# SAE 2.04

## installation:

sudo pip install scapy
sudo pip install jupyterlab

## TP N°1:

Pour ouvrir en mode admin jupyter notebook entrer la commande suivante:
sudo jupyter-notebook --allow-root
⚠️ si ça marche pas : sudo -E env "PATH=$PATH" jupyter notebook --allow-root
Première étape exporter scapy dans votre programme qui se fera à chaque fois qu'on ré-ouvre le fichier:
from scapy.all import *
pour générer un paquet avec scapy:
paquet1=IP()/UDP()
ceci est un exemple type on peut faire aussi:
paquet1=Ether(src=x ou"x", dst=xou"x")/IP(src=xou"x", dst=x ou"x")/UDP()
x= une variable prédéfinis
"x"= @mac ou @IP
les trames dans scapy sont traiter sous forme de tableau voir même de dictionnaire donc une clé=une valeur:
exemple:
trames=rdpcap("chemin ou et stocker votre fichier enregistrer de la capture wireshark/nom du fichier.(pcapng ou pcap)")# pour lire le fichier
trames.summary() #pour voir la totalité des éléments comportant le fichier.
exemple:
Ether / IP / ICMP 192.168.1.48 > 8.8.8.8 echo-request 0 / Raw
Ether / IP / ICMP 8.8.8.8 > 192.168.1.48 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.1.48 > 8.8.8.8 echo-request 0 / Raw
Ether / IP / ICMP 8.8.8.8 > 192.168.1.48 echo-reply 0 / Raw
exemple:
trames[0]# pour afficher tout le contenue de la première trames
ce qui affichera:
trames[0][IP].dst #ce qui affichera l'adresse de destination de la trame
trames[0][Raw] #affichera le contenue du champ de donnée de notre première trame
trames[0][Raw].load # nous permettra de les convertir en binaire pour pouvoir les exploiter par la suite d'où le b' pour signaler qu'on est passer en binaire.
trames[0][Raw].load.split(sep=None) #ça split tous les mots.

avant:
b'a\xc0\xbd\xd2\x00\x05\xa25\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\
x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

après:
[ b'a\xc0\xbd\xd2\x00\x05\xa25\x08',
b'\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1',
b'!"#$%&\'()*+,-./01234567']
trames[0][Raw].load.split(sep=None)[0].decode('UTF-8')# decoder en UTF-8.

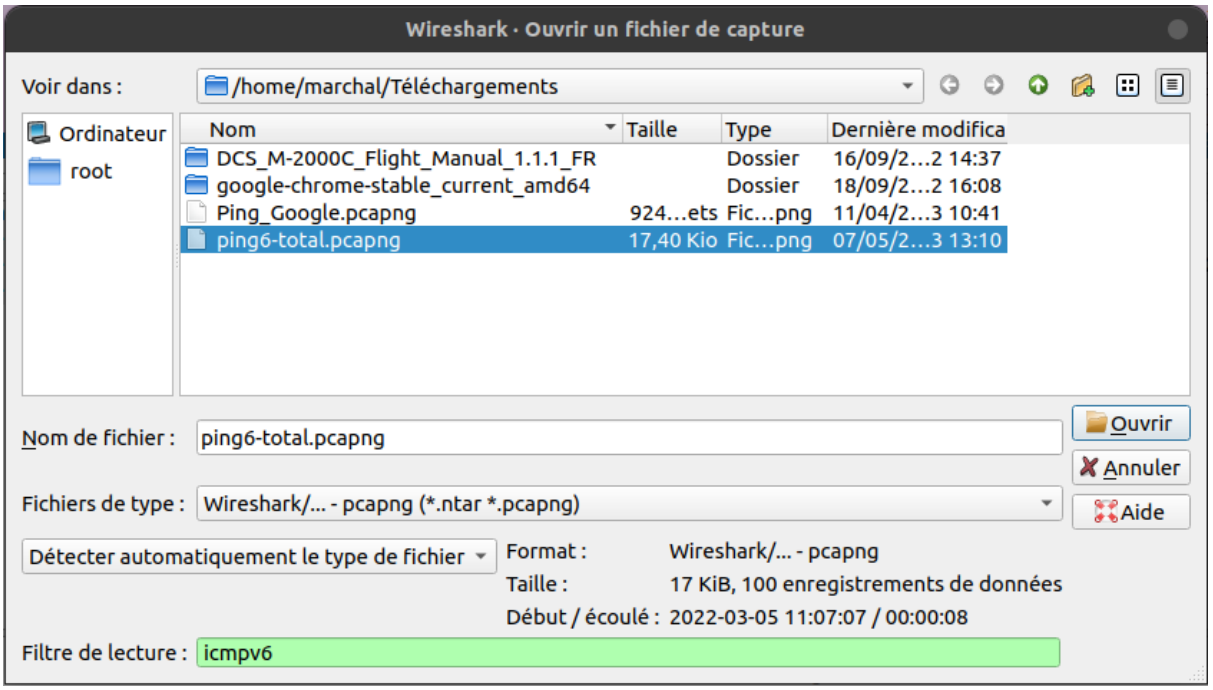
**émettre un paquet ou une trame avec sapy:**
Format:
ARP,Ether/IP (v4 ou v6)/ICMP (v4 ou v6)/UDP, TCP/ …
PING:
ping=IP(dst='10.2.12.4')/ICMP()#si il y a rien par défaut il fera un echo
send(ping)#pour envoyer
Option du send:
send(ping, loop=1)# ce qui enverra un ping en boucle.
send(ping,loop=1,inter=1)# ce qui enverra en boucle mais avec un intervalle de 1s.
send(ping, iface="ens3")# pour dire sur quelle interface.
Recevoir les réponses:
Niveau 3:
sr #pour voir simplement si le paquet à bien été reçus
srloop #pour envoyer en boucle et avoir un retour
sr1#pour voir le contenue du paquet envoyer
Niveau 2:
srp#pour voir simplement si la trame à bien été reçus
srploop#pour envoyer en boucle et avoir un retour
srp1#pour voir le contenue de la trame envoyer
exemple:
ping=IP(dst='192.168.1.254')/ICMP()
ok, nonok = srloop(ping, count=10, inter=1)
Il envoie un paquet 10 fois avec un intervalle de 1 seconde à chaque fois.
**Sniffer des paquets:**
sniff(filter="", count=0, prn=None, lfilter=None, timeout=None, ..)

**count :** nombre de paquet à capturer. La valeur par défaut est 0 ce qui veut dire qu'il n'y a pas de limite au nombre de paquets capturés.
**• prn :** nom de la fonction à appliquer à chaque paquet reçu. C'est dans cette fonction qu'on effectue les traitements sur les paquets (détection du login, mot de passe, …)
**• lfilter :** filtre les paquets à l'aide d'une fonction Python, on pourra utiliser une fonction lambda. Comme filter ce filtre est utilisé pour filtrer les paquets en utilisant la syntaxe Python/Scapy. On peut dont cibler n'importe quel champ d'un protocole par contre ce filtre est beaucoup plus lent car pas implémenté dans le noyau. Ex : lfilter = lambda pkt: TCP in pkt and(pkt[TCP].dport == 80 or pkt[TCP].sport == 80)
**• 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). Si la capture dure dans le temps et qu'on stocke les paquets, la RAM allouée au processus augmentera progressivement avec l'enregistrement des paquets.
**• stopfilter :** fonction à évaluer pour arrêter la capture (la fonction doit retourner true pour arrêter, false pour continuer)

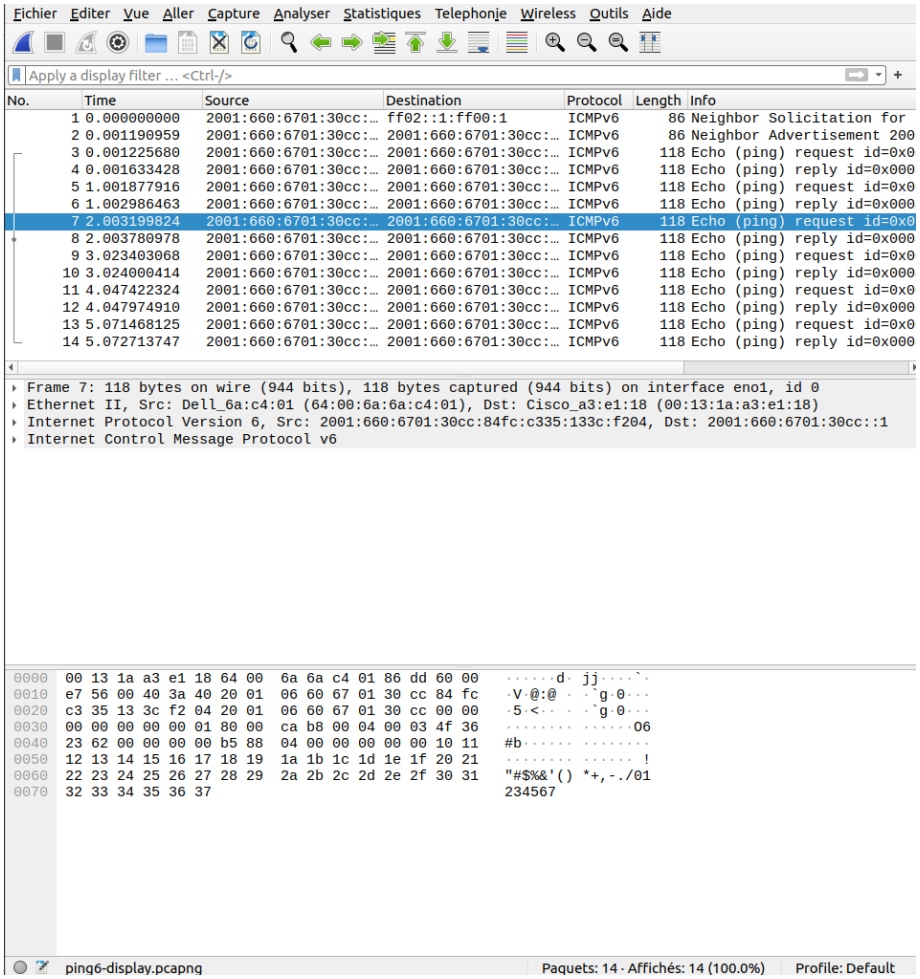
# Tp N°2:

## Wireshark:

pour le démarrer en root: sudo wireshark
Pour ouvrir le fichier demander avec les parametrage demander:




ensuite vous devrez avoir ceci afficher:




pour enregistrer les 14 trames il suffit d'aller dans fichier → exporter les paquets spécifiques puis sélectionner tous les paquet donc les 14 et données le nom suivant:

ping6-display

et au format pcapng.

Entrée [12]:

```
from scapy.all import *

trames=rdpcap("/home/marchal/Bureau/sae204/ping6-display.pcapng")

for trame in trames: # on fait une boucle pour les traiter une par une ...
    if(trame[0][1].version)==6: # on ne prend que les paquets en IPv6
        if (trame[0][1].nh)==58: # on ne prend que les paquets dont le NextHeader = ICMPv6
            if (trame[0][2].type)==135: # on ne prend que les paquets de type ICMP = 135 (NS)
                print(f"Ethernet: MAC Source : {trame[0][0].src}")
                print(f"Ethernet: MAC Destination : {trame[0][0].dst}")
                print(f"IPv6 : IP Source : {trame[0][1].src}")
                print(f"IPv6 : IP Destination : {trame[0][1].dst}")
                print(f"ICMPv6 : IP Target : {trame[0][2].tgt}")
                print(f"ICMPv6 : MAC Requested : {trame[0][3].lladdr}")
```

```
Ethernet: MAC Source : 64:00:6a:6a:c4:01
Ethernet: MAC Destination : 33:33:ff:00:00:01
IPv6 : IP Source : 2001:660:6701:30cc:84fc:c335:133c:f204
IPv6 : IP Destination : ff02::1:ff00:1
ICMPv6 : IP Target : 2001:660:6701:30cc::1
ICMPv6 : MAC Requested : 64:00:6a:6a:c4:01
```

voici un aperçu de ce qu'on peut faire avec scapy et notre fichier ping-dispay.pcapng maintenant nous allons essayer avec le fichier non filtré qui est l'original donc avec ping6-total.pcapng mes vous allez voir très vite un problème car la première trames de notre paquet et une requête ARP sachant quand IP n'est pas utilisez c'est pour ça qu'il aura un message d'erreur avec version comme ci dessous:
AttributeError: version
car il serait pas donc impossible pour votre programme de fonctionner car il ne trouve pas le champ IP donc version.

```
from scapy.all import *

trames=rdpcap("/home/marchal/Téléchargements/ping6-total.pcapng")

for trame in trames:
    try:
        trame[0][0].type
    except:
        continue
        if(trame[0][0].type==34525): # on prend les trames contenant de l'IPv6
            if(trame[0][1].version)==6: # on ne prend que les paquets en IPv6, mais ce test est maintenant redondant
                if (trame[0][1].nh)==58: # on ne prend que les paquets dont le NextHeader = ICMPv6
                    if (trame[0][2].type)==135: # on ne prend que les paquets de type ICMP = 135 (NS)
                        print(f"Ethernet: MAC Source : {trame[0][0].src}")
                        print(f"Ethernet: MAC Destination : {trame[0][0].dst}")
                        print(f"IPv6 : IP Source : {trame[0][1].src}")
                        print(f"IPv6 : IP Destination : {trame[0][1].dst}")
                        print(f"ICMPv6 : IP Target : {trame[0][2].tgt}")
                        print(f"ICMPv6 : MAC Requested : {trame[0][3].lladdr}")
```

Les trames 802.3 n'ont pas de champ type et génèrent une exception qu'il faut gérer d'où le l'utilisation de try et de except.
maintenant nous allons sniff notre réseau pour voir si il y a pas de requête ICMPv6 le programme est le suivant:

```
from scapy.all import *

ICMPv6_types={128 : 'Echo-request', 129 : 'Echo-reply',
              135 : 'Neighbor Solicitation',136 : 'Neighbor Advertisement',
              133 : 'Router Solicitation',134 : 'Routeur Advertisement'}
#notre dictionnaire pour les Différente requête ICMPv6

def print_icmpv6 (trame):#notre fonction pour afficher.
    print(trame.summary())
    type=trame[2].type
    if (type==135 or type==136):
        print(f"TYPE PACKET ICMP : {ICMPv6_types[type]}")
        print(f"Ethernet: MAC Source : {trame[0].src}")
        print(f"Ethernet: MAC Destination : {trame[0].dst}")
        print(f"IPv6 : IP Source : {trame[1].src}")
        print(f"IPv6 : IP Destination : {trame[1].dst}")
        print(f"ICMPv6 : IP Target : {trame[2].tgt}")
        print(f"ICMPv6 : MAC Requested : {trame[3].lladdr}")
        print ("\n")
    else:
        print(f"TYPE PACKET ICMP : {ICMPv6_types[type]}")
        print(f"Ethernet: MAC Source : {trame[0].src}")
        print(f"Ethernet: MAC Destination : {trame[0].dst}")
        print(f"IPv6 : IP Source : {trame[1].src}")
        print(f"IPv6 : IP Destination : {trame[1].dst}")
        print ("\n")

carte=conf.iface#interface réseau utiliser
print(f"On commence le 'sniffing' sur la carte {carte}:")
print("\n")
sniff(filter="ip6 proto 58", prn=print_icmpv6, store=0, iface=carte, count=6)
#vas capturer 6 trames en IPv6
```

Nous allons maintenant  créer un programme équivalent permettant de capturer et afficher les champs essentiels d'un ping en v4, incluant l'échange Req/Rep ARP. voici le programme demander:

```
from scapy.all import *

ICMP_types={ 8: 'Echo-request', 0 : 'Echo-reply'}
#notre dictionnaire pour les Différente requête ICMP

def print_icmp (trame):#notre fonction pour afficher.
   print(trame.summary())
   type=trame[2].type
   if (type==0 or type==8):
       print(f"TYPE PACKET ICMP : {ICMP_types[type]}")
       print(f"Ethernet: MAC Source : {trame[0].src}")
       print(f"Ethernet: MAC Destination : {trame[0].dst}")
       print(f"IPv4 : IP Source : {trame[1].src}")
       print(f"IPv4 : IP Destination : {trame[1].dst}")
       print(f"ICMP : IP Target : {trame[1].dst}")
       print ("\n")

def print_arp (trame):
   if trame[ARP].op == 1:
       return f"Request: {trame[ARP].psrc} qu'elle est l'@MAC de
{trame[ARP].pdst}"
   if trame[ARP].op == 2:
       return f"*Response: {trame[ARP].hwsrc} vois-ci l'@MAC de
{trame[ARP].psrc} \n"
def choix(trame):
   if (trame.haslayer("ARP")):
       print_arp(trame)
   elif(trame.haslayer("ICMP")):
      print_icmp(trame)

carte=conf.iface#interface réseau utiliser
print(f"On commence le 'sniffing' sur la carte {carte}:")
print("\n")
sniff(filter=" ", prn=choix, store=0, iface=carte,count=6)
#vas capturer 6 trames en IPv4 ICMP et arp
```

# Challenges FTP

## Etape 1:

On récupère le nom de l'utilisateur ainsi que son mot passe. Voici le programme:

```
from scapy.all import *
fichier=open("donné.txt","w")#On ouvre un fichier txt
list2=[]
list1=[]
list3=[]

trame=rdpcap("ftp-total.pcapng")#on importe notre fichier de capture
wireshark
trames=trame[TCP][Raw]#pour prendre que les fichier qui sont en tcp est
qui on une partie donné

for i in range(len(trames)):#pour pouvoir commencer le tri
   if trames[i][TCP].dport==21 or trames[i][TCP].sport==21:#pour avoir
que les ports 21
       try:#décodage
           list2.append(trames[i][TCP][Raw].load.decode('utf-8'))

       except:#si il trouve pas de donné il passe au suivant
           pass

for i in range (len(list2)):#pour pouvoir avoir que le nom utilisateur
est le mot de passe ainsi l'écrire dans notre fichier
   if list2[i][:4]=="USER" or list2[i][:4]=="PASS":
       fichier.write(list2[i])
   elif list2[i][:4]=="RETR":#pour pouvoir savoir le type de fichier
       list1.append(list2[i])

mot=list1[0].split(" ")
mot2=mot[1].split("\r")



fichier.close()#fermeture du fichier
```

On ne mais que le port 21 car c'est par celui-ci que ce type de donné son transmise et le port 20 ne sont que les fichier du style les pdf.

## Etape 2:(bonus)

On récupère le nom des fichiers qui sont transférés. Voici le programme:

```
from scapy.all import *
fichier=open("donné.txt","w")#On ouvre un fichier txt
list2=[]
list1=[]
list3=[]

trame=rdpcap("ftp-total.pcapng")#on importe notre fichier de capture
wireshark
trames=trame[TCP][Raw]#pour prendre que les fichier qui sont en tcp est
qui on une partie donné

for i in range(len(trames)):#pour pouvoir commencer le tri
   if trames[i][TCP].dport==21 or trames[i][TCP].sport==21:#pour avoir
que les ports 21
       try:#décodage
           list2.append(trames[i][TCP][Raw].load.decode('utf-8'))

       except:#si il trouve pas de donné il passe au suivant
           pass

for i in range (len(list2)):#pour pouvoir avoir que le nom utilisateur
est le mot de passe ainsi l'écrire dans notre fichier
   if list2[i][:4]=="USER" or list2[i][:4]=="PASS":
       fichier.write(list2[i])
   elif list2[i][:4]=="RETR":#pour pouvoir savoir le type de fichier
       list1.append(list2[i])

mot=list1[0].split(" ")
mot2=mot[1].split("\r")

fichier1=open(mot2[0],"wb")

for i in range(len(trames)):#pour pouvoir commencer le tri
   if trames[i][TCP].sport==20:#pour avoir que les ports 21
       try:#décodage
           list3.append(trames[i][TCP][Raw].load)
       except:#si il trouve pas de donné il passe au suivant
           pass
list3.pop(0)
for i in range (len(list3)):#pour récupérrer les fichier
   fichier1.write(list3[i])
```

```
print(list3)



fichier.close()#fermeture du fichier
```

## étape 3 (sniff):

```
from scapy.all import *

list1=[]
list2=[]
list3=[]

def ftp(trame):
   global list2,list3
   if trame[TCP].dport==21 or trame[TCP].sport==21 or
trame[TCP].sport==20:
       try:
           if trame[TCP].dport==21 or trame[TCP].sport==21:#pour avoir
que les ports 21
               list2.append(trame[TCP][Raw].load.decode('utf-8'))
           elif trame[TCP].sport==20:
               list3.append(trame[TCP][Raw].load)
       except:
           pass
   id()
   return list2,list3

def donnée():
   global list3,list1
   list3.pop(0)
   for i in range(len(list1)):
       mot=list1[i].split(" ")
   for i in range(len(list1)):
       mot2=mot[i].split("\r")
   for i in range(len(mot2)):
       fichier=open(mot2[i],"wb")
   for i in range (len(list3)):#pour récupérrer les fichier
       fichier.write(list3[i])
   fichier.close()

def id():
   global list2,list1
```

```
    fichier=open("id.txt","a")
    for i in range (len(list2)):#pour pouvoir avoir que le nom
utilisateur est le mot de passe ainsi l'écrire dans notre fichier
        if list2[i][:4]=="USER" or list2[i][:4]=="PASS":
            print(list2[i])
            fichier.write(list2[i])
        elif list2[i][:4]=="RETR":
            list1.append(list2[i])
            donnée()
    fichier.close()
    return list1

carte=conf.iface

def choix(trame):
    if (trame.haslayer("TCP")):
        ftp(trame)



carte=conf.iface#interface réseau utiliser
print(f"On commence le 'sniffing' sur la carte {carte}:")
print("\n")

sniff(filter=" ", prn=choix, store=0, iface=carte)
```

# Challenges TELNET

```
from scapy.all import *

def telnet(trame):
    fichier=open("login-telnet.txt","w")
    list3=[]
    list2=[]
    trame=rdpcap("1.pcapng")
    trames=trame[TCP]
    for i in range(len(trames)):
        if trames[i][TCP].sport==23 or trames[i][TCP].dport==23:
            try:
                list3.append(trames[i][TCP][Raw].load.decode("UTF-8"))
            except:
                pass
```

```
    for i in range(len(list3)):
        list2.append(list3[i])
        if list3[i][:10]=='Last login':
            break
    while i < len(list2):
        if list2[i][:10]=='Last login':
            list2.pop()
    list2.pop(0)
    list2.pop(0)
    for i in range(len(list2)):
        fichier.write(list2[i])
    print(list2)
    fichier.close()

carte=conf.iface


carte=conf.iface#interface réseau utiliser
print(f"On commence le 'sniffing' sur la carte {carte}:")
print("\n")


sniff(filter="tcp port 23", prn=telnet, iface=carte)
```

# Challenges HTTP

## étape 1 (récupération du login et du mot de passe dans le fichier de capture):

```
from scapy.all import *
import base64

trame=rdpcap("www-total.pcapng")

list2=[]

for i in range(len(trame)):
    try:
        if trame[i].dport==80:
            data =str(trame[i][Raw].load)
            if "Authorization: Basic" in data:
                new = data.split("\\r")
                current = []
                for element in new:
                    if "Host: " in element:
```

```
                    current.append(element[8:])
                if "Authorization" in element:
                    current.append(trame[i][IP].src)
                    current.append(trame[i][IP].dst)
                    userpass =
base64.b64decode(element[23:]).decode('utf-8').split(":")
                    current.append(userpass[0])
                    current.append(userpass[1])
            if current not in list2:
                list2.append(current)
                print("----------------------------------------")
                print("Connexion HTML")
                print(f"Site: {current[0]} / IP(serv): {current[2]}
/ IP(client): {current[1]}")
                print(f"User: {current[3]}")
                print(f"Password: {current[4]}")

        if "POST" in data:
            new = data.split("\\r")
            current = []
            for element in new:
                if "Referer: " in element:
                    current.append(element[11:])
                if "username" in element:
                    current.append(trame[i].src)
                    current.append(trame[i].dst)
                    logform = element.split("&")
                    current.append(logform[1][9:])
                    current.append(logform[0][11:])

            if current not in list2:
                list2.append(current)
                print("----------------------------------------")
                print("Connexion HTML.")
                print(f"Site: {current[0]} / IP(serv): {current[2]}
/ IP(client): {current[1]}")
                print(f"User: {current[3]}")
                print(f"Password: {current[4]}")


    except:
        pass
```

## étape 2 (sniff):

```
from scapy.all import *
import base64

trame=rdpcap("www-total.pcapng")

fichier=open("login.txt","a")
list2=[]

def http(trame):
   global list2,fichier
   try:
       if trame.dport==80:
           data =str(trame[Raw].load)
           if "Authorization: Basic" in data:
               new = data.split("\\r")
               current = []
               for element in new:
                   if "Host: " in element:
                       current.append(element[8:])
                   if "Authorization" in element:
                       current.append(trame[IP].src)
                       current.append(trame[IP].dst)
                       userpass =
base64.b64decode(element[23:]).decode('utf-8').split(":")
                       current.append(userpass[0])
                       current.append(userpass[1])
               if current not in list2:
                   list2.append(current)
                   print("----------------------------------------")

fichier.write("----------------------------------------")
                   print("Connexion HTML")
                   fichier.write("Connexion HTML")
                   print(f"Site: {current[0]} / IP(serv): {current[2]}
/ IP(client): {current[1]}")
                   fichier.write(f"Site: {current[0]} / IP(serv):
{current[2]} / IP(client): {current[1]}")
                   print(f"User: {current[3]}")
                   fichier.write(f"User: {current[3]}")
                   print(f"Password: {current[4]}")
                   fichier.write(f"Password: {current[4]}")
```

```
            if "POST" in data:
                new = data.split("\\r")
                current = []
                for element in new:
                    if "Referer: " in element:
                        current.append(element[11:])
                    if "username" in element:
                        current.append(trame.src)
                        current.append(trame.dst)
                        logform = element.split("&")
                        current.append(logform[1][9:])
                        current.append(logform[0][11:])

                if current not in list2:
                    list2.append(current)
                    print("----------------------------------------")

fichier.write("----------------------------------------")
                    print("Connexion HTML.")
                    fichier.write("Connexion HTML")
                    print(f"Site: {current[0]} / IP(serv): {current[2]}
/ IP(client): {current[1]}")
                    fichier.write(f"Site: {current[0]} / IP(serv):
{current[2]} / IP(client): {current[1]}")
                    print(f"User: {current[3]}")
                    fichier.write(f"User: {current[3]}")
                    print(f"Password: {current[4]}")
                    fichier.write(f"Password: {current[4]}")

    except:
        pass

def choix(trame):
  if (trame.haslayer("TCP")):
      http(trame)


carte=conf.iface#interface réseau utiliser
print(f"On commence le 'sniffing' sur la carte {carte}:")
print("\n")


sniff(filter=" ", prn=choix, store=0, iface=carte)
```