



SAÉ INTRO CYBER - GIRAULT Adrien et ARVIN-BÉROD Maxence

Bienvenue sur notre SAE INTRO CYBER qui sera sur la **Réalisation d'un CTF débutant orienté sur le réseau et sur l'utilisation de Python et Scapy**.

Qu'est-ce qu'un CTF ?

CTF signifie "Capture The Flag".

C'est une suite de challenges de cybersécurité à enchaîner pour récupérer des informations que l'on appelle le "drapeau".



DEF CON, la plus grande conférence de hackers

Le CTF à réaliser :

Salut jeune Padawan, les Sith préparent une attaque contre une planète localisée dans la bordure médiane. Heureusement le maître Jedi Guillemin a intercepté la communication chiffrée envoyée par Dark Plageuis à Dark Sidious qui indique la planète qui sera détruite et la date de l'attaque. Il est parti en mission sur Tatouine et te confie la mission de décoder ce message. Fais vite énormément de vies en dépendant ! La communication a été enregistrée dans un fichier nommé McDiarmid.pcapng

Pour ce CTF nous aurons besoins de :

- Wireshark
- FileZilla
- Python
- Scapy

Wireshark :

Wireshark est un **analyseur de protocoles réseau** libre et gratuit.

Il est utilisé dans le dépannage et l'analyse de réseaux informatiques, le développement de protocoles, l'éducation et la rétro-ingénierie



FileZilla :

FileZilla est un client **FTP, FTPS et SFTP**, développé sous la licence publique générale GNU.

Il est intégré à la liste des logiciels libres préconisés par l'État français dans le cadre de la modernisation globale de ses **systèmes d'informations** (S.I.).

Client : Mode de transaction entre plusieurs programmes et processus. L'un envoie des requêtes (le client), l'autre attend les requêtes des clients et y répond (le serveur).

FTP : "File Transfert Protocole" est un protocole de communication destiné au partage de fichiers sur un réseau TCP/IP.

FTPS : "File Transfert Protocole Secure" est un protocole de communication destiné à l'échange informatique de fichiers sur un réseau TCP/IP, variante du FTP, sécurisé avec les protocoles SSL ou TLS.



Python :

Python est un langage de programmation.

Il favorise la programmation structurée, fonctionnelle et orientée **objet** (la programmation orientée objet permet aux objets de communiquer entre eux).

- Il est doté d'un typage **dynamique fort** (il garantit que les types de données employés décrivent correctement les données manipulées)
- D'une **gestion automatique** de la mémoire par **ramasse-miettes** (sous-système informatique de gestion automatique de la mémoire)
- D'un système de gestion d'exceptions (permet de gérer les conditions exceptionnelles pendant l'exécution du programme).



Scapy :

Scapy est un logiciel **libre** de manipulation de **paquets réseau** écrit en **python**.

Il est capable d'**intercepter** le trafic sur un segment réseau, de **générer** des paquets dans un nombre important de protocoles, de réaliser une **prise d'empreinte** de la pile TCP/IP, de faire un **traceroute**, d'**analyser** le réseau informatique...



les

wave icon

OBJECTIFS

Chapitre 1 : Analyse du protocole de transfert de fichier

- Découverte de l'interception du transfert de fichier entre deux Siths sur Wireshark
- Récupération de login et password d'un Sith
- Découverte de FileZilla

Chapitre 2 : Installation de FileZilla

- Configuration d'un serveur FileZilla
- Création d'un utilisateur
- Transfert de fichier entre un PC serveur et PC client

Chapitre 3 : Récupération automatique du login et du mot de passe

- Création d'un script permettant de récupérer en **dur** un login et mot de passe

- Création d'un script permettant de récupérer **automatiquement** un login et un mot de passe

Chapitre 4 : Récupération du fichier transmis à partir du numéro de port TCP et décodage du message chiffrée

- Création d'un programme qui récupère toutes les données transférées dans la capture Wireshark et les stocker dans un fichier nommé SW2.pdf
- Création d'un programme qui permet de déchiffrer le message du Sith.

Chapitre 5 : Récupération automatique du numéro de port TCP négocié pour le transfert et du fichier

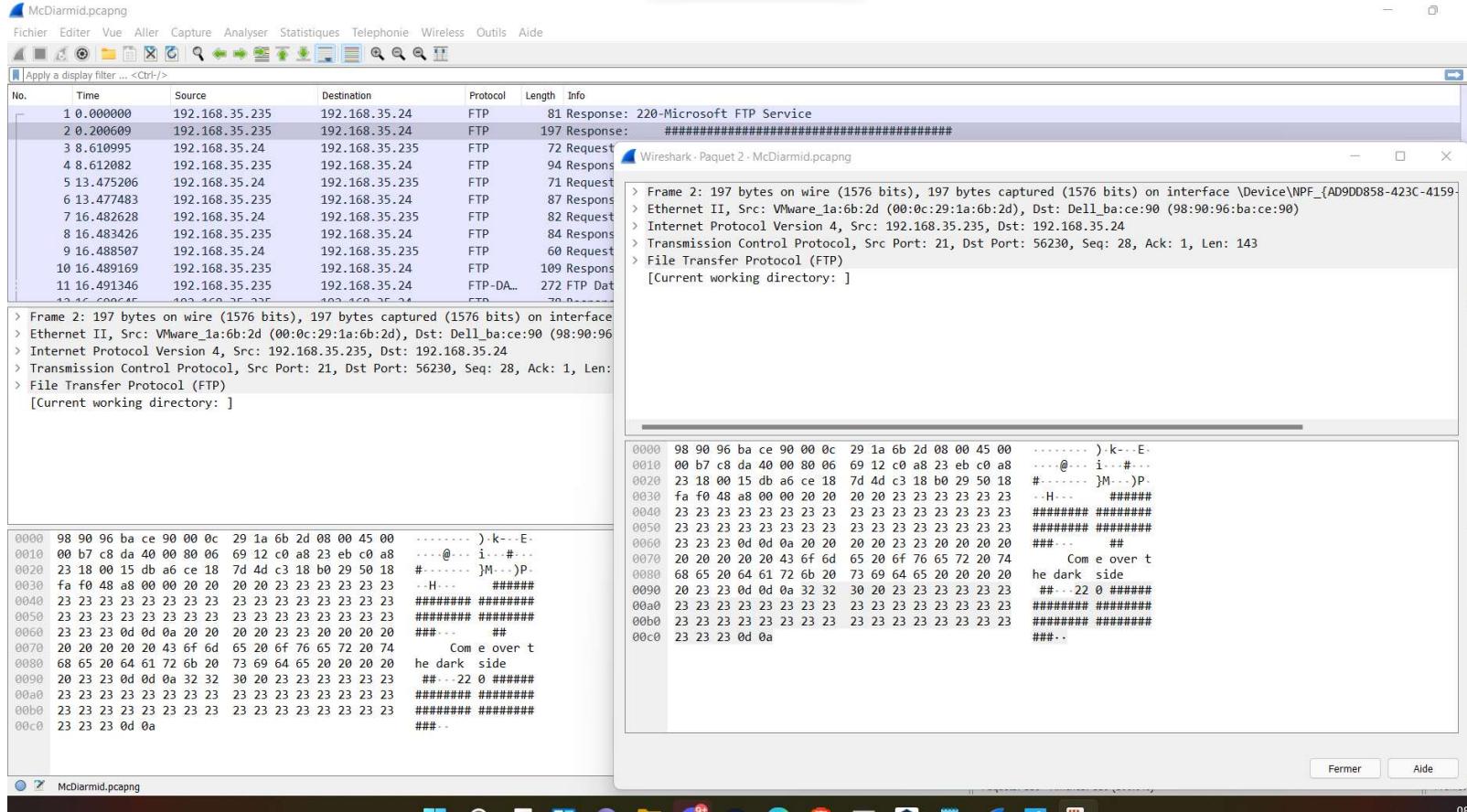
- Création d'un script pour récupérer le numéro de port négocié pour le transfert de données ainsi que le nom du fichier à transférer codé en **dur**.
- Récupérer automatiquement le numéro de port négocié pour le transfert, le nom du fichier et le fichier.

Chapitre 6 : Attaque MITM

- a. Sniff les paquets FTP
- b. Récupère le login et mot de passe de l'utilisateur
- c. Identifie le numéro de port négocié pour un transfert de fichier ainsi que le nom du fichier et la fin de transfert
- d. Enregistrer le fichier avec le nom de fichier utilisé pour le transfert
- Qu'est-ce qu'une attaque MITM ?

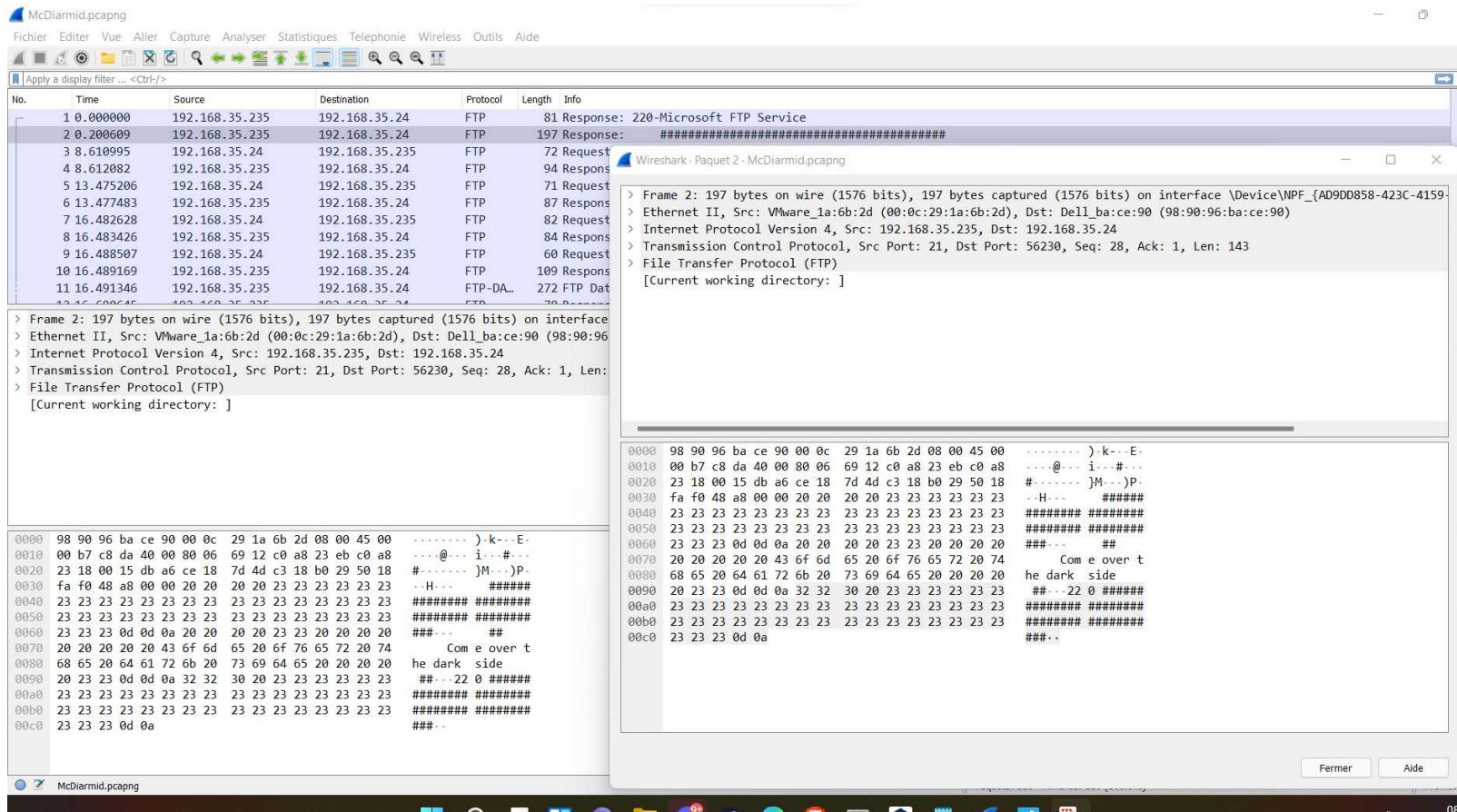
Chapitre 1 : Analyse du protocole de transfert de fichier

1) Ouverture du fichier McDiarmid.pcapng dans WireShark



Nous voyons qu'il s'agit d'un header qui envoie une réponse ou il y'a du texte.

2) Le protocole applicatif utilisée de cette communication

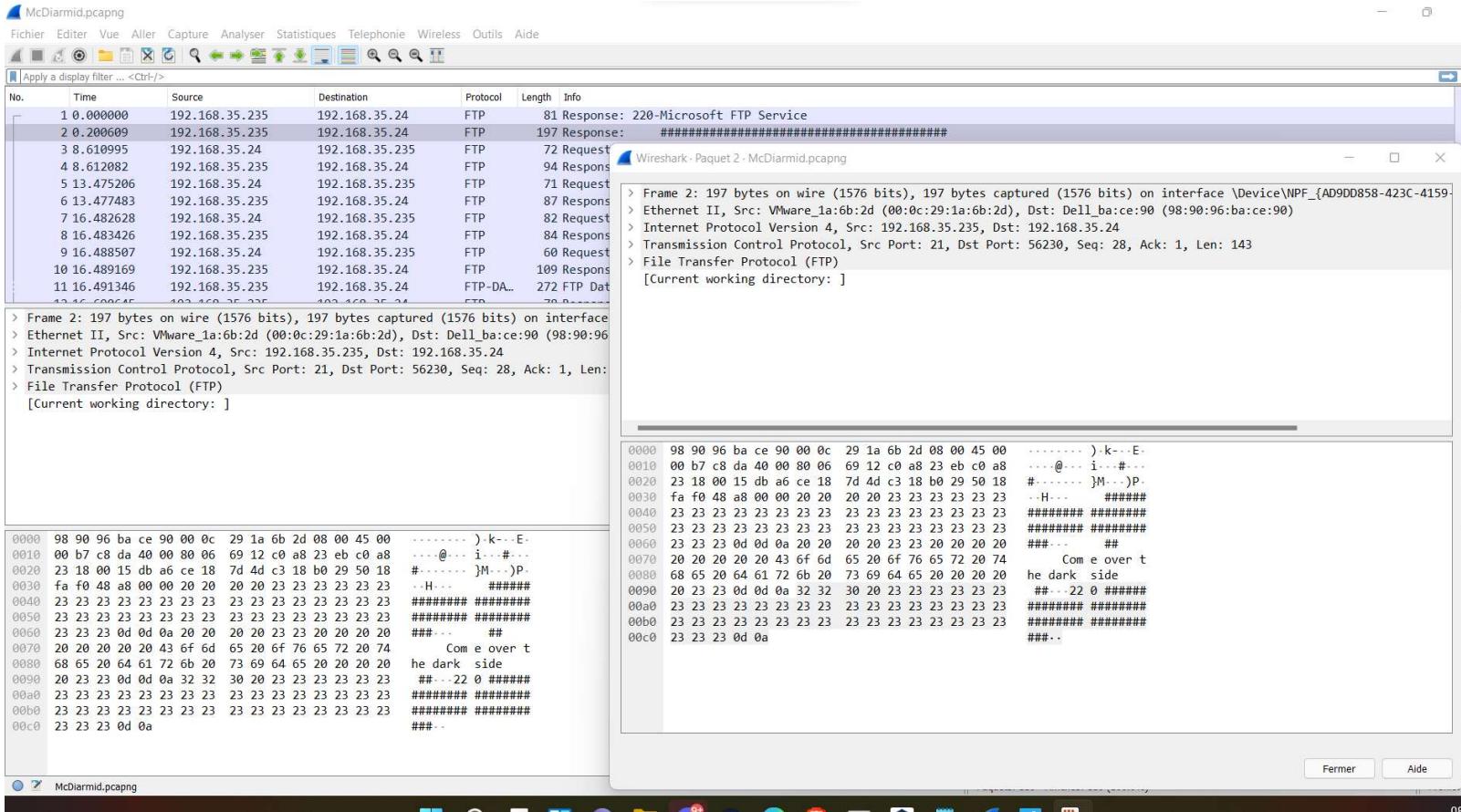


L'envoi de ce fichier utilise le protocole applicatif **FTP**

Il a comme port **source** : 21 et port **destination** : 56230

FTP pour *File Transfert Protocol* est un protocole de communication destiné au partage de fichiers sur un réseau TCP/IP.

3) Adresse IP client et serveur



- Adresse IP client : 192.168.35.24
- Adresse IP serveur : 192.168.35.235

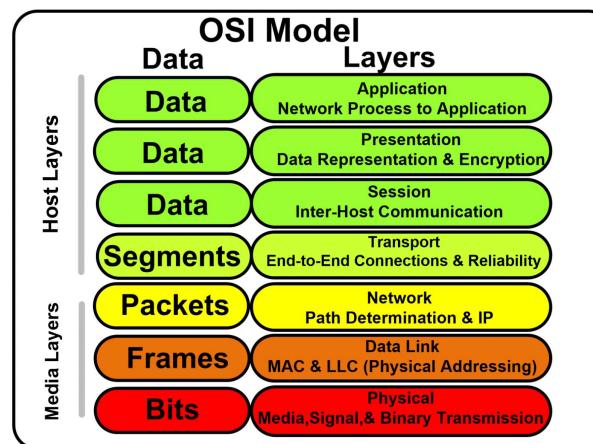
4) Protocole de transport utilisé

Il s'agit d'un protocole **TCP**, Wireshark permet de ce repérer sur le modèle **OSI**. Nous pouvons apercevoir de haut en bas la couche 1 (Physique) à la couche 5 (Sessions).

Protocole TCP :

"Transmission Control Protocol" est un protocole de contrôle de transmissions permettant d'établir une connexion et de transmettre des données.

- TCP permet de remettre en ordre les datagrammes en provenance du protocole IP
- TCP permet de vérifier le flot de données afin d'éviter une saturation du réseau
- TCP permet de formater les données en segments de longueur variable afin de les "remettre" au protocole IP
- TCP permet de faire circuler simultanément des informations provenant de sources (applications par exemple) distinctes sur une même ligne
- TCP permet enfin l'initialisation et la fin d'une communication de manière courtoise



Le numéro de ports **TCP** utilisé pour le protocole applicatif **FTP** par défaut est **21**

5) Message de bienvenue de ce site

Lorsque nous nous connectons, le serveur nous renvoie le message :

```
#####
```

```
## Come over the dark side ##
```

```
#####
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.35.235	192.168.35.24	FTP	81	Response: 220-Microsoft FTP Service
2	0.200699	192.168.35.235	192.168.35.24	FTP	197	Response: #####
3	8.610995	192.168.35.24	192.168.35.235	FTP	72	Request: USER Dark_Sidius
4	8.612082	192.168.35.235	192.168.35.24	FTP	94	Response: 331 Password required for Dark_Sidius.
5	13.475206	192.168.35.24	192.168.35.235	FTP	71	Request: PASS Dark_Force
6	13.477483	192.168.35.235	192.168.35.24	FTP	87	Response: 230 User Dark_Sidius logged in.
7	16.482628	192.168.35.24	192.168.35.235	FTP	82	Request: PORT 192,168,35,24,219,167
8	16.483426	192.168.35.235	192.168.35.24	FTP	84	Response: 200 PORT command successful.
9	16.488507	192.168.35.24	192.168.35.235	FTP	60	Request: NLST
10	16.489169	192.168.35.235	192.168.35.24	FTP	109	Response: 150 Opening ASCII mode data connection for file list.
11	16.491346	192.168.35.235	192.168.35.24	FTP-DA..	272	FTP Data: 218 bytes (PORT) (NLST)
12	16.49945	192.168.35.235	192.168.35.24	FTP	30	Response: 226 Transfer complete.

```

> Frame 2: 197 bytes on wire (1576 bits), 197 bytes captured (1576 bits) on interface \Device\NPF_{AD9DD858-423C-4159-81FD-26B98CEB77D6}, id 0
> Ethernet II, Src: VMware_1a:6b:2d (00:0c:29:1a:6b:2d), Dst: Dell_be:ce:90 (98:9b:96:ba:ce:90)
> Internet Protocol Version 4, Src: 192.168.35.235, Dst: 192.168.35.24
> Transmission Control Protocol, Src Port: 21, Dst Port: 56230, Seq: 28, Ack: 1, Len: 143
└> File Transfer Protocol (FTP)
    > #####\r\n
    \|n
    ##      Come over the dark side     ##\r\n
    \|n
    220 #####\r\n
[Current working directory: ]

```

6) Login et mot de passe

- Le **USER** est **Dark_Sidius**
- Le **PASS** est **Dark_Force**

3 8.610995	192.168.35.24	192.168.35.235	FTP	72 Request: USER Dark_Sidius
4 8.612082	192.168.35.235	192.168.35.24	FTP	94 Response: 331 Password required for Dark_Sidius.
5 13.475206	192.168.35.24	192.168.35.235	FTP	71 Request: PASS Dark_Force
6 13.477483	192.168.35.235	192.168.35.24	FTP	87 Response: 230 User Dark_Sidius logged in.

Le **USER** et le **PASS** sont transmis dans une *requête* provenant du client allant au *serveur*. Ensuite, le *serveur répond* qu'il faut entrer le **PASSWORD** et l'utilisateur transmet une *requête* avec le mot de passe écrit.

7) IP et PORTS

Le message utilisé pour préciser sur quelle adresse IP et quel numéro de port envoyer les données est : *Request: USER*

- IP source (client) : 192.168.35.24
- IP destination (serveur) : 192.168.35.235

On envoie les données sur l'adresse IP destination du serveur et le port destination du serveur : **21**

L'adresse IP et le numéro de de port sont indiqués sur la **couche 4** du modèle OSI pendant le **Transport**

Transmission Control Protocol, Src Port: 56230, Dst Port: 21, Seq: 1, Ack: 171, Len: 18

8) Protocole passif ou actif ?

FTP Actif :

En mode actif, c'est le client FTP qui détermine le port de connexion à utiliser pour permettre le transfert des données. Port : 20

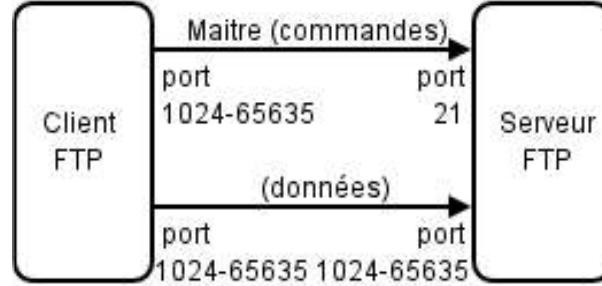
- Le port de commande du client contacte le port de commande du serveur et lui donne son port de données.
- Le serveur donne un accusé de réception au port de commande du client.
- Le serveur établit une connexion entre son port de données et le port de données du client.
- Enfin, le client envoie un accusé de réception au serveur.

FTP Passif :

En mode passif, le serveur FTP détermine lui-même le port de connexion à utiliser pour permettre le transfert des données (data connexion) et le communique au client. Port : 21

- Le client contacte le port de commande du serveur et émet une commande PASV pour indiquer qu'il s'agit d'une connexion passive.
- Ensuite, le serveur donne son port de données d'écoute au client.
- Ensuite, le client établit une connexion de données entre le serveur et lui-même à l'aide du port indiqué. (le port est donné par le serveur)
- Enfin, le serveur envoie un accusé de réception au client.

Le mode utilisé est le mode **FTP Passif** car le port est le 21 et le serveur FTP détermine lui-même le port de connexion à utiliser pour permettre le transfert des données.



9-10) Transfert de données et commandes

Dans la capture, le transfert de données est représenté par le protocole applicatif FTP-DATA .

Nous allons les filtrer pour en savoir le nombres :

No.	Time	Source	Destination	Protocol	Length	Info
11	16.491346	192.168.35.235	192.168.35.24	FTP-DA...	272	FTP Data: 218 bytes (P
17	34.786945	192.168.35.235	192.168.35.24	FTP-DA...	1514	FTP Data: 1460 bytes (
18	34.786947	192.168.35.235	192.168.35.24	FTP-DA...	1514	FTP Data: 1460 bytes (
19	34.788953	192.168.35.235	192.168.35.24	FTP-DA...	1514	FTP Data: 1460 bytes (
20	34.788958	192.168.35.235	192.168.35.24	FTP-DA...	1514	FTP Data: 1460 bytes (
21	34.788959	192.168.35.235	192.168.35.24	FTP-DA...	1514	FTP Data: 1460 bytes (
22	34.792595	192.168.35.235	192.168.35.24	FTP-DA...	1514	FTP Data: 1460 bytes (
23	34.792597	192.168.35.235	192.168.35.24	FTP-DA...	1514	FTP Data: 1460 bytes (

- Il y'a **90** transfert de données.

Elles sont initiées après ces trames :

- FTP Response: 150 Opening ASCII mode data connection for file list.

Et les données transférées sont des **octets** dont le fichier pdf se nomme *attaque_sith.pdf*

Ce fichier contient 128 718 **bytes** soit 128 718 **octets** soit 128,718 **kilooctets**

Il y'a : **87** paquets de 1460 bytes qui vont être transférés

Si les paquets envois que **maximum** 1460 bytes, c'est parce que la taille **maximale** MTU(Maximum Transmission Unit) de l'UDP que nous pouvons recevoir sans fragmentation est de **1460 bytes (octets)**.

11) Fin du téléchargement

Le message qui marque la fin du téléchargement est :

Response: 226 Transfer complete

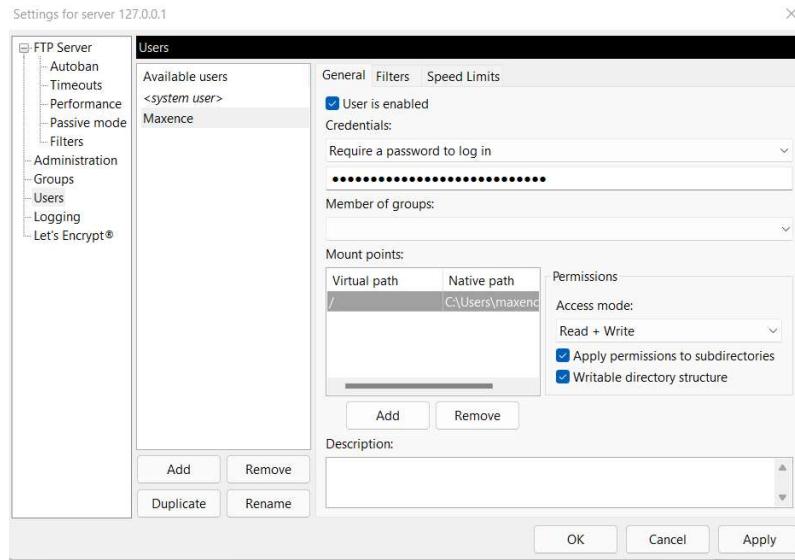


Chapitre 2 : Installation et configuration de FileZilla

1-2) Configuration de FileZilla

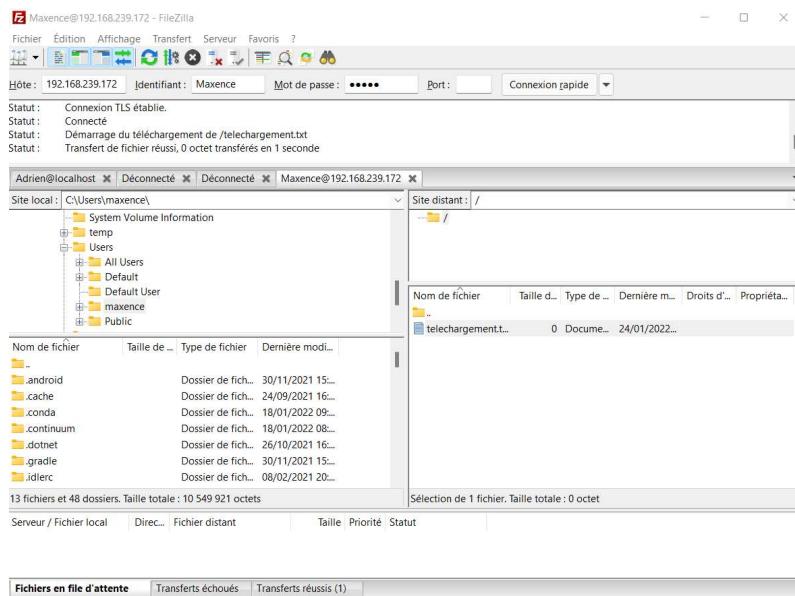
Nous nous connectons au serveur FileZilla et nous allons créer un utilisateur :

The screenshot shows two windows of the FileZilla Server Administration interface. On the left, the 'Users' settings window is open, showing a list of available users including '<system user>'. It includes fields for 'User is enabled', 'Credentials', 'Mount points', and 'Permissions'. On the right, the main administration window shows a log entry: 'Successfully connected to server 127.0.0.1'. Below the log is a table of session details, with one row visible: 'Connected to 127.0.0.1' with columns for Date, Session ID, Username, Host, Transfer st..., Path, Size, Inst. Spe..., Avg. Spe..., and Progress.



Notes : Nous avons du enlever les pare-feu côte à serveur pour que je puisse accéder au FTP de Adrien

Ensuite, je peux télécharger son fichier :



Voici les logs :

Administration interface - FileZilla Server 1.2.0

Date	Info	Type	Message
24/01/20...	FTP Sessi...	Comma...	TYPE I
24/01/20...	FTP Sessi...	Response	200 Type set to I
24/01/20...	FTP Sessi...	Comma...	EPSV
24/01/20...	FTP Sessi...	Response	229 Entering Extended Passive Mode (65184)
24/01/20...	FTP Sessi...	Comma...	MLSD
24/01/20...	FTP Sessi...	Response	150 About to start data transfer.
24/01/20...	FTP Sessi...	Response	226 Operation successful
24/01/20...	FTP Sessi...	Error	GnuTLS error -110 in gnutls_record_recv: The TLS connection was non-properly terminated.
24/01/20...	FTP Sessi...	Status	Client did not properly shut down TLS connection
24/01/20...	FTP Sessi...	Error	Control channel closed with error from source 0. Reason: ECONNABORTED - Connection aborted.
24/01/20...	FTP Server	Error	Session 1 ended with error from source 0. Reason: ECONNABORTED - Connection aborted.

Date	Session ID	Username	Host	Transfer st...	Path	Size	Inst. Spe...	Avg. Spe...	Progress
3 8.610995	192.168.35.24	192.168.35.235	FTP	72 Request: USER Dark_Sidius					

3 8.610995	192.168.35.24	192.168.35.235	FTP	72 Request: USER Dark_Sidius
4 8.612082	192.168.35.235	192.168.35.24	FTP	94 Response: 331 Password required for Dark_Sidius.
5 13.475206	192.168.35.24	192.168.35.235	FTP	71 Request: PASS Dark_Force
6 13.477483	192.168.35.235	192.168.35.24	FTP	87 Response: 230 User Dark_Sidius logged in.

Le user est **Dark_Sidius** et le password est **Dark_Force**

2) Création des commandes pour récupérer le login et mdp de l'utilisateur

```
Entrée [9]: from scapy.all import *

# rdpcap lit un fichier pcap ou pcapng (format Wireshark) et renvoie une liste de paquets
paquets=rdpcap("Wireshark/McDiarmid.pcapng")
print("#####")
print("Récupération de l'utilisateur et le mot de passe :")
print("#####\n")

# On récupère le 2ème paquets en données brutes indiqué par "RAW" (bytes notés b)
print("USER :", paquets[2][Raw].load.split(sep=None)) #affiche le contenu du paquet FTP ENCODÉ et séparer les chaines en
print("USER :", paquets[2][Raw].load.split(sep=None)[1].decode('UTF8'))#affiche le contenu du paquet FTP DÉCODÉ
# On récupère le 4ème paquets en données brutes indiqué par "RAW" (bytes notés b)
print("PASSWORD :", paquets[4][Raw].load.split(sep=None))#affiche le contenu du paquet FTP ENCODÉ et séparer les chaines
print("PASSWORD :", paquets[4][Raw].load.split(sep=None)[1].decode('UTF8'))#affiche le contenu du paquet FTP DÉCODÉ

#####
Récupération de l'utilisateur et le mot de passe :
#####

USER : [b'USER', b'Dark_Sidious']
USER : Dark_Sidious
PASSWORD : [b'PASS', b'Dark_Force']
PASSWORD : Dark_Force
```

Decode UTF8 nous permet de décoder notre format hexadécimal brute noté **b'**.

3) Écrire un script pour récupérer automatiquement le login et mot de passe de l'utilisateur

Entrée [69]: paquets=rdpcap("Wireshark/McDiarmid.pcapng")

```
for i in range(len(paquets)) : # parcours la capture wireshark :  
    if paquets[i][Raw].load.split(sep=None)[0].decode('latin1') == "USER" : # si dans la première valeur de la liste paquet[i]  
        user = paquets[i][Raw].load.split(sep=None)[1].decode('latin1') # prends la deuxième valeur de la liste paquet[i]  
    if paquets[i][Raw].load.split(sep=None)[0].decode('latin1') == "PASS" : #même chose pour le password#  
        password = paquets[i][Raw].load.split(sep=None)[1].decode('latin1') #même chose pour le password#  
print("le nom de l'utilisateur est : ",user)  
print("le mot de passe est : ",password)
```

```
le nom de l'utilisateur est : Dark_Sidius  
le mot de passe est : Dark_Force
```

Chapitre 4 : Récupération du fichier transmis à partir du numéro de port TCP et décodage du message chiffrée

1) Identifier le paquet transportant les données du fichier transféré

Pour identifier le plus simplement le paquet transportant les données du fichier transféré dans Wireshark, il suffit de trouver le **port** source. En l'occurrence, le port n° 20 (port lorsqu'il y'a des transferts de fichier sur le FTP).

2) Créer un programme qui récupère toutes les données transférées dans la capture Wireshark et les stocker dans un fichier nommé SW2.pdf

Entrée [49]: `from scapy.all import *`

```
#ouverture/création de mon fichier pdf SW2
fichier = open('SW2.pdf', 'ab')# a : ajouter du contenu et b : pour le format bytes

# rdpcap lit un fichier pcap ou pcapng (format Wireshark)
paquets=rdpcap("Wireshark/McDiarmid.pcapng")
print("#####")
print("Récupération de toutes les données transférées en cours...")
print("#####\n")

# pour i dans compris dans Les paquets
for i in range(len(paquets)):
    if paquets[i][TCP].sport == 20: #si les paquets[i] de protocole TCP de port source 20
        fichier.write(bytes(paquets[i][Raw]))#alors ça écrit dans 'fichier' en bytes, Les paquets du port source 20
fichier.close() #ferme 'fichier'

print("\n#####")
print("Récupération terminé !")
print("#####")
```

```
#####
Récupération de toutes les données transférées en cours...
#####
```

```
#####
Récupération terminé !
#####
```

3) Ouverture du fichier :

*La paix est un mensonge, il n'y a que la passion.
Par la passion, j'ai la puissance.
Par la puissance, j'ai le pouvoir.
Par le pouvoir, j'ai la victoire.
Par la victoire, je brise mes chaînes.
La Force me libérera.*



RWIG OTRTULO

HULO RDMUHO DEIWODI VDO YDRT DK KULKDO VDO NDIOUHHDO JLT VDO
OLNNUIKDHK. TV DOK KDBNO RD XIWNNDI LH CIWHR EULN DK XWTID
EUBNIDHRID W HUO DHHDBTO JL'TVO RUTMDHK OD OULBDKKID W HUKID
NLTOOWHED.

IDYUTHO HUKID XVUKKD RWHO VD OZOKDBD RD VW SUIRLID BDRTWHD
RWHO KIUTO YULIO DK BDHD LHD WKKJLD EUHKID VW NVWHDKD WHRU.
TV HD RUTK NVLO IDOKDI WLELH OLIMTMWHK OLI EDKKD NVWHDKD DK
VW IWED WJLWVTOF RUTK RTONWIWTKID.

RD NVLO VD YDRT RUHK YD KD YUTHO VW NFUKU DOK LH YDRT I&K. TV W
RDO JLWVTKDO DAEDNK TUHHDVVDO NULI VTHKDIEDNK TUH RDO
EUBBLHTEWKTUHO. KL RUTO VD KLDI.

OUTO OWHO NTKTD.

RWIG NVWCDLTO

Regarder ! Nous avons bel et bien intercepté une communication entre deux siths ! Nous pouvons apercevoir un message codé que nous allons devoir déchiffrer !

4) Le Jedi menacé est :



Le Maître Jedi Guillemin est menacé ! Il faut le sauver avant qu'il ne soit trop tard !

5) Décodons le message des Siths

Le chiffrement utilisée est un chiffrement par substitution (une lettre du message est remplacée par une autre dans le message chiffré)

Heureusement, une personne infiltrée nous a fourni au prix de sa vie, la correspondance de la substitution

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
W	S	E	R	D	X	C	F	T	Y	G	V	B	H	U	N	J	I	O	K	L	M	P	A	Z	Q

Entrée [2]: `from scapy.all import *`

```
#ouverture/création de mon fichier pdf SW2
fichier = open('SW2.pdf', 'r')# r : lire le fichier

texte = input(str("Entrez la chaîne à traduire :"))

decrypt={'W' : 'A',
          'S' : 'B',
          'E' : 'C',
          'R' : 'D',
          'D' : 'E',
          'X' : 'F',
          'C' : 'G',
          'F' : 'H',
          'T' : 'I',
          'Y' : 'J',
          'G' : 'K',
          'V' : 'L',
          'B' : 'M',
          'H' : 'N',
          'U' : 'O',
          'N' : 'P',
          'J' : 'Q',
          'I' : 'R',
          'O' : 'S',
          'K' : 'T',
          'L' : 'U',
          'M' : 'V',
          'P' : 'W',
          'A' : 'X',
          'Z' : 'Y',
          'Q' : 'Z',
          "''" : "''",
          "--" : "--",
          "&" : "&",
          ":" : ".",
          ":" : ":"}
```


`l= [] #liste vide`
`for i in texte : #parcours tout le texte`

```
    l.append(decrypt[str(i)]) #on ajoute dans notre liste, la clé de décryptage pour chaque lettre
l = "".join(l) #permet de réunir toutes les "valeur de la liste dans une valeur"
print("\n")
print(l) #affiche la liste

print("\n#####")
print("Récupération terminé !")
print("#####")
```

Entrez la chaîne à traduire :RWIG OTRTULO HULO RDMUHO DEIWODI VDO YDRT DK KULKDO VDO NDIOUHHDO JLT VDO OLNNUIKDHK. TV DOK KDBNO RD XIWNNDI LH CIWHR EULN DK XWTID EUBNIDHRID W HUO DHHDBTO JL'TVO RUTMDHK OD OULBDKKID W HUKID NLTOOWHED. I DYUTHO HUKID XVUKKD RWHO VD OZOKDBD RD VW SUIRLID BDRTWHD RWHO KIUTO YULIO DK BDHD LHD WKKWJLD EUHKID VW NVWHDKD WHRU. TV HD RUTK NVLO IDOKDI WLELH OLIMTMWHK OLI EDKKD NVWHDKD DK VW IWED WJLWVTOF RUTK RTONWIWTKID. RD NVLO VD YDRT RUHK YD KD YUTHO VW NFUKU DOK LH YDRT I&K. TV W RDO JLWVTKDO DAEDNK TUHHDV VDO NULI V'THKDIEDNKTUH RDO EUBBLHTEWKUHO. KL RUTO VD KLDI. OUTO OWHO NTKTD. RWIG NVWCDLTO

DARK SIDIOUS NOUS DEVONS ECRASER LES JEDI ET TOUTES LES PERSONNES QUI LES SUPPORTENT. IL EST TEMPS DE FRAPPER UN GRAND COUP ET FAIRE COMPRENDRE A NOS ENNEMIS QU'ILS DOIVENT SE SOUMETTRE A NOTRE PUISSANCE. REJOINS NOTRE FLOTTE DANS LE SYSTEME DE LA BORDURE MEDIANE DANS TROIS JOURS ET MENE UNE ATTAQUE CONTRE LA PLANETE ANDO. IL NE DOIT PLUS RESTER AUCUN SURVIVANT SUR CETTE PLANETE ET LA RACE AQUALISH DOIT DISPARAÎTRE. DE PLUS LE JEDI DONT JE TE JOINS LA PHOTO EST UN JEDI R&T. IL A DES QUALITES EXCEPTIONNELLES POUR L'INTERCEPTION DES COMMUNICATIONS. TU DOIS LE TUER. SOIS SANS PITIE. DARK PLAGEUIS

```
#####
Récupération terminé !
#####
```

Nous avons bien décrypté le message et le Maître Guillemin est en DANGER !

Nous lui avons envoyé un **hologram** sur son adresse willy.guillemin@iut-velizy.uvsq.fr à temps !



Maxence Gam <maxence.ab@gmail.com>

À willy.guillemain ▾

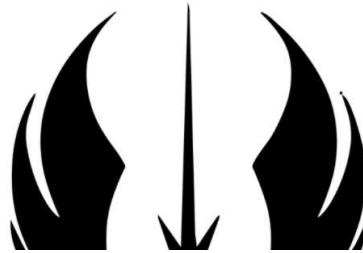
MAITRE GUILLEMIN,

Vous êtes en danger !!! Nous avons intercepté un message de l'Empire !
Regarder par vous même !

DARK SIDIOUS NOUS DEVONS ECRASER LES JEDI ET TOUTES LES PERSONNES QUI LES SUPPORTENT. IL EST TEMPS DE FRAPPER UN GRAND COUP ET FAIRE COMPRENDRE A NOS ENNEMIS QU'ils DOIVENT SE SOUMETTRE A NOTRE PUISSANCE. REJOINS NOTRE FLOTTE DANS LE SYSTEME DE LA BORDURE MEDIANE DANS TROIS JOURS ET MENER UNE ATTAQUE CONTRE LA PLANETE ANDO. IL NE DOIT PLUS RESTER AUCUN SURVIVANT SUR CETTE PLANETE ET LA RACE AQUALISH DOIT DISPARAÎTRE. DE PLUS LE JEDI DONT JE TE JOINS LA PHOTO EST UN JEDI R&T. IL A DES QUALITES EXCEPTIONNELLES POUR L'INTERCEPTION DES COMMUNICATIONS. TU DOIS LE TUER. SOIS SANS Pitié. DARK PLAGUEUIS

Récupération terminé !

De la part du Padawan Maxence et du Padawan Adrien



Chapitre 5 : Récupération automatique du numéro de port TCP négocié pour le transfert et du fichier

1) Crédit pas à pas des commandes pour récupérer le numéro de port négocié pour le transfert de données ainsi que le nom du fichier à transférer

Entrée [69]: `from scapy.all import *`

```
n = 0 #variable n initialisé à 0
# rdpcap lit un fichier pcap ou pcapng (format Wireshark)
paquets=rdpcap("Wireshark/McDiarmid.pcapng")
print("#####")
print("Récupération du numéro de port pour le transfert de données...")
print("#####\n")

#####
# RÉCUPÉRATION DU PORT NÉGOCIÉ
#####

for i in range(len(paquets)): # pour i dans compris dans les paquets
    if paquets[6][Raw].load.split(sep=None)[0].decode('latin1') == 'PORT' : #si les paquets[6] sont égaux à 'PORT'
        n = n+1 # ajouter 1 à n à chaque fois que nous passons dans la boucle
        port1 = paquets[6][Raw].load.decode('latin1') # port1 sera le paquet numéro 6 de ma capture en données brute décodé
        port11 = port1.split(",")[4] # Séparer la 6eme case par des ',' jusqu'au 4eme caractère pour obtenir le premier port
        port2 = paquets[6][Raw].load.decode('latin1')
        port22 = port1.split(",")[5] # Séparer la 6eme case par des ',' jusqu'au 5eme caractère pour obtenir le deuxième port
        portneg = int(port11)*256+int(port22) # Formule pour calculer le port négocié : Port1 * 256 + port2
        # Afficher le numéro de la trame de transfert ainsi que son port

    print(f"Le port 1 est {port11} et le deuxième est {port22}")
    print(f"Le port négocié est : {portneg}")

print("\n#####")
print("Récupération du nom du fichier...")
print("#####\n")

print("Le nom du fichier est :", paquets[14][Raw].load.split(sep=None)[1].decode('latin1'))

print("\n#####")
print("Récupération terminé !")
print("#####")
```

```
#####
Récupération du numéro de port pour le transfert de données...
```

```
#####
#####
```

Le port 1 est 219 et le deuxième est 167

Le port négocié est : 56231

```
#####
#####
```

Récupération du nom du fichier...

```
#####
#####
```

Le nom du fichier est : attaque_sith.pdf

```
#####
#####
```

Récupération terminé !

```
#####
#####
```

- Nous avons bien **récupérer** le numéro de port négocié pour le **transfert** de données (mais en dur).
Le port négocié est **56231**. Cela permet d'éviter les conflits de ports.
- Nous avons bien **récupérer** le nom du fichier mais nous l'avons fais en dur.
Nous voulons créer un programme pour récupérer automatiquement le nom du fichier et le fichier en plus !

Maître Jedi Guillemin a besoin de notre aide pour intercepter les futurs messages des Siths (s'ils sont assez bête pour garder leur méthode de cryptage) !

2) Récupérer automatiquement le numéro de port négocié pour le transfert, le nom du fichier et le fichier

- Nous avons bien ajouté la **récupération** du nom du fichier **automatiquement** et nous avons bien **demandé** si l'utilisateur voulait **récupérer le** fichier **en plus**.

Désormais, nous devons réaliser l'**automatisation** du port **négocié**.

Toutes les explications sont dans le code sous forme de **commentaires** :

```
Entrée [74]: from scapy.all import *
```

```

x = 0 # variable utilisé pour trouver le nom du fichier
n = 0 #variable n initialisé à 0
# rdpcap lit un fichier pcap ou pcapng (format Wireshark)
paquets=rdpcap("Wireshark/McDiarmid.pcapng")
print("#####")
print("Récupération du numéro de port pour le transfert de données...")
print("#####\n")
# Choix du port par l'utilisateur
print("#####")
paquet = input(str("Port numéro 20 ou 21 ? "))
print("#####\n")

#####
# RÉCUPÉRATION DU PORT NEGOCIÉ
#####
for x in range(len(paquets)): # pour i dans compris dans les paquets
    if paquets[x][Raw].load.split(sep=None)[0].decode('latin1') == 'PORT' : #si les paquets[x] au caractère 0 sont égaux
        n = n+1 # ajouter 1 à n à chaque fois que nous passons dans la boucle
        port1 = paquets[x][Raw].load.decode('latin1') # port1 sera le paquet numéro x de ma capture en données brute décodé
        port11 = port1.split(",")[4] # Séparer la 6eme case par des ',' jusqu'au 4eme caractère pour obtenir le premier
        port2 = paquets[x][Raw].load.decode('latin1')
        port22 = port1.split(",")[5] # Séparer la 6eme case par des ',' jusqu'au 5eme caractère pour obtenir le deuxième
        portneg = int(port11)*256+int(port22) # Formule pour calculer le port négocié : Port1 * 256 + port2
        # Afficher le numéro de la trame de transfert ainsi que son port

    print(f"Le port 1 est {port11} et le deuxième est {port22}")
    print(f"Le port négocié est : {portneg}")

#####
# RÉCUPÉRATION DU NOM DU FICHIER SI LE PORT EST LE NUMERO 20
#####

# Si le paquet est du port '20'
if paquet == '20':
    print("#####")
    print("Récupération du nom du fichier...")
    print("#####\n")

    for x in range(len(paquets)): # Pour x compris dans les paquets

```

```
#####
Récupération du numéro de port pour le transfert de données...
#####
```

```
Port numéro 20 ou 21 ? 20
#####
#
```

Le port 1 est 219 et le deuxième est 172

```
Le port négocié est : 56236
#####
Récupération du nom du fichier...
#####
```

Le nom du fichier est : attaque_sith.pdf

```
#####
Souhaitez-vous récupérer le fichier ? (o/n) o
```

Chapitre 6 : Attaque MITM (Man In The Middle)

Pour éviter que les Jedis me transmettent les fichiers qu'ils subtilisent à l'ennemi.

Il est souhaitable que nous puissions récupérer le login/mot de passe et les fichiers en temps réel lorsque nous interceptons les communications des Siths avec nos attaques MITM

Tout d'abord, qu'est-ce qu'une attaque **Man In The Middle** ?

Une attaque **Man In The Middle** est une cyberattaque dans laquelle l'attaquant relaie secrètement et éventuellement modifie les communications entre deux parties qui croient communiquer directement l'une avec l'autre, car l'attaquant s'est inséré entre les deux parties

1) Création d'un script qui :

- a. Sniff les paquets FTP
- b. Récupère le login et mot de passe de l'utilisateur
- c. Identifie le numéro de port négocié pour un transfert de fichier ainsi que le nom du fichier et la fin de transfert
- d. Enregistrer le fichier avec le nom de fichier utilisé pour le transfert

Qu'est-ce que le sniffing ?

Le sniffing est une technique qui consiste à se mettre en écoute sur le réseau afin de voir quels sont les échanges de données.

Pour cette partie, nous **voulons** sniffer les paquets FTP :

sniff permet de capturer le trafic réseau à partir d'une ou plusieurs interfaces. `sniff()` a 6 options :

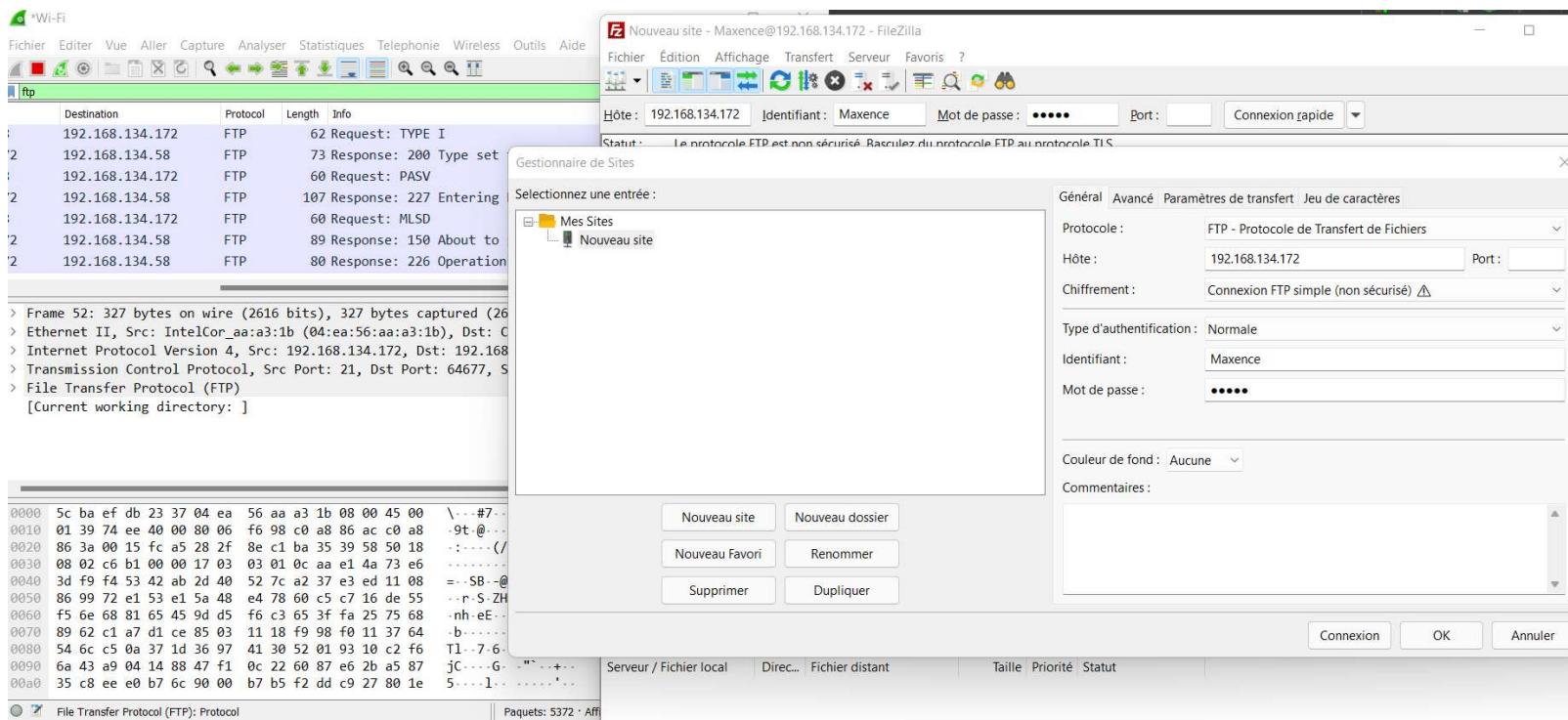
- `filter` : filtre les paquets à l'intérieur du noyau Linux ce qui rend le filtrage très rapide.
- `count` : nombre de paquet à capturer.
- `prn` : nom de la fonction à appliquer à chaque paquet reçu.
- `lfilter` : filtre les paquets à l'aide d'une fonction Python, on pourra utiliser une fonction lambda.
- `timeout` : durée de la capture, par défaut `None=∞ (^C pour arrêter)`
- `iface` : interface sur laquelle on souhaite capturer les paquets (défaut `:all`)
- `store` : s'il faut stocker les paquets capturés ou les supprimer (`store=0`).
- `stopfilter` : fonction à évaluer pour arrêter la capture (la fonction doit retourner `true` pour arrêter, `false` pour continuer)

Tout d'abord, nous avons remarqué que le **FTP** était crypté via le chiffrement **TLS**.

TLS signifie « Transport Layer Security », c'est-à-dire « sécurité de la couche de transport ».

Il permet de **chiffrer** les flux de données sur Internet afin que ces données ne puissent être lues **que par les destinataires autorisés**.

Du coup, nous nous sommes connectés sur le serveur via un mode dit *simple sans chiffrement* pour éviter d'avoir des paquets cryptés.



a) Interception d'une connexion FTP entre le serveur FTP et le client

```
Entrée [22]: from scapy.all import *

def loginpass (paquets):
    try:
        if paquets.haslayer(Raw): # si les paquets ont la couche Raw (données brutes) :
            if paquets[Raw].load.split(sep=None)[0].decode('latin1') == "USER": # si dans la première valeur de la liste
                user = paquets[Raw].load.split(sep=None)[1].decode('latin1') # prends la deuxième valeur de la liste pa
                print("le nom de l'utilisateur est : ", user)
            if paquets[Raw].load.split(sep=None)[0].decode('latin1') == "PASS": # même chose pour le password#
                password = paquets[Raw].load.split(sep=None)[1].decode('latin1') # même chose pour le password#
                print("le mot de passe est : ", password)
    except:
        pass
sniff(prn=loginpass,filter ="tcp port 20 or 21")
```

```
Out[22]: <Sniffed: TCP:0 UDP:0 ICMP:0 Other:0>
```

Voici un aperçu du résultat de notre programme dans *PyCharm*, cela ne fonctionne malheureusement pas dans le Notebook :

```
from scapy.all import *
paquets = sniff(prn=loginpass.filter = "tcp port 20 or 21")
def loginpass(paquets):
    try:
        if paquets.haslayer(Raw):
            if paquets[Raw].load.split(sep=None)[0].decode('latin1') == "USER": # si dans la première valeur de la liste paquet[i][Raw]
                user = paquets[Raw].load.split(sep=None)[1].decode('latin1') # prends la deuxième valeur de la liste paquet[i][Raw]#
                print("le nom de l'utilisateur est : ", user)
            if paquets[Raw].load.split(sep=None)[0].decode('latin1') == "PASS": # même chose pour le password#
                password = paquets[Raw].load.split(sep=None)[1].decode('latin1') # même chose pour le password#
                print("le mot de passe est : ", password)
    except:
        pass
```

Run: main x
C:\Users\adrie\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/adrie/PycharmProjects/pythonProject/main.py
le nom de l'utilisateur est : Maxence
le mot de passe est : admin

CONCLUSION

Nous n'avons malheureusement pas pu finir sachant que notre code paraît fonctionnel.

Nous sommes allés chercher de l'aide pour voir si nos camarades padawan avaient réussi et le code paraissait semblable au nôtre, ce qui nous frustrait énormément.

Nous avons préférés terminer le notebook, le paufiner pour qu'il soit agréable et intuitif à lire et bien expliqué ce que nous avons compris et fait durant cette SAE.

Nous avons fais le maximum pour détailler et expliquer les programmes, les applications, nos méthodes, nos chapitres, ...

Notre SAE se résume par le fait que :

- Nous avons bien découvert l'interception du transfert de fichier entre deux Siths sur WireShark
- Nous avons bien récupéré le login et password d'un Sith
- Nous avons bien installé et configuré FileZilla
- Nous avons bien crée un script permettant de récupérer en dur un login et mot de passe
- Nous avons bien crée un script permettant de récupérer automatiquement un login et un mot de passe
- Nous avons bien crée un programme qui récupère toutes les données transférées dans la capture Wireshark et les stockes dans un fichier nommé SW2.pdf
- Nous avons bien crée un programme qui permet de déchiffrer le message du Dark Sidious.
- Nous avons bien récupéré automatiquement le numéro de port négocié pour le transfert, le nom du fichier et le fichier.
- Nous avons bien récupéré le login et mot de passe de l'utilisateur automatiquement via une connexion FTP

Ce que nous n'avons pas terminés dans le dernier chapitre:

- a. Sniff les paquets FTP
- c. Identifie le numéro de port négocié pour un transfert de fichier ainsi que le nom du fichier et la fin de transfert
- d. Enregistrer le fichier avec le nom de fichier utilisé pour le transfer



PS : McDiarmid est un acteur mondialement célèbre pour avoir joué le rôle de l'empereur Sheev Palpatine / Dark Sidious dans la saga Star Wars. Il apparaît dans 6 des épisodes de la saga ainsi que dans la série d'animation Star Wars Rebels.