Scapy

sebastien.larinier@esiea.fr

esiea

Agenda

- Introduction et Installation
- concepts
- Ping avec Scapy et Wireshark
- Lire un fichier pcap
- Developper un client DNS
- Man in the middle

esiea

Introduction

- module python
- Crafteur de paquets
- testeur de réseau
- analyseur de réseau

esiea

Préparation de l'environnement

- Création d'un virtualenv
- Installation de scapy

```
pip install scapy
```

- Installation de wireshark

esiea

Concepts (1/5)

- Construction des packets couche par couche
- Chaque couche est manipulable indépendamment
- chaque champs de paquets a une/des valeur(s) par défaut

```
>>> a = IP(dst="www.google.com")
>>> a.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = ip
  chksum    = None
  src       = 10.8.63.140
  dst       = Net("google.com/32")
  \options   \
```

esiea

```
b = TCP(dport=80)
c=a/b
c.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 10.8.63.140
  dst       = Net("google.com/32")
  \options   \
###[ TCP ]###
     sport     = ftp_data
     dport     = http
     seq       = 0
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = ''

>>>
```

Concepts (2/5)

- l'ensemble des champs sont multivalués
- on ne peut pas modifier la source des IPs, des mac adresses
- de base le port source TCP est 20 et 80 pour la destination
- pour UDP c'est 53
- pour acceder l'ensemble des champs d'un protocole

```
>>> ls(IP)
```

esiea

Concepts (3/5)

pour avoir la trame ethernet en raw

```
>>> raw(c)
b'E\x00\x00(\x00\x01\x00\x00@\x06\xd8\x92\n\x08?\x8c\x8e\xfa\xc9\xae\x00\x14\x00P\x00\x00\x00\x00\x00\x00\x00\x00P\x02 \x00\xedA\x00\x00'
```

Concepts (4/5)

- hexdump(pkt) pour avoir un dump hexa
- ls(pkt) pour avoir la liste des valeurs des champs
- pkt.summary() pour un résumé en une ligne
- pkt.show() pour une vue développée du paquet
- pkt.show2() même chose que show mais sur le paquet assemblé (la somme de contrôle
  est calculée, par exemple)
- pkt.sprintf() remplit une chaîne de format avec les valeurs des champs du paquet
- pkt.decode payload as() change la façon dont la charge utile est décodée
- pkt.psdump() dessine un postscript avec une dissection expliquée
- pkt.pdfdump() dessine un PDF avec une dissection expliquée

Concepts (5/5)

- Lire un fichier pcap, faire du sniffing est lire une liste de paquets
- Le resultat d'une probe est un tuple (paquet recu, paquet envoyé)

Methodes

- summary() affiche une liste de chaque paquet
  nsummary() identique au précédent, avec le numéro du paquet
  sr() liste les paquets avec les réponses associées à l'envoie
  conversations() affiche un graphique des conversations entre les hôtes
  sessions() retourne un dictionnaire indexé par "flows". (ip source, port source, ip destination, port destination)
  replace() remplace une valeur de champ par une autre
  show() affiche la représentation du packet (habituellement nsummary())
  filter() retourne une liste de paquets filtrée avec une fonction lambda
  hexdump() retourne un hexdump de tous les paquets
  hexraw() retourne un fichier hexdump de la couche Raw de tous les paquets
  padding() retourne un vidage hexadécimal de tous les paquets avec padding
  nzpadding() renvoie un fichier hexadécimal des paquets avec un remplissage non nul.
  plot() trace une fonction lambda appliquée à la liste de paquets
  make_table() affiche un tableau selon une fonction lambda

Concepts send and receive

- sr() envoie et reçoit au niveau de la couche 3
- sr1() envoie et reçoit en couche 3, renvoie uniquement la première réponse
- srloop() envoi et réception en boucle à la couche 3
- srp() envoi et réception en couche 2
- srp1() envoie et reçoit au niveau de la couche 2, renvoie seulement la première réponse
- srploop() envoi et réception en boucle à la couche 2

PING avec Scapy et Wireshark

```
import sys

import scapy.all as scapy
ip = sys.argv[1]
count = sys.argv[2]
for i in range(int(count)):
    scapy.send(scapy.IP(dst=ip)/scapy.ICMP(id=1, seq=1))
```

19… 69.947…	8.8.8.8	10.8.63.140	ICMP	60 Echo (ping) reply	id=0x0001, seq=1/256, ttl=120 (request in 1993)
19… 69.947…	10.8.63.140	8.8.8.8	ICMP	42 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (reply in 1996)
19… 69.948…	8.8.8.8	10.8.63.140	ICMP	60 Echo (ping) reply	id=0x0001, seq=1/256, ttl=120 (request in 1995)
19… 69.949…	10.8.63.140	8.8.8.8	ICMP	42 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (reply in 1998)
19… 69.950…	8.8.8.8	10.8.63.140	ICMP	60 Echo (ping) reply	id=0x0001, seq=1/256, ttl=120 (request in 1997)
19… 69.952…	10.8.63.140	8.8.8.8	ICMP	42 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (reply in 2000)
20… 69.953…	8.8.8.8	10.8.63.140	ICMP	60 Echo (ping) reply	id=0x0001, seq=1/256, ttl=120 (request in 1999)
20… 69.954…	10.8.63.140	8.8.8.8	ICMP	42 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (reply in 2002)
20… 69.955…	8.8.8.8	10.8.63.140	ICMP	60 Echo (ping) reply	id=0x0001, seq=1/256, ttl=120 (request in 2001)
20… 69.955…	10.8.63.140	8.8.8.8	ICMP	42 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (no response found!)
20… 69.957…	10.8.63.140	8.8.8.8	ICMP	42 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (reply in 2005)
20… 69.957…	8.8.8.8	10.8.63.140	ICMP	60 Echo (ping) reply	id=0x0001, seq=1/256, ttl=120 (request in 2004)
20… 69.958…	8.8.8.8	10.8.63.140	ICMP	60 Echo (ping) reply	id=0x0001, seq=1/256, ttl=120
20… 69.958…	10.8.63.140	8.8.8.8	ICMP	42 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (reply in 2008)
20… 69.959…	8.8.8.8	10.8.63.140	ICMP	60 Echo (ping) reply	id=0x0001, seq=1/256, ttl=120 (request in 2007)
20… 69.960…	10.8.63.140	8.8.8.8	ICMP	42 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (reply in 2010)

Question:

est ce que le TTL est modifié par rapport à la stack de l'OS ?

Si oui, pourquoi d'après vous ?

tests

- Envoyer des paquets avec la commande sr et analyser les résultats
- Envoyer des paquets avec la commande sr1 et analyser les résultats

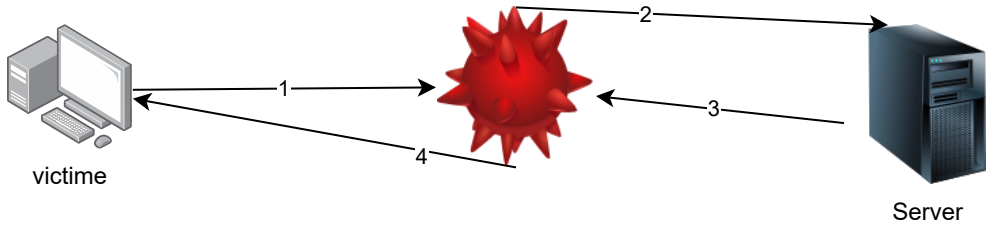
Exercice lire un pcap

- Lire un fichier pcap avec la commande rdpcap
- Afficher les IPs et les domaines des requêtes DNS
- Exporter tous les fichiers lors de la transactions HTTPs

Faire un client DNS

- Envoyer une requête DNS (A, AAAA, MX, NS, CNAME, TXT, PTR, SOA, SRV, HINFO, NAPTR, SPF, ANY)
- Afficher les différentes réponses

Man in the middle



Process

- Modifier les tables ARP
- Verifier les tables ARP qui ont modifiées
- Afficher les liens entre la victime et le serveur

--