Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Audit&SécuRés

Corrections TP n°1

Introduction à Scapy

Contrôle d'erreur

1 – Différence « checksum » et « CRC » :

a. Créez à l'aide des fonctions et objets disponibles dans Scapy deux trames Ethernet contenant :
   1. un paquet UDP contenant les données : `'\x00\x00'` ;
   2. un paquet UDP contenant les données : `'\xFF\xFF'` ;

b. Comparez les valeurs *calculées* des checksums présents dans la couche UDP : sont ils identiques ? Pourquoi ?
Les valeurs sont identiques, car avec le checksum, l'addition de 65535 est équivalent à ajouter 0 !

c. Calculez le CRC-32 de chacune de ces trames : sont-ils différents ? Pourquoi ?
Les valeurs sont différentes car le CRC est capable de détecter une modification de la trame d'une longueur de 16 bits.

Détection de trafic d'audit

2 – Écrire un programme utilisant Scapy permettant de détecter si on est la cible d'un scannage de port :

```
#!/usr/bin/python3

from scapy.all import *

traces = {}

# traces est un dico de dico {@IPsrc:{port:temps, ...port:temps}, ...}
def verifier_attaque(a,t):
    if len(t)<10:
        return
    temps_min = min(t.values())
    temps_max = max(t.values())
    if (temps_max-temps_min) < 10:
        print ("Attaque possible depuis %s"%a)

def traiter_trame(t):
    if (not TCP in t):
        return
    if (t.sprintf("%TCP.flags%")!='S'):
        return
    adresse_source = t[IP].src
    port_destination = t[TCP].dport
    temps = t.time
    if (adresse_source in traces):
        table_ports = traces[adresse_source]
        table_ports[port_destination] = temps
        print (traces)
        verifier_attaque(adresse_source,table_ports)
    else :
        traces[adresse_source]={port_destination:temps}

sniff(count=0,filter="tcp",prn=traiter_trame)
```

3 – Réalisation d'un serveur de « *Port Knocking* ».

Vous écrirez le programme Python utilisant Scapy et réalisant :

- ⋆ la détection de cette séquence de tentative de connexion correspondant à du « port knocking » ;
- ⋆ l'autorisation à l'aide de NetFilter de la connexion vers l'application (le travail du « PK daemon ») :
  - ◦ la séquence de tentative de connexion TCP est la suivante : `[2027, 3230, 2001, 17377]` ;
  - ◦ l'application attend en TCP sur le port `16400` ;
  - ◦ le serveur hébergeant l'application et votre programme possède l'@IP `192.168.1.137`.

```
#!/usr/bin/python3

from scapy.all import *
import commands

sequence = [2027, 3230, 2001, 17377]
adresse_serveur = '192.168.1.137'
adresse_client = ''
port_application = 16400
rang_sequence = 0

def traiter_paquet(p):
  global rang_sequence
  if (p.sprintf('%TCP.flags%') == 'S'):
     if ((rang_sequence == 0) and (p[TCP].dport == sequence[0])):
      adresse_client = p[IP].src
    if ((p[TCP].dport == sequence[rang_sequence]) and (p[IP].src == adresse_client)):
      rang_sequence += 1
      print ("Knock...",rang_sequence)
      if (rang_sequence == len(sequence)):
         rang_sequence = 0
         commands.getoutput('sudo iptables -A INPUT -s %s -p tcp --dport %d -m state --state NEW, ESTABLISHED -j ACCEPT'%(adresse_client,port_application))
   sniff(filter="(dst host %s) and tcp"%adresse_serveur, prn=traiter_paquet)
```