



lobster

An IST Project



Information Society
Technologies

<http://www.ist-lobster.org/>

passive monitoring with pcap a mini-tutorial

Herbert Bos

Vrije Universiteit Amsterdam

[herbertb @ cs.vu.nl](mailto:herbertb@cs.vu.nl)



Herbert Bos, VU, <http://ww.cs.vu.nl/~herbertb>



lobster

An IST Project

pcap



Information Society
Technologies

<http://www.ist-lobster.org/>

- Extremely popular
 - tcpdump, snort, ntop, ngrep, ethereal, Bro, honeyd
- Widely supported
 - *nix
 - Windows
- Very convenient way of expressing filters
 - `tcpdump ip host nordkapp and not alzheimer`
 - `tcpdump 'gateway flits and (port ftp or ftp-data)'`
- Based on BPF
- Powerful
- Ancient





2 views on programming



1. filter programming

- `tcpdump 'gateway flits and (port ftp or ftp-data)'`
- `tcpdump dst host spider and "(udp or proto 51)" and not "(src host daffy or src host fulton)"`

2. application programming

- how to implement tcpdump
 - specifying the device
 - setting up the filter
 - capturing the packet
 - etc.





connecting the 2 views



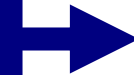
Information Society
Technologies

lobster

An IST Project

<http://www.ist-lobster.org/>

```
main () {  
  // let's filter packets  
  use filter "proto TCP and  
    port 54321"  
  for (;;)   
    get packet  
}
```



compile



BPF code



kernel

```
main () {  
  for (;;)   
    get packet  
}
```



```
proto TCP and  
port 54321
```





lobster

An IST Project

writing filters



Information Society
Technologies

<http://www.ist-lobster.org/>

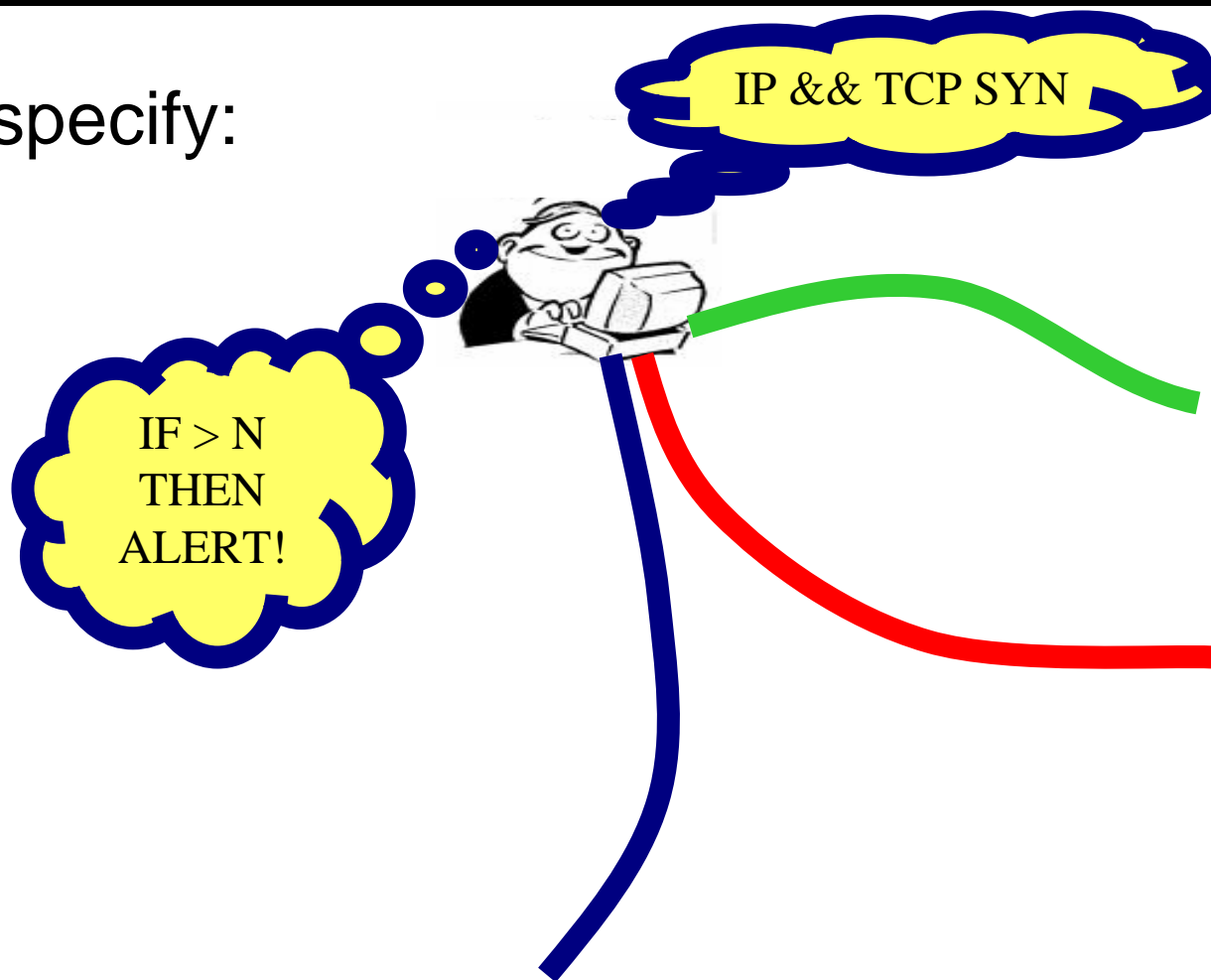
- not covered in this tutorial
 - `man tcpdump`
 - tons of tutorials on the web



Herbert Bos, VU, <http://ww.cs.vu.nl/~herbertb>



- we need to specify:
 - device(s)
 - filters
 - action





actual programming



practice

```
dev = pcap_lookupdev(err);
```

```
// say we want to monitor device dev
```

```
// open the device:
```

```
pcap_t *descr = pcap_open_live(dev, BUFSIZ, 0, -1, errbuf);
```

```
// grab a packet:
```

```
char *pkt = pcap_next(descr, &hdr);
```

```
// what did we catch?
```

```
printf("Catched packet: %s\n", pkt);
```

```
i = 0; while (i < 10) {  
    pcap_open_live(char *dev, int snaplen,  
    int promisc, int to_ms, char *ebuf)  
    do {
```

- **snaplen** - maximum size of packets to capture
- **promisc** - set card in promiscuous mode?
- **to_ms** - time to wait for packets in milliseconds before read times out
- **errbuf** - if something happens, place error string here





Most programmers use a callback function

// Open the device:

```
pcap_t *descr = pcap_open_live(dev, BUFSIZ, 0, -1, errbuf);
```

// now register callback and start looping:

```
pcap_loop(descr, 1000, my_callback, NULL);
```

// callback function that is passed to pcap_loop(..) and

// called each time * a packet is recieved */

```
void my_callback (u_char *user, const struct pcap_pkthdr*  
                 pkthdr, const u_char* packet)
```

```
{  
    fprintf(stdout, "%d, ", count);
```

```
}
```





now set a filter:

// Open the device:

```
pcap_t *descr = pcap_open_live(dev, BUFSIZ, 0, -1, errbuf);
```

// compile filter:

```
pcap_compile(descr, &fp, argv[1], 0, netp);
```

// set filter:

```
pcap_setfilter(descr, &fp);
```

// now register callback and start looping:

```
pcap_loop(descr, 1000, my_callback, NULL);
```

// callback function is passed to pcap_loop(..) and called each time a pkt is received

```
void my_callback (u_char *user, const struct pcap_pkthdr*  
                 pkthdr, const u_char* packet)
```

```
{  
    fprintf(stdout, "%d, ", count);
```

```
}
```





lobster

An IST Project

my view



Information Society
Technologies

<http://www.ist-lobster.org/>

- **pcap is**
 - simple
 - often slow (but can be made a lot faster)
- **drawbacks**
 - no persistent state
 - limited expressiveness:
 - no loops
 - no way to exploit external hardware and software

