Jérémy Barrette – 1736976
Alexis Vailles – 1742139

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

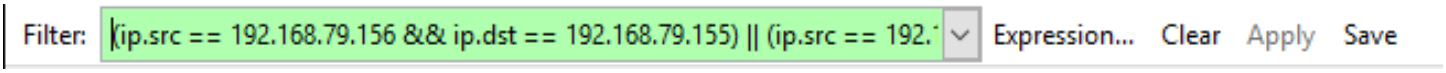
Soumis à : Kadi, Mehdi
INF3405 (01 – B1) – Réseaux informatiques
Session Automne 2018

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

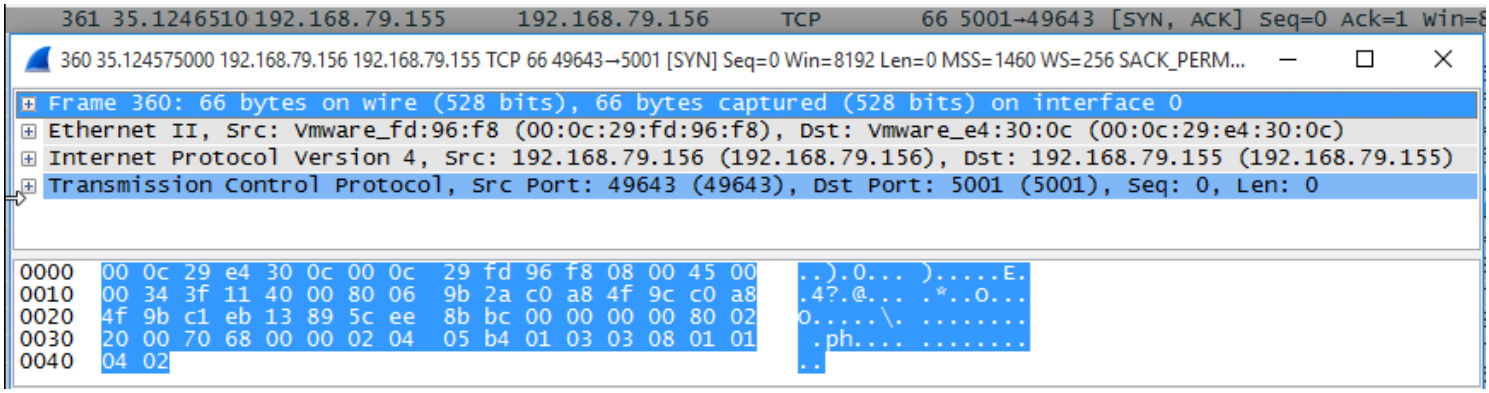


Il est à noter que le filtre utilisé sur la capture d'écran est :

(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

Display

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

Display

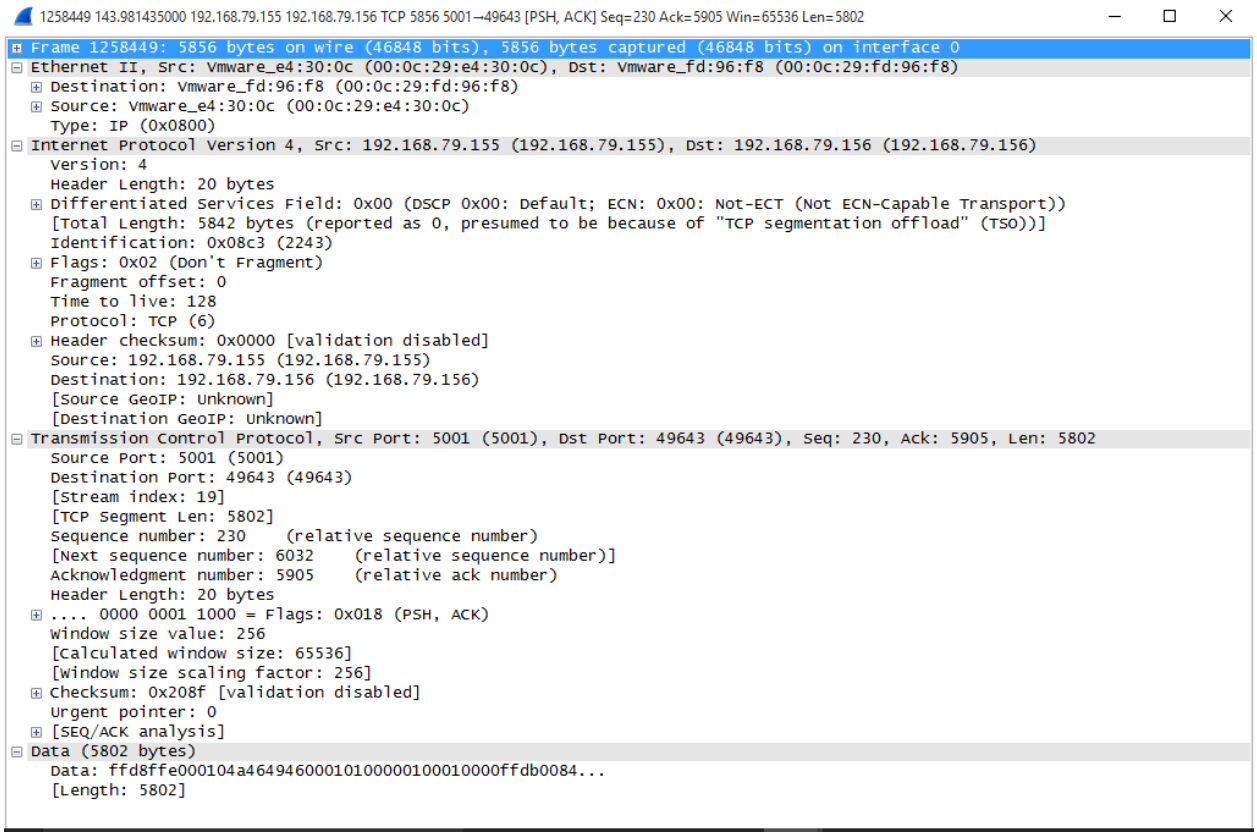
Display filter: (ip.src == 192.168.79.155 && ip.dst == 192.168.79.156)
Ignored packets: 0 (0,000%)

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;

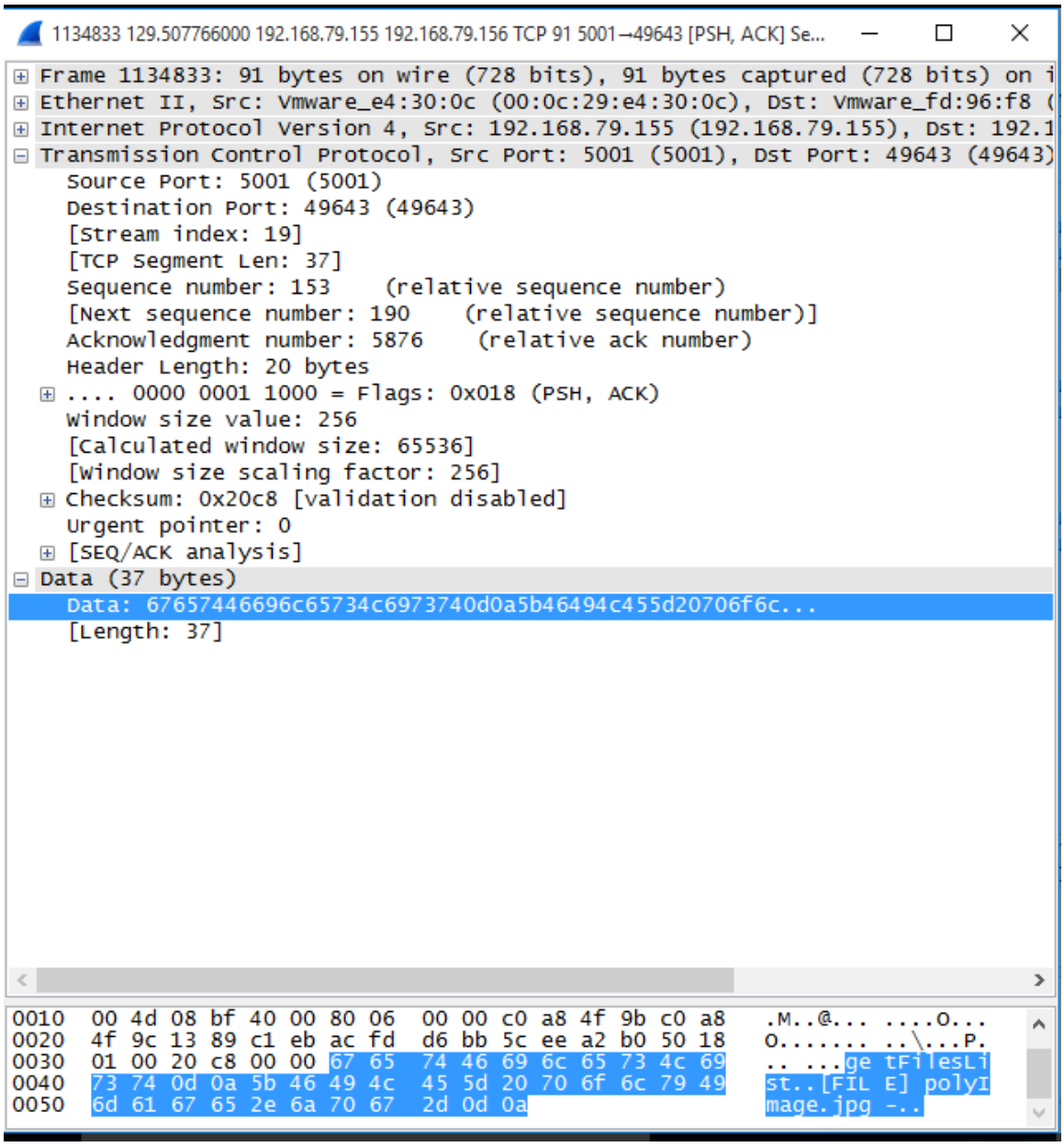
Il y a **6961** octets envoyés du serveur vers le client.

4)



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**.

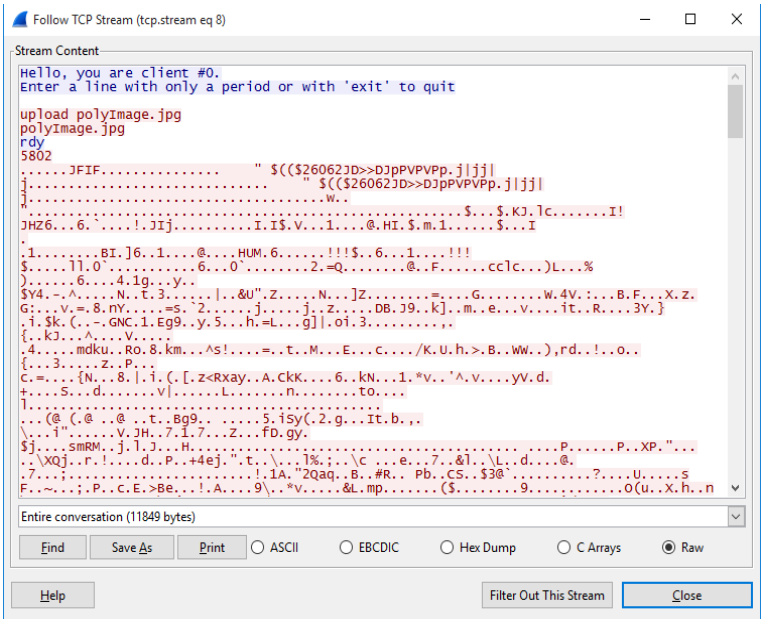
5)

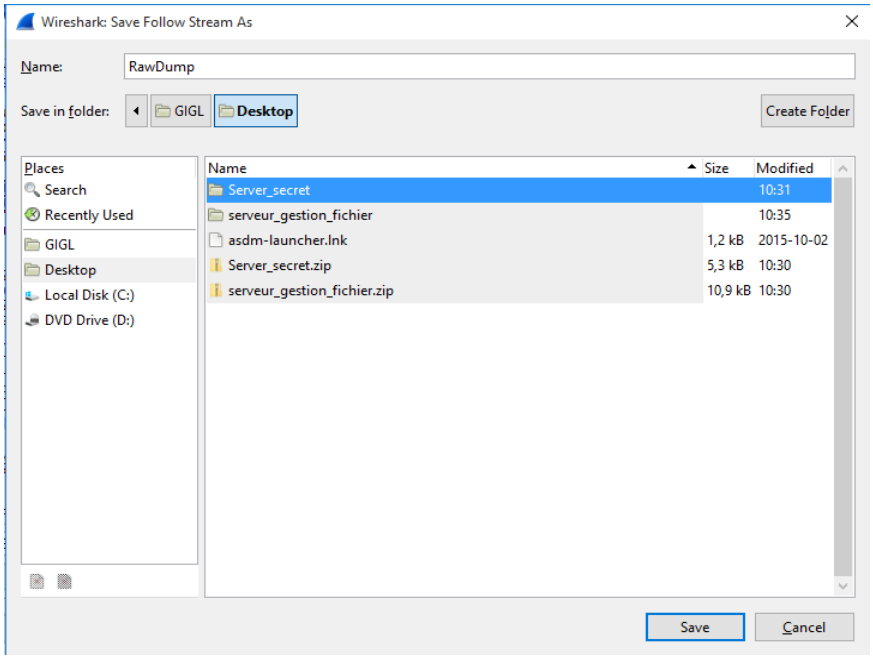


La commande « ls » nous permet d'intercepter la liste de fichiers et dossiers contenus dans le serveur.

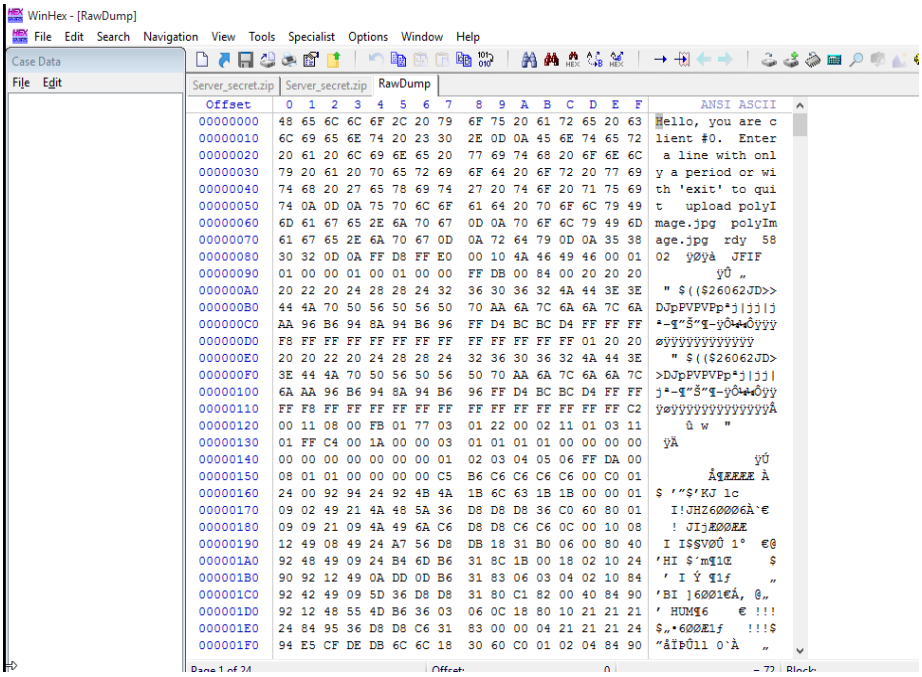
6) Comme nous ne sommes pas parvenus à effectuer cet exercice correctement durant les heures de laboratoire, nous avons emprunté les captures d'écran d'une autre équipe. Voici donc comment, avec Wireshark, on peut extraire l'image envoyée par le client ou l'image envoyer par le serveur vers le client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

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



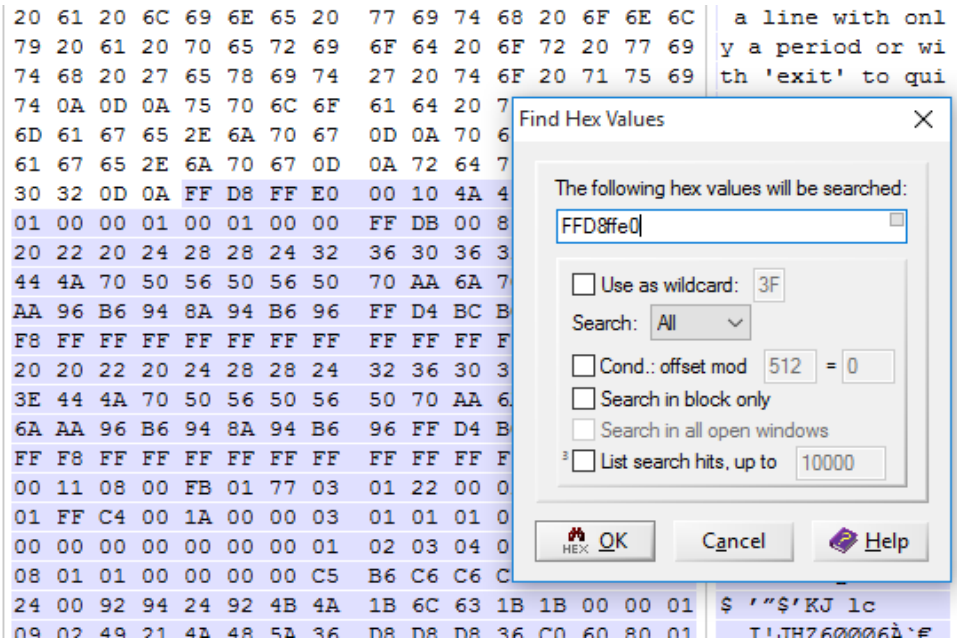


Étape 2 : On ouvre le *TCP stream* dans *WinHex* :

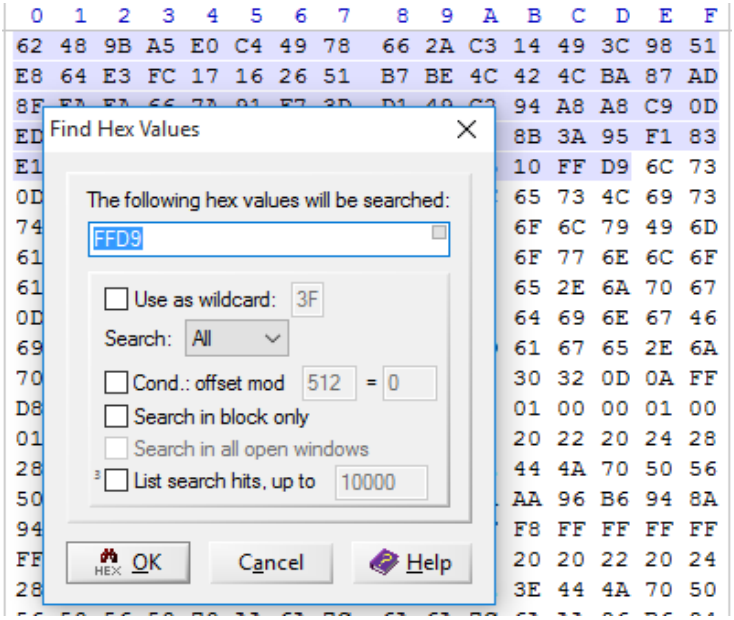


Étape 3 : Parmi les données représentées, on repère où commencent les données associées à l'image (après l'en-tête), et où elles se terminent :

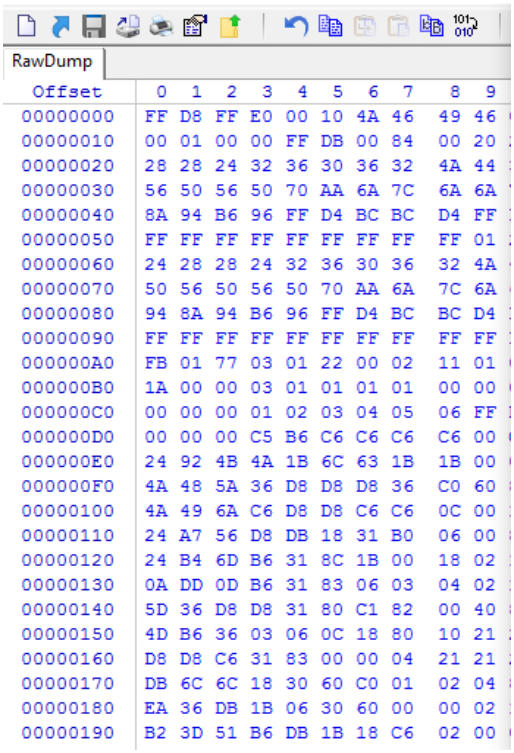
Début :



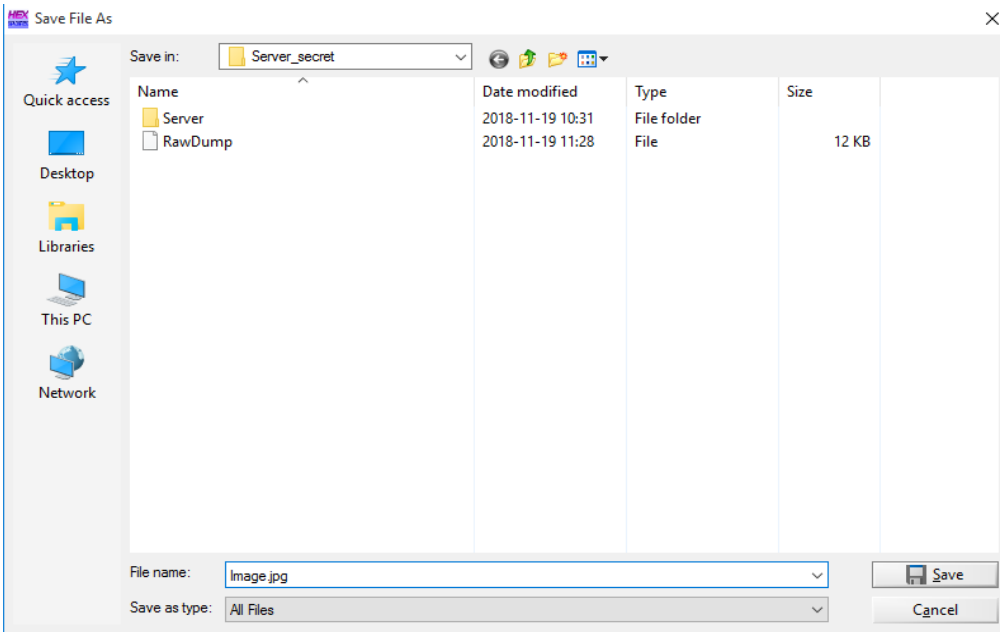
Fin :



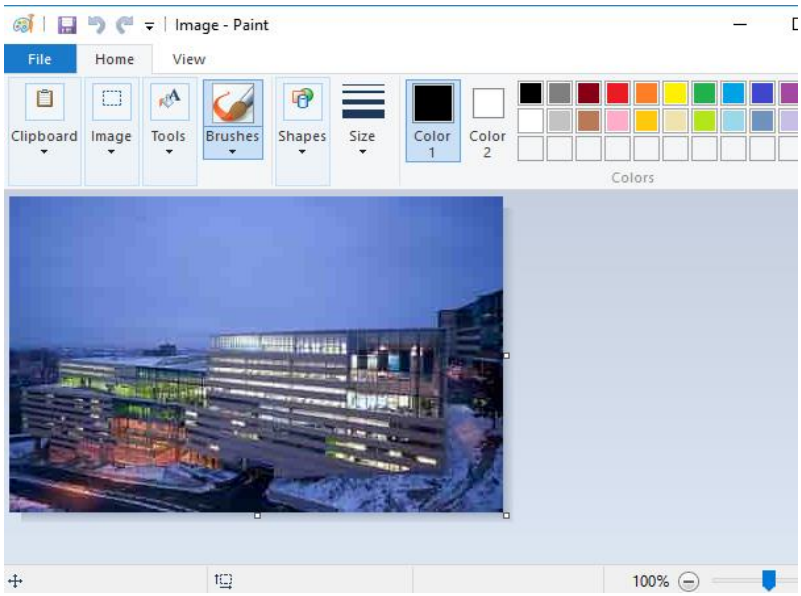
Étape 4 : On ne conserve que les données associées à l'image (donc on supprime le reste) :



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* :

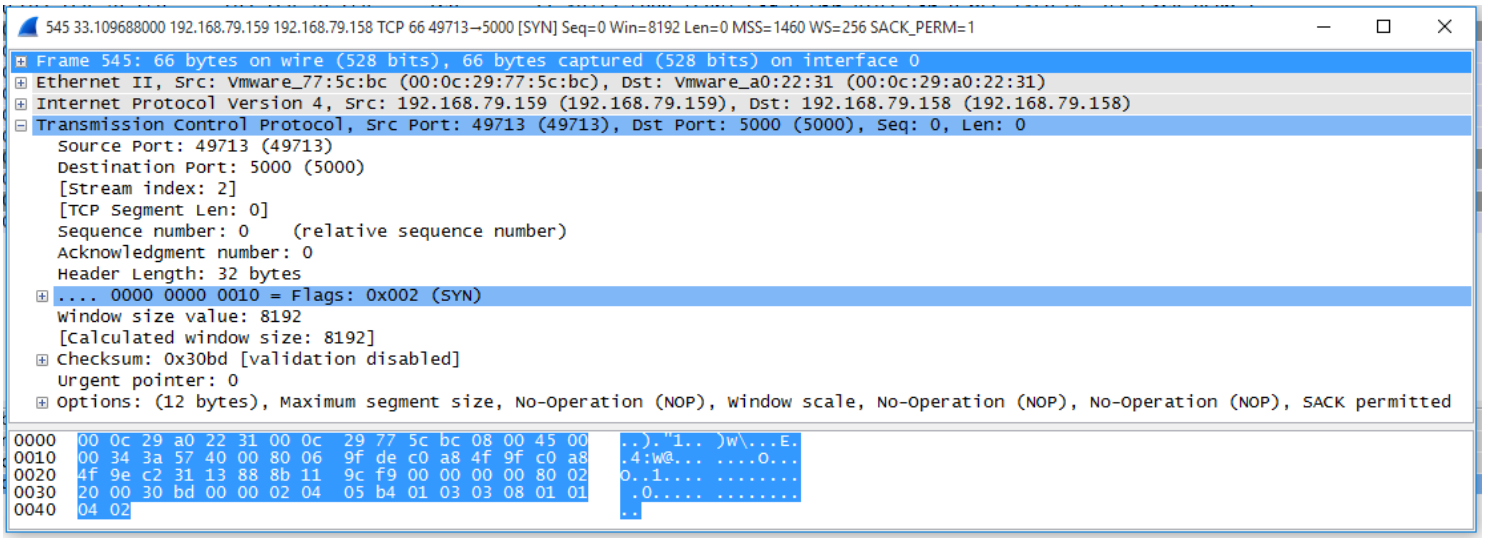


7) Suite à notre analyse, nous constatons qu'il est très facile d'intercepter et de récupérer les données et informations transmises entre le serveur et le client. La sécurité de la connexion est donc plutôt faible.

Analyse d'une application client-serveur "secrète"

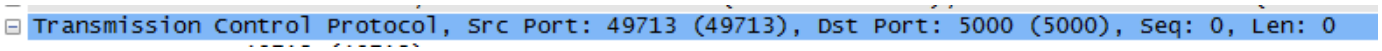
Mode secret 1

1)



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

2)



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)

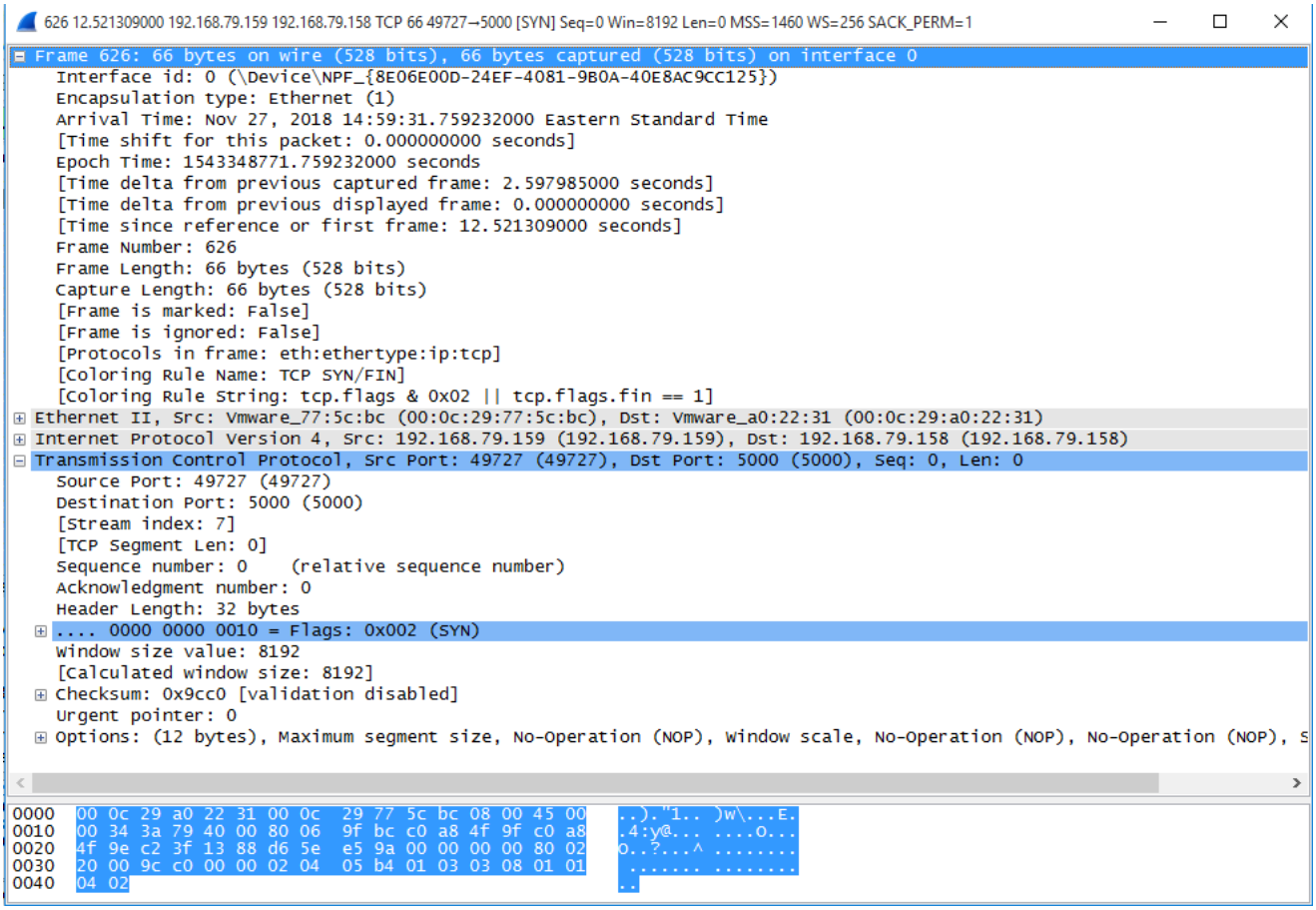
```
548 33.1125310 192.168.79.159    192.168.79.158    IPA      1514 unknown 0x6c [Malformed Packet]
549 33.1125320 192.168.79.159    192.168.79.158    IPA      1514 unknown 0x50 [Malformed Packet]
550 33.1125330 192.168.79.159    192.168.79.158    IPA      1134 unknown 0x6c [Malformed Packet]
```

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)



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

2)

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

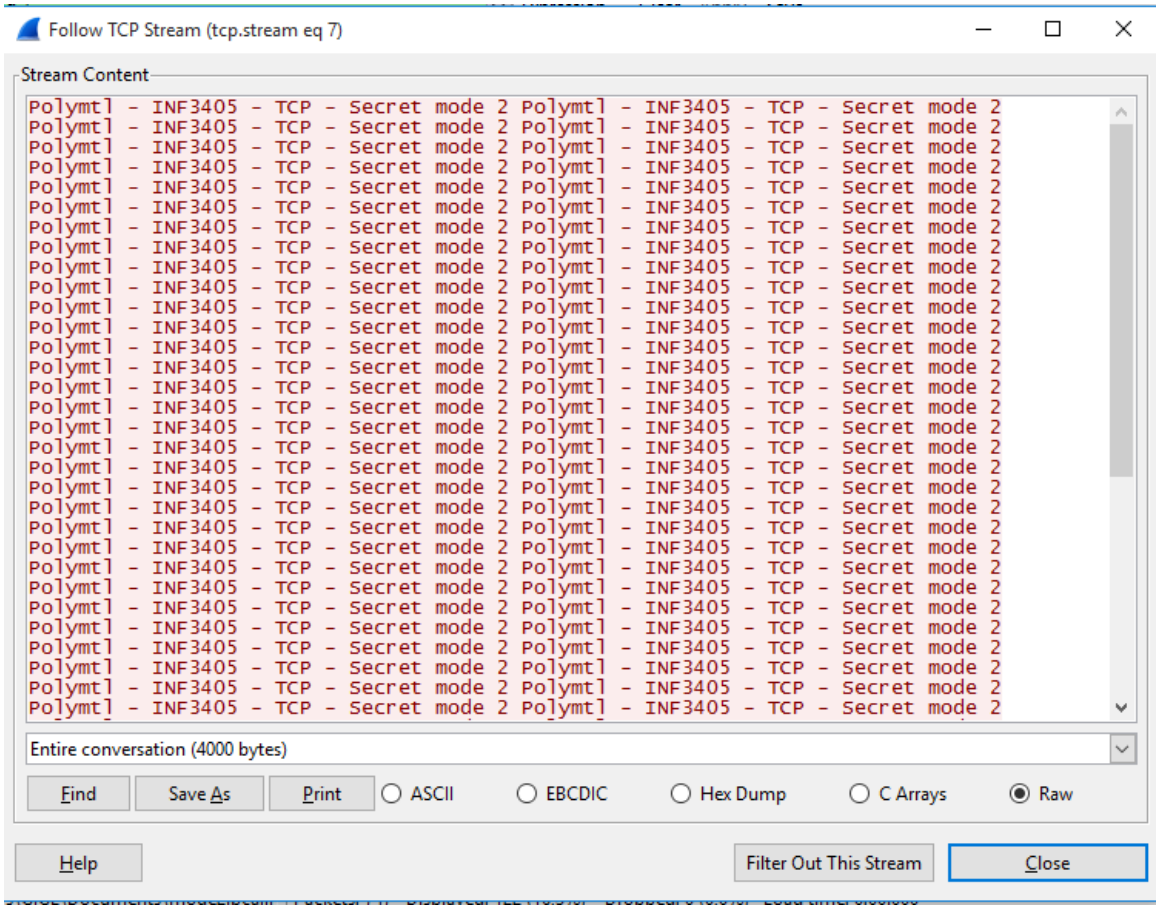
Ignored packets: 0 (0,000%)

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)



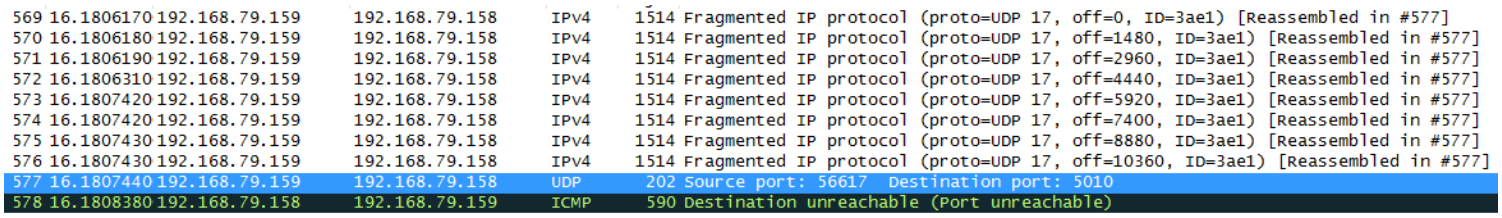
626	12.5213090	192.168.79.159	192.168.79.158	TCP	66 49727→5000 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
627	12.5213750	192.168.79.158	192.168.79.159	TCP	66 5000→49727 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
628	12.5216300	192.168.79.159	192.168.79.158	TCP	60 49727→5000 [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)



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)

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)

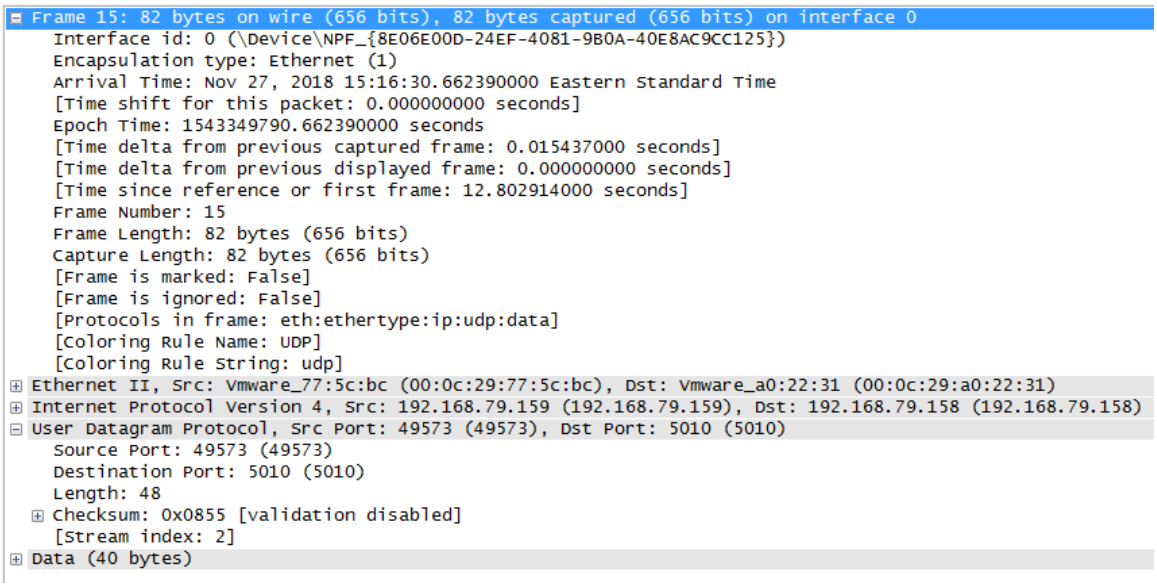
No.	Time	Source	Destination	Protocol	Length	Info
569	16.1806170	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=3ae1) [Reassembled in #577]
570	16.1806180	192.168.79.159	192.168.79.158	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=3ae1) [Reassembled 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.

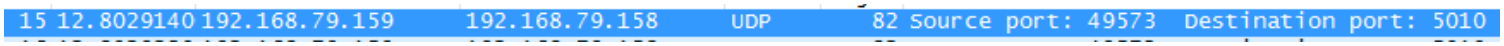
Mode secret 4

1)



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)



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

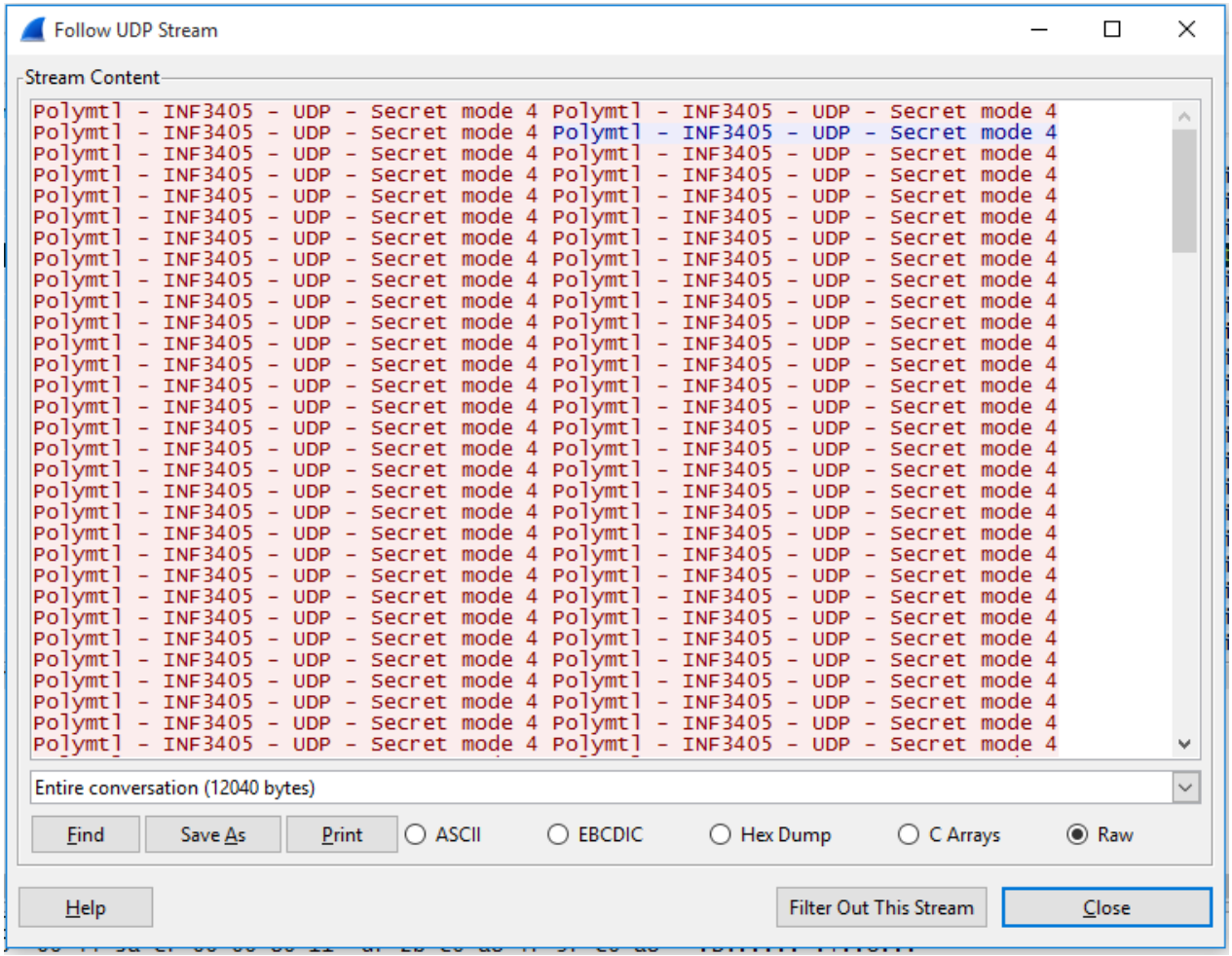
Display filter: ip.src == 192.168.79.158 && ip.dst == 192.168.79.159
Ignored packets: 0 (0,000%)

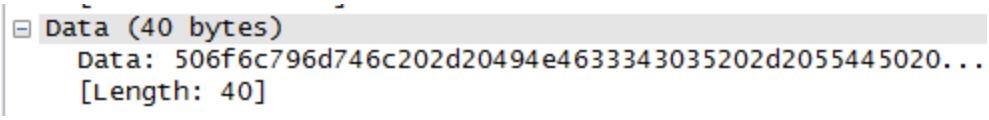
Traffic	Captured	Displayed	Displayed %	Marked	Marked %
Packets	893	1	0,112%	0	0,000%
Between first and last packet	13,099 sec				
Avg. packets/sec	68,175				
Avg. packet size	740 bytes				
Bytes	661084	110	0,017%	0	0.000%
Avg. bytes/sec	50469,976				
Avg. MBit/sec	0,404				

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

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

4)





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 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 ».