Packet Crafting

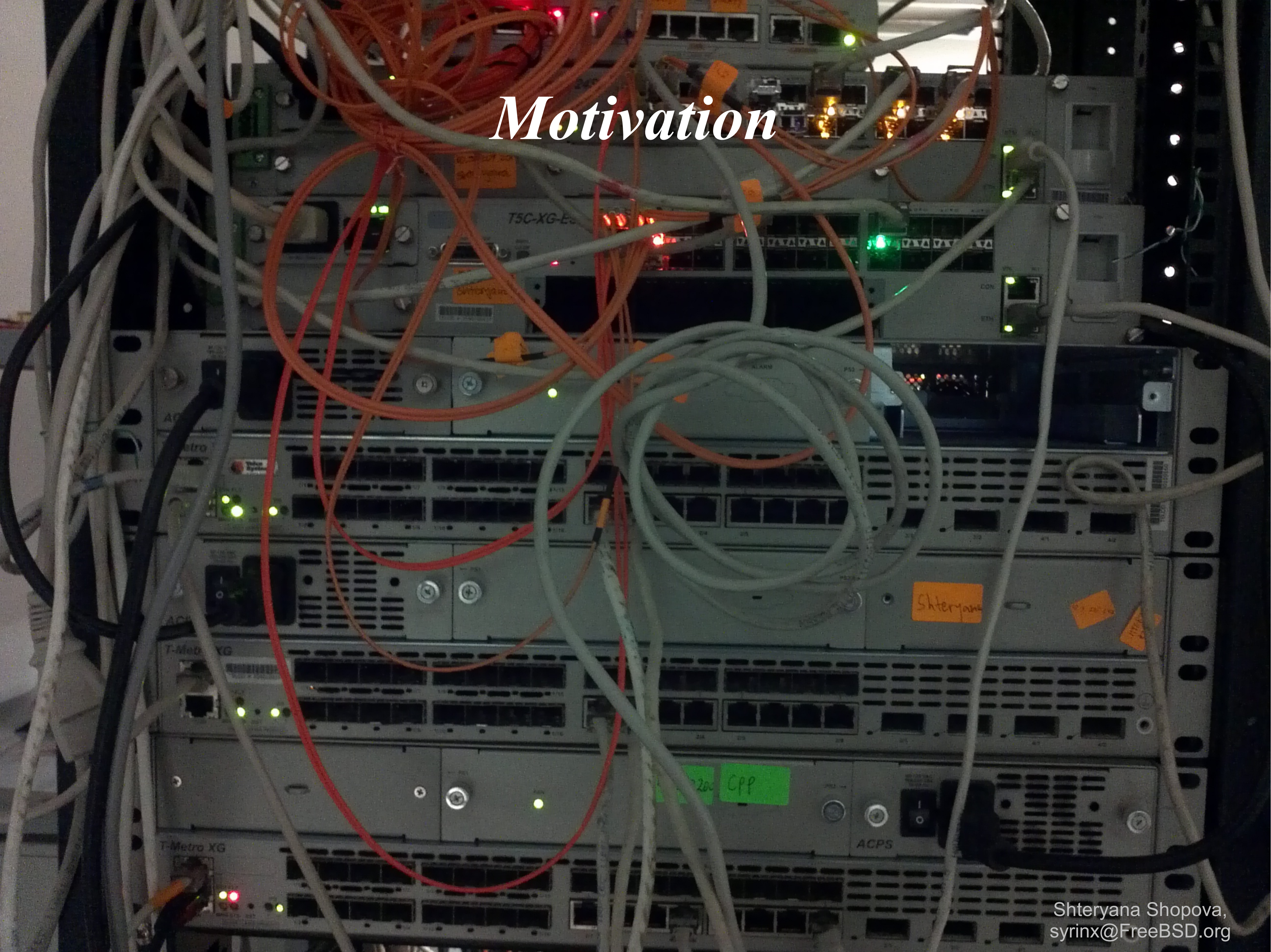
OpenFest, Sofia
November, 2013

Shteryana Shopova,
syrinx@FreeBSD.org

Disclaimer

!!! Use at your own risk !!!

Motivation



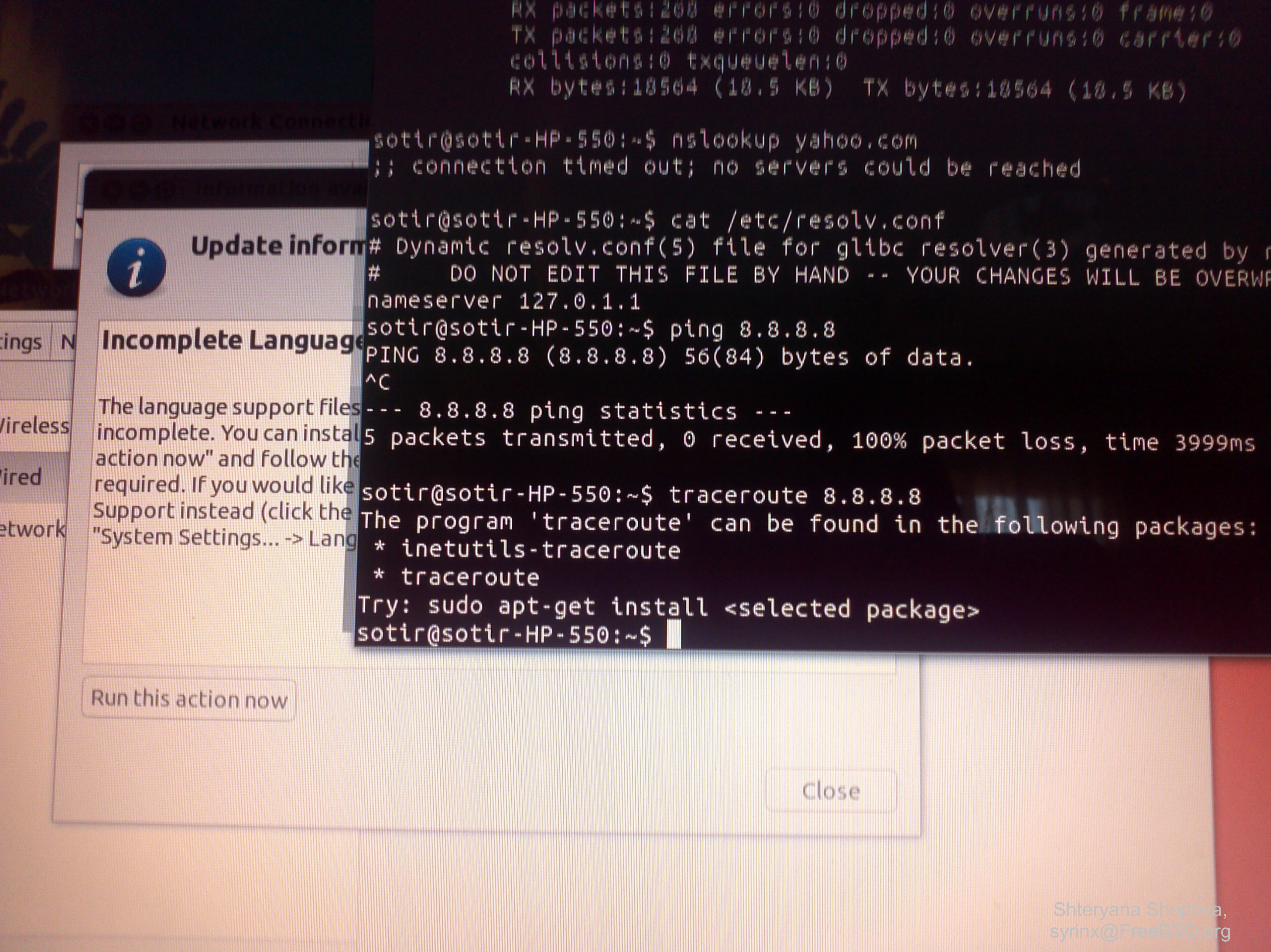
Motivation (2)

Agenda

- ping(8), traceroute(8), telnet(1), nmap(1)
- ng_source(4), tcpdump(1)
- bridge(4), vlan(4), svlan(4), trunk(4)/lagg(4)
- yersinia(8)
- nemesis(1), libnet
- hyenae
- pierf, Scapy
- netmap(4)
- iperf
- PF_PACKET, etc... sockets, C code
- OS network stack & daemons

Let's start

- ping(8)
  - options: flood, quiet, multicast, source route, etc
- traceroute(8)
  - incrementing IP TTL probes
  - UDP, but may be random IP protocol number
- telnet(1)
  - TCP port 23, unless custom specified
- nmap(1)
  - network security scanner
- and more
  - arping, nslookup/host/dig, ...



```
          RX packets:268 errors:0 dropped:0 overruns:0 frame:0
          TX packets:268 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:18564 (18.5 KB)  TX bytes:18564 (18.5 KB)

sotir@sotir-HP-550:~$ nslookup yahoo.com
;; connection timed out; no servers could be reached

sotir@sotir-HP-550:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by r
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWR
nameserver 127.0.1.1
sotir@sotir-HP-550:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 3999ms

sotir@sotir-HP-550:~$ traceroute 8.8.8.8
The program 'traceroute' can be found in the following packages:
 * inetutils-traceroute
 * traceroute
Try: sudo apt-get install <selected package>
sotir@sotir-HP-550:~$ 
```

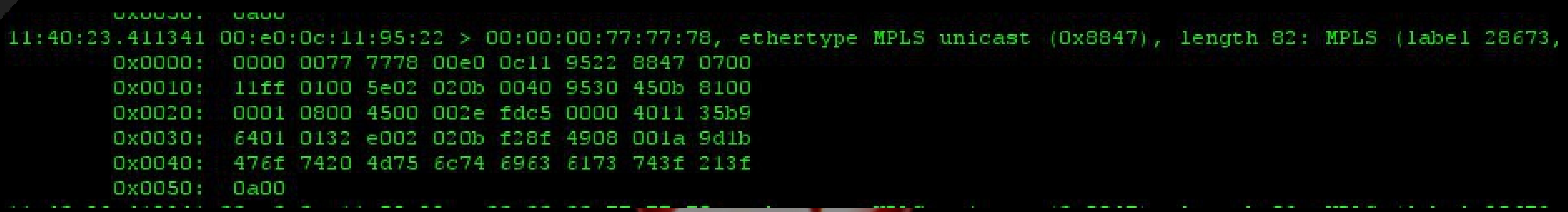
Update inform

Incomplete Languag

The language support files incomplete. You can instal action now" and follow the required. If you would like Support instead (click the "System Settings... -> Lang

ng_source(4)

- tcpdump(8)



```
11:40:23.411341 00:e0:0c:11:95:22 > 00:00:00:77:77:78, ethertype MPLS unicast (0x8847), length 82: MPLS (label 28673,
        0x0000:  0000 0077 7778 00e0 0c11 9522 8847 0700
        0x0010:  11ff 0100 5e02 020b 0040 9530 450b 8100
        0x0020:  0001 0800 4500 002e fdc5 0000 4011 35b9
        0x0030:  6401 0132 e002 020b f28f 4908 001a 9d1b
        0x0040:  476f 7420 4d75 6c74 6963 6173 743f 213f
        0x0050:  0a00
```

  - there's wireshark too
- netgraph(3)
  - graph based kernel networking subsystem of FreeBSD
- ng_ether(4)
  - a node automatically created for every Ethernet interface in the system

ng_source(4) contd.

```
#!/bin/sh
##########################################################
# TODO: insert some nice comment here
##########################################################
ECHO=/bin/echo
NGHOOK=/usr/sbin/nghook
HOOKNAME=input

usage () {
$ECHO     "Usage: $0 <ng-node> <count> <packet-gen> <initial-arg> <increment-gen>"
}

...

$ECHO "Injecting $LIMIT packets into $NGNODE node"

# Loop from 1 to 255
while [ "$var1" -le $LIMIT ]
do
DSTIP=${ARGS}
$ECHO "Injecting packet with DST IP ${DSTIP}"

${PACKGEN} ${DSTIP} | ${NGHOOK} ${NGNODE} ${HOOKNAME}
let "var1 = var1 + 1" >> /dev/null  # let "var1+=1"
ARGS=`${INCGEN} ${DSTIP}`
done
exit 0
```

Ether-like interfaces

- bridge(4)
- vlan(4), svlan(4)
- trunk(4)/lagg(4)
- tap(4)

Yersinia

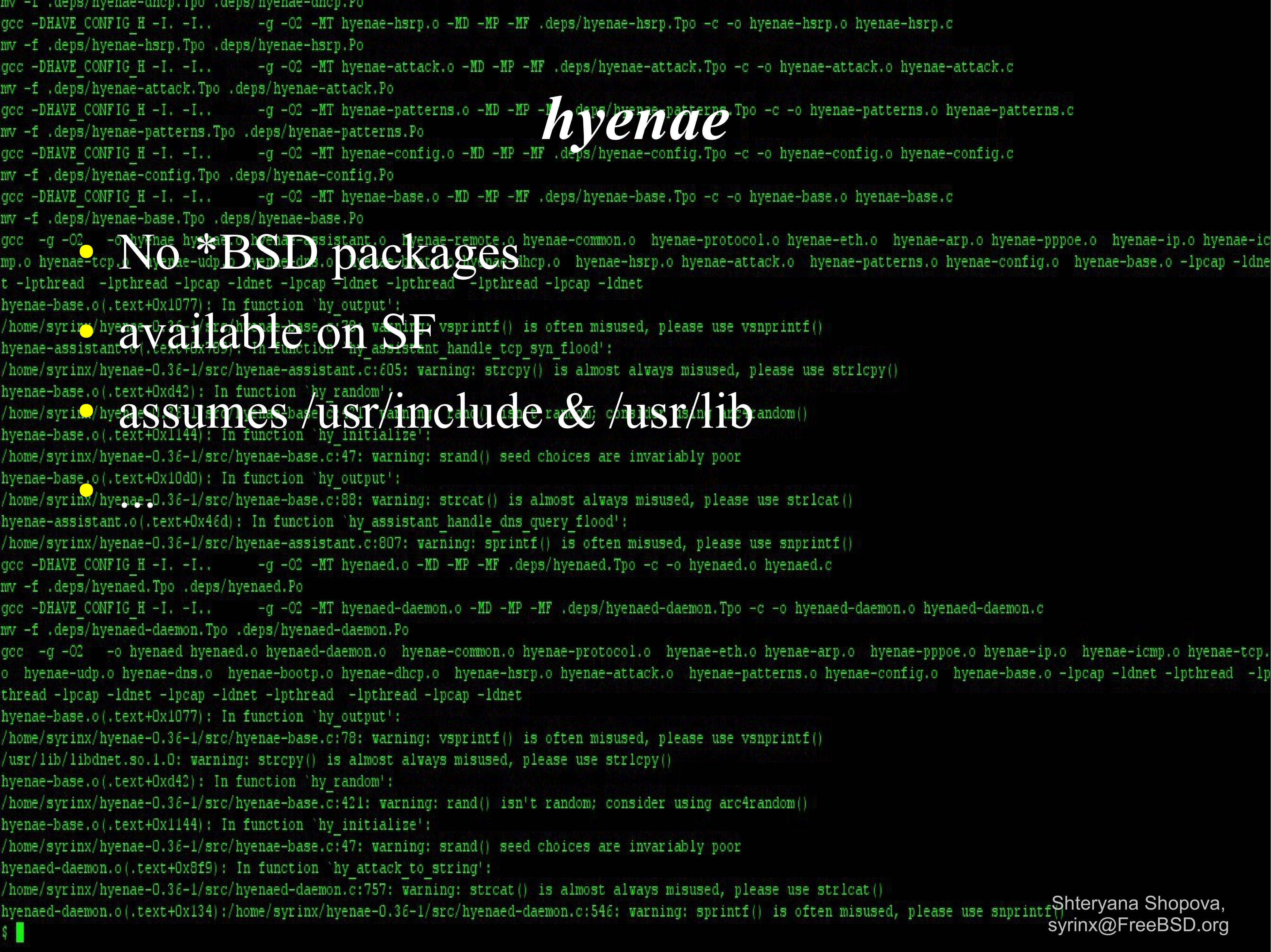
- http://www.yersinia.net
- Supported protocols
  - STP, CDP, DTP, DHCP, HSRP, IEEE 802.1Q, IEEE 802.1X, ISL, VTP
- UI
  - GTK, ncurses, command line, netclient
- predefined scenarios
- netclient
  - TCP port 12000
  - passwd/enable passwd : root/tomac

nemesis(1)

```
NEMESIS -=- The NEMESIS Project Version 1.4 (Build 26)


NEMESIS Usage:
  nemesis [mode] [options]

NEMESIS modes:
  arp
  dns
  ethernet
  icmp
  igmp
  ip
  ospf (currently non-functional)
  rip
  tcp
  udp
```

hyenae

- No *BSD packages
- available on SF
- assumes /usr/include & /usr/lib
- ...



```
mv -f .deps/hyenae-dhcp.Tpo .deps/hyenae-dhcp.Po
gcc -DHAVE_CONFIG_H -I. -I..      -g -O2 -MT hyenae-hsrp.o -MD -MP -MF .deps/hyenae-hsrp.Tpo -c -o hyenae-hsrp.o hyenae-hsrp.c
mv -f .deps/hyenae-hsrp.Tpo .deps/hyenae-hsrp.Po
gcc -DHAVE_CONFIG_H -I. -I..      -g -O2 -MT hyenae-attack.o -MD -MP -MF .deps/hyenae-attack.Tpo -c -o hyenae-attack.o hyenae-attack.c
mv -f .deps/hyenae-attack.Tpo .deps/hyenae-attack.Po
gcc -DHAVE_CONFIG_H -I. -I..      -g -O2 -MT hyenae-patterns.o -MD -MP -MF .deps/hyenae-patterns.Tpo -c -o hyenae-patterns.o hyenae-patterns.c
mv -f .deps/hyenae-patterns.Tpo .deps/hyenae-patterns.Po
gcc -DHAVE_CONFIG_H -I. -I..      -g -O2 -MT hyenae-config.o -MD -MP -MF .deps/hyenae-config.Tpo -c -o hyenae-config.o hyenae-config.c
mv -f .deps/hyenae-config.Tpo .deps/hyenae-config.Po
gcc -DHAVE_CONFIG_H -I. -I..      -g -O2 -MT hyenae-base.o -MD -MP -MF .deps/hyenae-base.Tpo -c -o hyenae-base.o hyenae-base.c
mv -f .deps/hyenae-base.Tpo .deps/hyenae-base.Po
gcc  -g -O2   -o hyenae hyenae.o hyenae-assistant.o hyenae-remote.o hyenae-common.o  hyenae-protocol.o hyenae-eth.o  hyenae-arp.o hyenae-pppoe.o  hyenae-ip.o hyenae-ic
mp.o hyenae-tcp.o hyenae-udp.o hyenae-dns.o hyenae-bootp.o hyenae-dhcp.o  hyenae-hsrp.o hyenae-attack.o  hyenae-patterns.o hyenae-config.o  hyenae-base.o -lpcap -ldne
t -lpthread  -lpthread -lpcap -ldnet -lpcap -ldnet -lpthread  -lpthread -lpcap -ldnet
hyenae-base.o(.text+0x1077): In function `hy_output':
/home/syrinx/hyenae-0.36-1/src/hyenae-base.c:78: warning: vsprintf() is often misused, please use vsnprintf()
hyenae-assistant.o(.text+0x789): In function `hy_assistant_handle_tcp_syn_flood':
/home/syrinx/hyenae-0.36-1/src/hyenae-assistant.c:605: warning: strcpy() is almost always misused, please use strlcpy()
hyenae-base.o(.text+0xd42): In function `hy_random':
/home/syrinx/hyenae-0.36-1/src/hyenae-base.c:421: warning: rand() isn't random; consider using arc4random()
hyenae-base.o(.text+0x1144): In function `hy_initialize':
/home/syrinx/hyenae-0.36-1/src/hyenae-base.c:47: warning: srand() seed choices are invariably poor
hyenae-base.o(.text+0x10d0): In function `hy_output':
/home/syrinx/hyenae-0.36-1/src/hyenae-base.c:88: warning: strcat() is almost always misused, please use strlcat()
hyenae-assistant.o(.text+0x46d): In function `hy_assistant_handle_dns_query_flood':
/home/syrinx/hyenae-0.36-1/src/hyenae-assistant.c:807: warning: sprintf() is often misused, please use snprintf()
gcc -DHAVE_CONFIG_H -I. -I..      -g -O2 -MT hyenaed.o -MD -MP -MF .deps/hyenaed.Tpo -c -o hyenaed.o hyenaed.c
mv -f .deps/hyenaed.Tpo .deps/hyenaed.Po
gcc -DHAVE_CONFIG_H -I. -I..      -g -O2 -MT hyenaed-daemon.o -MD -MP -MF .deps/hyenaed-daemon.Tpo -c -o hyenaed-daemon.o hyenaed-daemon.c
mv -f .deps/hyenaed-daemon.Tpo .deps/hyenaed-daemon.Po
gcc  -g -O2   -o hyenaed hyenaed.o hyenaed-daemon.o  hyenae-common.o hyenae-protocol.o  hyenae-eth.o hyenae-arp.o  hyenae-pppoe.o hyenae-ip.o  hyenae-icmp.o hyenae-tcp.
o  hyenae-udp.o hyenae-dns.o  hyenae-bootp.o hyenae-dhcp.o  hyenae-hsrp.o hyenae-attack.o  hyenae-patterns.o hyenae-config.o  hyenae-base.o -lpcap -ldnet -lpthread  -lp
thread -lpcap -ldnet -lpcap -ldnet -lpthread  -lpthread -lpcap -ldnet
hyenae-base.o(.text+0x1077): In function `hy_output':
/home/syrinx/hyenae-0.36-1/src/hyenae-base.c:78: warning: vsprintf() is often misused, please use vsnprintf()
/usr/lib/libdnet.so.1.0: warning: strcpy() is almost always misused, please use strlcpy()
hyenae-base.o(.text+0xd42): In function `hy_random':
/home/syrinx/hyenae-0.36-1/src/hyenae-base.c:421: warning: rand() isn't random; consider using arc4random()
hyenae-base.o(.text+0x1144): In function `hy_initialize':
/home/syrinx/hyenae-0.36-1/src/hyenae-base.c:47: warning: srand() seed choices are invariably poor
hyenaed-daemon.o(.text+0x8f9): In function `hy_attack_to_string':
/home/syrinx/hyenae-0.36-1/src/hyenaed-daemon.c:757: warning: strcat() is almost always misused, please use strlcat()
hyenaed-daemon.o(.text+0x134):/home/syrinx/hyenae-0.36-1/src/hyenaed-daemon.c:546: warning: sprintf() is often misused, please use snprintf()
$ 
```

hyenae (contd.)

- “ERROR: Root privileges required”
- Supported protocols
  - ARP, PPPoE, ICMP, TCP, UDP, DNS, DHCP, HSRP
- UI
  - command line, QT-frontend
- Example

```
$ sudo hyenae -a pppoe-discover -i em0 -s 01:00:82:00:00:c2 -d
01:00:82:00:00:c3
* Initializing
* Opening network interface (em0)
* Launching attack

  Press any key to stop
```

Scapy

- “Powerful interactive packet manipulation program in python”
- Python intrerpreter disguised as a Domain Specific Language
- Fast Packet Designing
- Interactive packet and result manupulation
- Fast packet generator ? :dd

Scapy - ls()

```
Welcome to Scapy (2.2.0)
>>>
>>> ls()
ARP        : ARP
ASN1_Packet : None
BOOTP      : BOOTP
CookedLinux : cooked linux
DHCP       : DHCP options
DHCP6      : DHCPv6 Generic Message)
DHCP6OptAuth : DHCP6 Option - Authentication
DHCP6OptBCMCSDomains : DHCP6 Option - BCMCS Domain Name List
DHCP6OptBCMCSServers : DHCP6 Option - BCMCS Addresses List
DHCP6OptClientFQDN : DHCP6 Option - Client FQDN
DHCP6OptClientId : DHCP6 Client Identifier Option
DHCP6OptDNSDomains : DHCP6 Option - Domain Search List option
DHCP6OptDNSServers : DHCP6 Option - DNS Recursive Name Server
DHCP6OptElapsedTime : DHCP6 Elapsed Time Option
DHCP6OptGeoConf :
DHCP6OptIAAddress : DHCP6 IA Address Option (IA_TA or IA_NA suboption)
...
<6 more screens of output>
```

Scapy - lcs()

```
>>> lsc()
bind_layers         : Bind 2 layers on some specific fields' values
corrupt_bits        : Flip a given percentage or number of bits from a string
corrupt_bytes       : Corrupt a given percentage or number of bytes from a string
defrag              : defrag(plist) -> ([not fragmented], [defragmented],
defragment          : defrag(plist) -> plist defragmented as much as possible
etherleak           : Exploit Etherleak flaw
fragment            : Fragment a big IP datagram
fuzz                : Transform a layer into a fuzzy layer by replacing some default values by
random objects
hexdiff             : Show differences between 2 binary strings
hexdump             : --
hexedit             : --

rdpcap              : Read a pcap file and return a packet list
send                : Send packets at layer 3
sendp               : Send packets at layer 2
sendpfast           : Send packets at layer 2 using tcpreplay for performance
sniff               : Sniff packets
split_layers        : Split 2 layers previously bound
sr                  : Send and receive packets at layer 3
srbt                : send and receive using a bluetooth socket
srflood             : Flood and receive packets at layer 3
srloop              : Send a packet at layer 3 in loop and print the answer each time
srp                 : Send and receive packets at layer 2
traceroute          : Instant TCP traceroute
wrpcap              : Write a list of packets to a pcap file

<some commands ommitted, such as wireshark>
```

Scapy – sending a packet

```
Welcome to Scapy (2.2.0)
>>> a=Ether()/IP(dst="www.slashdot.org")/TCP()/"GET /index.html
HTTP/1.0 \n\n"
>>> hexdump(a)
0000   00 22 68 5A AA E4 74 E5  0B E3 39 68 08 00 45 00   ."hZ..t...9h..E.
0010   00 43 00 01 00 00 40 06  41 3B AC 14 00 12 D8 22   .C....@.A;....."
0020   B5 30 00 14 00 50 00 00  00 00 00 00 00 00 50 02   .0...P........P.
0030   20 00 84 38 00 00 47 45  54 20 2F 69 6E 64 65 78    ..8..GET /index
0040   2E 68 74 6D 6C 20 48 54  54 50 2F 31 2E 30 20 0A   .html HTTP/1.0 .
0050   0A                                                 .
>>> sendp(Ether()/IP(dst="1.2.3.4",ttl=(1,4)), iface="wlan0")
....
Sent 4 packets.
```

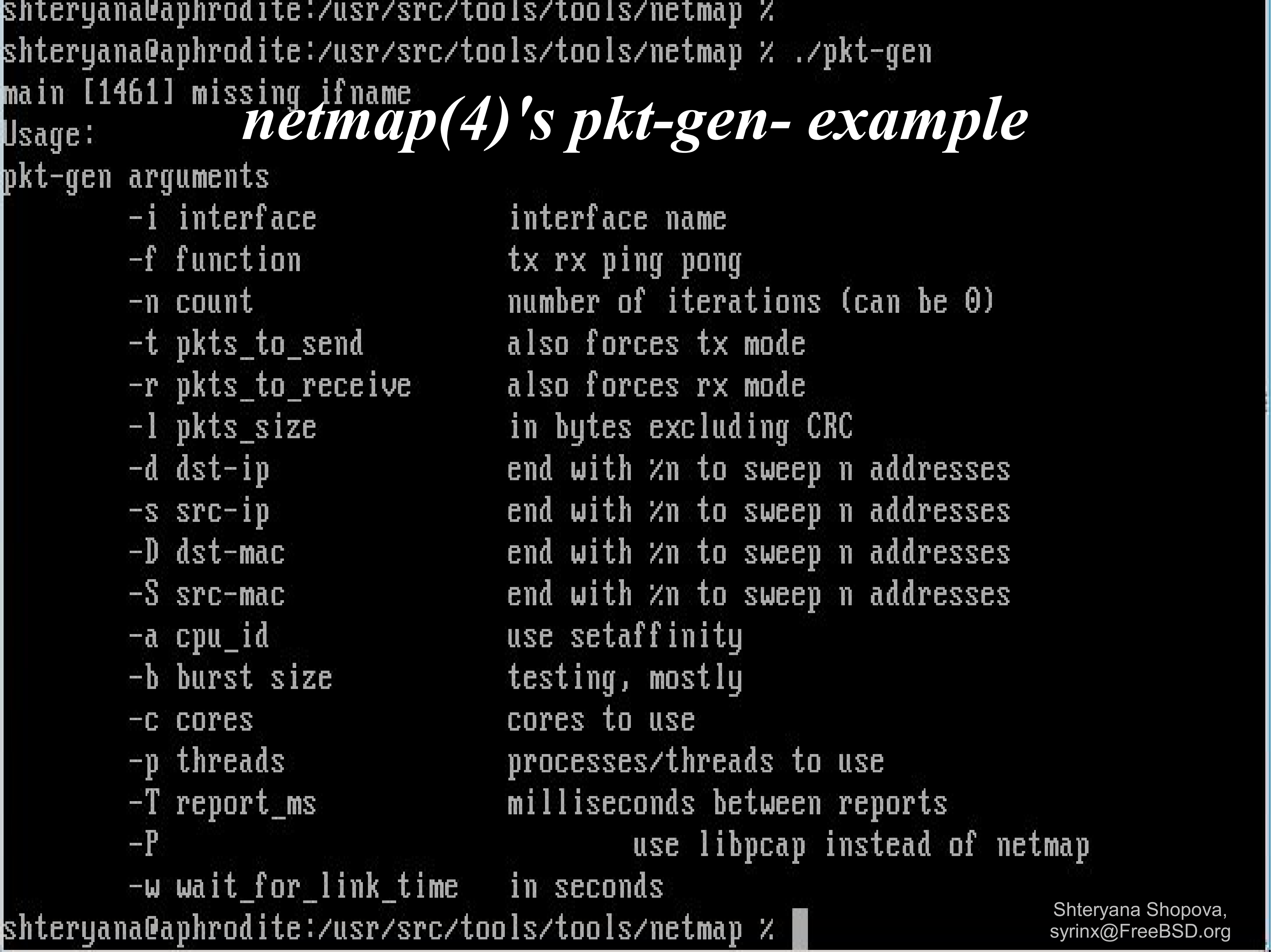
Scapy – writing your own class

- TBD
- (homework)

netmap(4)

- fast packet I/O framework
- reduce syscalls to improve packet processing performance
- in many cases, shared buffers between user and kernel space
- open(“/dev/netmap”)
- ioctl(.., NIOCREGIF, ..) to bind to an interface
- fill in available buffers with data
- non-blocking ioctl(.., NIOCTXSYNC) /ioctl(.., NIOCRXSYNC) to transmit/receive data
- supported drivers - em(4), igb(4), ixgbe(4), re(4)

netmap(4)'s pkt-gen- example



```
shteryana@aphrodite:/usr/src/tools/tools/netmap %
shteryana@aphrodite:/usr/src/tools/tools/netmap % ./pkt-gen
main [1461] missing ifname
Usage:
pkt-gen arguments
        -i interface            interface name
        -f function             tx rx ping pong
        -n count                number of iterations (can be 0)
        -t pkts_to_send         also forces tx mode
        -r pkts_to_receive      also forces rx mode
        -l pkts_size            in bytes excluding CRC
        -d dst-ip               end with %n to sweep n addresses
        -s src-ip               end with %n to sweep n addresses
        -D dst-mac              end with %n to sweep n addresses
        -S src-mac              end with %n to sweep n addresses
        -a cpu_id               use setaffinity
        -b burst size           testing, mostly
        -c cores                cores to use
        -p threads              processes/threads to use
        -T report_ms            milliseconds between reports
        -P                              use libpcap instead of netmap
        -w wait_for_link_time   in seconds
shteryana@aphrodite:/usr/src/tools/tools/netmap %
```

netmap(4) - example

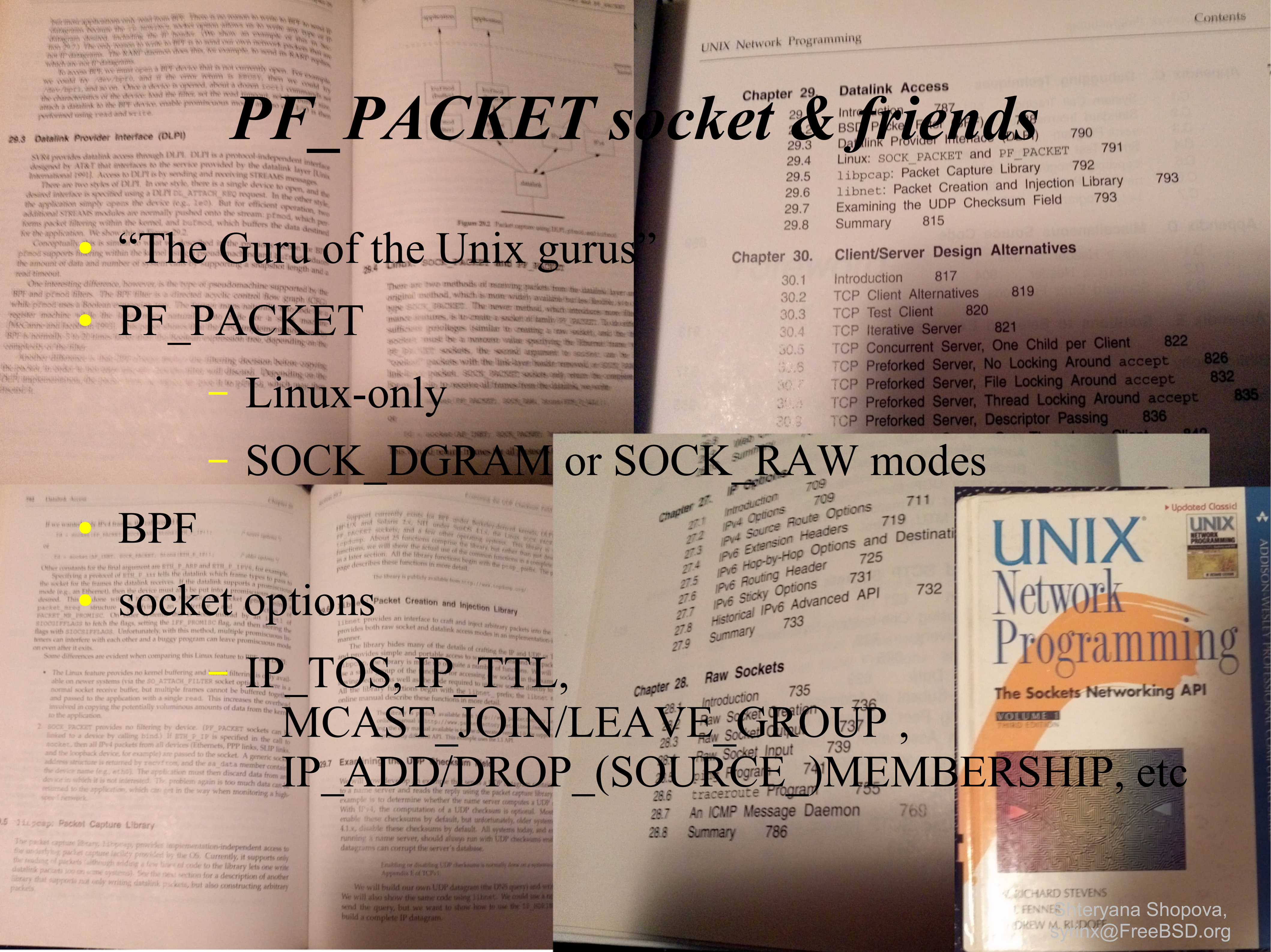
```
char *buf;
int fd;
struct pollfd      fds;
struct nmreq       nmr;
struct netmap_if *nifp;

fd = open(“/dev/netmap”, O_RDWR);

strlcpy(nmr.nr_name, “em0”, sizeof(nmr.nr_name));
nmr.nr_version = NETMAP_API;
ioctl(ng.fds.fd , NIOCGINFO, &nmr);

buf = mmap(NULL, nmr.nr_memsize, PROT_READ | PROT_WRITE,
    MAP_SHARED, fd, 0));

ioctl(fd , NIOCREGIF, &nmr);

fds.events = POLLOUT | POLLWRNORM | POLLWRBAND;
fds.fd = fd;

/* prepare packet in buf */
poll(&fds, 1, INFTIM);

ioctl(fd, NIOCTXSYNC, NULL); /* optional */
```

iperf

- commonly used network testing tool
- creates UDP/TCP streams to measure throughput
- iperf (July, 2010) vs iperf3 (March, 2013)
- client and server
- uni-/bi-directional tests

PF_PACKET socket & friends

- "The Guru of the Unix gurus"
- PF_PACKET
  - Linux-only
  - SOCK_DGRAM or SOCK_RAW modes
- BPF
- socket options
  - IP_TOS, IP_TTL, MCAST_JOIN/LEAVE_GROUP , IP_ADD/DROP_(SOURCE_)MEMBERSHIP, etc



Chapter 29. Datalink Access
29.1 Introduction 787
29.2 BSD Packet Filter (BPF)
29.3 Datalink Provider Interface (DLPI) 790
29.4 Linux: SOCK_PACKET and PF_PACKET 791
29.5 libpcap: Packet Capture Library 792
29.6 libnet: Packet Creation and Injection Library 793
29.7 Examining the UDP Checksum Field 793
29.8 Summary 815
Chapter 30. Client/Server Design Alternatives
30.1 Introduction 817
30.2 TCP Client Alternatives 819
30.3 TCP Test Client 820
30.4 TCP Iterative Server 821
30.5 TCP Concurrent Server, One Child per Client 822
30.6 TCP Preforked Server, No Locking Around accept 826
30.7 TCP Preforked Server, File Locking Around accept 832
30.8 TCP Preforked Server, Thread Locking Around accept 835
30.9 TCP Preforked Server, Descriptor Passing 836
Chapter 27. IP Options
27.1 Introduction 709
27.2 IPv4 Options 709
27.3 IPv4 Source Route Options 711
27.4 IPv6 Extension Headers 719
27.5 IPv6 Hop-by-Hop Options and Destinati
27.6 IPv6 Routing Header 725
27.7 IPv6 Sticky Options 731
27.8 Historical IPv6 Advanced API 732
27.9 Summary 733
Chapter 28. Raw Sockets
28.1 Introduction 735
28.2 Raw Socket Creation 736
28.3 Raw Socket Output 737
28.4 Raw Socket Input 739
28.5 ping Program 741
28.6 traceroute Program 755
28.7 An ICMP Message Daemon 769
28.8 Summary 786

Updated Classic

UNIX Network Programming

The Sockets Networking API

VOLUME 1 THIRD EDITION

W. RICHARD STEVENS
BILL FENNER
ANDREW M. RUDOFF

ADDISON-WESLEY PROFESSIONAL COMPUTING

Shteryana Shopova,
syrinx@FreeBSD.org

References

http://www.kohala.com/start/unpv12e.html

http://www.unpbook.com/src.html

http://www.secdev.org/projects/scapy/

http://info.iet.unipi.it/~luigi/netmap

http://www.secdev.org/conf/scapy_pacsec05.pdf

http://docs.python.org/2/tutorial/introduction.html

http://www.secdev.org/projects/scapy/doc/usage.html#interactive-tu

Questions?

Thank you!

* sources from this presentation available at http://people.freebsd.org/~syrinx/pktgen/

Shteryana Shopova,
syrinx@FreeBSD.org