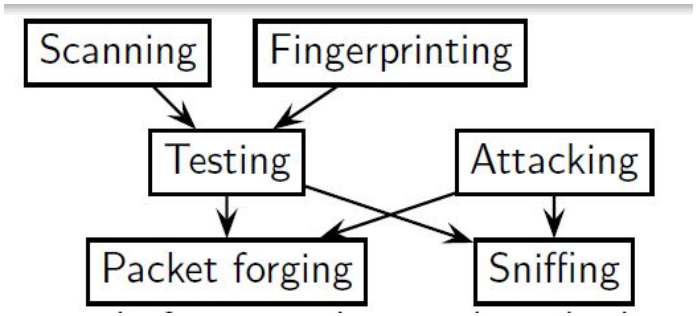
SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS
DEPUIS 1257

Scapy: outil de manipulation avancée des paquets réseaux

Nesrine Ammar – ProgRes 2018

nesrine.ammar@etu.upmc.fr

Objectives de manipulation des paquets réseaux



Packet forging tool

Forge des paquets et envoient les dans le réseaux

Sniffing tool

Capture des paquets réseaux

Testing tool

Essaye de répondre à une question précise (oui/non) ex. ping

Objectives de manipulation des paquets réseaux

Scanning tool

Faire des tests utilisant différents paramètres

Fingerprinting tool

Faire des tests prédéfinis afin de discriminer un pair

Attacking tool

Utilise des valeurs inattendus d'un protocole

Bibliothèque d'analyse de trafic

Pcap

- Interface de programmation (API) permettant de capturer le trafic réseau
- Implémenté sous Unix et Mac OS par la bibliothèque Libpcap et sous windows par la bibliothèque Winpcap

Libpcap et winpcap:

- Assure le filtrage, la capture et l'analyse de paquets,
- Supporte plusieurs protocoles: IP(), UDP(), DNS(), DHCP(), HTTP()…

Libpcap

Programme utilisant Libpcap:

- Tcpdump
- Wireshark
- Nmap

Bibliothèque d'enveloppe pour Libpcap(wrapper librairies):

- Python: pylibpcap, scapy
- Java: jpcap, jNetPcap
- Ruby: PacketFu

…

Python

- This is an **int** (signed, 32bits) : 42
- This is a **long** (signed, infinite): 42L
- This is a **str** : "bell\x07\n" or 'bell\x07\n' (" $\Leftrightarrow$ ')
- This is a **tuple** : (1,4,"42")
- This is a **list** : [4,2,"1"]
- This is a **dict** : { "one":1 , "two":2 }

Python

```
If cond1:
  instr
  instr
Elif cond2:
  instr
Else
  instr
```

```
Try:
  instr
Except exception:
  instr
Else
  instr
```

```
For var in set:
  instr
```

```
While cond:
  instr
  instr
```

```
Def func(x):
  if x==0:
    return 1
  else
    return x*2
```

Scapy

Un outil de manipulation avancée des paquets réseaux, écrit en python

Objectives:

- Intercepter le trafic sur un segment réseau
- Générer des paquets
- Réaliser une prise d'empreinte (fingerprinting)
- Faire une traceroute
- Analyser le trafic réseau

…

Scapy

- **Points forts :**

Langage interactif de haut niveau

Forge et analyse de paquets très simples

Passe le firewall local

- **Points faibles :**

Fournit les résultats bruts, ne les interprète pas

Support partiel de certains protocoles complexes

Scapy

Liste de couches

```
>>> ls()
ARP : ARP
DHCP : DHCP options DNS : DNS
Dot11 : 802.11
[...]
```

Liste de commandes

```
>>> lsc()
sr : Send and receive packets at layer 3
sr1 : Send packets at layer 3 and return only the fi
srp : Send and receive packets at layer 2
[...]
```

Valeurs par défaut pour le packet IP

```
>>> ls(IP)
version : BitField = (4)
ihl : BitField = (None)
tos : XByteField = (0)
len : ShortField = (None)
id : ShortField = (1)
flags : FlagsField = (0)
frag : BitField = (0)
ttl : ByteField = (64)
proto : ByteEnumField = (0)
chksum : XShortField = (None)
src : Emph = (None)
dst : Emph = ('127.0.0.1')
options : IPoptionsField = ('')
```

Scapy

- Chaque paquet est crée couche par couche
- Chaque couche peut être empiler sur une autre
- Chaque couche et chaque paquet peuvent être manipulés indépendamment
- Chaque champ dans l'entête des protocoles possède une valeur par défaut
- Chaque champ peut contenir une valeur ou plusieurs à la fois

```
>>> a=IP(dst="www.target.com", id=0x42)
>>> a.ttl=12
>>> b=TCP(dport=[22,23,25,80,443])
>>> c=a/b
```

Scapy: valeurs par défaut

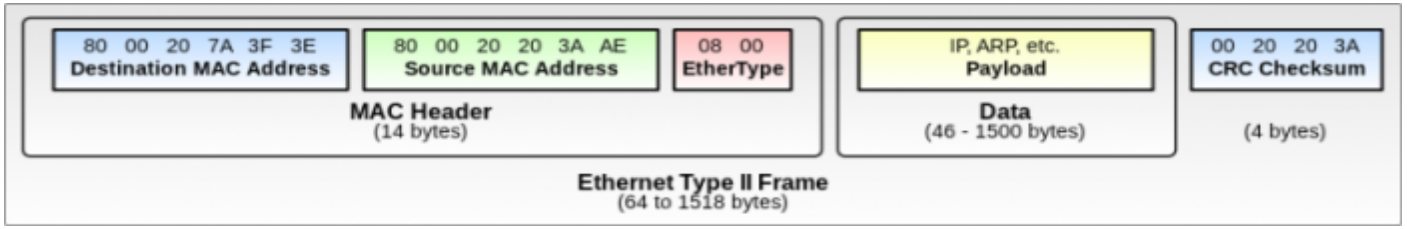
- IP source est choisie selon l'adresse destination et la table de routage
- La valeur de checksum est calculée
- MAC source est choisie selon l'interface de sortie
- Le type du paquet Ethernet et le protocole IP est choisie selon les couches supérieures
- …

D'autres champs les plus utilisés:

- Port TCP source est 20, TCP destination est 80
- Port UDP source et destination est 53
- Le type de ICMP est echo request
- …

Création d'une trame

```
>>> ma_trame = Ether()
>>> ma_trame.show()
###[ Ethernet ]###
WARNING: Mac address to reach destination not found. Using broadcast.
dst= ff:ff:ff:ff:ff:ff
src= 00:00:00:00:00:00
type= 0x0
>>>
```




```
>>> ma_trame = Ether(dst='00:19:4b:10:38:79')
>>>
```

Sniffer

```
>>> rep = sniff(filter="host 192.168.1.10")
>>>
```

```
$ ping 192.168.1.10 -c 1
```

```
>>> rep.show()
0000 Ether / IP / ICMP 192.168.1.14 > 192.168.1.10 echo-request 0 / Raw
0001 Ether / IP / ICMP 192.168.1.10 > 192.168.1.14 echo-reply 0 / Raw
>>>
```

Ecrire et lire un fichier pcap

>>> wrpcap('file_name.pcap', packet)
>>>
>>> packet = rdpcap('file_name.pcap')

>>> packet[Ether].dst → adresse mac dst
Packet[IP].dst → adresse ip dst

```
>>> packet
<Ether  dst=00:19:4b:10:38:79 src=00:26:5e:17:00:6e type=0x800 |<IP  version=4L ihl=5L tos=0x0 len=64
id=37095 flags=DF frag=0L ttl=64 proto=udp chksum=0x2666 src=192.168.1.14 dst=192.168.1.1
options=[] |<UDP  sport=38897 dport=domain len=44 chksum=0x65a |<DNS  id=28184 qr=0L
opcode=QUERY aa=0L tc=0L rd=1L ra=0L z=0L rcode=ok qdcount=1 ancount=0 nscount=0 arcount=0
qd=<DNSQR  qname='www.siteduzero.com.' qtype=A qclass=IN |> an=None ns=None ar=None |>>>>
```

Navigation entre les couches réseaux

```
If UDP in pkt:
    print pkt[UDP].chksum
```

```
If pkt.haslayer(UDP):
    print pkt[UDP].chksum
```

→Permet de vérifier la présence de la couche UDP

→Permet de retourner la couche demandée (UDP)

→Le code est indépendant des couches inférieures

Calculer le nombre de paquets UDP

```
#! /usr/bin/python
From scapy.all import *

Def  compter(filename):
  nb_paquets = 0
  file  =  rdpcap(filename)
  for  pkt  in  file:
    if  pkt.haslayer(UDP):
      nb_paquets+=1
  print  nb_paquets

compter('captures.pcap')
```

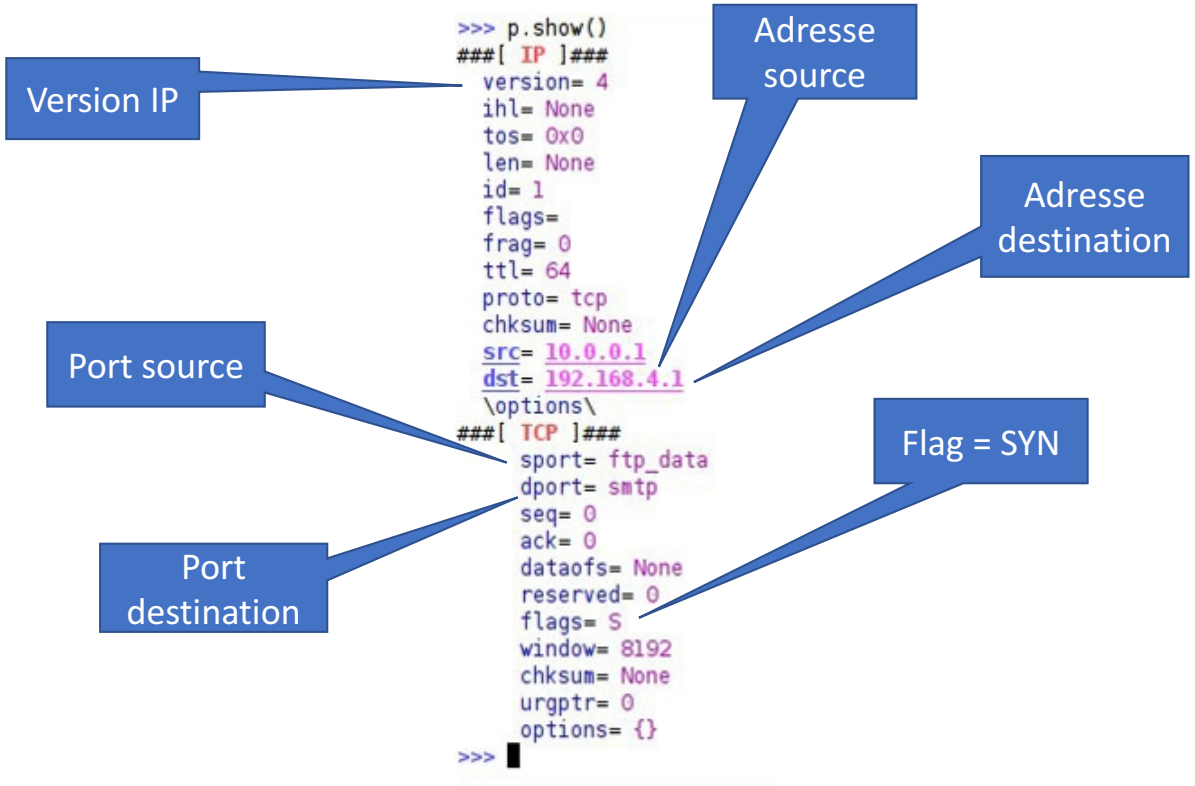
Pour une vue développée du paquet

```
#! /usr/bin/python
From scapy.all import *

def afficher_tcp(filename):
  file = rdpcap(filename)
  for pkt in file:
    if pkt.haslayer(TCP):
      pkt.show()
      break:

afficher_tcp('captures.pcap')
```

Pour une vue développée du paquet

```
>>> p.show()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= tcp
  chksum= None
  src= 10.0.0.1
  dst= 192.168.4.1
  \options\
###[ TCP ]###
     sport= ftp_data
     dport= smtp
     seq= 0
     ack= 0
     dataofs= None
     reserved= 0
     flags= S
     window= 8192
     chksum= None
     urgptr= 0
     options= {}
>>>
```

Pour une vue développée du paquet



Afficher sport du premier paquet TCP

```
#! /usr/bin/python
From scapy.all import *

File = rdpcap('captures.pcap')
for pkt in file:
 try:
   if pkt.haslayer(TCP):
     print pkt[TCP].sport
     break:
```

Scapy: commandes

Command	Effect
raw(pkt)	assemble the packet
hexdump(pkt)	have an hexadecimal dump
ls(pkt)	have the list of fields values
pkt.summary()	for a one-line summary
pkt.show()	for a developed view of the packet
pkt.show2()	same as show but on the assembled packet (checksum is calculated, for instanc
pkt.sprintf()	fills a format string with fields values of the packet
pkt.decode_payload_as()	changes the way the payload is decoded
pkt.psdump()	draws a PostScript diagram with explained dissection
pkt.pdfdump()	draws a PDF with explained dissection
pkt.command()	return a Scapy command that can generate the packet

Sprintf()

Grâce à la methode sprintf():

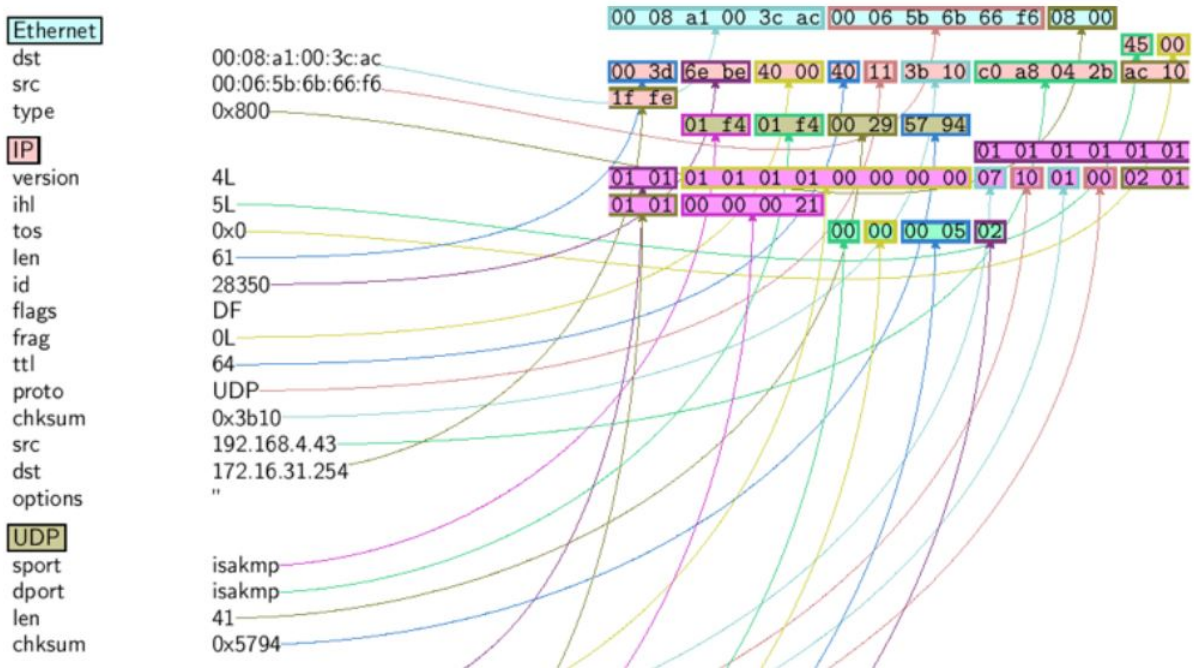
- Créer notre propre résumé du paquet
- Extraire les différentes couches et focaliser sur les champs nécessaires

```
>>> pkt = IP(dst="192.168.8.1",ttl=12)/UDP(dport=123)
>>> pkt.sprintf("The source is %IP.src%")
’The source is 192.168.8.14’
```

- “%”, “{” and “}” sont des caractères spéciaux
- Ils sont remplacés par “%%”, “%(” and “%)”

pdfdump()

```
>>> pkt.pdfdump()
```



Summary()

```
>>> file = rdpcap("test.pcap")
>>> pkt = file[0]
>>> pkt.summary()
Ether / IP / TCP 42.2.5.3:3021 > 192.168.8.14:22 PA / Raw
```

Bytes()

```
>>> bytes(pkt[IP])
b'E\x00\x00\x14\x00\x01\x00\x00@\x00|\xe7\x7f\x00\x00\x01\x7f\x00\x00\x01'
```

Hexdump()

La fonction hexdump() donne une représentation hexadécimale du paquet (ainsi qu'une représentation ASCII) et s'utilise ainsi :

```
>>> hexdump(packet)

0000   00 19 4B 10 38 79 00 26  5E 17 00 6E 08 00 45 00   ..K.8y.&^..n..E.
0010   00 40 90 E7 40 00 40 11  26 66 C0 A8 01 0E C0 A8   .@..@.@.&f......
0020   01 01 97 F1 00 35 00 2C  06 5A 6E 18 01 00 00 01   .....5.,.Zn.....
0030   00 00 00 00 00 00 03 77  77 77 0A 73 69 74 65 64   .......www.sited
0040   75 7A 65 72 6F 03 63 6F  6D 00 00 01 00 01         uzero.com.....
```

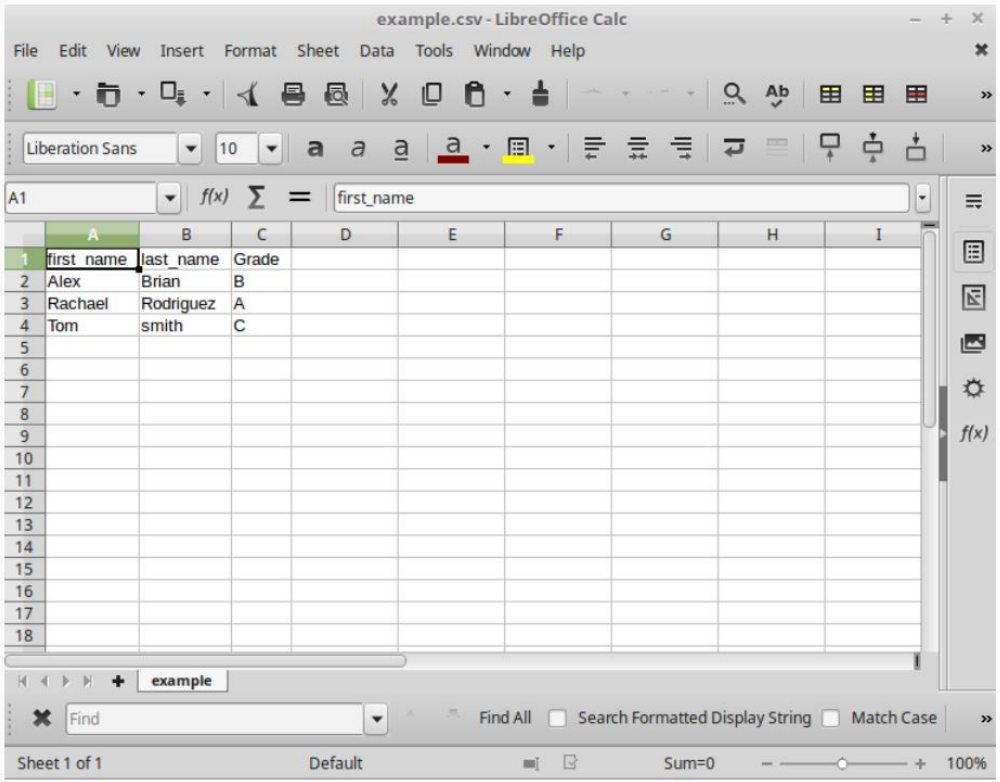
Python: CSV

```
import csv

with open('example.csv', newline='') as File:
    reader = csv.reader(File)
    for row in reader:
        print(row)
```

Python: CSV

```
import csv

myData = [["first_name", "second_name", "Grade"],
        ['Alex', 'Brian', 'A'],
        ['Tom', 'Smith', 'B']]

myFile = open('example2.csv', 'w')
with myFile:
    writer = csv.writer(myFile)
    writer.writerows(myData)

print("Writing complete")
```

Python: CSV


example.csv - LibreOffice Calc
File Edit View Insert Format Sheet Data Tools Window Help
Liberation Sans
10
A1
f(x)
first_name
A B C D E F G H I
1 first_name last_name Grade
2 Alex Brian B
3 Rachael Rodriguez A
4 Tom smith C
5 6 7 8 9 10 11 12 13 14 15 16 17 18
example
Find
Find All
Search Formatted Display String
Match Case
Sheet 1 of 1
Default
Sum=0
100%