

BSM 458-AĞ PROGRAMLAMA

Hafta2: PCAP (Packet Capture)

Dr. Öğr. Üyesi Musa BALTA
Bilgisayar Mühendisliği Bölümü
Bilgisayar ve Bilişim Bilimleri Fakültesi

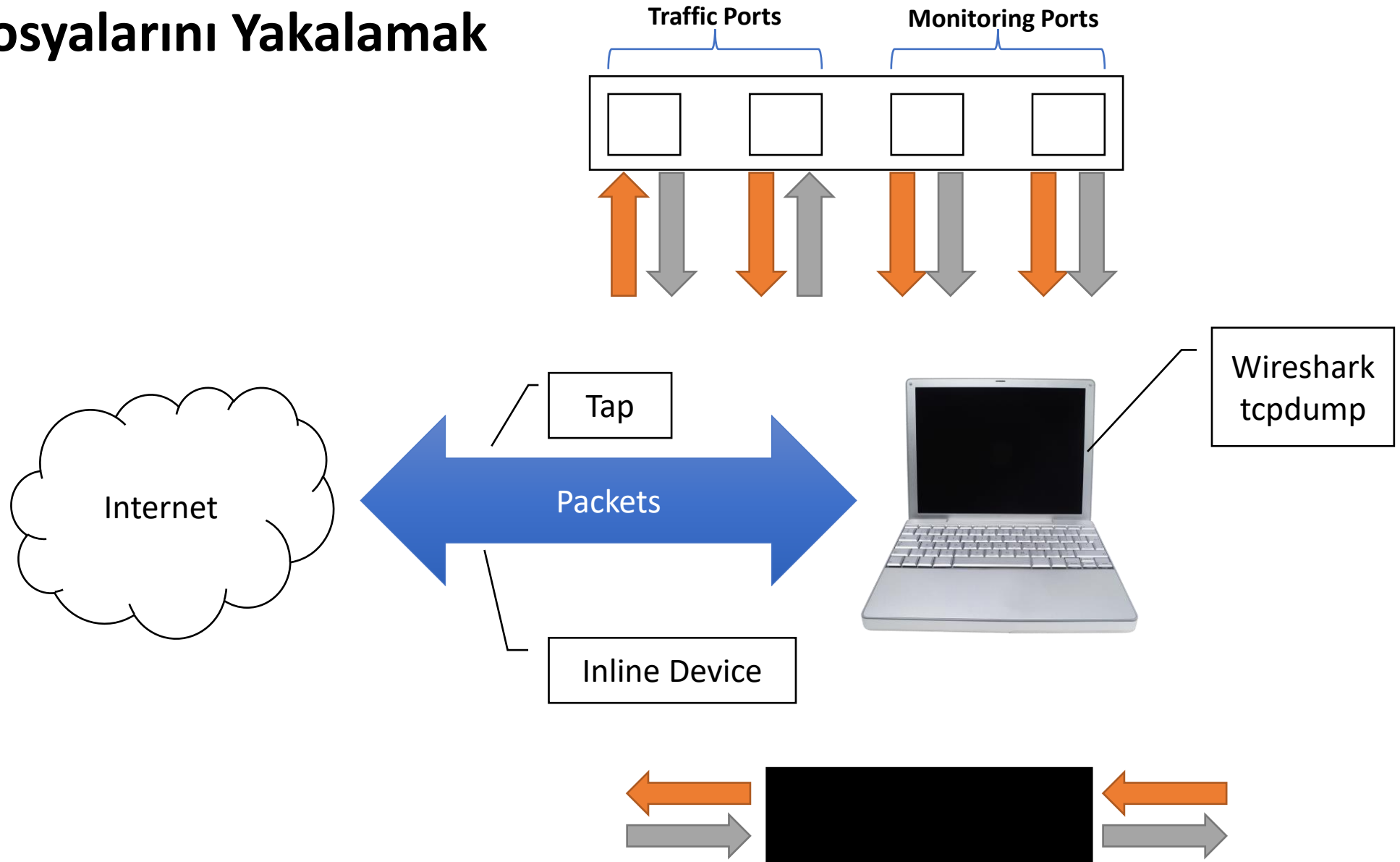
Haftalık İçerik

- PCAP Kavramı
- Libpcap Kavramı
- Ağ Dinleme Araçları
- Libpcap Syntax Yapısı

PCAP Kavramı

- PCAP == **P**acket **C**apture
- Ağ aktivitelerin tüm kaydı
 - Katman 2 – 7
- En yaygın kullanım şekli: `libpcap`
 - Açık-kaynak
 - *nix ve Windows sistemlere uyumlu
 - C kütüphanesi ve birçok dile uyumluluk
 - Others proprietary formats not covered
- Kullanım Alanlar;
 - Ağ trafiğini dinlemek
 - Bant genişliği kullanımı kontrol etme
 - Pasif DNS çözümlemesi
 - Saldırı girişimlerini görüntüleme
 - Araştırma çalışmaları için testler

PCAP Dosyalarını Yakalamak



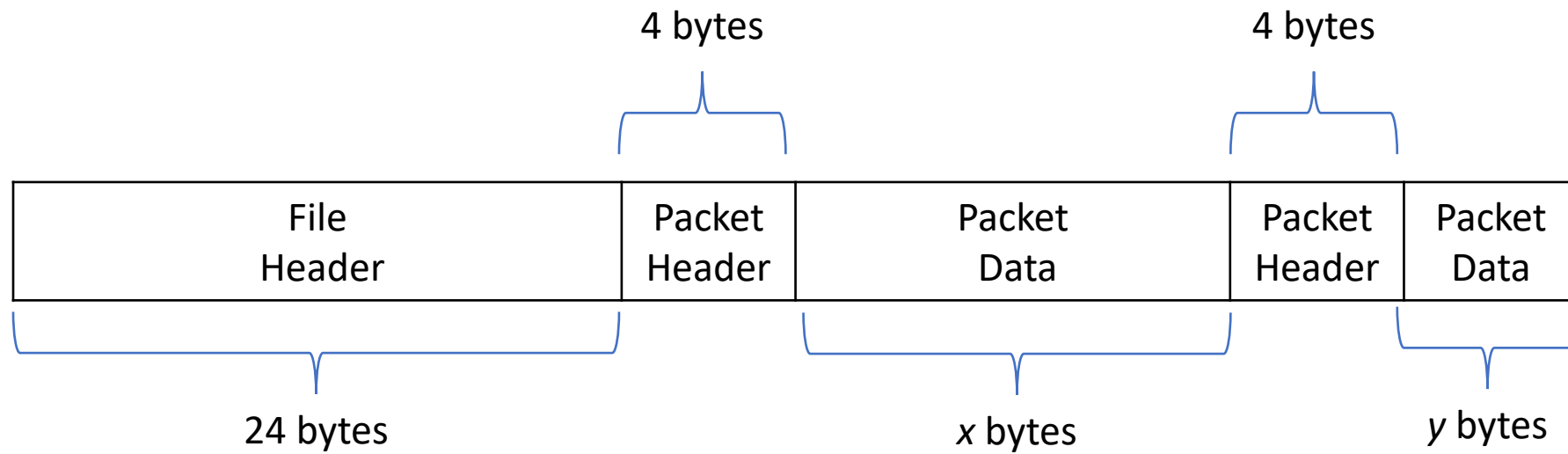
Ham PCAP Depolama

$$\begin{aligned} 1\text{gbps} \times 3600 \times 24 &= 86400 \text{ gigabits} \\ 86400 \div 8 &= 10800 \text{ gigabytes} \end{aligned}$$

$$\begin{aligned} 1\text{gbps} \div (64\text{ bytes} \times 8\text{ bits}) &= \\ 10^9 \text{ bits/s} \div 512 \text{ bits} &= 1953125\text{pps} \end{aligned}$$

$$N \text{ pps} \times H_{\text{PCAP}} = NH \text{ Bps}$$

libpcap Format



libpcap Overhead

Avg. Packet Size (bytes)	Packets Per Second	Overhead (MB/s)	Overhead (GB / day)
64	1,953,125	7.45	628.64
1500	83,333	0.32	26.82
7981	15,662	0.06	5.04
9000	13,888	0.05	4.47

Ağ Dinleme Araçları

- Ağ üzerindeki trafiği sezebilen, çözebilen ve değiştirebilen donanım ve yazılım araçlarının bir kombinasyonudur
 - Pasif izleme (sezme)
 - Aktif (atak yapma)
- Hem ticari hem de ücretsiz sürümleri bulunmaktadır
- Genelde yazılım tabanlıdır
- Sniffer olarak da bilinirler
 - Ağ üzerindeki veriyi pasif olarak izleyen bir programdır
 - Makineniz üzerinde çalışan uygulamalar ve protokoller tarafından gönderilen ya da alınan paketlerin bir kopyasını alır
- Yaygın kullanılan ağ protokolü analiz programları
 - Wireshark , Ethereal , Windump, Ve diğerleri
- Sniffer programlarını efektif bir şekilde kullanmak için iyi bir ağ bilgisine sahip olmak gerekmektedir.

Ağ Dinleme Araçları (devam)

- Sistem yöneticileri
 - Sistem problemlerini ve performansını anlama
 - Saldırıları tespit etme
 - Uygulama operasyonlarını test etme
- Saldırganlar (Kötü niyetli kişiler)
 - Protokoller üzerinden pasif olarak veri toplar
 - FTP, POP3, IMAP, SMTP, rlogin, HTTP, vb.
 - VoIP data
 - Hedef ağın trafiğini keşfeder
 - Trafik desenini keşfeder
 - Ağ içerisine dahil olur (backdoor teknikleri ile)

Ağ Dinleme Araçları (Wireshark)

- Gerald Combs tarafından Ethereal ismi ile başlatılan bir projedir
- İlk versiyonu 1998 yılında yayınlanmıştır, Wireshark ismi haziran 2006'da verilmiştir
- Paket sniffer uygulamasıdır, dolayısıyla bir ağın haritasını çıkartmak için kullanılamaz
- Fonksiyonelliği tcpdump'a çok benzerdir ve diğer bir çok sniffer ile de uyumludur
- Birçok bilgiyi sıralayan ve filtreleyen özelliklere, komut satırına ve grafik arabirimine sahiptir
- 750'nin üzerinde protokol destekler ve bu protokollerin yapısını gösterir
- Kapsüllemeli bir yapıda görünüm sunar ve anlamlarını yorumlar
- Sadece pcap tarafından desteklenen ağlar üzerinde veri yakalama yapabilir
- Açık kaynak kodludur ve farklı işletim sistemleri üzerinde çalışabilir
- İnternet ortamında bir çok online kaynak mevcuttur
- Pasif bir izleme aracıdır, dolayısıyla ağ verisi üretmez

Ağ Dinleme Araçları (Wireshark-WinPcap&libpcap)

Wireshark – Paketleri Snif etmek için uygulama

WinPcap – Paket yakalamak için açık kaynak kütüphanesi

Operating System – Windows & Unix/Linux

Network Card Drivers – Ethernet/WiFi Kartı

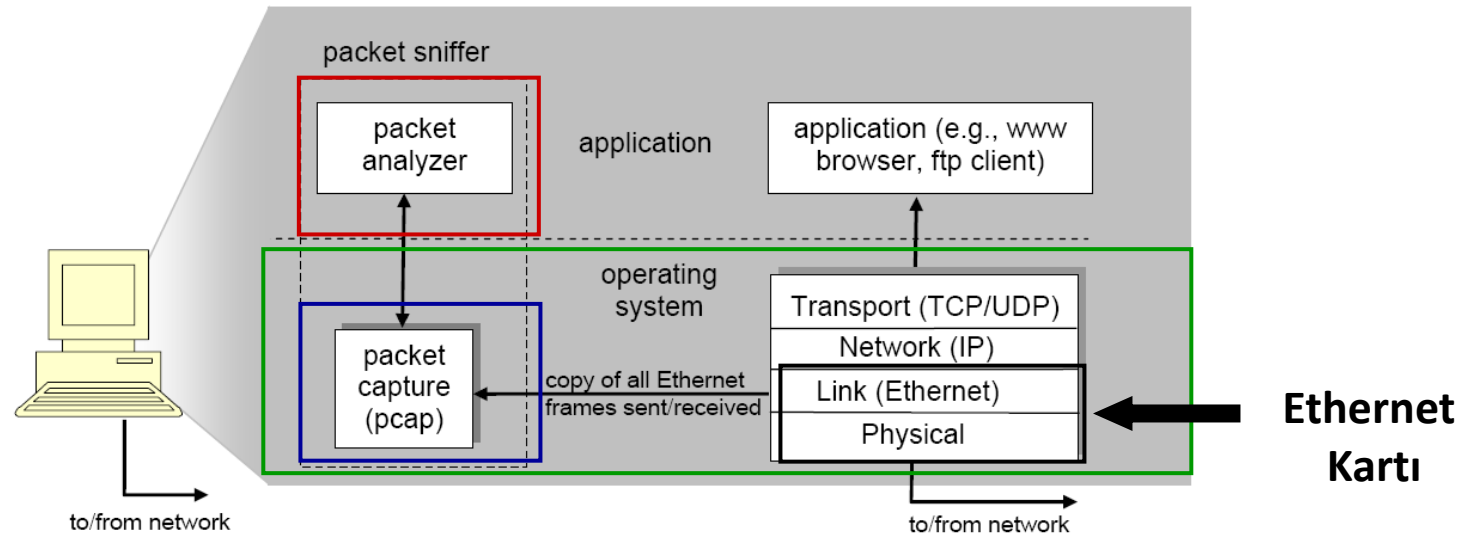
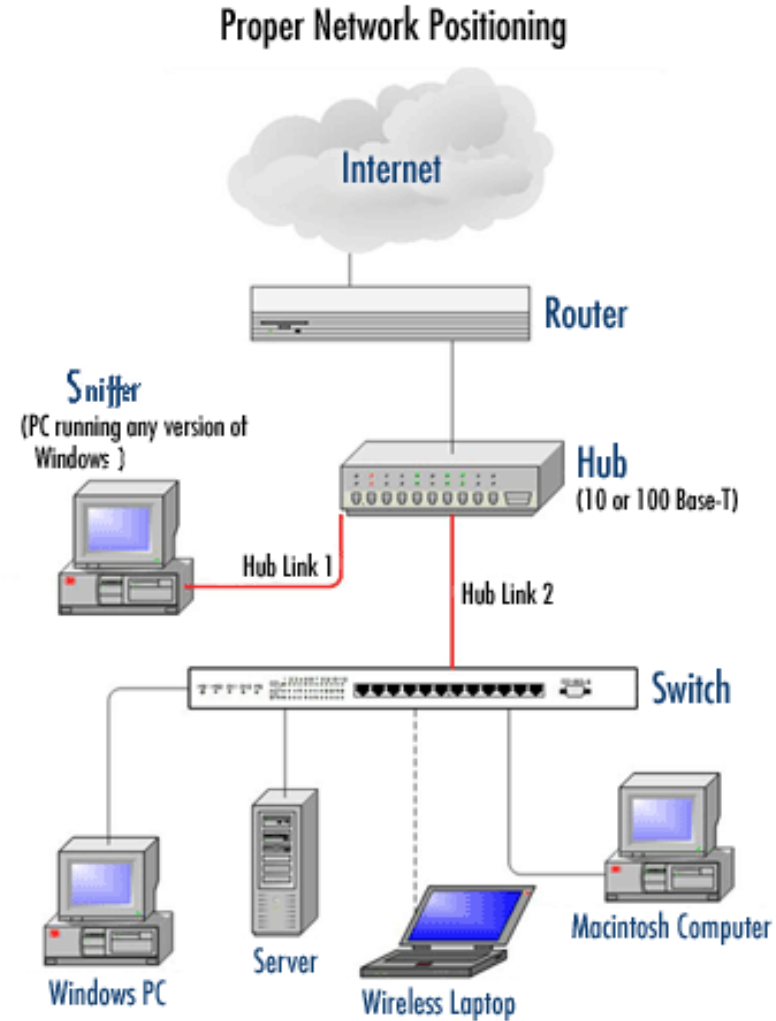
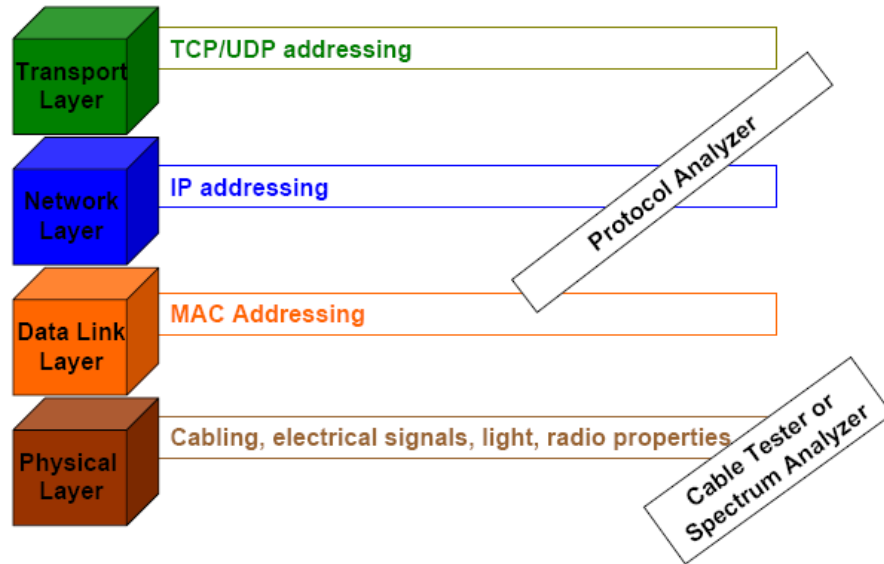


Figure 1: Packet sniffer structure

Ağ Dinleme Araçları (Sniffer ve Topolojideki Yeri)



Ağ Dinleme Araçları (Wireshark Kullanıcı Arayüzü)

The image shows the Wireshark interface with a packet capture of an HTTP GET request. The packet list shows a GET request from 192.168.1.103 to 64.233.169.99. The packet details pane shows the frame structure with MAC, IP, and TCP headers. The packet bytes pane shows the raw data in hexadecimal and ASCII.

Packet List:

No.	Time	Source	Destination	Protocol	Info
2	0.002401	Cisco-Li_f6:53:23	Cisco-Li_7c:6a:35	ARP	192.168.1.1 is at 00:1a:70:f6:53:23
3	0.002432	192.168.1.103	68.105.28.12	DNS	Standard query A www.google.com
4	0.014577	68.105.28.12	192.168.1.103	DNS	Standard query response CNAME ww
5	0.015472	192.168.1.103	64.233.169.99	TCP	49591 > http [SYN] Seq=0 Len=0 M
6	0.031934	64.233.169.99	192.168.1.103	TCP	http > 49591 [SYN, ACK] Seq=0 Ac
			233.169.99	TCP	49591 > http [ACK] Seq=1 Ack=1 W
			233.169.99	HTTP	GET / HTTP/1.1
		192.168.1.103	64.233.169.99	TCP	http > 49591 [ACK] Seq=1 Ack=524
		192.168.1.103	64.233.169.99	TCP	[TCP segment of a reassembled PC
		192.168.1.103	64.233.169.99	TCP	[TCP segment of a reassembled PC

Packet Details:

- Frame 6 (577 bytes on wire (577 bytes captured) on interface 0)
- Ethernet II, Src: Cisco-Li_7c:6a:35 (00:14:bf:7c:6a:35), Dst: Cisco-Li_f6:53:23 (00:1a:70:f6:53:23)
- Internet Protocol, Src: 192.168.1.103 (192.168.1.103), Dst: 64.233.169.99 (64.233.169.99)
- Transmission Control Protocol, Src Port: 49591 (49591), Dst Port: http (80), Seq: 1, Ack: 1, Len: 523

Packet Bytes:

Offset	Hex	ASCII
00b0	28 63 6f 6d 70 61 74 69	(compatible; MSI
00c0	45 20 37 2e 30 3b 20 57	E 7.0; Windows N
00d0	54 20 36 2e 30 3b 20 53	T 6.0; S LCC1; .N
00e0	45 54 20 43 4c 52 20 32	ET CLR 2 .0.50727

Annotations:

- MAC Header: Points to the Ethernet II header.
- IP Header: Points to the Internet Protocol header.
- TCP Header: Points to the Transmission Control Protocol header.
- Data: Points to the application data.
- MS IE7.0: Points to the user-agent string in the ASCII data.
- ASCII equivalent of data: Points to the ASCII data pane.

File: "C:\Users\Joe\AppData\Local\Temp\etherXXXXa01124" 5905 Bytes 00:00... P: 17 D: 17 M: 0 Drops: 0

Libpcap-Cihaz Ayarlama

```
#include <stdio.h>
#include <pcap.h>

int main(int argc, char *argv[])
{
    char *dev = argv[1];

    printf("Device: %s\n", dev);
    return(0);
}
```

```
#include <stdio.h>
#include <pcap.h>

int main(int argc, char *argv[])
{
    char *dev, errbuf[PCAP_ERRBUF_SIZE];

    dev = pcap_lookupdev(errbuf);
    if (dev == NULL) {
        fprintf(stderr, "Couldn't find default device: %s\n", errbuf);
        return(2);
    }
    printf("Device: %s\n", dev);
    return(0);
}
```

Libpcap-Cihaz Koklaması

```
pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms,  
char *ebuf)
```

```
#include <pcap.h>  
...  
pcap_t *handle;  
  
handle = pcap_open_live(dev, BUFSIZ, 1, 1000, errbuf);  
if (handle == NULL) {  
    fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);  
    return(2);  
}
```

Libpcap-Paket Filtreleme (Örnek Kod)

```
#include <pcap.h>

...
pcap_t *handle;          /* Session handle */
char dev[] = "r10";      /* Device to sniff on */
char errbuf[PCAP_ERRBUF_SIZE]; /* Error string */
struct bpf_program fp;    /* The compiled filter expression */
char filter_exp[] = "port 23"; /* The filter expression */
bpf_u_int32 mask;         /* The netmask of our sniffing device */
bpf_u_int32 net;          /* The IP of our sniffing device */

if (pcap_lookupnet(dev, &net, &mask, errbuf) == -1) {
    fprintf(stderr, "Can't get netmask for device %s\n", dev);
    net = 0;
    mask = 0;
}
handle = pcap_open_live(dev, BUFSIZ, 1, 1000, errbuf);
if (handle == NULL) {
    fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
    return(2);
}
if (pcap_compile(handle, &fp, filter_exp, 0, net) == -1) {
    fprintf(stderr, "Couldn't parse filter %s: %s\n", filter_exp, pcap_geterr(handle));
    return(2);
}
if (pcap_setfilter(handle, &fp) == -1) {
    fprintf(stderr, "Couldn't install filter %s: %s\n", filter_exp, pcap_geterr(handle));
    return(2);
}
```