Basics of Firewalls

Josh Richard
(jrichard@dsluug.org)

Alex Jokela
(alex@dsluug.org)

Presented on January 28, 2005
Duluth Superior Linux UNIX Users Group

Firewall Stuff

- Introduction
- Types of firewalls
  - host
  - perimeter
  - proxy
- Setting up firewalls the open source way
  - Ipfw
  - ipfilter
  - pf
  - Iptables
- Firewall verification

Introduction to Firewalls

- What is a firewall?
  - A) Protective barrier between you and the engine in your car.
  - B) Hardware or software designed and implemented to control the flow of network traffic.

- If you answered 'A', please leave. You want “Introduction to Motor Vehicles.”

Why Use a Firewall

- Enter the alarmists!!!
  - no firewall == University?
  - malware
  - malicious mobile code...viri, worms
- Enter the jerk!!!
  - Room/House mates not paying their share for Internet access
- Really though, you should make an effort to protect your equipment. Firewalls can help.

Types of Firewalls

- Host based
  - Run on end user machine.
  - Can be used to augment perimeter firewalls.
- Perimeter
  - Protect people from outside attack.
  - Does not protect people from inside attack.
- Proxy based
  - Not usually kernel level -> slower than kernel based.
  - Specific to application level protocols (HTTP, FTP, IMAP...)

ipfw

- FreeBSD 4 firewall.
  - Requires kernel recompilation
  - Filter by src ip, mac, TOS, you name it
  - Dynamic rulset support
  - Easy to use, once kernel is setup for ipfw.
- FreeBSD 5
  - ipfw enabled by default!

ipfw example

- Kernel options required for ipfw to work:

```
#ipfw hooks
options IPFIREWALL
options IPFIREWALL_VERBOSE
options IPFIREWALL_VERBOSE_LIMIT=300
```

- Add to /etc/rc.conf (BSD 4 or 5)

```
firewall_type="client"
firewall_quiet="YES"
firewall_logging="YES"
```

ipfw syntax /etc/rc.firewall

```
[Cc][Ll][Ii][Ee][Nn][Tt])
# set these to your network and netmask and ip
        net="10.2.3.0"
        mask="255.255.255.0"
        ip="10.2.3.1"
        setup_loopback
        ${fwcmd} add pass tcp from any to any established

        ${fwcmd} add pass all from any to any frag
        ${fwcmd} add pass icmp from  any to ${ip}
        ${fwcmd} add pass icmp from ${ip} to any
        # Allow setup of outgoing TCP connections only
        ${fwcmd} add pass tcp from ${ip} to any setup
        # Disallow setup of all other TCP connections
        ${fwcmd} add deny tcp from any to any setup
        # Everything else is denied by default, unless the
        # IPFIREWALL_DEFAULT_TO_ACCEPT option is set in your kernel
        # config file.
        ;;
```

ipfilter

- Simple Syntax
- Not as Robust as IPFW, PF, or iptables
- Available for use in
  - NetBSD
  - FreeBSD
  - OpenBSD (2.0 – 3.5)
  - Solaris
  - Linux
  - QNX
  - HP-UX
  - Tru64
- http://coombs.anu.edu.au/~avalon/

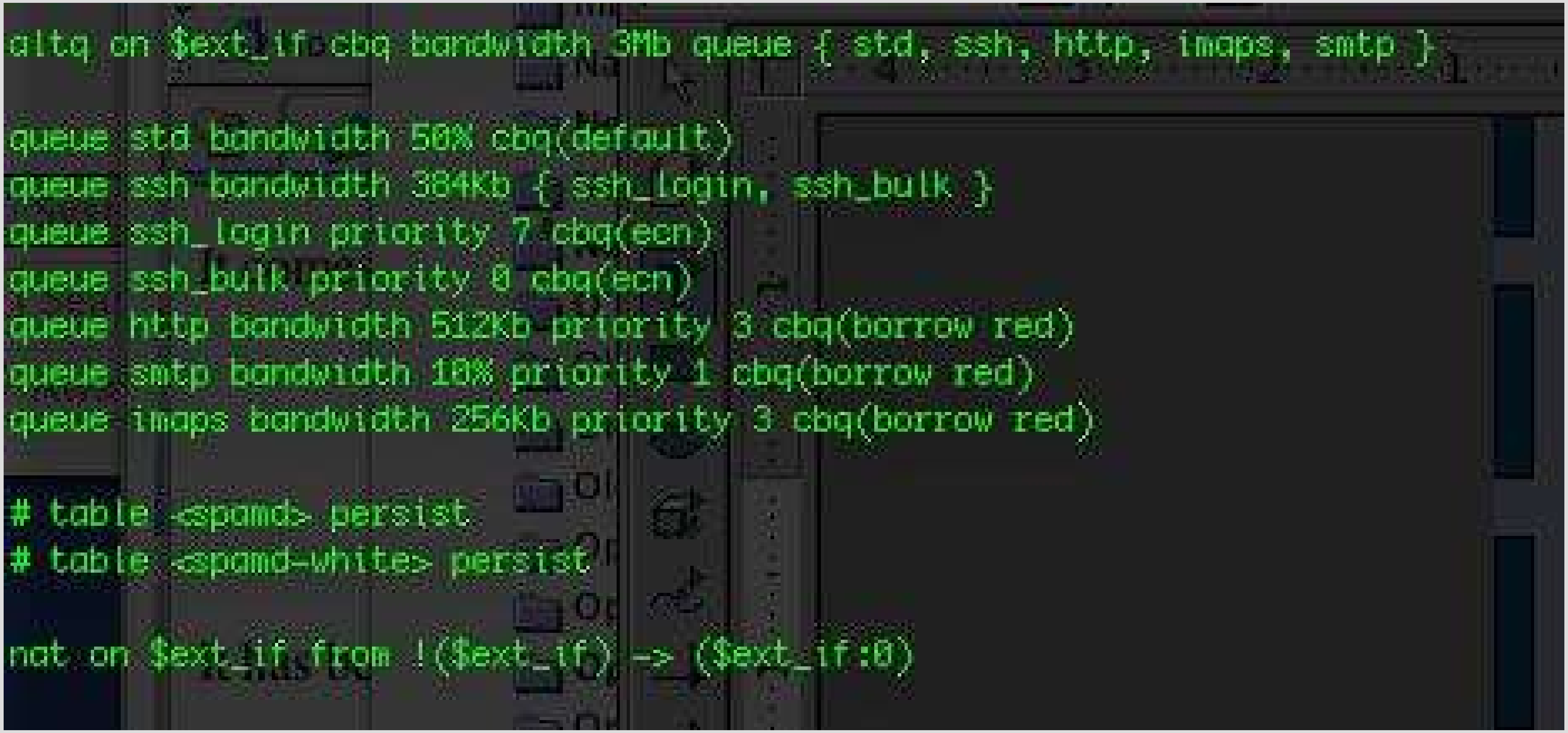
Ipfilter syntax

```
block in all
pass in quick from any to any port = 25 keep state
block out proto icmp on fxp0 from 192.168.0.1
```

OpenBSD's PF

- Included in FreeBSD 5.3, requires kernel recompile
- OpenBSD, no recompile
- Complete package: QoS, NAT and Filtering
- Syntax nearly identical to ipfilter
- Cool Things
  - Tables
  - Anchors – authorization via ssh connection
  - ALTQ
  - Passive OS Finger Printing

PF: Cool Things

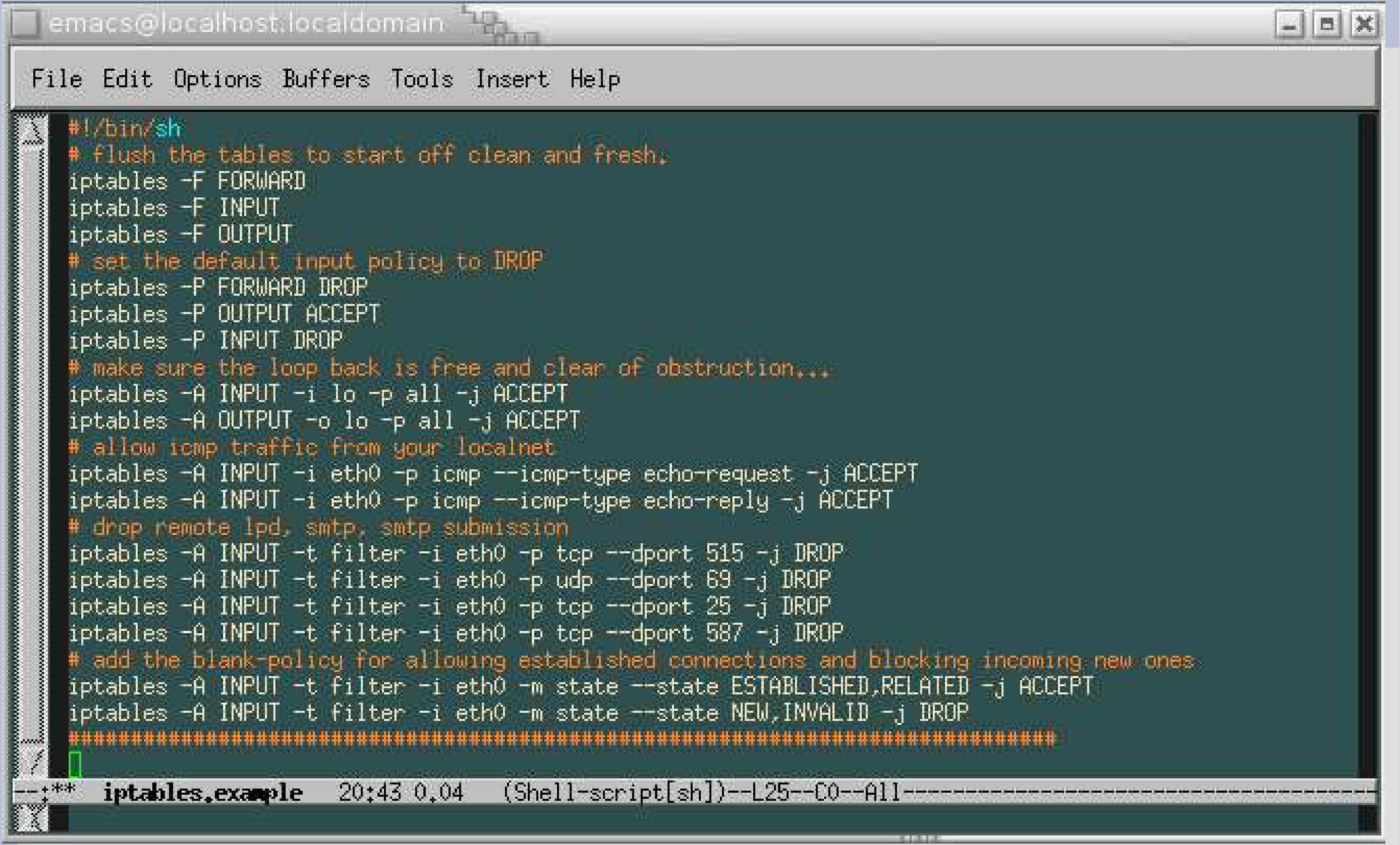


```
altq on $ext_if cbq bandwidth 3Mb queue { std, ssh, http, imaps, smtp }

queue std bandwidth 50% cbq(default)
queue ssh bandwidth 384Kb { ssh_login, ssh_bulk }
queue ssh_login priority 7 cbq(ecn)
queue ssh_bulk priority 0 cbq(ecn)
queue http bandwidth 512Kb priority 3 cbq(borrow red)
queue smtp bandwidth 10% priority 1 cbq(borrow red)
queue imaps bandwidth 256Kb priority 3 cbq(borrow red)

# table <spamd> persist
# table <spamd-white> persist

nat on $ext_if from !($ext_if) -> ($ext_if:0)
```

iptables

- Used for defining rulesets for packet mangling in linux 2.4 and 2.6 kernels.
- Dynamic ruleset support
- Create a shell script with your rules, chmod 0700 as root and run the script. Test and you are done.

iptables syntax


emacs@localhost.localdomain
File Edit Options Buffers Tools Insert Help

```
#!/bin/sh
# flush the tables to start off clean and fresh.
iptables -F FORWARD
iptables -F INPUT
iptables -F OUTPUT
# set the default input policy to DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
iptables -P INPUT DROP
# make sure the loop back is free and clear of obstruction...
iptables -A INPUT -i lo -p all -j ACCEPT
iptables -A OUTPUT -o lo -p all -j ACCEPT
# allow icmp traffic from your localnet
iptables -A INPUT -i eth0 -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -i eth0 -p icmp --icmp-type echo-reply -j ACCEPT
# drop remote lpd, smtp, smtp submission
iptables -A INPUT -t filter -i eth0 -p tcp --dport 515 -j DROP
iptables -A INPUT -t filter -i eth0 -p udp --dport 69 -j DROP
iptables -A INPUT -t filter -i eth0 -p tcp --dport 25 -j DROP
iptables -A INPUT -t filter -i eth0 -p tcp --dport 587 -j DROP
# add the blank-policy for allowing established connections and blocking incoming new ones
iptables -A INPUT -t filter -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -t filter -i eth0 -m state --state NEW,INVALID -j DROP
##########################################################################

--:**  iptables.example   20:43 0.04   (Shell-script[sh])--L25--C0--All---------------------------
```

firewall verification

- firewall verification tools should be used to verify your setup. (ie: the native tools, lsof, nmap)

```
~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     icmp --  anywhere             anywhere            icmp echo-request
ACCEPT     icmp --  anywhere             anywhere            icmp echo-reply
DROP       tcp  --  anywhere             anywhere            tcp dpt:printer
DROP       udp  --  anywhere             anywhere            udp dpt:tftp
DROP       tcp  --  anywhere             anywhere            tcp dpt:smtp
DROP       tcp  --  anywhere             anywhere            tcp dpt:submission
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
DROP       all  --  anywhere             anywhere            state INVALID,NEW

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
```

```
# lsof -i
COMMAND    PID   USER   FD   TYPE DEVICE SIZE NODE NAME
lpd       1693   root    6u  IPv4   2258       TCP *:printer (LISTEN)
sshd      1727   root    3u  IPv4   2301       TCP *:ssh (LISTEN)
dhclient  2566   root    9u  IPv4   6196       UDP *:bootpc

# nmap -P0 localhost
Starting nmap 3.75 ( http://www.insecure.org/nmap/ ) at 2005-01-26 20:57
   CST
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1661 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
22/tcp  open  ssh
515/tcp open  printer

Nmap run completed -- 1 IP address (1 host up) scanned in 0.366 seconds
```