



Jérémy Barrette – 1736976  
Alexis Vailles – 1742139

**Rapport TP #3 :  
Analyse d'applications client-serveur avec WireShark**

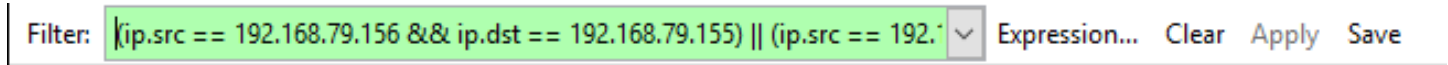
Soumis à : Kadi, Mehdi  
INF3405 (01 – B1) – Réseaux informatiques  
Session Automne 2018

École Polytechnique de Montréal  
Mercredi le 5 décembre 2018

# Analyse de l'application client/serveur du laboratoire 1

1) Le filtre appliqué est :

ip.addr == 192.168.79.156 && ip.addr == 192.168.79.155

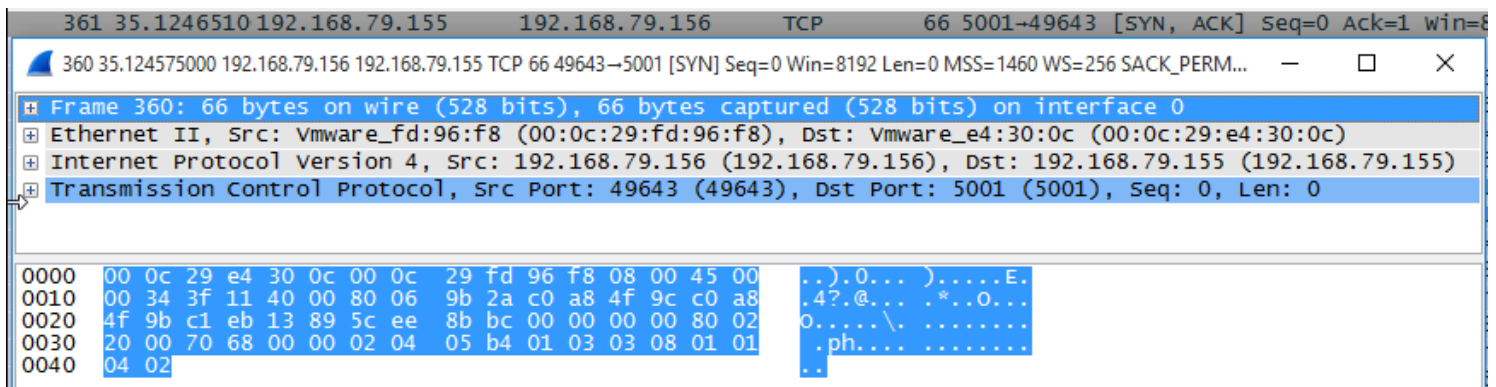


Il est à noter que le filtre utilisé sur la capture d'écran est :

(ip.src == 192.168.79.156 && ip.dst == 192.168.79.155) || (ip.src == 192.168.79.155 && ip.dst == 192.168.79.156)

Ce filtre est un équivalent au filtre énoncé plus haut, mais est passablement plus compliqué à écrire. C'est pourquoi nous l'avons changé après avoir fait la capture d'écran.

2) Le protocole de la couche 4 utilisé est le protocole **TCP**.



3)

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Source IP Addresses	49				0,0004	100%	0,1000	123,047
192.168.79.156	32				0,0003	65,31%	0,0700	123,047
192.168.79.155	17				0,0001	34,69%	0,0300	123,110

### Client => Serveur

Filtre appliqué : ip.src == 192.168.79.156 && ip.dst == 192.168.79.155

<b>Display</b>						
Display filter:		(ip.src == 192.168.79.156 && ip.dst == 192.168.79.155)				
Ignored packets:		0 (0,000%)				
Traffic	Captured	Displayed	Displayed %	Marked	Marked %	
Packets	2425496	32	0,001%	0	0,000%	
Between first and last packet	297,576 sec	127,731 sec				
Avg. packets/sec	8150,838	0,251				
Avg. packet size	1332 bytes	261 bytes				
Bytes	3230051167	8355	0,000%	0	0,000%	
Avg. bytes/sec	10854531,348	65,411				
Avg. MBit/sec	86,836	0,001				

Il y a **32** paquets envoyés du client vers le serveur;

Il y a **8355** octets envoyés du client vers le serveur.

### Serveur => Client

Filtre appliqué : ip.src == 192.168.79.155 && ip.dst == 192.168.79.156

<b>Display</b>						
Display filter:		(ip.src == 192.168.79.155 && ip.dst == 192.168.79.156)				
Ignored packets:		0 (0,000%)				
Traffic	Captured	Displayed	Displayed %	Marked	Marked %	
Packets	2425496	17	0,001%	0	0,000%	
Between first and last packet	297,576 sec	108,857 sec				
Avg. packets/sec	8150,838	0,156				
Avg. packet size	1332 bytes	409 bytes				
Bytes	3230051167	6961	0,000%	0	0,000%	
Avg. bytes/sec	10854531,348	63,946				
Avg. MBit/sec	86,836	0,001				

Il y a **17** paquets envoyés du serveur vers le client;

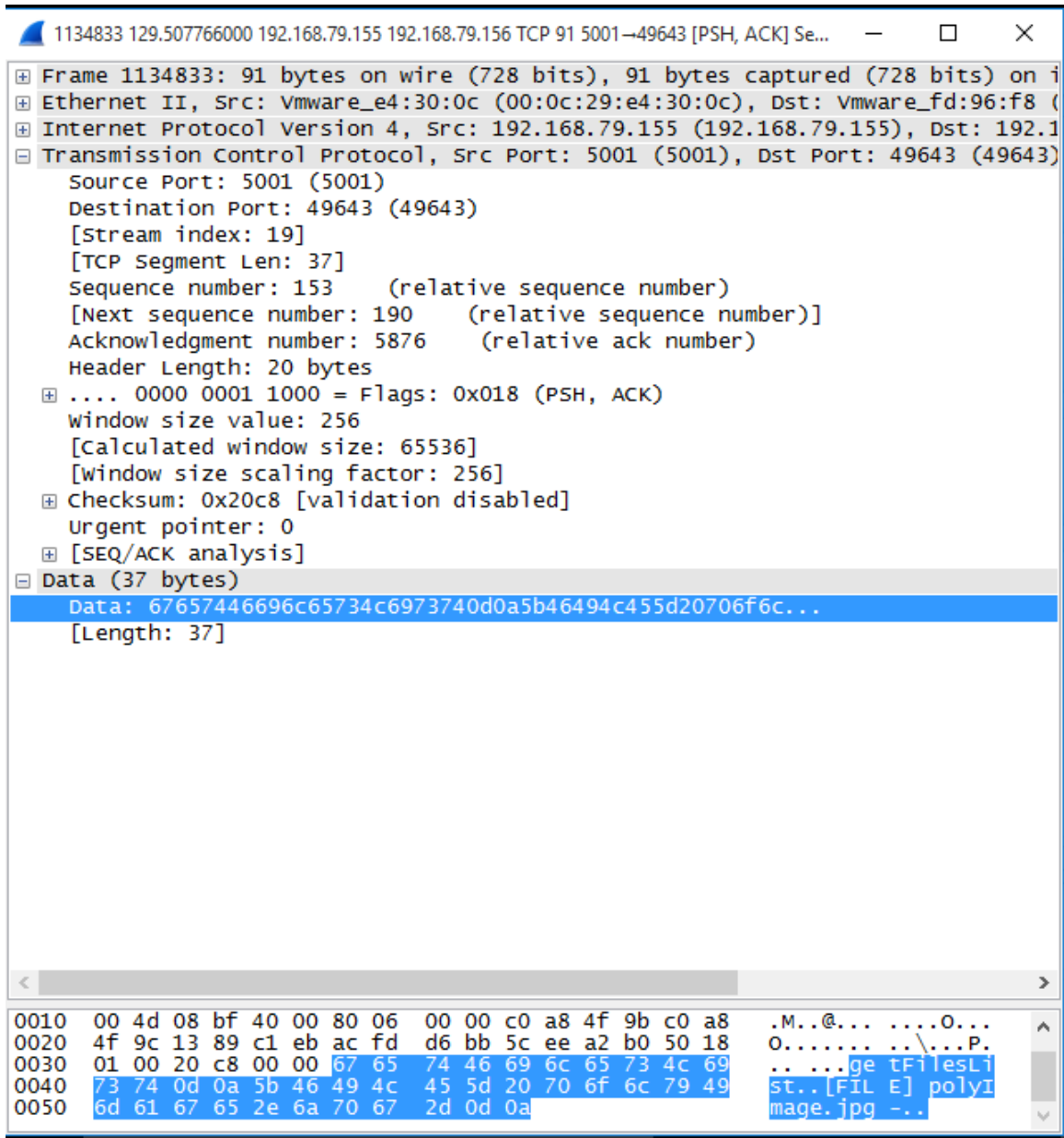
Il y a **6961** octets envoyés du serveur vers le client.

4)

```
1258449 143.981435000 192.168.79.155 192.168.79.156 TCP 5856 5001→49643 [PSH, ACK] Seq=230 Ack=5905 Win=65536 Len=5802
Frame 1258449: 5856 bytes on wire (46848 bits), 5856 bytes captured (46848 bits) on interface 0
Ethernet II, Src: Vmware_e4:30:0c (00:0c:29:e4:30:0c), Dst: Vmware_fd:96:f8 (00:0c:29:fd:96:f8)
  Destination: Vmware_fd:96:f8 (00:0c:29:fd:96:f8)
  Source: Vmware_e4:30:0c (00:0c:29:e4:30:0c)
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 192.168.79.155 (192.168.79.155), Dst: 192.168.79.156 (192.168.79.156)
  Version: 4
  Header Length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
  [Total Length: 5842 bytes (reported as 0, presumed to be because of "TCP segmentation offload" (TSO))]
  Identification: 0x08c3 (2243)
  Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 128
  Protocol: TCP (6)
  Header checksum: 0x0000 [validation disabled]
  Source: 192.168.79.155 (192.168.79.155)
  Destination: 192.168.79.156 (192.168.79.156)
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: 5001 (5001), Dst Port: 49643 (49643), Seq: 230, Ack: 5905, Len: 5802
  Source Port: 5001 (5001)
  Destination Port: 49643 (49643)
  [Stream index: 19]
  [TCP Segment Len: 5802]
  Sequence number: 230 (relative sequence number)
  [Next sequence number: 6032 (relative sequence number)]
  Acknowledgment number: 5905 (relative ack number)
  Header Length: 20 bytes
  ... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
  window size value: 256
  [Calculated window size: 65536]
  [window size scaling factor: 256]
  Checksum: 0x208f [validation disabled]
  Urgent pointer: 0
  [SEQ/ACK analysis]
Data (5802 bytes)
  Data: ffd8ffe000104a46494600010100000100010000ffdb0084...
  [Length: 5802]
```

Le serveur envoie un paquet de 5856 octets vers le client, ce qui est plus élevé que 1518. Ce paquet a pu être envoyé parce que ses données n'étaient pas fragmentables. Il a été envoyé selon le format **jumbo frame**.

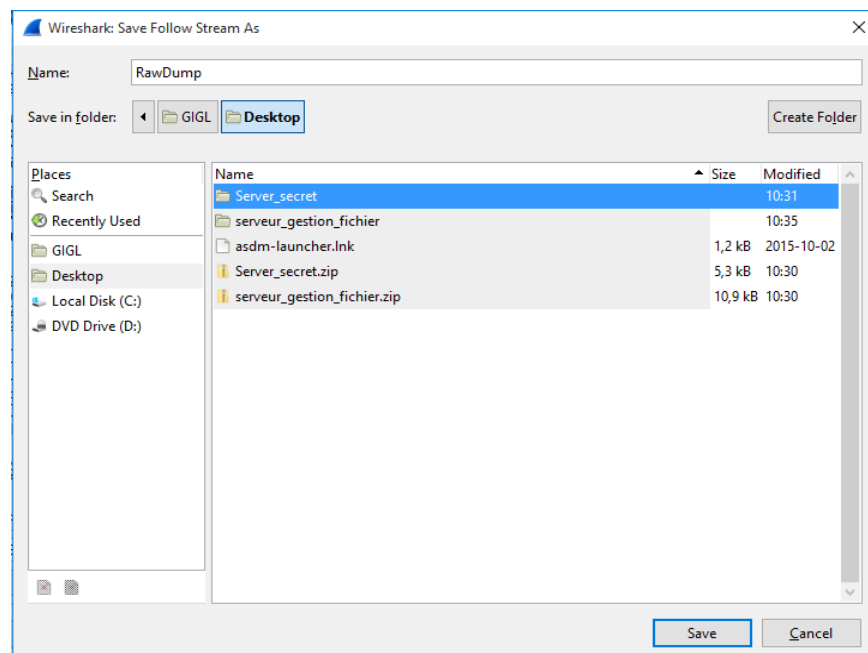
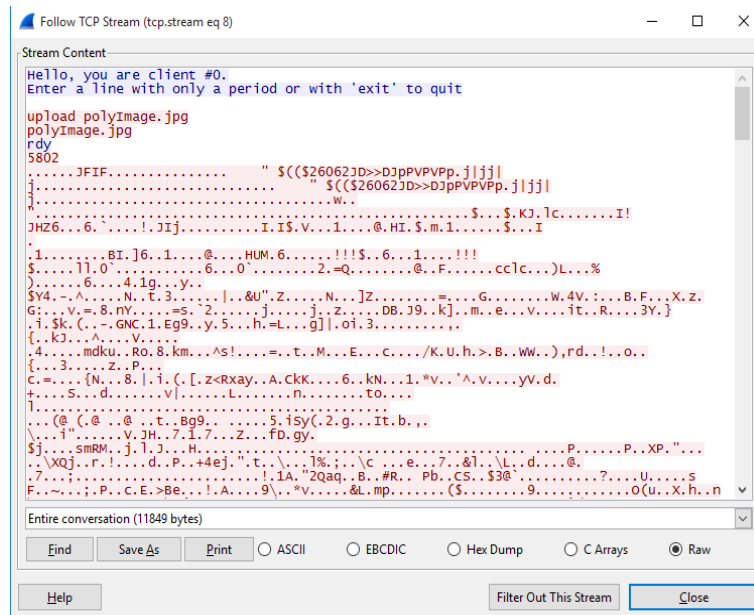
5)



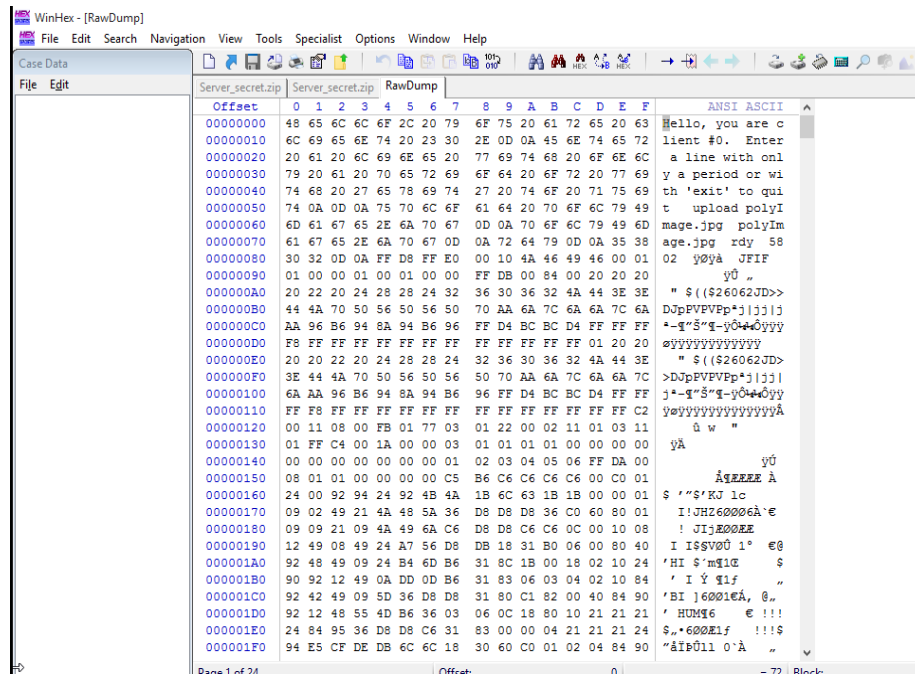
La commande « ls » nous permet d'intercepter la liste de fichiers et dossiers contenus dans le serveur.

6) Comme nous ne sommes pas parvenus à effectuer cet exercice correctement durant les heures de laboratoire, nous avons emprunté les captures d'écran d'une autre équipe. Voici donc comment, avec Wireshark, on peut extraire l'image envoyée par le client ou l'image envoyer par le serveur vers le client :

**Étape 1 :** On sauvegarde le *TCP stream* en format *.raw* :

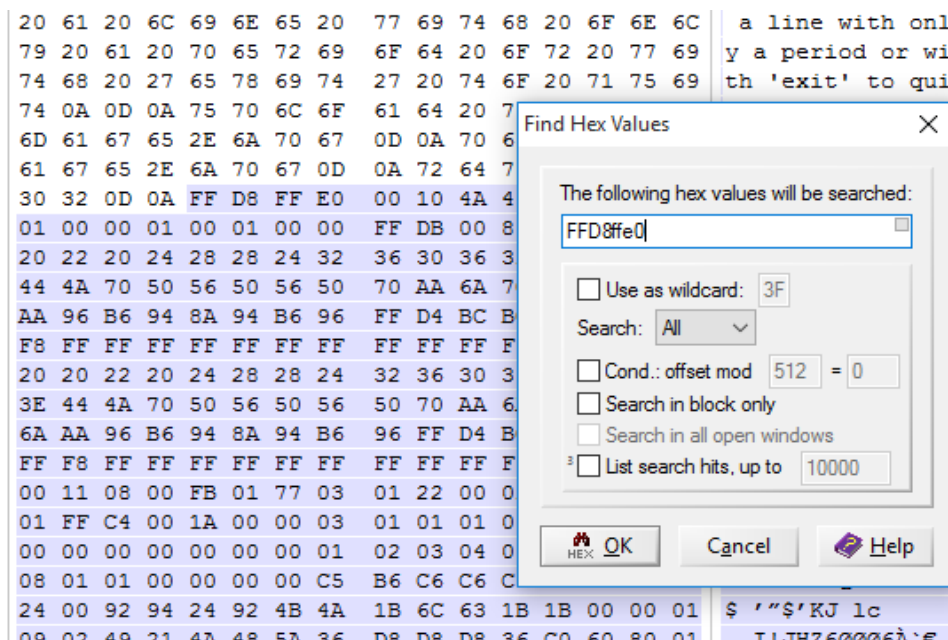


**Étape 2 :** On ouvre le *TCP stream* dans *WinHex* :

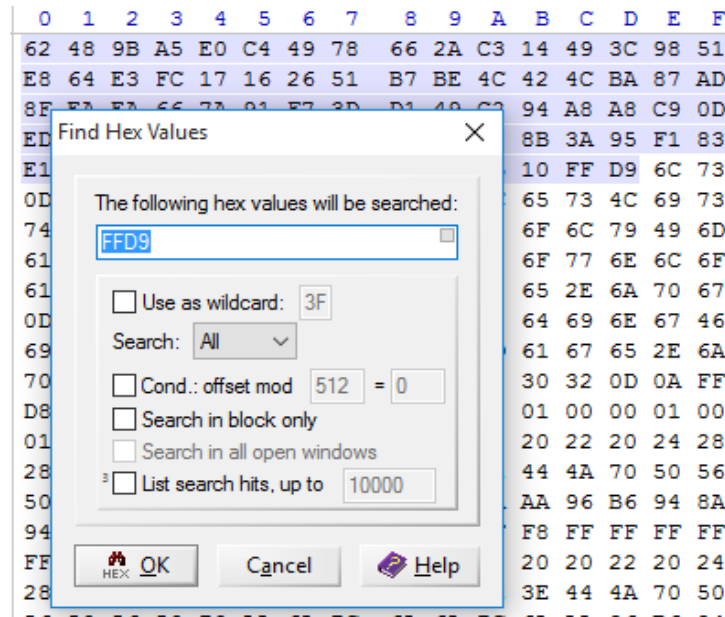


**Étape 3 :** Parmi les données représentées, on repère où commencent les données associées à l'image (après l'en-tête), et où elles se terminent :

Début :



Fin :

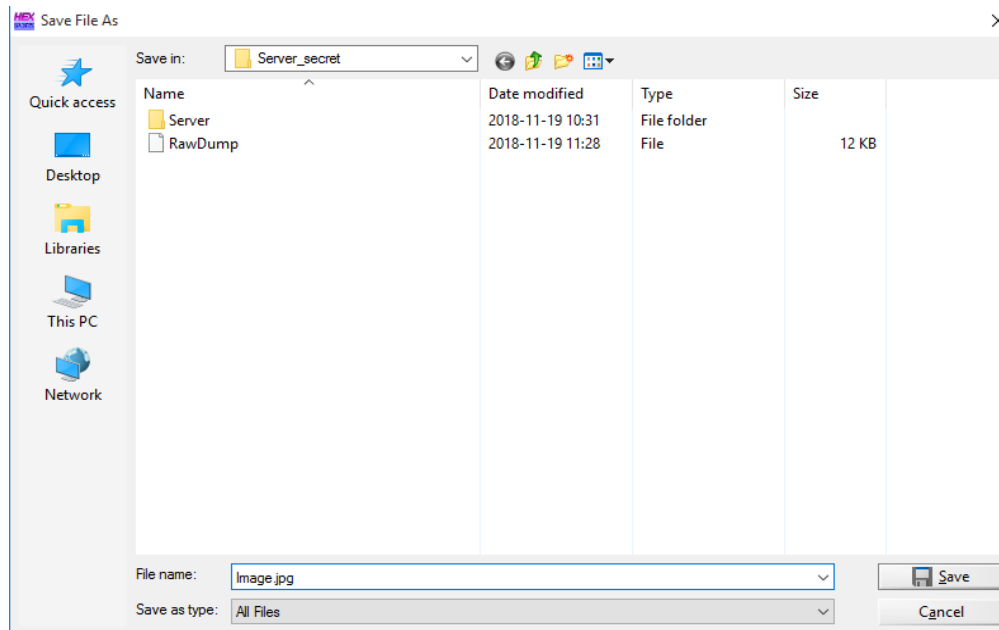


**Étape 4 :** On ne conserve que les données associées à l'image (donc on supprime le reste) :

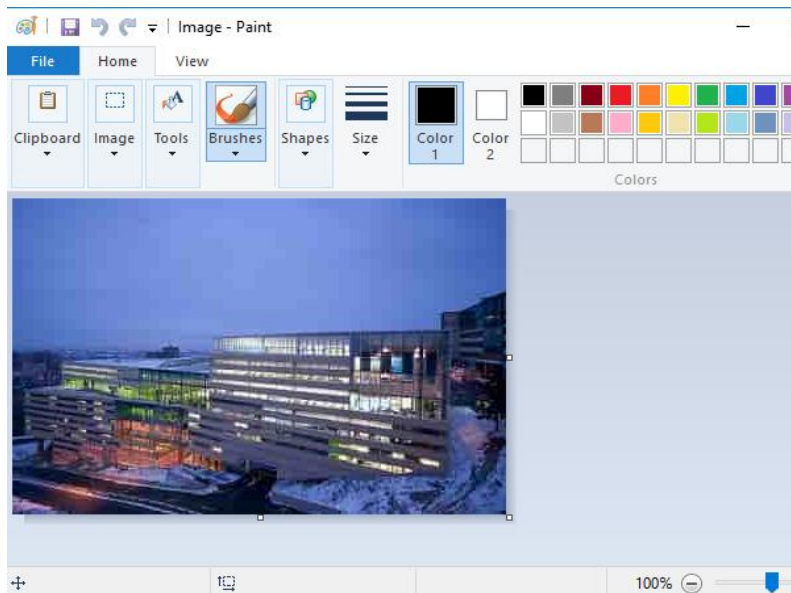
RawDump										
Offset	0	1	2	3	4	5	6	7	8	9
00000000	FF	D8	FF	E0	00	10	4A	46	49	46
00000010	00	01	00	00	FF	DB	00	84	00	20
00000020	28	28	24	32	36	30	36	32	4A	44
00000030	56	50	56	50	70	AA	6A	7C	6A	6A
00000040	8A	94	B6	96	FF	D4	BC	BC	D4	FF
00000050	FF	FF	FF	FF	FF	FF	FF	FF	FF	01
00000060	24	28	28	24	32	36	30	36	32	4A
00000070	50	56	50	56	50	70	AA	6A	7C	6A
00000080	94	8A	94	B6	96	FF	D4	BC	BC	D4
00000090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
000000A0	FB	01	77	03	01	22	00	02	11	01
000000B0	1A	00	00	03	01	01	01	01	00	00
000000C0	00	00	00	01	02	03	04	05	06	FF
000000D0	00	00	00	C5	B6	C6	C6	C6	C6	00
000000E0	24	92	4B	4A	1B	6C	63	1B	1B	00
000000F0	4A	48	5A	36	D8	D8	D8	36	C0	60
00000100	4A	49	6A	C6	D8	D8	C6	C6	0C	00
00000110	24	A7	56	D8	DB	18	31	B0	06	00
00000120	24	B4	6D	B6	31	8C	1B	00	18	02
00000130	0A	DD	0D	B6	31	83	06	03	04	02
00000140	5D	36	D8	D8	31	80	C1	82	00	40
00000150	4D	B6	36	03	06	0C	18	80	10	21
00000160	D8	D8	C6	31	83	00	00	04	21	21
00000170	DB	6C	6C	18	30	60	C0	01	02	04
00000180	EA	36	DB	1B	06	30	60	00	00	02
00000190	B2	3D	51	B6	DB	1B	18	C6	02	00



**Étape 5 :** On sauvegarde les données de l'image en format *.jpeg* :



**Étape 6 :** On peut maintenant visualiser l'image en l'ouvrant avec un logiciel qui permet de l'afficher, comme *Paint* :

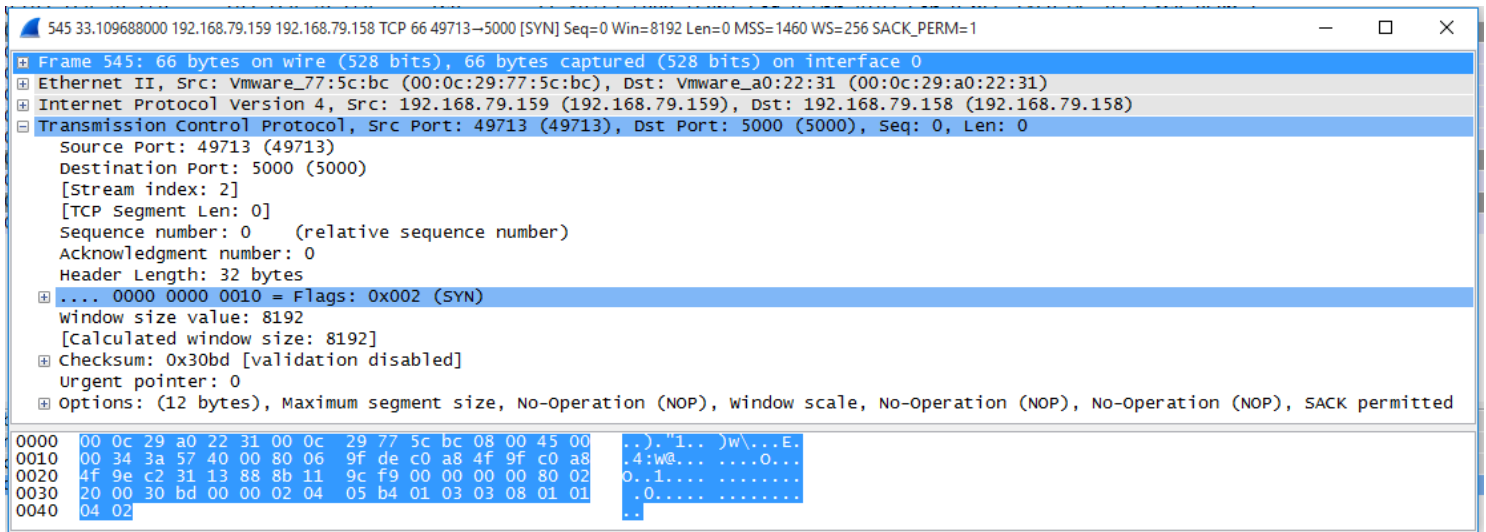


7) Suite à notre analyse, nous constatons qu'il est très facile d'intercepter et de récupérer les données et informations transmises entre le serveur et le client. La sécurité de la connexion est donc plutôt faible.

# Analyse d'une application client-serveur "secrète"

## Mode secret 1

1)



C'est le protocole TCP qui est utilisé pour la couche de transport. Ce protocole est un SYN (pour synchronisation).

2)

Transmission Control Protocol, Src Port: 49713 (49713), Dst Port: 5000 (5000), seq: 0, Len: 0

Port source : 49713

Port destination : 5000

3)

### Client => Serveur

Filtre appliqué : ip.src == 192.168.79.159 && ip.dst == 192.168.79.158

Display						
Display filter:		ip.dst == 192.168.79.158 && ip.src == 192.168.79.159				
Ignored packets:		0 (0,000%)				
Traffic	Captured	Displayed	Displayed %	Marked	Marked %	
Packets	555	7	1,261%	0	0,000%	
Between first and last packet	33,195 sec	0,085 sec				
Avg. packets/sec	16,719	82,145				
Avg. packet size	1146 bytes	630 bytes				
Bytes	635982	4408	0,693%	0	0,000%	
Avg. bytes/sec	19159,026	51728,071				
Avg. MBit/sec	0,153	0,414				

Il y a 7 paquets envoyés du client vers le serveur;

Il y a 4408 octets envoyés du client vers le serveur;

### Serveur => Client

Filtre appliqué : ip.src == 192.168.79.158 && ip.dst == 192.168.79.159

Display						
Display filter:		ip.src == 192.168.79.158 && ip.dst == 192.168.79.159				
Ignored packets:		0 (0,000%)				
Traffic	Captured	Displayed	Displayed %	Marked	Marked %	
Packets	555	4	0,721%	0	0,000%	
Between first and last packet	33,195 sec	0,084 sec				
Avg. packets/sec	16,719	47,618				
Avg. packet size	1146 bytes	57 bytes				
Bytes	635982	228	0,036%	0	0,000%	
Avg. bytes/sec	19159,026	2714,221				
Avg. MBit/sec	0,153	0,022				

Il y a 4 paquets envoyés du serveur vers le client;

Il y a 228 octets envoyés du serveur vers le client.

4)

548	33.1125310	192.168.79.159	192.168.79.158	IPA	1514	unknown	0x6c	[Malformed Packet]
549	33.1125320	192.168.79.159	192.168.79.158	IPA	1514	unknown	0x50	[Malformed Packet]
550	33.1125330	192.168.79.159	192.168.79.158	IPA	1134	unknown	0x6c	[Malformed Packet]

On observe 1 seule itération, divisée en 3 paquets envoyés simultanément.

Cela signifie donc que le client pour ce mode envoie toute l'information au serveur d'un bloc.

## Mode secret 2

1)

```
626 12.521309000 192.168.79.159 192.168.79.158 TCP 66 49727→5000 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1

Frame 626: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
  Interface id: 0 (\Device\NPF_{8E06E00D-24EF-4081-9B0A-40E8AC9CC125})
  Encapsulation type: Ethernet (1)
  Arrival Time: Nov 27, 2018 14:59:31.759232000 Eastern Standard Time
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1543348771.759232000 seconds
  [Time delta from previous captured frame: 2.597985000 seconds]
  [Time delta from previous displayed frame: 0.000000000 seconds]
  [Time since reference or first frame: 12.521309000 seconds]
  Frame Number: 626
  Frame Length: 66 bytes (528 bits)
  Capture Length: 66 bytes (528 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp]
  [Coloring Rule Name: TCP SYN/FIN]
  [Coloring Rule String: tcp.flags & 0x02 || tcp.flags.fin == 1]
  Ethernet II, Src: Vmware_77:5c:bc (00:0c:29:77:5c:bc), Dst: Vmware_a0:22:31 (00:0c:29:a0:22:31)
  Internet Protocol Version 4, Src: 192.168.79.159 (192.168.79.159), Dst: 192.168.79.158 (192.168.79.158)
  Transmission Control Protocol, Src Port: 49727 (49727), Dst Port: 5000 (5000), Seq: 0, Len: 0
    Source Port: 49727 (49727)
    Destination Port: 5000 (5000)
    [Stream index: 7]
    [TCP Segment Len: 0]
    Sequence number: 0 (relative sequence number)
    Acknowledgment number: 0
    Header Length: 32 bytes
    .... 0000 0000 0010 = Flags: 0x002 (SYN)
    window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0x9cc0 [validation disabled]
    Urgent pointer: 0
    Options: (12 bytes), Maximum segment size, No-Operation (NOP), window scale, No-Operation (NOP), No-Operation (NOP), S
0000 00 0c 29 a0 22 31 00 0c 29 77 5c bc 08 00 45 00 ..). "1.. )w\...E.
0010 00 34 3a 79 40 00 80 06 9f bc c0 a8 4f 9f c0 a8 .4:y@... ..O...
0020 4f 9e c2 3f 13 88 d6 5e e5 9a 00 00 00 00 80 02 O..?..^ .....
0030 20 00 9c c0 00 00 02 04 05 b4 01 03 03 08 01 01 .....
0040 04 02 ..
```

C'est à nouveau le protocole TCP qui est utilisé pour la couche de transport. Ce protocole est aussi un SYN.

2)

```
Transmission Control Protocol, Src Port: 49727 (49727), Dst Port: 5000 (5000), Seq: 0, Len: 0
```

Port source : 49727

Port destination : 5000

3)

### Client => Serveur

Display						
Display filter:		ip.dst == 192.168.79.158 && ip.src == 192.168.79.159				
Ignored packets:		0 (0,000%)				
Traffic	Captured	Displayed	Displayed %	Marked	Marked %	
Packets	747	104	13,922%	0	0,000%	
Between first and last packet	12,527 sec	0,006 sec				
Avg. packets/sec	59,629	16812,134				
Avg. packet size	889 bytes	93 bytes				
Bytes	663805	9646	1,453%	0	0.000%	
Avg. bytes/sec	52987,847	1559325,383				
Avg. MBit/sec	0,424	12,475				

Il y a **104** paquets envoyés du client vers le serveur;

Il y a **9646** octets envoyés du client vers le serveur.

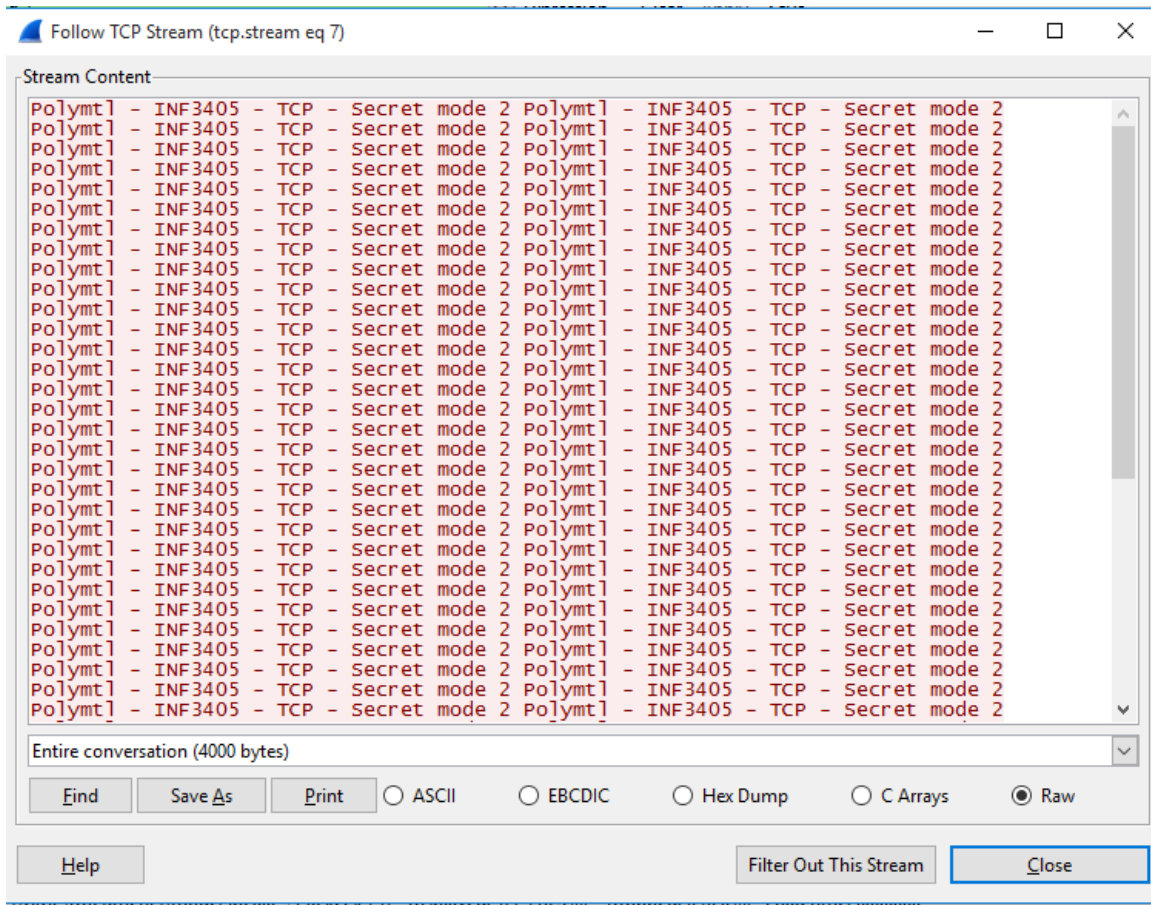
### Serveur => Client

Display						
Display filter:		ip.src == 192.168.79.158 && ip.dst == 192.168.79.159				
Ignored packets:		0 (0,000%)				
Traffic	Captured	Displayed	Displayed %	Marked	Marked %	
Packets	747	18	2,410%	0	0,000%	
Between first and last packet	12,527 sec	0,005 sec				
Avg. packets/sec	59,629	3308,246				
Avg. packet size	889 bytes	55 bytes				
Bytes	663805	984	0,148%	0	0.000%	
Avg. bytes/sec	52987,847	180850,757				
Avg. MBit/sec	0,424	1,447				

Il y a **18** paquets envoyés du serveur vers le client;

Il y a **984** octets envoyés du serveur vers le client.

4)



626	12.5213090	192.168.79.159	192.168.79.158	TCP	66 49727-5000 [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
627	12.5213750	192.168.79.158	192.168.79.159	TCP	66 5000-49727 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
628	12.5216300	192.168.79.159	192.168.79.158	TCP	60 49727-5000 [ACK] Seq=1 Ack=1 win=65536 Len=0
629	12.5230450	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
630	12.5230460	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
631	12.5230460	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
632	12.5230460	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
633	12.5230460	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
634	12.5230460	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
635	12.5230470	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
636	12.5230470	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
637	12.5230470	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
638	12.5230470	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
639	12.5230470	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
640	12.5230470	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
641	12.5230470	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
642	12.5230480	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
643	12.5230480	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]
644	12.5230480	192.168.79.159	192.168.79.158	IPA	94 unknown 0x6c [Malformed Packet]

Le client envoie au serveur un total de 4000 bytes, dont 40 octets par paquet, donc 100 itérations.

Cela signifie donc que le client envoie les données au serveur par de nombreux petits blocs, au lieu d'un seul grand bloc.

## Mode secret 3

1)

```
569 16.1806170 192.168.79.159 192.168.79.158 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=3ae1) [Reassembled in #577]
570 16.1806180 192.168.79.159 192.168.79.158 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=3ae1) [Reassembled in #577]
571 16.1806190 192.168.79.159 192.168.79.158 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=2960, ID=3ae1) [Reassembled in #577]
572 16.1806310 192.168.79.159 192.168.79.158 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=4440, ID=3ae1) [Reassembled in #577]
573 16.1807420 192.168.79.159 192.168.79.158 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=5920, ID=3ae1) [Reassembled in #577]
574 16.1807420 192.168.79.159 192.168.79.158 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=7400, ID=3ae1) [Reassembled in #577]
575 16.1807430 192.168.79.159 192.168.79.158 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=8880, ID=3ae1) [Reassembled in #577]
576 16.1807430 192.168.79.159 192.168.79.158 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=10360, ID=3ae1) [Reassembled in #577]
577 16.1807440 192.168.79.159 192.168.79.158 UDP 202 Source port: 56617 Destination port: 5010
578 16.1808380 192.168.79.158 192.168.79.159 ICMP 590 Destination unreachable (Port unreachable)
```

C'est le protocole UDP qui est utilisé pour la couche de transport. Il n'y a pas d'échange de synchronisation, parce qu'il n'y a pas de synchronisation en UDP.

2)

```
577 16.1807440 192.168.79.159 192.168.79.158 UDP 202 Source port: 56617 Destination port: 5010
578 16.1808380 192.168.79.158 192.168.79.159 ICMP 590 Destination unreachable (Port unreachable)
```

Port source : 56617

Port destination : 5010

3)

### Client => Serveur

Display						
Display filter:		ip.dst == 192.168.79.158 && ip.src == 192.168.79.159				
Ignored packets:		0 (0,000%)				
Traffic	Captured	Displayed	Displayed %	Marked	Marked %	
Packets	580	10	1,724%	0	0,000%	
Between first and last packet	17,796 sec	0,000 sec				
Avg. packets/sec	32,592	45245,998				
Avg. packet size	1114 bytes	1290 bytes				
Bytes	645891	12904	1,998%	0	0.000%	
Avg. bytes/sec	36294,712	58385435,616				
Avg. MBit/sec	0,290	467,083				

Il y a **10** paquets envoyés du client vers le serveur;

Il y a **12904** octets envoyés du client vers le serveur.

### Serveur => Client

Display						
Display filter:		ip.src == 192.168.79.158 && ip.dst == 192.168.79.159				
Ignored packets:		0 (0,000%)				
Traffic	Captured	Displayed	Displayed %	Marked	Marked %	
Packets	580	1	0,172%	0	0,000%	
Between first and last packet	17,796 sec					
Avg. packets/sec	32,592					
Avg. packet size	1114 bytes					
Bytes	645891	590	0,091%	0	0.000%	
Avg. bytes/sec	36294,712					
Avg. MBit/sec	0,290					

Il y a **1** paquets envoyés du serveur vers le client;

Il y a **590** octets envoyés du serveur vers le client.

4)

No.	Time	Source	Destination	Protocol	Length	Info
569	16.1806170	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=3ae1) [Reassembled in #577]
570	16.1806180	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=3ae1) [Reassembled in #577]
571	16.1806190	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=2960, ID=3ae1) [Reassembled in #577]
572	16.1806310	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=4440, ID=3ae1) [Reassembled in #577]
573	16.1807420	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=5920, ID=3ae1) [Reassembled in #577]
574	16.1807420	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=7400, ID=3ae1) [Reassembled in #577]
575	16.1807430	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=8880, ID=3ae1) [Reassembled in #577]
576	16.1807430	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=10360, ID=3ae1) [Reassembled in #577]

On observe 1 seule itération, divisée en 8 paquets envoyés simultanément.

Cela signifie donc que le client pour ce mode envoie toute l'information au serveur d'un bloc.



## Mode secret 4

1)

```
Frame 15: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
  Interface id: 0 (\Device\NPF_{8E06E00D-24EF-4081-9B0A-40E8AC9CC125})
  Encapsulation type: Ethernet (1)
  Arrival Time: Nov 27, 2018 15:16:30.662390000 Eastern Standard Time
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1543349790.662390000 seconds
  [Time delta from previous captured frame: 0.015437000 seconds]
  [Time delta from previous displayed frame: 0.000000000 seconds]
  [Time since reference or first frame: 12.802914000 seconds]
  Frame Number: 15
  Frame Length: 82 bytes (656 bits)
  Capture Length: 82 bytes (656 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:udp:data]
  [Coloring Rule Name: UDP]
  [Coloring Rule String: udp]
  Ethernet II, Src: Vmware_77:5c:bc (00:0c:29:77:5c:bc), Dst: Vmware_a0:22:31 (00:0c:29:a0:22:31)
  Internet Protocol Version 4, Src: 192.168.79.159 (192.168.79.159), Dst: 192.168.79.158 (192.168.79.158)
  User Datagram Protocol, Src Port: 49573 (49573), Dst Port: 5010 (5010)
    Source Port: 49573 (49573)
    Destination Port: 5010 (5010)
    Length: 48
    Checksum: 0x0855 [validation disabled]
    [Stream index: 2]
  Data (40 bytes)
```

C'est à nouveau le protocole UDP qui est utilisé pour la couche de transport. Encore une fois, il n'y a pas d'échange de synchronisation, parce qu'il n'y a pas de synchronisation en UDP.

2)

15	12.8029140	192.168.79.159	192.168.79.158	UDP	82	Source port: 49573	Destination port: 5010
----	------------	----------------	----------------	-----	----	--------------------	------------------------

Port source : 49573

Port destination : 5010

3)

**Client => Serveur**

Display						
Display filter:		ip.dst == 192.168.79.158 && ip.src == 192.168.79.159				
Ignored packets:		0 (0,000%)				
Traffic	◄	Captured ◄	Displayed ◄	Displayed % ◄	Marked ◄	Marked % ◄
Packets		893	301	33,707%	0	0,000%
Between first and last packet		13,099 sec	0,009 sec			
Avg. packets/sec		68,175	35340,971			
Avg. packet size		740 bytes	82 bytes			
Bytes		661084	24710	3,738%	0	0.000%
Avg. bytes/sec		50469,976	2901247,147			
Avg. MBit/sec		0,404	23,210			

Il y a **301** paquets envoyés du client vers le serveur;

Il y a **24710** octets envoyés du client vers le serveur.

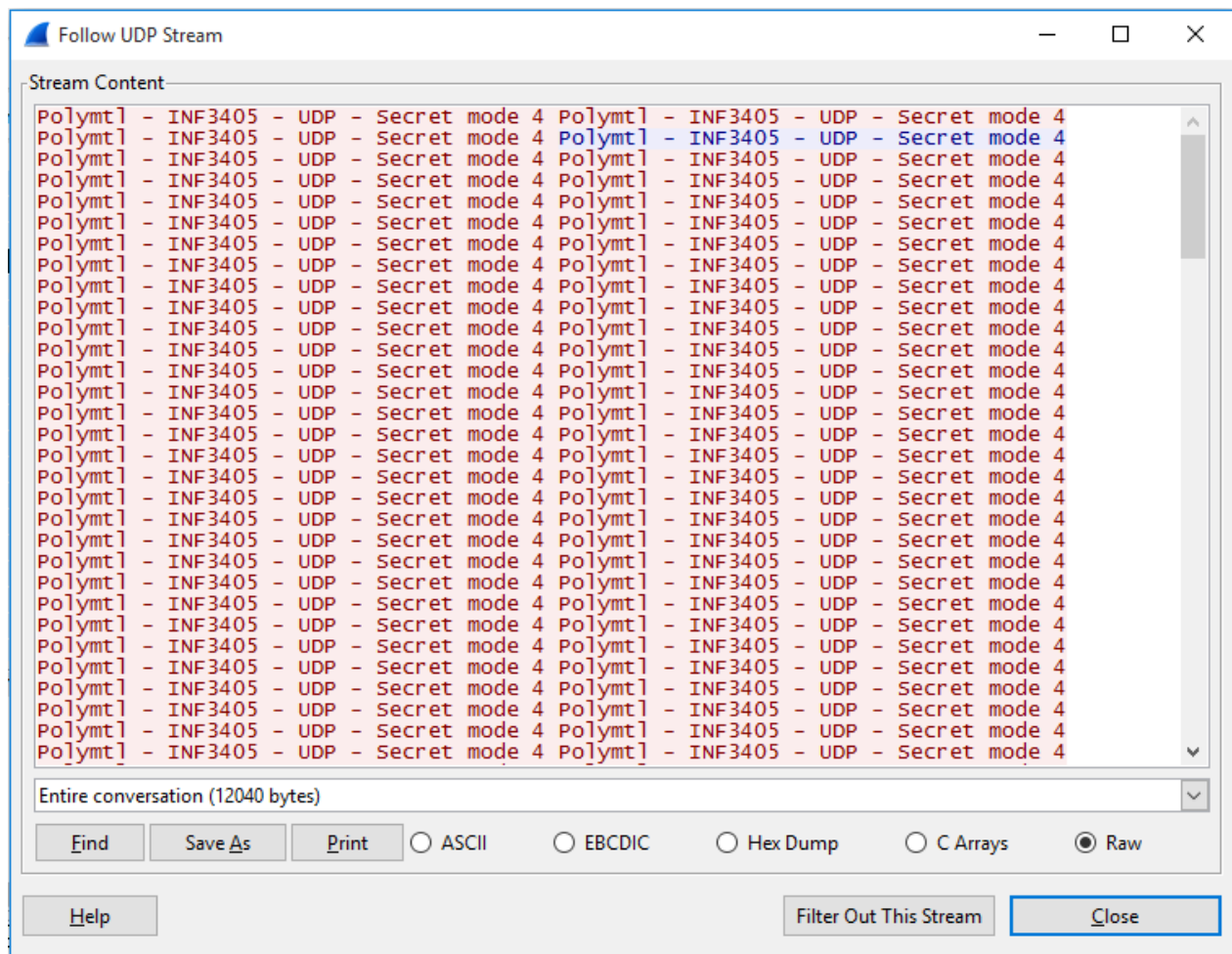
**Serveur => Client**

Display						
Display filter:		ip.src == 192.168.79.158 && ip.dst == 192.168.79.159				
Ignored packets:		0 (0,000%)				
Traffic	◄	Captured ◄	Displayed ◄	Displayed % ◄	Marked ◄	Marked % ◄
Packets		893	1	0,112%	0	0,000%
Between first and last packet		13,099 sec				
Avg. packets/sec		68,175				
Avg. packet size		740 bytes				
Bytes		661084	110	0,017%	0	0.000%
Avg. bytes/sec		50469,976				
Avg. MBit/sec		0,404				

Il y a **1** paquets envoyés du serveur vers le client;

Il y a **110** octets envoyés du serveur vers le client.

4)



```
Data (40 bytes)
Data: 506f6c796d746c202d20494e4633343035202d2055445020...
[Length: 40]
```

Le client envoie au serveur un total de 12040 bytes en 301 itérations, donc 40 octets par paquet.

Cela signifie donc que le client envoie les données au serveur par de nombreux petits blocs, au lieu d'un seul grand bloc.

## Analyse des performances et protocole TCP

1)

Le premier mode envoie toute l'information d'un bloc, mais comme le message envoyé est plus grand que la limite de taille d'un paquet, le protocole est forcé de fragmenter le bloc en trois divisions.

Le second mode n'a pas ce problème, car il envoie toute l'information en petits blocs séparés. Par contre, chacun de ces petits blocs nécessite son propre en-tête, ce qui en bout de ligne consomme plus de bande passante.

Selon nous, le mode le plus performant est le premier mode, car il consomme beaucoup moins de bande passante en envoyant qu'un seule en-tête pour toutes les données.

2)

Tout comme le premier mode, le troisième mode envoie toute l'information d'un bloc, mais comme le message envoyé est plus grand que la limite de taille d'un paquet, le protocole est forcé de fragmenter le bloc en huit divisions.

Le quatrième mode n'a pas ce problème, car il envoie toute l'information en petits blocs séparés. Par contre, chacun de ces petits blocs nécessite son propre en-tête, ce qui en bout de ligne consomme plus de bande passante, comme pour le deuxième mode.

Selon nous, le mode le plus performant est le troisième mode, car il consomme beaucoup moins de bande passante en envoyant qu'un seule en-tête pour toutes les données.

3)

Selon nous, les modes les plus performants sont le deuxième et le quatrième mode, car ils envoient les données en de nombreux petits paquets pré-séparés, ce qui fait que si l'un d'entre eux n'est pas reçu correctement, il est plus facile et moins coûteux de le renvoyer, comparativement à un gros message à renvoyer en entier.

4)

L'échange FIN, ACK est la confirmation de terminaison de connexion (ACK) et l'information de fin de connexion de la part du récepteur (FIN) afin de compléter le processus de fin de connexion entre le serveur et le client par un système de « handshaking ».