622 → 493
  127 228.789256  127.0.0.1   127.0.0.1      TDS          480 Response
  128 228.789273  127.0.0.1   127.0.0.1      TCP           84 49337 → 496
```

```
◢ Token - ColumnMetaData
    Columns: 5
  ▷ Column 1
  ▷ Column 2
  ▷ Column 3
  ▷ Column 4
  ◢ Column 5
      Usertype: 0
    ▷ Flags: 9
      Type: 38 (INTNTYPE)
      Type size: 4
      Column name length: 7
      Column Name: isadmin
◢ Token - Row
  ▷ Field 1 (INT4TYPE - Int (4 byte data representation))
  ▷ Field 2 (BIGVARCHRTYPE - VarChar)
  ▷ Field 3 (BIGVARCHRTYPE - VarChar)
  ▷ Field 4 (BIGVARCHRTYPE - VarChar)
  ▷ Field 5 (INTNTYPE)
◢ Token - Done
  ▷ .... ...0 .001 0000 = Status flags: 0x010, Row count valid
    Operation: 0x00c1
    Row count: 1
```

```
0000  02 00 00 00 45 00 00 ee  0d 1a 40 00 80 06 00 00   ····E··· ··@·····
0010  7f 00 00 01 7f 00 00 01  c1 d6 c0 b9 ab 04 b9 c7   ········ ········
0020  21 fc c8 cd 50 18 00 1b  56 97 00 00 04 01 00 c6   !···P··· V·······
0030  00 33 01 00 e3 03 00 12  00 00 81 05 00 00 00 00   ·3······ ········
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



```
0040  00 10 00 38 02 69 00 64  00 00 00 00 00 08 00 a7   ···8·i·d ········
0050  64 00 09 04 d0 00 34 08  75 00 73 00 65 00 72 00   d·····4· u·s·e·r·
0060  6e 00 61 00 6d 00 65 00  00 00 00 00 08 00 a7 64   n·a·m·e· ·······d
0070  00 09 04 d0 00 34 08 70  00 61 00 73 00 73 00 77   ·····4·p ·a·s·s·w
0080  00 6f 00 72 00 64 00 00  00 00 00 09 00 a7 64 00   ·o·r·d·· ······d·
0090  09 04 d0 00 34 05 65 00  6d 00 61 00 69 00 6c 00   ····4·e· m·a·i·l·
00a0  00 00 00 00 09 00 26 04  07 69 00 73 00 61 00 64   ······&· ·i·s·a·d
00b0  00 6d 00 69 00 6e 00 d1  01 00 00 00 07 00 72 65   ·m·i·n·· ······re
00c0  62 65 63 63 61 07 00 72  65 62 65 63 63 61 10 00   becca··r ebecca··
00d0  72 65 62 65 63 63 61 40  74 65 73 74 2e 63 6f 6d   rebecca@ test.com
00e0  04 00 00 00 00 fd 10 00  c1 00 01 00 00 00 00 00   ········ ········
00f0  00 00                                              ··
```

Login query response

Admin Login

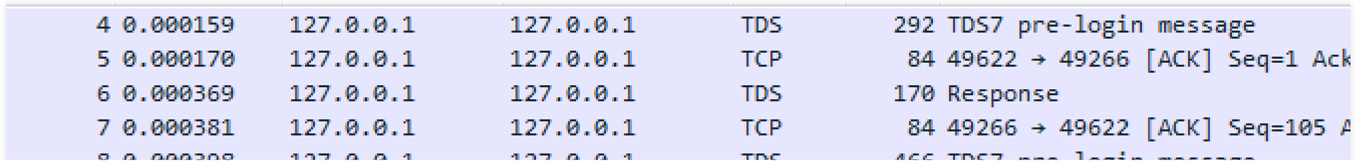
We know administrators can login to the application and backup data to an FTP server. We want to observe this traffic with Wireshark.

Logout and login with `admin:admin123`. Note admin interface has only one button, `Backup Data to FTP Server`. This should give us the clue that FTP credentials are hardcoded.

Looking at Wireshark, we will see two different streams of traffic:

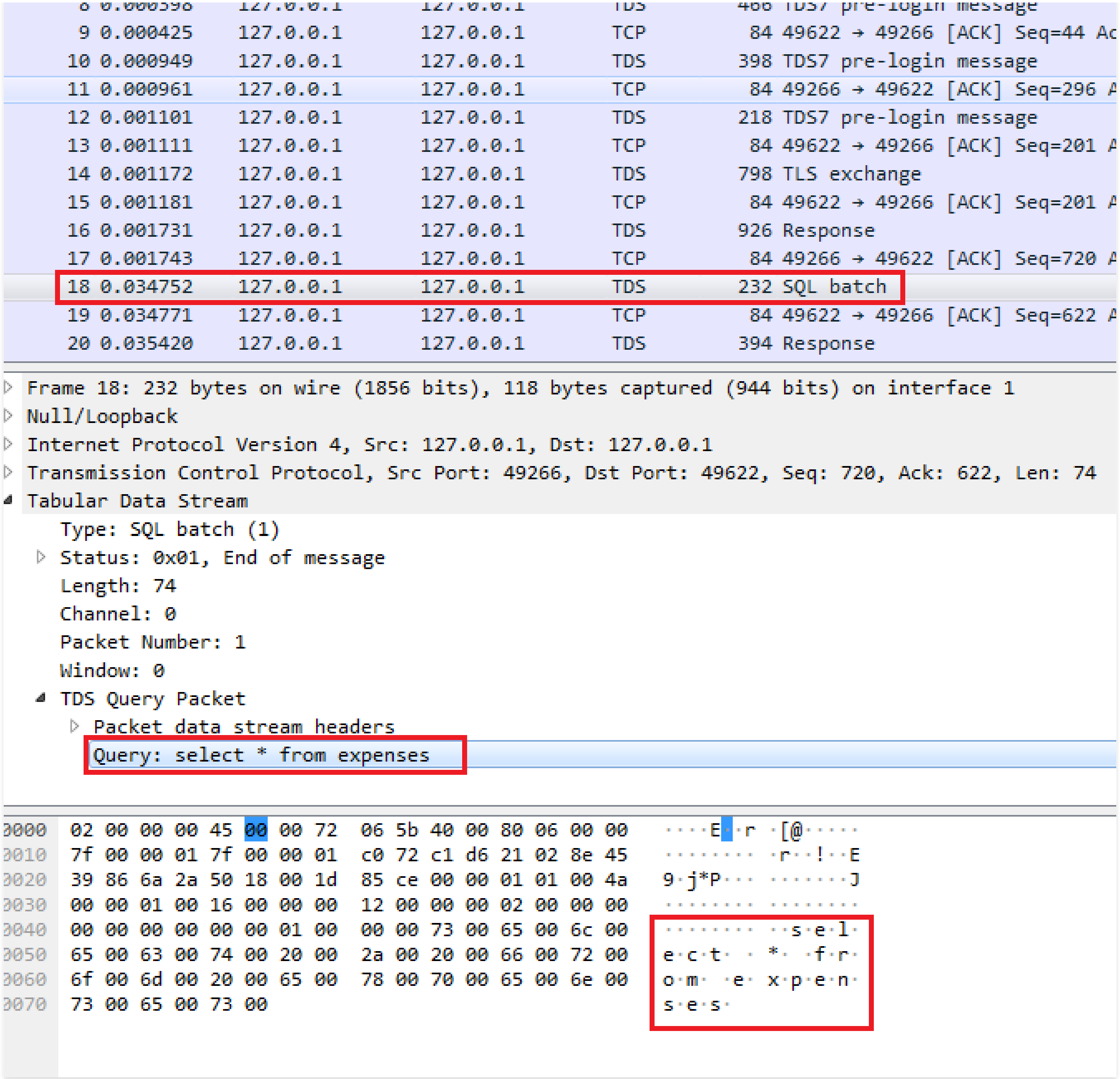
1. Connection to the MSSQL server.
2. Connection to the FTP server.

The connection to the MSSQL server is similar to what we have seen before (port `49622`).



```
      4 0.000159    127.0.0.1        127.0.0.1        TDS        292 TDS7 pre-login message
      5 0.000170    127.0.0.1        127.0.0.1        TCP         84 49622 → 49266 [ACK] Seq=1 Ack
      6 0.000369    127.0.0.1        127.0.0.1        TDS        170 Response
      7 0.000381    127.0.0.1        127.0.0.1        TCP         84 49266 → 49622 [ACK] Seq=105 A
      8 0.000398    127.0.0.1        127.0.0.1        TDS        466 TDS7 pre-login message
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



```
  8 0.000398   127.0.0.1   127.0.0.1   TDS   466 TDS7 pre-login message
  9 0.000425   127.0.0.1   127.0.0.1   TCP    84 49622 → 49266 [ACK] Seq=44 Ac
 10 0.000949   127.0.0.1   127.0.0.1   TDS   398 TDS7 pre-login message
 11 0.000961   127.0.0.1   127.0.0.1   TCP    84 49266 → 49622 [ACK] Seq=296 A
 12 0.001101   127.0.0.1   127.0.0.1   TDS   218 TDS7 pre-login message
 13 0.001111   127.0.0.1   127.0.0.1   TCP    84 49622 → 49266 [ACK] Seq=201 A
 14 0.001172   127.0.0.1   127.0.0.1   TDS   798 TLS exchange
 15 0.001181   127.0.0.1   127.0.0.1   TCP    84 49622 → 49266 [ACK] Seq=201 A
 16 0.001731   127.0.0.1   127.0.0.1   TDS   926 Response
 17 0.001743   127.0.0.1   127.0.0.1   TCP    84 49266 → 49622 [ACK] Seq=720 A
 18 0.034752   127.0.0.1   127.0.0.1   TDS   232 SQL batch
 19 0.034771   127.0.0.1   127.0.0.1   TCP    84 49622 → 49266 [ACK] Seq=622 A
 20 0.035420   127.0.0.1   127.0.0.1   TDS   394 Response
```

```
▷ Frame 18: 232 bytes on wire (1856 bits), 118 bytes captured (944 bits) on interface 1
▷ Null/Loopback
▷ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▷ Transmission Control Protocol, Src Port: 49266, Dst Port: 49622, Seq: 720, Ack: 622, Len: 74
◢ Tabular Data Stream
     Type: SQL batch (1)
   ▷ Status: 0x01, End of message
     Length: 74
     Channel: 0
     Packet Number: 1
     Window: 0
   ◢ TDS Query Packet
      ▷ Packet data stream headers
        Query: select * from expenses
```

```
0000  02 00 00 00 45 00 00 72  06 5b 40 00 80 06 00 00   ····E··r ·[@·····
0010  7f 00 00 01 7f 00 00 01  c0 72 c1 d6 21 02 8e 45   ········ ·r··!··E
0020  39 86 6a 2a 50 18 00 1d  85 ce 00 00 01 01 00 4a   9·j*P··· ·······J
0030  00 00 01 00 16 00 00 00  12 00 00 00 02 00 00 00   ········ ········
0040  00 00 00 00 00 00 01 00  00 00 73 00 65 00 6c 00   ········ ··s·e·l·
0050  65 00 63 00 74 00 20 00  2a 00 20 00 66 00 72 00   e·c·t· · *· ·f·r·
0060  6f 00 6d 00 20 00 65 00  78 00 70 00 65 00 6e 00   o·m· ·e· x·p·e·n·
0070  73 00 65 00 73 00                                  s·e·s·
```

Backup traffic to MSSQL server

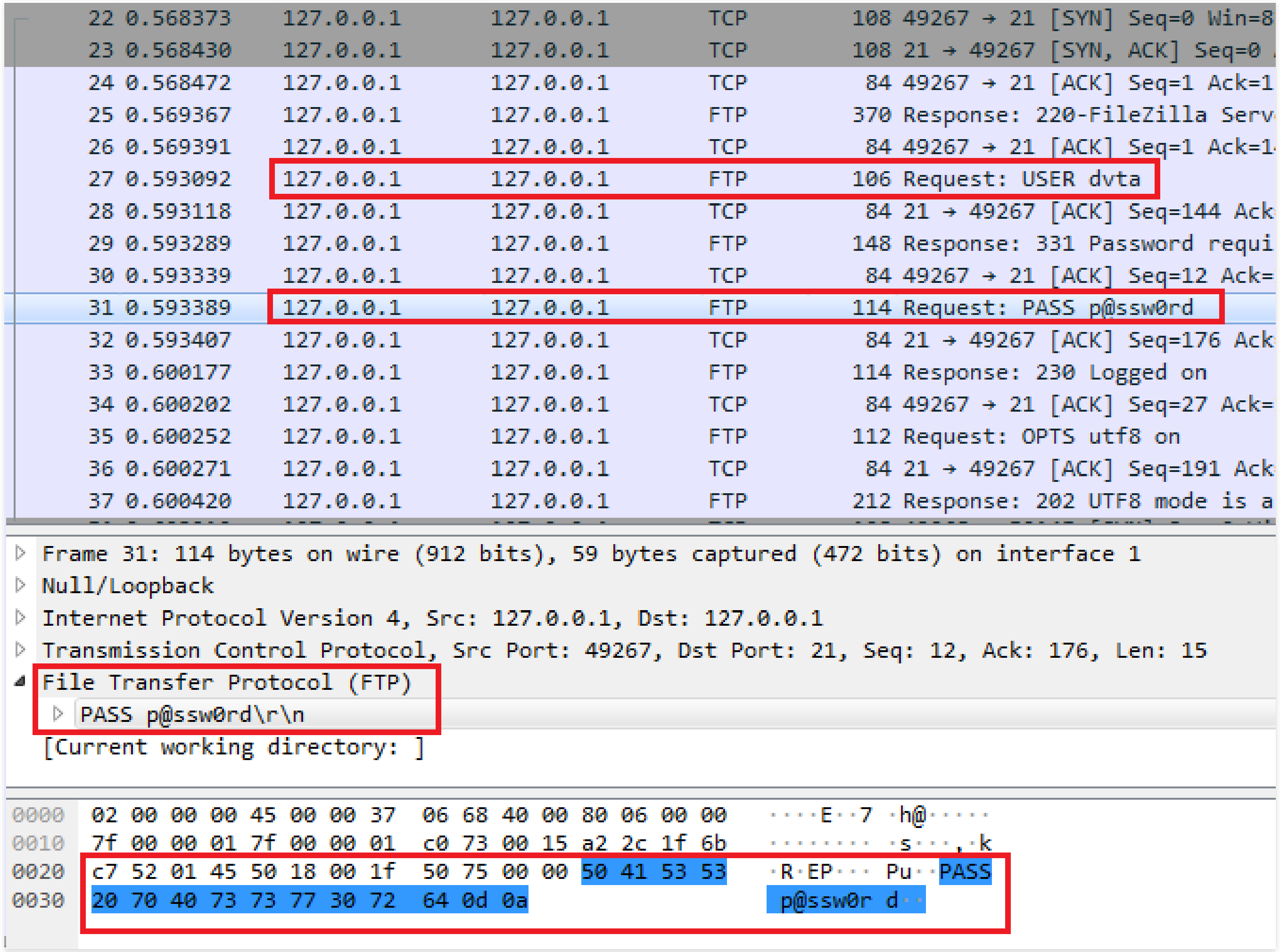
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

The application connects and runs the following query:

- `select * from expenses`

Next is the FTP connection to `localhost:22`. We can see it's in cleartext and user/pass is visible.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

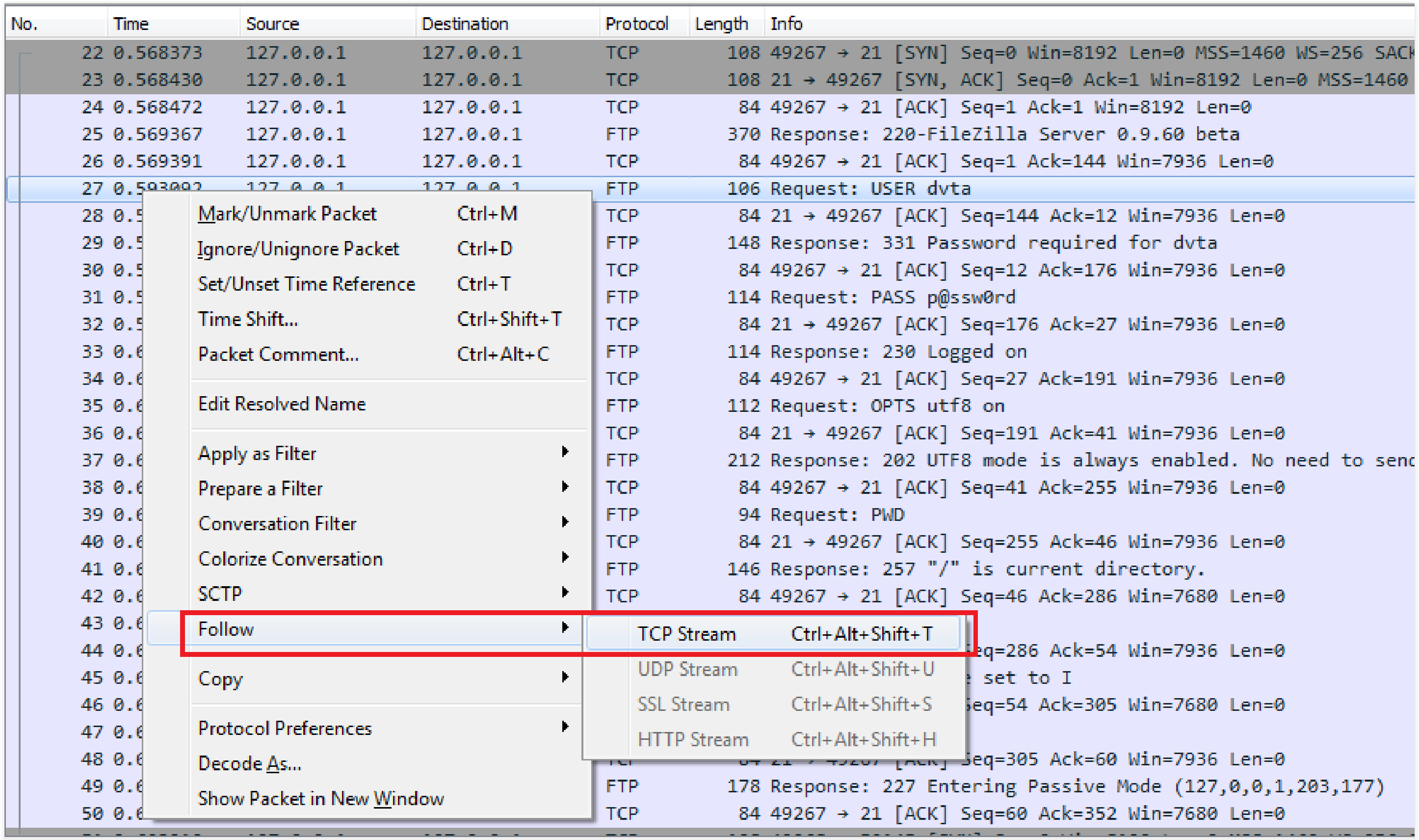


```
 22 0.568373   127.0.0.1   127.0.0.1   TCP   108 49267 → 21 [SYN] Seq=0 Win=8
 23 0.568430   127.0.0.1   127.0.0.1   TCP   108 21 → 49267 [SYN, ACK] Seq=0
 24 0.568472   127.0.0.1   127.0.0.1   TCP    84 49267 → 21 [ACK] Seq=1 Ack=1
 25 0.569367   127.0.0.1   127.0.0.1   FTP   370 Response: 220-FileZilla Serv
 26 0.569391   127.0.0.1   127.0.0.1   TCP    84 49267 → 21 [ACK] Seq=1 Ack=1
 27 0.593092   127.0.0.1   127.0.0.1   FTP   106 Request: USER dvta
 28 0.593118   127.0.0.1   127.0.0.1   TCP    84 21 → 49267 [ACK] Seq=144 Ack
 29 0.593289   127.0.0.1   127.0.0.1   FTP   148 Response: 331 Password requi
 30 0.593339   127.0.0.1   127.0.0.1   TCP    84 49267 → 21 [ACK] Seq=12 Ack=
 31 0.593389   127.0.0.1   127.0.0.1   FTP   114 Request: PASS p@ssw0rd
 32 0.593407   127.0.0.1   127.0.0.1   TCP    84 21 → 49267 [ACK] Seq=176 Ack
 33 0.600177   127.0.0.1   127.0.0.1   FTP   114 Response: 230 Logged on
 34 0.600202   127.0.0.1   127.0.0.1   TCP    84 49267 → 21 [ACK] Seq=27 Ack=
 35 0.600252   127.0.0.1   127.0.0.1   FTP   112 Request: OPTS utf8 on
 36 0.600271   127.0.0.1   127.0.0.1   TCP    84 21 → 49267 [ACK] Seq=191 Ack
 37 0.600420   127.0.0.1   127.0.0.1   FTP   212 Response: 202 UTF8 mode is a

▷ Frame 31: 114 bytes on wire (912 bits), 59 bytes captured (472 bits) on interface 1
▷ Null/Loopback
▷ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▷ Transmission Control Protocol, Src Port: 49267, Dst Port: 21, Seq: 12, Ack: 176, Len: 15
◢ File Transfer Protocol (FTP)
  ▷ PASS p@ssw0rd\r\n
  [Current working directory: ]

0000  02 00 00 00 45 00 00 37  06 68 40 00 80 06 00 00   ····E··7 ·h@·····
0010  7f 00 00 01 7f 00 00 01  c0 73 00 15 a2 2c 1f 6b   ········ ·s···,·k
0020  c7 52 01 45 50 18 00 1f  50 75 00 00 50 41 53 53   ·R·EP··· Pu··PASS
0030  20 70 40 73 73 77 30 72  64 0d 0a                   p@ssw0r d··
```

FTP traffic and password displayed in Wireshark

For easier visualization, right-click on any packet in the stream and select `Follow > TCP Stream`.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



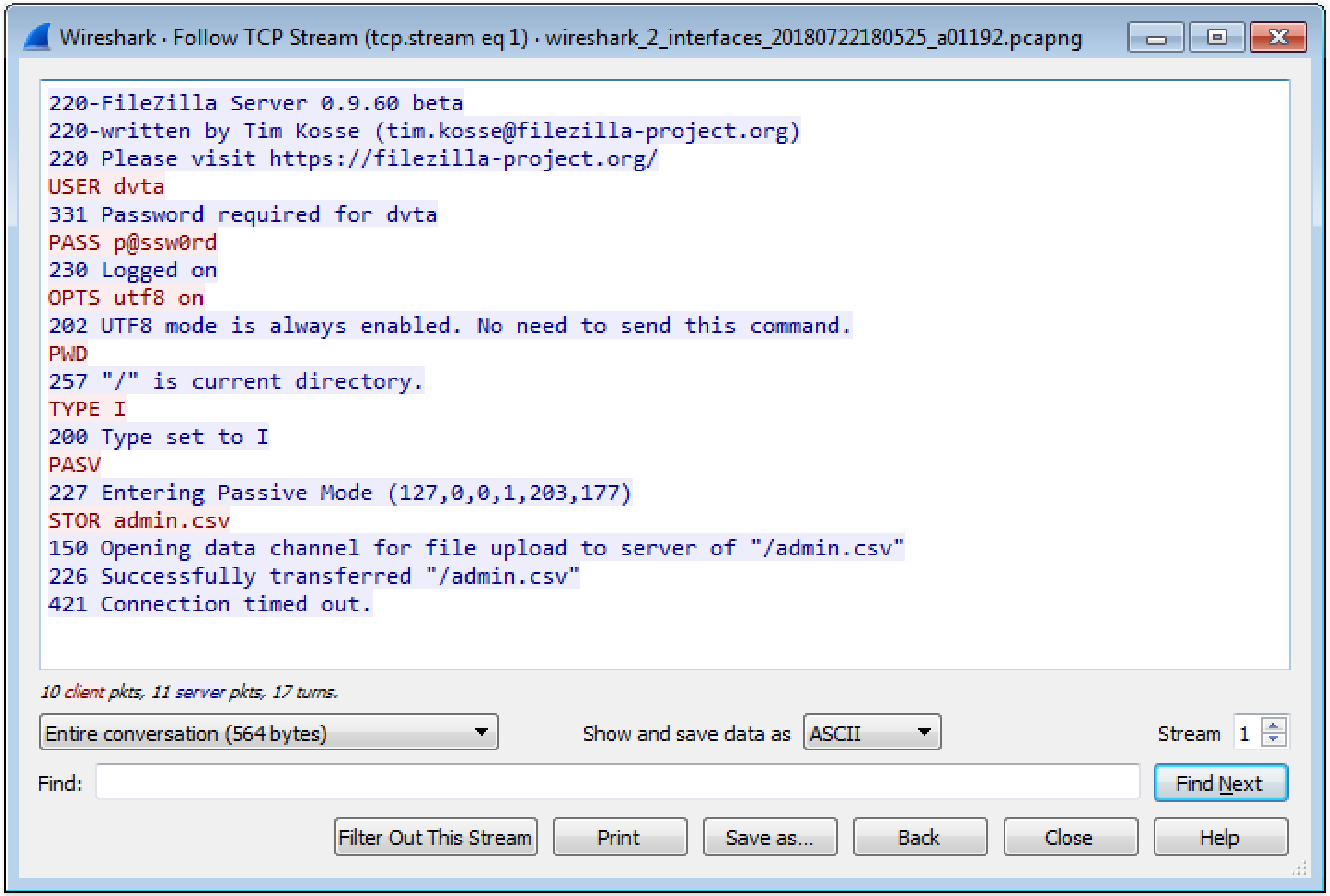
```
No.   Time        Source       Destination   Protocol  Length  Info
   22 0.568373    127.0.0.1    127.0.0.1     TCP          108 49267 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK
   23 0.568430    127.0.0.1    127.0.0.1     TCP          108 21 → 49267 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
   24 0.568472    127.0.0.1    127.0.0.1     TCP           84 49267 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0
   25 0.569367    127.0.0.1    127.0.0.1     FTP          370 Response: 220-FileZilla Server 0.9.60 beta
   26 0.569391    127.0.0.1    127.0.0.1     TCP           84 49267 → 21 [ACK] Seq=1 Ack=144 Win=7936 Len=0
   27 0.593092    127.0.0.1    127.0.0.1     FTP          106 Request: USER dvta
   28 0.5                                    TCP           84 21 → 49267 [ACK] Seq=144 Ack=12 Win=7936 Len=0
   29 0.5                                    FTP          148 Response: 331 Password required for dvta
   30 0.5                                    TCP           84 49267 → 21 [ACK] Seq=12 Ack=176 Win=7936 Len=0
   31 0.5                                    FTP          114 Request: PASS p@ssw0rd
   32 0.5                                    TCP           84 21 → 49267 [ACK] Seq=176 Ack=27 Win=7936 Len=0
   33 0.6                                    FTP          114 Response: 230 Logged on
   34 0.6                                    TCP           84 49267 → 21 [ACK] Seq=27 Ack=191 Win=7936 Len=0
   35 0.6                                    FTP          112 Request: OPTS utf8 on
   36 0.6                                    TCP           84 21 → 49267 [ACK] Seq=191 Ack=41 Win=7936 Len=0
   37 0.6                                    FTP          212 Response: 202 UTF8 mode is always enabled. No need to send
   38 0.6                                    TCP           84 49267 → 21 [ACK] Seq=41 Ack=255 Win=7936 Len=0
   39 0.6                                    FTP           94 Request: PWD
   40 0.6                                    TCP           84 21 → 49267 [ACK] Seq=255 Ack=46 Win=7936 Len=0
   41 0.6                                    FTP          146 Response: 257 "/" is current directory.
   42 0.6                                    TCP           84 49267 → 21 [ACK] Seq=46 Ack=286 Win=7680 Len=0
   43 0.6
   44 0.6                                                                                  eq=286 Ack=54 Win=7936 Len=0
   45 0.6                                                                                  set to I
   46 0.6                                                                                  eq=54 Ack=305 Win=7680 Len=0
   47 0.6
   48 0.6                                                                                  eq=305 Ack=60 Win=7936 Len=0
   49 0.6                                    FTP          178 Response: 227 Entering Passive Mode (127,0,0,1,203,177)
   50 0.6                                    TCP           84 49267 → 21 [ACK] Seq=60 Ack=352 Win=7680 Len=0
```

Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
TCP Stream Ctrl+Alt+Shift+T
UDP Stream Ctrl+Alt+Shift+U
SSL Stream Ctrl+Alt+Shift+S
HTTP Stream Ctrl+Alt+Shift+H

Following the TCP stream in Wireshark

Application logins with `dvta:p@ssw0rd` and then stores `admin.csv` on the FTP server (which we can assume contains information from the `expenses` table).

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Wireshark · Follow TCP Stream (tcp.stream eq 1) · wireshark_2_interfaces_20180722180525_a01192.pcapng

```
220-FileZilla Server 0.9.60 beta
220-written by Tim Kosse (tim.kosse@filezilla-project.org)
220 Please visit https://filezilla-project.org/
USER dvta
331 Password required for dvta
PASS p@ssw0rd
230 Logged on
OPTS utf8 on
202 UTF8 mode is always enabled. No need to send this command.
PWD
257 "/" is current directory.
TYPE I
200 Type set to I
PASV
227 Entering Passive Mode (127,0,0,1,203,177)
STOR admin.csv
150 Opening data channel for file upload to server of "/admin.csv"
226 Successfully transferred "/admin.csv"
421 Connection timed out.
```

10 client pkts, 11 server pkts, 17 turns.

Entire conversation (564 bytes)

Show and save data as: ASCII

Stream: 1

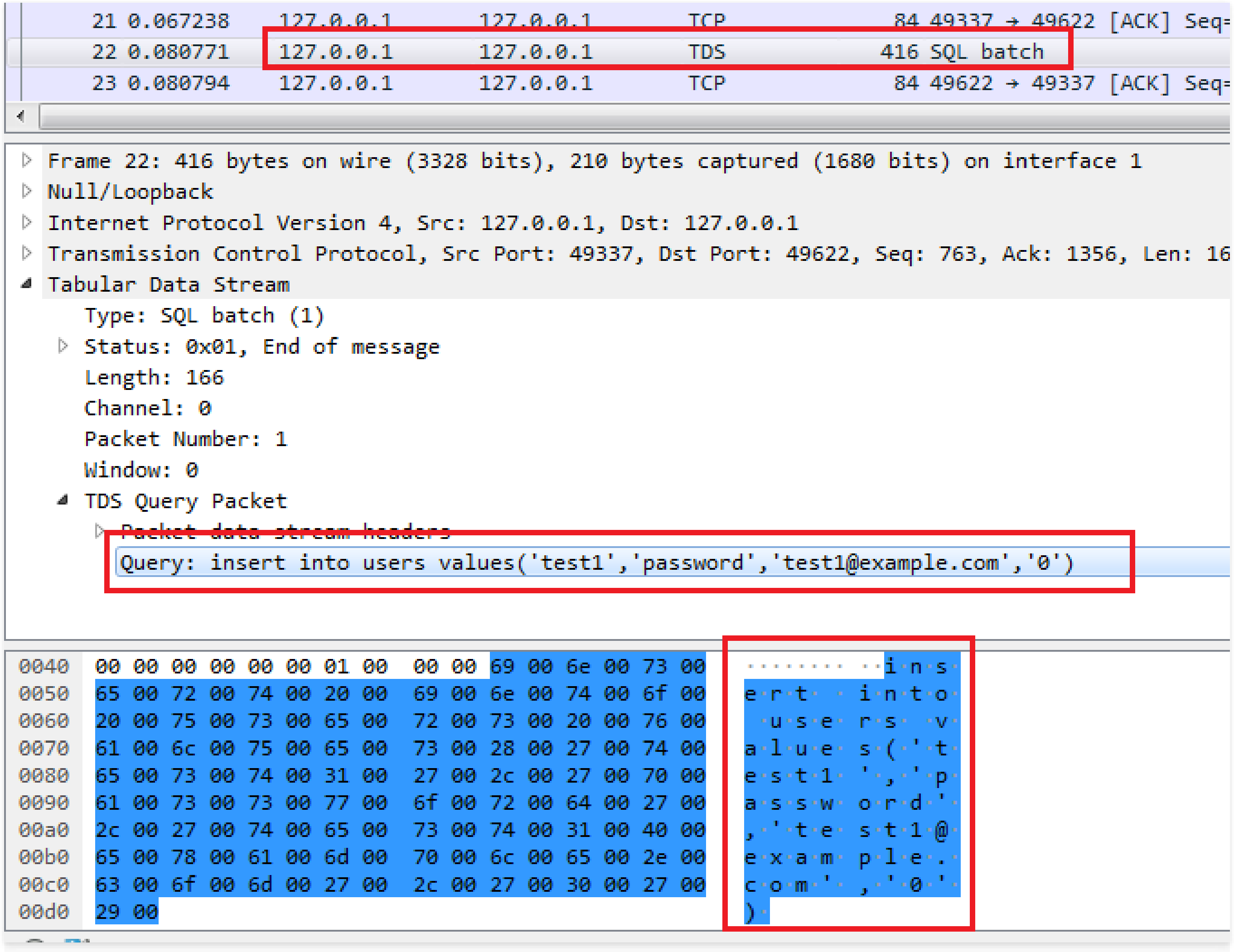
Find:

All FTP traffic in stream displayed in Wireshark

Register Functionality

We can also register new users. Users will not be administrators. Let's look at that traffic too.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



```
21 0.067238   127.0.0.1   127.0.0.1   TCP    84 49337 → 49622 [ACK] Seq=
22 0.080771   127.0.0.1   127.0.0.1   TDS   416 SQL batch
23 0.080794   127.0.0.1   127.0.0.1   TCP    84 49622 → 49337 [ACK] Seq=
```

```
▷ Frame 22: 416 bytes on wire (3328 bits), 210 bytes captured (1680 bits) on interface 1
▷ Null/Loopback
▷ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▷ Transmission Control Protocol, Src Port: 49337, Dst Port: 49622, Seq: 763, Ack: 1356, Len: 16
◢ Tabular Data Stream
    Type: SQL batch (1)
  ▷ Status: 0x01, End of message
    Length: 166
    Channel: 0
    Packet Number: 1
    Window: 0
  ◢ TDS Query Packet
    ▷ Packet data stream headers
      Query: insert into users values('test1','password','test1@example.com','0')
```

```
0040  00 00 00 00 00 00 01 00  00 00 69 00 6e 00 73 00   ........ ..i.n.s.
0050  65 00 72 00 74 00 20 00  69 00 6e 00 74 00 6f 00   e.r.t. . i.n.t.o.
0060  20 00 75 00 73 00 65 00  72 00 73 00 20 00 76 00    .u.s.e. r.s. .v.
0070  61 00 6c 00 75 00 65 00  73 00 28 00 27 00 74 00   a.l.u.e. s.(.'.t.
0080  65 00 73 00 74 00 31 00  27 00 2c 00 27 00 70 00   e.s.t.1. '.,.'.p.
0090  61 00 73 00 73 00 77 00  6f 00 72 00 64 00 27 00   a.s.s.w. o.r.d.'.
00a0  2c 00 27 00 74 00 65 00  73 00 74 00 31 00 40 00   ,.'.t.e. s.t.1.@.
00b0  65 00 78 00 61 00 6d 00  70 00 6c 00 65 00 2e 00   e.x.a.m. p.l.e...
00c0  63 00 6f 00 6d 00 27 00  2c 00 27 00 30 00 27 00   c.o.m.'. ,.'.0.'.
00d0  29 00                                              ).
```

SQL statement to register a new user

As we can see, traffic is similar to the previous parts. This time we are sending an `insert` query:

- `insert into users values('test1','password','test1@example.com','0')`

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

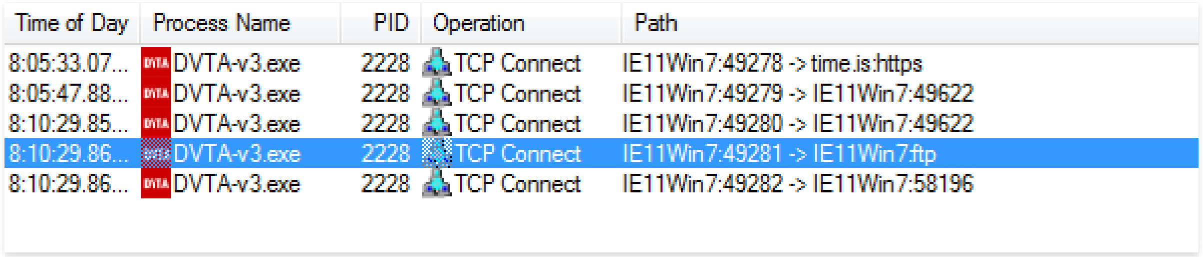
Note the `0` in the end. It's setting the `isadmin` column that we observed earlier.

Recon with Procmon

We can do the same with Sysinternals Procmon. We can see the traffic but we can identify the endpoints. For the record, Procmon does a lot more than what we are using it for.

Quit the application, run it again, login as admin and backup the data. Then run Procmon and set the following filters similar to what we did in part 1 to identify the FTP endpoint:

- `Process Name contains dvta`. I have set this to `contains` because I have versioned patched executables from part 2.
- `Operation is TCP Connect`. Or you could only enable network activity like part 1 (DVTA - Part 1 - Setup - Discover the FTP Address).



Time of Day	Process Name	PID	Operation	Path
8:05:33.07...	DVTA-v3.exe	2228	TCP Connect	IE11Win7:49278 -> time.is:https
8:05:47.88...	DVTA-v3.exe	2228	TCP Connect	IE11Win7:49279 -> IE11Win7:49622
8:10:29.85...	DVTA-v3.exe	2228	TCP Connect	IE11Win7:49280 -> IE11Win7:49622
8:10:29.86...	DVTA-v3.exe	2228	TCP Connect	IE11Win7:49281 -> IE11Win7:ftp
8:10:29.86...	DVTA-v3.exe	2228	TCP Connect	IE11Win7:49282 -> IE11Win7:58196

Application endpoints displayed in Procmon

We can see connections to:

- Fetching login token from `time.is:443`.
- MSSQL server at `localhost:49622`.
- FTP at `localhost:22` and `54823` (ephemeral port for actual `STOR` action).

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Conclusion

We learned how to identify network endpoints using two tools. We did some limited traffic analysis. In the next part, we will learn how to manipulate traffic in different ways.

Posted by Parsia • Jul 30, 2018 • Tags: Wireshark Procmon

DVTA - Part 2 - Cert Pinning and Login Button

DVTA - Part 4 - Traffic Tampering with dnSpy

0 Comments Parsiya 1 Login

♡ Recommend Tweet Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

✉ Subscribe Add Disqus to your site Disqus' Privacy Policy DISQUS

Copyright © 2019 Parsia - License - Powered by Hugo and Hugo-Octopress theme.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD