*nix Araçlarıyla Derinlemesine Paket Analizi

1.Giriş:

Günümüzde kopmaz bir şekilde bağlandığımız internet ve bununla beraber gelen diğer ağ teknolojileriyle birlikte, ağ içersinden veya dısarısından meydana gelebilecek çeşitli saldırılar, bantgenişliğini yanlış kullanımını vb sorunlar artmaktadır. Bunları engellemek için kullanılan bazı araçlar vardır. Bu araçlarla isterse paket içersinde ki bayraklar seviyesine inerek veyahut gelen paketler içersinden isim çözümlemesi yaparak analiz yapabilmekteyiz. Bu yazımızda bunlardan özellikle

- tcpdump
  - sniff
  - tcpdfilter

- ngrep
- p0f

sniff ve tcpdfilter aracları tcpdump ile beraber kullanılmaktadır. Şimdi bu araçları en basit kullanımından başlayarak derinlemesine inceleyelim

Tcpdump temel kullanımı

En basit kullanım olarak söyle başlayabiliriz

```
# tcpdump

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
17:00:30.743692 IP 192.168.1.6.ssh > 192.168.1.2.3574: P 4216237590:4216237706(116) ack 1317638344 win 7504
17:00:30.745055 IP 192.168.1.2.3574 > 192.168.1.6.ssh: P 1:53(52) ack 116 win 63868
17:00:30.745158 IP 192.168.1.6.ssh > 192.168.1.2.3574: . ack 53 win 7504
17:00:30.748649 IP 192.168.1.6.ssh > 192.168.1.2.3574: P 116:232(116) ack 53 win 7504
17:00:30.752549 IP 192.168.1.2.3574 > 192.168.1.6.ssh: P 53:105(52) ack 232 win 63752
17:00:30.759639 IP 192.168.1.6.32771 > 192.168.1.1.domain:  9184+ PTR? 2.1.168.192.in-addr.arpa. (42)
17:00:30.792271 IP 192.168.1.6.ssh > 192.168.1.2.3574: . ack 105 win 7504
17:00:30.899779 IP 64.233.183.89.www > 192.168.1.2.3224: P 1262527807:1262527819(12) ack 4162857010 win
18980
17:00:30.901539 IP 192.168.1.2.3224 > 64.233.183.89.www: P 1:23(22) ack 12 win 64216
17:00:30.958759 IP 85.104.177.57.15696 > 192.168.1.2.4500: . 2881874158:2881875618(1460) ack 3777403483 win
65251
17:00:30.967437 IP 64.233.183.89.www > 192.168.1.2.3224: . ack 23 win 18980
17:00:31.001754 IP 85.104.177.57.15696 > 192.168.1.2.4500: P 1460:2600(1140) ack 1 win 65251
17:00:31.001959 IP 192.168.1.2.4500 > 85.104.177.57.15696: . ack 2600 win 64240
17:00:31.137328 IP 88.Red-81-35-40.dynamicIP.rima-tde.net.4672 > 192.168.1.2.4510: UDP, length 35
17:00:31.140333 IP 192.168.1.2.4510 > 88.Red-81-35-40.dynamicIP.rima-tde.net.4672: UDP, length 69
17:00:31.226278 IP 213.199.198.38.4672 > 192.168.1.2.4510: UDP, length 35
17:00:31.234762 IP 192.168.1.2.4510 > 213.199.198.38.4672: UDP, length 119
17:00:31.285270 IP 192.168.1.2.4510 > 85.103.146.197.4672: UDP, length 32
17:00:31.287267 IP 192.168.1.2.4510 > 85.101.134.225.4672: UDP, length 32
17:00:31.467208 IP 85.104.177.57.15696 > 192.168.1.2.4500: . 2600:4060(1460) ack 1 win 65251
17:00:31.473936 IP 200.232.208.225.4672 > 192.168.1.2.4510: UDP, length 35
17:00:31.476274 IP 192.168.1.2.4510 > 200.232.208.225.4672: UDP, length 119
17:00:31.508903 IP 85.104.177.57.15696 > 192.168.1.2.4500: P 4060:5200(1140) ack 1 win 65251
17:00:31.508913 IP 192.168.1.2.4500 > 85.104.177.57.15696: . ack 5200 win 64240
17:00:31.888036 IP  85.99.130.114.6462 > 192.168.1.2.3360: P 1624414437:1624414523(86) ack 4265982471 win
65406

25 packets captured
1496 packets received by filter
1290 packets dropped by kernel
```

Ctrl + C ile sonlandırırsak

Bu en temel kullanımdır ve o anki ağ üzerindeki gecen paketlerin tam alan isimlerini çözerek zaman damgasını, kullanılan protokolü vb. gösterir. Başlığın geri kalan kısımlarını göstermez.

Biz bunun çıktısını almak istersek klasik unix kuralları gereği redirection yaparak bunu bi dosya içerisine kaydedebiliriz :

```
# tcpdump > tcpdump.dosyasi
```

Yapılanların çıktısını hem görmek hamde kaydetmek istersek :

```
# tcpdump -l | tee textfile
```

Dosyaya kaydetme işini tcpdump -w parametresi kullanarak kendi binary formatında kaydedebiliriz. & ile bunu arkaplanda yapıyoruz.

```
 # tcpdump -w tracefile &
```

Burada hem daha az işlem zamanı ve daha az yer işgal etmiş olacak. Fakat bu paketleri acıp okuyamayız okumak için -r parametresinden sonra dosyanın adını yazmalıyız

```
# tcpdump -r tracefile
```

tcpdumpta gösterilen paketin içeriği

Kullanılan parametreye göre paket içerisinde görünen kısımlarda değişmektedir örneğin

```
#tcpdump -q -e -c 5
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
17:35:52.178642  00:0c:29:13:62:10  (oui  Unknown)  >  00:e0:50:00:06:62  (oui
Unknown), IPv4, length 170: 192.168.1.6.ssh > 192.168.1.2.3574: tcp 116
17:35:52.180269  00:e0:50:00:06:62  (oui  Unknown)  >  00:0c:29:13:62:10  (oui
Unknown), IPv4, length 106: 192.168.1.2.3574 > 192.168.1.6.ssh: tcp 52
17:35:52.180268  00:0c:29:13:62:10  (oui  Unknown)  >  00:e0:50:00:06:62  (oui
Unknown), IPv4, length 54: 192.168.1.6.ssh > 192.168.1.2.3574: tcp 0
17:35:52.180269  00:c0:49:d3:48:2c  (oui  Unknown)  >  00:e0:50:00:06:62  (oui
Unknown), IPv4, length 90: 205-252-0-11.btnaccess.net.4674 > 192.168.1.2.4510:
UDP, length 48
17:35:52.180269  00:e0:50:00:06:62  (oui  Unknown)  >  00:c0:49:d3:48:2c  (oui
Unknown), IPv4, length 69: 192.168.1.2.4510 > 83-148-204-81.dyndsl.ssp.fi.4672:
UDP, length 27
5 packets captured
80 packets received by filter
0 packets dropped by kernel
```

-e parametresiyle mac adresi (kaynak ve hedef adreslerini) ve c parametresi ile kac paketin yakalanacağını belirtiyoruz.

şimdi birde paketin hexadecimal olarak cıktısını görmek istersek parametre olarak -x kullancağız

```
# tcpdump c1 -x

1023873914.125606 192.168.1.6.ssh > 192.168.1.2.2659: P
3066603742:3066603806(64) ack 1646168027  win 17520 [tos 0x10]
                          4510 0068 7e87 4000 4006 3862 c0a8 011e
                          c0a8 0128 0016 0479 b6c8 a8de 621e 87db
                          5018 4470 1813 0000 e492 152f 23c3 8a2b
                          4ee7 dbf8 0d48 88e8 0110 2b01 4295 39f4
                          52c9 a05b 31d7 e3ae 1c62 2dbd d955 d604
                          b5d2 63d1 8fbc 4ab7 1615 b382 571c 70e0
                          a368 a03f 425b 6211
```

burada bulunan renklere göre paket içeriği

- **Siyah kısım** ağ kartımızdan geçen paketlerin zamanın belirtir.
- **Koyu mavi kısım** İletişimin geçtiği yerdeki Kaynak – kaynak port, Hedef - Hedef port belirtir
- **Kırmız kısım** TCP *Bayraklar'ıdır.*
- **Kahverengi kısım** Sıra- aralık
- **Açık Mavi kısım** Çerçeve büyüklüğünü
- **Yeşil kısım** TCP servis tipini (type of service) belirtir.

Not: Paketin yapısı ve bilgisi paketin doğasına bağlıdır yani buradaki gibi paket TCP , SSH oturumunun ortasında yapılan veriler gibi. Daha fazla bilgi icin RFC793 ve RFC791 e müracaat ediniz

TCP bayraklarına genel olarak bakarsak:

TCP Bayrağı	**tcpdump'ta karşılığı**	**Bayrağın anlamı**
SYN	s	Syn paketi, oturum kurma isteğidir. TCP bağlantılarının ilk kısmını oluşturur.
ACK	ack	Kabul bayrağı.Diğer bayraklarla birleşerek görünebilir.
FIN	f	Oturum sonlandırmak icin kullanılır.
RESET	r	Baglantıyı resetlemek yarıda için kullanılır.
PUSH	p	Verinin acilen iletilmesini sağlar. Telnet gibi uygulamalarda ana unsur acil cevap süresidir ki buda PUSH bayrağı sinyali ile olur
URGENT	urg	Acil olan verinin diger verilerden önce yapılmasını sağlar Ctrl-C FTP download kesmesi
Placeholder	.	Bağlantı syn, finish, reset, push bayrakları ile

		ayarlanmamışsa ayıraç hedef port tan sonraya gelir.bazen ack bayrağıyla beraber kullanılır

En son ve işe yarar olarak paketin veri kısmının ASCII formatına dönüştürmek istersek X parametresini kullanmamız gerekir

```
# tcpdump -qec1xX

1023873914.125606 192.168.1.6.ssh > 192.168.1.2.2659: P
3066603742:3066603806(64) ack 1646168027  win 17520 [tos 0x10]
       4500 0054 7e87 4000 4006 3862 c0a8 0106    E...LP@.@.j.....
       c0a8 0128 0016 0df6 fb50 488a 4e8b 0074    .........PH.N..t
       5018 3ed0 892b 0000 0a15 8f5b b4e2 f68a    P.>..+.....[....
       50df 609f 37a7 4322 d814 48bd fe05 0a13    P.`.7.C"..H.....
       5893 d8b6 e7ce 12a3 f24e a6f1 3fcf 05ac    X........N..?...
       dc53                                       .S
```

Paketin içeriğini byte byte derinlemesine incelemek gerekirse

4500

4 IP V4 paketi

00 ayrılmıs servis kod noktası QoS için kullanılır buradaki deger 0 dır

0054 datagram ın boyutu

7e87 Tanımlama bilgisi

40 Bayraklar
00 Fragment offset

40 TTL (time to live)
06
Protokol
1=ICMP
2=IGMP
6=TCP
17=UDP

3862 Baslık sağlama toplamı (checksum)

c0a80106
Kaynak Adresi (IP)
c0 = 192 (16*12)
a8 = 168 (16*10+8)
01 = 1 (16*0+1)
06 = 6 (16*0+6)
buradaki kaynak adres : 192.168.1.6

c0a80102
Hedef Adresi (IP)
buradaki hedef adres : 192.168.1.2

00	
16	22 (1*16+6) burada paketin ssh paketi olduğu görülüyor
0df6	Sağlama Toplamı
fb50	Tanımlayıcı
488a	Dizi numarası

şeklinde paket içersindeki hexadecimal olarak gösterilen bölümlerin ne anlama geldiğini gördük.

Adres filterleme

- Buradaki adres bir host, network, multicast/broadcast, yada mac/ethernet adresi olabilir
- Adres kaynak yada hedef olabilir.

Şimdi exik adlı bilgisayar uzerindeki tüm trafiği görüntülemek için

```
# tcpdump host exik
```

yada şöyle kullanırsak exik adlı bilgisayarın hedef makinemiz olduğunu düşünelim

```
# tcpdump dst host exik
```

yada ethernet adresini kullanarakta trafigi izleyebiliriz

```
# tcpdump ether src host 0:a0:3b:3:e1:1d
```

Protokol filtereleme

- Beş temel protokol isimleriyle beraber belirtilmiştir :
  - ip
  - tcp
  - udp
  - icmp
  - igmp

- Diğer protokolleri numaralarıyla dinleyebiliriz

```
# tcpdump udp
```

Göründüğü gibi bütün udp paketleri dinlenmektedir. Işi biraz daha özelleştirirsek diyelim ki 1000 adet udp paketinin her bir paketin 300 bytelık verisini bir dosyaya binary olarak kaydeden komut için aşağıdakini kullanbiliriz :

```
# tcpdump -c 1000 -s 300 -w udp1000hits.dump udp
```

yukarıdaki 5 protokol dışındakileri numarasıyla belirterek yakalyabiliyorduk.

```
# tcpdump proto 8
```

ile /etc/protocols içerisinde görüldüğü gibi egp paketlerini yakalayabiliriz.**proto**

Port filtreleme

Buradada yine protokol filtrelemedeki gibi /etc/services içerisinde bulunan port numaralarına göre paketleri yakalayabiliriz.

tcpdump port 23

ile telnet paketlerini dinleyebiliriz.

Paket karakteristik filtreleme

Yine aynı yolu izleyerek filtreleme yapabiliriz.

Kullanılabilinen protokoller: ip, tcp, udp, icmp, ether, arp, rarp, and fddi.

Mesela icmp trafiğini dolaylı bir yolla yapmak istersek şu şekilde yapabiliriz.

```
# tcpdump "ip[9]=1"

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

19:24:42.107912 IP 192.168.1.2 > 192.168.1.6: ICMP echo request, id 1024,
seq 1280, length 40
19:24:42.108839 IP 192.168.1.6 > 192.168.1.2: ICMP echo reply, id 1024, seq
1280, length 40
19:24:42.686771 IP 192.168.1.2 > 192.168.1.6: ICMP echo request, id 1024,
seq 1536, length 40
19:24:42.686878 IP 192.168.1.6 > 192.168.1.2: ICMP echo reply, id 1024, seq
1536, length 40
19:24:43.222077 IP 192.168.1.2 > 192.168.1.6: ICMP echo request, id 1024,
seq 1792, length 40
19:24:43.222186 IP 192.168.1.6 > 192.168.1.2: ICMP echo reply, id 1024, seq
1792, length 40
19:24:43.835688 IP 192.168.1.2 > 192.168.1.6: ICMP echo request, id 1024,
seq 2048, length 40
19:24:43.835705 IP 192.168.1.6 > 192.168.1.2: ICMP echo reply, id 1024, seq
2048, length 40
```

Gördüğümüz gibi çalıştı. Çünkü IP başlığının 10. byte'ı protokolü belirtir. Ayrıca biz biliyoruz ki icmp protokol numarası 1 dir. Bu kısmı biraz daha açarsak. Protokollerin ve başlıkların içeriğini derinlemesine bakarsak.

Paketlerin içeriğini belirten kullanımlar

İfade	Anlamı
`[x:y]`	Paketin basında itibaren x offsete kadar basla ve y byte oku
`[x]`	[x:1] için kısaltma
`proto[x:y]`	Proto isimli başlıkta x ten baslıyarak y kadar byte oku
`p[x:y] & z = 0 p[x:y]`	z tarafından hiçbir bit seçilmemiş
`p[x:y] & z != 0 p[x:y]`	z tarafından herhangi bir bit seçilmiş
`p[x:y] & z = z p[x:y]`	z tarafından bütün bitler seçilmiş

`p[x:y] = z` `p[x:y]` z tarafından sadece seçilmiş

IP Paket içeriği ve açıklaması

`ip[1]`	Servis tipi/QoS/DiffServ
`ip[2:2]`	Datagram toplam boyutu
`ip[4:2]`	IP ID numarası
`ip[6] & 0x80`	Ayrılmış bit
`ip[6:2] & 0x1fff`	Fragment offset
`ip[8]`	Yaşam süresi (Time to live - Ttl)
`ip[9]`	Protokol
`ip[10:2]`	Başlık checksum
`ip[12:4]`	Kaynak IP
`ip[16:4]`	Hedef IP

ICMP Mesajları ve içeriği

`icmp[0]`	Tip
`icmp[1]`	Kod
`icmp[2:2]`	Sağlama toplamı
`icmp[4...]`	Payload

UDP başlık kısımları

`udp[0:2]`	Kaynak port
`udp[2:2]`	Hedef port
`udp[4:2]`	Datagram boyutu
`udp[6:2]`	UDP sağlama toplamı

TCP başlık kısımları

`tcp[0:2]`	Kaynak port
`tcp[2:2]`	Hedef port
`tcp[4:4]`	Sıra number
`tcp[8:4]`	Onay numarası
`tcp[12]`	Başlık boyutu
`tcp[13]`	TCP bayrakları
`tcp[14:2]`	Çerçeve boyutu
`tcp[16:2]`	Sağlama toplamı
`tcp[18:2]`	Acil işaretçisi
`tcp[20..60]`	Veri yada seçenekler

Bayraklar	Sayısal olarak	Anlamı
---- --S-	0000 0010 = 0x02	normal syn
---A –S-	0001 0010 = 0x12	normal syn-ack
---A ----	0001 0000 = 0x10	normal ack
--UA P---	0011 1000 = 0x38	psh-urg-ack. Shh gibi uygulamarda
---A -R--	0001 0100 = 0x14	rst-ack.

---- --SF	0000 0011 = 0x03	syn-fin tarama
--U- P--F	0010 1001 = 0x29	urg-psh-fin. Nmap fingerprint paketi
-Y-- ----	0100 0000 = 0x40	hiçbir şey >= 0x40 ayrılmış bit seti
XY-- ----	1100 0000 = 0xC0	Ayrılmış bit seti
XYUA PRSF	1111 1111 = 0xFF	FULL_XMAS scan

ICMP tipleri ve kodları

0	ECHOREPLY
3	UNREACHABLE
3:0	NET
3:1	HOST
3:2	PROTOCOL
3:3	PORT
3:4	NEEDFRAG
3:5	SRC_ROUTE_FAILED
3:6	NET_UNKNOWN
3:7	HOST_UNKNOWN
3:8	SRC_HOST_ISOLATED
3:9	NET_PROHIB
3:10	HOST_PROHIB
3:11	BAD_TOS_FOR_NET
3:12	BAD_TOS_FOR_HOST
3:13	FILTER_PROHIB
3:14	HOST_PRECEDENCE_VIOLATION
3:15	PRECEDENCE_CUTOFF
4	SOURCEQUENCH
5	REDIRECT
5:0	NET
5:1	HOST
5:2	TOSNET
5:3	TOSHOST
8	ECHO
9	ROUTERADVERT
10	ROUTERSOLICIT
11	TIME_EXCEEDED
11:0	IN_TRANSIT
11:1	DURING_FRAG_REASSEMBLY
12	PARAMETER_PROBLEM
12:1	MISSING_OPT_FOR_REQUEST
13	TSTAMP_REQ
14	TSTAMP_REPLY
15	INFO_REQ
16	INFO_REPLY
17	NETMASK_REQ
18	NETMASK_REPLY

Aşağıdaki örnekleri tcpdump filtrelemesi için olusturulan Filtre dosyası olarak kaydedebilir ve bunları tcpdump'ta parametre kullanmaksızın (sadece -F parametresi kullanmanız gerekiyor) yapmak istediklerinizi yapabilirsiniz. Yani parametreleri hepsini dosyaya yazarak oradan okuyor.

```
#tcpdump -F filtre.dosyasi
```

Tcpdump filtre Örnekleri

TCP

1.Bayrakların durumlarına gore

PUSH bayrağıyla seçilmiş bütün paketler

```
tcp[13] & 8 != 0
```

RST bayrağıyla seçilmiş bütün paketler

```
tcp[13] & 4 != 0
```

Hiçbir bayrak seçilmemiş boş paket

```
tcp[13] & 0x3f = 0
```

SYN-FYN

```
tcp[13] = 3
```

SYN-FYN heriki bayrak birden seçilmiş

```
(tcp[13] & 0x03) = 3
```

Sadece SYN

```
tcp[13] & 0x02) != 0
```

Ayrılmış bitler

```
tcp[14] >= 64
```

2.Servis tiplerine gore :

`ssh filtreleme :`

```
tcp[(tcp[12]>>2):4] = 0x5353482D && (tcp[((tcp[12]>>2)+4):2] = 0x312E || \
       tcp[((tcp[12]>>2)+4):2] = 0x322E)
```

ftp filtreleme

```
tcp[(tcp[12]>>2):4] = 0x3232302d || tcp[(tcp[12]>>2):4] = 0x32323020
```

telnet filtreleme

```
tcp[2:2] = 23
```

Yine telnet fakat baska bir yolla

```
(tcp[(tcp[12]>>2):2] > 0xfffa) && (tcp[(tcp[12]>>2):2] < 0xffff)
```

SMB

```
dst port 139 && tcp[13:1] & 18 = 2
```

Portlara göre

Hedef portları 1024 ten küçük olanlar

```
tcp[2:2] < 1024
```

DNS zone transfer

```
tcp && dst port 53
```

X11 portları

```
(tcp[2:2] >= 6000) && (tcp[2:2] < 7000)
```

Http istekeri

```
(tcp[13:1]&18 = 2) && (port 80) && (ip dst 192.168.1.40)
```

RIP bilgisi

```
-s 1024 port routed
```

IP

Payload'ı 20 byte'tan fazla olan paketler

```
(ip[2:2] - ((ip[0]&0x0f)<<2) - (tcp[12]>>2)) <= 20
```

TTL değeri 5 'ten küçük olan paketler

```
ip[8] < 5
```

IP seçenekleri

```
(ip[0] & 0x0f) != 5
```

Broadcast mesajı xxx.xxx.xxx.255 || xxx.xxx.xxx.0

```
 (ip[19]=0xff) || (ip[19]=0x00)
```

IPv4 olmayan diğer IP versiyonları

```
ip && (ip[0] & 0xf0 != 0x40)
```

Unroutable addresler

```
not ((ip[12] < 3) || net 5 || net 10 || net 127 || net 172.16 \
       || net 192.168 || (ip[12] > 239))
```

ICMP

fragmentasyon gerekli fakat DF

```
(icmp[0] = 3) && (icmp[1] = 4)
```

fragmented ICMP

```
icmp && (ip[6:1] & 0x20 != 0)
```

ping beklenen bütün ICMP paketleri

```
icmp && icmp[0] != 8 && icmp[0] != 0
```

UDP

Teardrop atağı

```
udp && (ip[6:1] & 0x20 != 0)
```

500 udp portuna gelen herseyi yakala

```
-n -vv udp && dst port 500
```

Udp paket uzunluğuna uymayan paketleri yakalama

```
(udp[4:2] < 0) || (udp[4:2] > 1500)
```

Udp port tarama

```
udp && src port = dst port
```

iki bilgisayar arasında ping çektikten sonra sadece echo replay mesajlarını yakalamak istiyorsak (192.168.1.2 > 192.168.1.6)

```
#tcpdump -x "(icmp[0]=0)"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
22:57:02.452145 IP 192.168.1.6 > 192.168.1.2: ICMP echo reply, id 1024, seq
2304, length 40
 0x0000:  4500 003c 5965 0000 4001 9e03 c0a8 0106
 0x0010:  c0a8 0102 0000 485c 0400 0900 6162 6364
 0x0020:  6566 6768 696a 6b6c 6d6e 6f70 7172 7374
 0x0030:  7576 7761 6263 6465 6667 6869
```

Birleşik filtreleme

Birleşik filtrelemeden kasıt birçok filtre seçeneğini tcpdump' ta aynı komut içerisinde kullanarak gerçekleşmesidir. Burada hangi özelliklerin ön plana çıkmasını hangilerini istemediğimizi belirtebiliriz. Bunlar için kullanabileceğimiz seçenekler

- NOT
- OR
- AND

Buraya Kadar yaptığımız örneklerde hep ssh bağlantısı üzerinden gerçekleştirdik örnekleri. Şimdi var olan ssh baglantısı dışındaki trafiği göstermek istersek

```
# tcpdump -w tracefile not port 22
```

Fakat bekleyin burada bi hata var. şimdi şöyle düşünelim ya o ağ içerisinde bizim dışımızda başka birisi ssh bağlantısı kullanmaktaysa. O zaman bu seçeneğin içerisine hangi hostlar arasında

gerçekleştiğinide belirtmemiz lazım

```
# tcpdump -w tracefile not port 22 and host exik and host ubuntu
```

Fikir güzel fakat sözdizimi yanlış. Doğru komut

```
# tcpdump -w tracefile not "(port 22 and host exik and host ubuntu)"
```

Yukarıda da bahsettiğimiz gibi boyle birleşik komutları filtre dosyalarına kaydedersek işimiz daha kolaylaşır. Filtre dosyasından okumak için -F parametresini kullanacağımızı belirtmiştik.

```
# cat > filterfile
dst host exik and "(udp or proto 51)" and not "(src host 192.168.1.2 or src host
192.168.1.6)"
Ctrl-D

# tcpdump -F filterfile
```

Yukarıdaki komut kaynak ip ler 192.168.1.2 veya 192.168.1.6 olanlar hariç hedef host exik olan bütün udp trafiğini gösterir.
Buradaki trafiği sıkıştılımış olarak kaydetmek istersek

```
#gunzip -c Filter.gz | tcpdump -F filterfile
```

Filter.gz şeklinde sıkıştırlmış olarak kaydedebiliriz

Şimdide tcpdump ile beraber çalışabilen bi kaç küçük aracı inceleyelim.

1. Tcpdfilter

www.packetstormsecurity.org/ sitesinden indirebileceğiniz bu araç sayesinde tcpdump çıktısını daha anlaşılabilir hala getirebilirsiniz.
Aşağıdaki eth0 arayüzünde echo replay mesajını daha okunabilir bir formatta görmek için

```
#tcpdump -i eth0 -x -l "(icmp[0]=0)" | tcpdfilter -x -d
03:33:19.730266 IP 192.168.1.6 > 192.168.1.2: ICMP echo reply, id 1024, seq
5888, length 40

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
IP packet header
IP Version : 0
IP Header Length (32 bit quantities) : 0
Service Type
        Precedence : 0
        Type of Service : 0
IP Packet Length (bytes) :  56
Packet Identifier : 0
Flags
        Allow Fragmentation? : Yes
        Last Fragment? : Yes
```

```
Fragment Offset : 0
Time to Live (TTL) : 64
Protocol Number : 0
Source IP Address : 192.168.1.2
Destination IP Address :  192.168.1.6
----------------------------------------
        0x0000:  4500 003c fa85 0000 4001 fce2 c0a8 0106
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
IP packet header
IP Version : 0
IP Header Length (32 bit quantities) :
Service Type
        Precedence : 0
        Type of Service : 0
IP Packet Length (bytes) :  56
Packet Identifier : 0
Flags
        Allow Fragmentation? : Yes
        Last Fragment? : Yes
Fragment Offset : 0
Time to Live (TTL) : 64
Protocol Number : 0
Source IP Address :  192.168.1.2
Destination IP Address :  192.168.1.6
----------------------------------------
        0x0010:  c0a8 0102 0000 3a5c 0400 1700 6162 6364
```

2. Sniff

http://www.thedumbterminal.co.uk sitesinden indirebileceğimiz sniff ise tcpdump çıkışını renklendirerek okunabilirliği artırmaktadır.

```
# ./sniff  "(icmp[0]=0)"
Running tcpdump with the following options: -lnx -s 1024
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1024 bytes

------------------------------------------------------------
2005-12-16
03:39:33.428213
IP 192.168.1.2.1521 > 192.168.1.6.22
P1:S(R) k<bz0x0:  4T¥y5$0x@:  4K:
+(?:  PJD
0xP:  3<~
```

2005-12-16
03:39:33.431081
IP 192.168.1.6.22 > 192.168.1.2.4299
0x : PPw{z"70x0: u0x@: G"Q`|#F0xP: ¥e 0x: ^}r~
0x`: SR¤,0xp: !8,3u0x: lt1T]

2005-12-16
03:39:33.431535
IP 192.168.1.2.4299 > 192.168.1.6.22
CG0x : P%d$00x: E(O6@(A0x: r

Yardımı çalıştırmak için asagıdaki komutla baslayabiliriz.

```
./sniff -h
```

sniff seçenekleri tcpdump seçeneklerinden -- ile ayrılmalıdır. Örnek kullanımı:

eth0 a gelen bütün FTP bağlantılarını yakalamak için **:**

```
./sniff -c -- -i eth1 tcp port 21
```

gördüğünüz gibi tcpdump seçeneklerini "--" den sonra kullanabiliyoruz.

CSV dosyasında gerçek zamanlı log tutma
Aşağıdaki örnek şu seçenekleri kullanır:

verileri " ile bitirmek için (-e\") kullanılır.
Satırları , ile ayırmak için (-n,) kullanılır.
Sadece bir kez ayıraç kullanmak istiyorsak (-t0)
Renk kullanmak istemiyorsak (-c)

```
./sniff -e\" -n, -s -t0 -c > /tmp/dump.csv
```

tcpdump dumb dosyasını CSV formatına dönüştürmek

İlkönce tcpdump ' ı "-lx -s 1024" kullanarak dumb file olusturun. Örnek olarak

```
tcpdump -lx -s 1024 -w /tmp/dump.txt
```

Dahsa sonra sniff i gerekli seçeneklerle çalıstırın. Tabi burada tcpdump tarafından oluşturulan dosyayı okuyarak devam ediyoruz.

```
./sniff -e\" -n, -s -t0 -c -- -r /tmp/dump.txt > /tmp/dump.csv
```

NGREP

Ngrep i sadece root hesabıyla çalıştırabiliriz. Eğer herhangi bir seçenek belirtmeseniz bütün trafiği dinleyecektir. Bizim yapmak istediğimiz tabiki bu değildir. Mesela google ile arama yapanları listelemek istersek işimiz daha kolaylaşacaktır.

```
# ngrep google port 80
interface: wlan0 (192.168.0.0/255.255.255.0)
filter: ip and ( port 80 )
match: google
#########################
T 192.168.0.100:33020 -> 216.239.39.99:80 [AP]
GET / HTTP/1.1..Host: google.com..User-Agent: Mozilla/5.0 (X11; U; Linux i6
86; en-US; rv:1.7.6) Gecko/20050419 OpenLX/1.7.6-1.olx..Accept: text/xml,ap
plication/xml,application/xhtml+xml,text/html;q=0. 9,text/plain;q=0.8,image/
png,*/*;q=0.5..Accept-Language: en-us,en;q=0.5..Accept-Encoding: gzip,defla
te..Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7..Keep-Alive: 300..Connec
tion: keep-alive..Cookie: PREF=ID=6bfa2ae9c8bf1894:CR=1:TM=1118348709:LM=11
18400738:GM=1:S=NWBKfMYi55QzWD_y....
```

işareti o anki trafik içerisinde eşleşmeyen durumu bildirmektedir

şimdide internet bağlantısını kötüye kullananları tespit etmek istersek

```
# ngrep -i 'game*|chat|' -W byline > bad_user.txt
```

burada içerisinde game chat geçen kullanıcıları baduser.txt dosyası içersine kaydediyoruz. " | " ile her bir kelimeyi ayırıyoruz. –i ile aramayı küçük büyük harflere karsı duyarsız yapıyoruz.

-W ile dosyaya yazıyoruz

Örnek çıktısı:

```
interface: wlan0 (192.168.0.0/255.255.255.0)
match: game*|chat|recipe
##############################
T 192.168.0.100:33035 -> 66.249.85.104:80 [AP]
GET /search?hl=en&safe=off&q=online+games&btnG=Search& meta= HTTP/1.1.
Host: www.google.co.in.
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.6) Gecko/20050419
OpenLX/1.7.
6-1.olx.
Accept: text/xml,application/xml,application/xhtml+xml,tex
t/html;q=0.9,text/plain;q=0.8,
image/png,*/*;q=0.5.
Accept-Language: en-us,en;q=0.5.
Accept-Encoding: gzip,deflate.
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7.
Keep-Alive: 300.
Connection: keep-alive.
Referer: http://www.google.co.in/search?hl=en&q=hello&btnG= Google+Search&meta=.
Cookie: PREF=ID=7c5bf916f28d16c7:FF=4:LD=en:NR=10:TM=11183
48709:LM=1118348731:S=20ZkQG0Y
sMDDsXsW.
```

şimdide o anda yapılan bir email işleminin göstermek ve adresini belirtmek için

```
# ngrep -i 'rcpt to|mail from' tcp port smtp

interface: wlan0 (192.168.0.0/255.255.255.0)
filter: ip and ( tcp port smtp )
match: rcpt to|mail from
T 192.168.0.100:1043 -> 200.40.174.30:25 [AP]
MAIL From: SIZE=192..
T 192.168.0.100:1043 -> 200.40.174.30:25 [AP]
RCPT To:..
```

ngrep port adreslerini /etc/services dosyasında ki port isimleriyle eşleştirerek çözebilir.
mesela 25 smtp olarak algılar.
Eğer zaman damgası eklemek istersek –t parametresini kullanabiliriz.

```
# ngrep -q -t -wi "login" port 23
```

Bu komutla telnet trafiğini 23 port ile ve “login” kelimesiyle ve zaman damgasının formatı olarakta YYYY/AA/GG SS:DD:SS.UUUUUU cinsinden görebiliriz.

Şimdide 53. (DNS) port üzerindeki bütün trafiği zaman damgasıyla beraber pcap dosyası içerisine yazmak istersek şu komutu kullanabiliriz. Buradaki –d parametresiyle makine üzerindeki bütün ara birimleri (eğer makine üzerinde birden fazla ara yüz cihazı varsa) dahil ediyoruz.

```
# ngrep -O ~/logs/traffic.dump -d any -T port 53
```

Eğer sakladığımız pcap dosyası içerisinde bir şeyler aramak istersek –I parametresini kullanabiliriz.
Mesela aşağıdaki komutla içerisinde .net geçen bütün domainleri bulabiliriz

```
# ngrep -tv '*.net' -I ~/logs/traffic.dump
```

P0f (passive OS fingerprinting)

Paketleri inceleyerek hangi işletim sistemi olduğunu bulmak için kullanılır. http://lcamtuf.coredump.cx/p0f.shtml adresinden indirilebilir. Bunu yaparken `3 farklı test uygulayarak işletim sisteminin tipini belirleyebilir.`

1. Varsayılan olarak Syn paketkerini kullanır. Yerel ağa gelen SYN paketlerine bakar

2. -A parametresi kullanılarak SYN+ACK aktif hale getirir. Bu testte p0f uzaktaki açık portlardan dönen cevaplara göre işletim sisteminin tipine karar verir.

3. -R parametresi iste RST +ACK testini aktif hale getirir. Ki bu testte ise p0f RST + ACK paketleri ile kapalı portlardan dönen cevaplara bakar

P0f üç testi iki modda yapar. Ilk mod varsayılan olarak gelen moddur. Bunda ağ kartı üzerinden gerçek zamanlı işlem yapmaktadır. Aşağıdaki gibi p0f i 22. port dinleyecek şekilde açarsak uzaktaki bilgisayarlardan gelen SYN paketlerini dinlemeye baslar.

```
# p0f -i fxp0 'port 22'

p0f - passive os fingerprinting utility, version 2.0.3
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns
  <wstearns@pobox.com>
p0f: listening (SYN) on 'fxp0', 206 sigs (12 generic), rule:
  'port 22'.
10.10.10.2:57868 - HP-UX 11.00-11.11
  -> 172.27.20.3:22 (distance 1, link: ethernet/modem)
10.10.10.3:1085 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
  -> 172.27.20.3:22 (distance 1, link: ethernet/modem)
10.10.10.5:32805 - Solaris 8 (1)
  -> 172.27.20.3:22 (distance 1, link: ethernet/modem)
172.27.20.11:44521 - OpenBSD 3.0-3.4 [high throughput]
  (up: 5613 hrs)
  -> 172.27.20.3:22 (distance 0, link: ethernet/modem)
192.168.60.3:34744 - Linux 2.4/2.6 [high throughput]
  (up: 532 hrs)
  -> 172.27.20.3:22 (distance 1, link: ethernet/modem)
172.27.20.5:1376 - FreeBSD 4.6-4.8 [high throughput]
  (up: 26 hrs)
  -> 172.27.20.3:22 (distance 0, link: ethernet/modem)
```

Syn testinde p0f bazı özel versiyonlar hariç bütün işletim sistemlerini bulabilmektedir

aşağıdaki komutta ise SYN + ACK testi uygulayacağımızı belirtiyoruz. Uzaktaki makinanın 22 ve 139. portlarından SYN+ACK paketlerini arıyoruz.

```
# p0f -i eth0 -f p0fa.fp -A 'port 22 or port 139'

p0f - passive os fingerprinting utility, version 2.0.3
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns
```

```
  <wstearns@pobox.com>
p0f: listening (SYN+ACK) on 'fxp0', 57 sigs (1 generic), rule:
  'port 22 or port 139'.
10.10.10.2:22 - UNKNOWN [32768:63:1:60:M1460,W0,N,N,N,T:AT:?:?]
  (up: 446 hrs)
  -> 172.27.20.3:4492 (link: ethernet/modem)
10.10.10.3:139 - UNKNOWN [S12:127:1:60:M1460,N,W0,N,N,T0:A:?:?]
  -> 172.27.20.3:4493 (link: ethernet/modem)
10.10.10.5:22 - UNKNOWN [24616:63:1:60:N,N,T,N,W0,M1460:AT:?:?]
  (up: 446 hrs)
  -> 172.27.20.3:4494 (link: ethernet/modem)
172.27.20.5:22 - FreeBSD 4.6-4.8 (RFC1323) (up: 27 hrs)
  -> 172.27.20.3:4496 (distance 0, link: ethernet/modem)
172.27.20.11:22 - UNKNOWN [17376:64:1:60:M1460,N,W0,N,N,T:AT:?:?]
  (up: 5613 hrs)
  -> 172.27.20.3:4497 (link: ethernet/modem)
192.168.60.3:22 - UNKNOWN [5792:63:1:60:M1460,N,N,T,N,W0:ZAT:?:?]
  (up: 532 hrs)
  -> 172.27.20.3:4498 (link: ethernet/modem)
```

Gördüğümüz gibi bu paketlerde birçok sorun vardır. Bundan dolayı SYN+ACK paketleriyle sistem öğrenirken ihtiyatlı olmak gerekir

Son olarakta p0f'i RST+ACk testiyle çalıştıralım. Burada uzaktaki makinanın 81 tcp portunu dinleyelim.

```
# p0f -i fxp0 –f p0fr.fp -R 'port 81'

p0f - passive os fingerprinting utility, version 2.0.3
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns
  <wstearns@pobox.com>
p0f: listening (RST+) on 'fxp0', 46 sigs (3 generic), rule:
  'port 81'.
10.10.10.2:81 - UNKNOWN [0:63:1:51:.:K0AD:?:?] (refused)
  -> 172.27.20.3:4423 (link: unspecified)
```

```
10.10.10.3:81 - Windows XP/2000 (refused)
  -> 172.27.20.3:4424 (distance 1, link: unspecified)
10.10.10.5:81 - FreeBSD 4.8 (refused)
  -> 172.27.20.3:4425 (distance 1, link: unspecified)
172.27.20.5:81 - FreeBSD 4.8 (refused)
  -> 172.27.20.3:4429 (distance 0, link: unspecified)
172.27.20.11:81 - FreeBSD 4.8 (refused)
  -> 172.27.20.3:4427 (distance 0, link: unspecified)
192.168.60.3:81 - Linux recent 2.4 (refused)
  -> 172.27.20.3:4428 (distance 1, link: unspecified)
```

Buradaki sonuçlar SYN + ACK testindeki sonuçlar kadar kötü değildir.

Ikinci mod ise verileri trace dosyasından okumasıdır. -s ile trace dosyasından okunmaktadır – x ile ise paket içeriğini ekrana getirebilmekteyiz.

```
# p0f -s em0.lpc -x | less

p0f - passive os fingerprinting utility, version 2.0.3
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns
  <wstearns@pobox.com>
p0f: listening (SYN) on 'em0.lpc', 206 sigs\
  (12 generic),  rule: 'all'.
[+] End of input file.
192.168.60.3:34720 - Linux 2.4/2.6 (up: 69 hrs)
  -> 10.10.10.3:3389 (distance 1, link: ethernet/modem)
45 00 00 3c 65 91 40 00 3f 06 c5 72 c0 a8 3c 03 |E..<e.@.?..r..<.
0a 0a 0a 03 87 a0 0d 3d 8c 59 87 20 00 00 00 00 |.......=.Y. ....
a0 02 16 d0 93 f8 00 00 02 04 05 b4 04 02 08 0a |................
01 7d e2 b1 00 00 00 00 01 03 03 00             |.}..........
10.10.10.3:1075 - Windows 2000 SP2+, XP SP1 (seldom 98 4.10.2222)
  -> 172.27.20.5:21 (distance 0, link: ethernet/modem)
45 00 00 30 04 15 40 00 80 06 22 86 0a 0a 0a 03 |E..0..@..."".....
ac 1b 14 05 04 33 00 15 8a 33 c3 13 00 00 00 00 |.....3...3......
70 02 40 00 1d 63 00 00 02 04 05 b4 01 01 04 02 |p.@..c..........
```

aşağıda p0f seçenekleri ve bunların ne anlama geldiği belirtilmiştir.

- -w SYN ve SYN ACK paketlerini analizi daha sonra yapmak için libpcap formatında trace dosyasına yazar.

- -o P0f kuralların ASCII metin dosayasına yazar

- -N ise p0f'in sadece ip ve işletim sistemi bilgisini ekrana getirmesini söylüyoruz.

- -D ise işletim sistemi versiyonlarını gizlemektedir.

- -p seçeneğiyle rasgele moduna geçer (promiscuous mode) yani p0f ağ içerisindeki bütün paketlerin içeriğine bakmaya çalışır. Bu seçeneği kullanmassak sadece yerel makina üzerindeki paketleri yakalayıp işletim sistemini bulabilir.

- -d seçeneğini kullanarak p0f'ı servis olarak çalıştırabiliriz -o kullanırsak alınan bu verileri çıkış dosyasına yazabiliriz.

Bâkır EMRE
EMRE@EnderUNIX.org