Iptables - practical

Oskar Andreasson
blueflux@koffein.net

Introduction

- The speaker
- A brief Table of Contents

The speaker - that is me

- Oskar Andreasson
- From Sweden
- Used Linux since 1994.
- Written about Iptables since 2.4 kernels
- My reliability
  - I am here as a private person.
  - No companies in my back.
    - I will say what I like and don’t like.
    - I am not here to sell.

A brief Table of Contents

Table of Content

- Introduction - finished after this slide
- Iptables - what is it
- Packet traversal
- Iptables syntax
- A simple example ruleset
- Final notes

Iptables - what is it

Table of Content

Iptables - The Linux 2.4 IP filter solution

Where did it come from

- BSD -> Linux 2.0 (ipfw)
- Linux 2.0 (ipfw) -> Linux 2.2 (ipchains)
  - Rusty Russell
- Linux 2.2 (ipchains) -> Linux 2.4 (iptables)
  - Rusty Russell
  - Netfilter core team
  - Others

Iptables - The Linux 2.4 IP filter solution

By whom was it written

- The Netfilter core team
  - A small group of large contributors (single persons).
  - The main group of people developing Netfilter/iptables.
  - Governs the main iptables tree.

- The core team consists of:
  - Rusty Russell
  - Jozsef Kadleczic
  - Marc Boucher
  - James Morris
  - Harald Welte

- Others:
  - Anyone with the time or will to contribute.

Iptables - Basic functionalities - IP Filter

IP Filter

- Used to filter packets
- The command to enter a rule is called iptables
- The framework inside kernel is called Netfilter
- Full matching on IP, TCP, UDP and ICMP packet headers
- Lesser matching on other packet headers possible
- Exception in TCP is the Options field

IP Filter rule

- Insertion point
- Match
- Target

Iptables - Basic functionalities - Stateful firewalling

Full state matching

- TCP
- UDP
- ICMP

Other protocols

- Uses a generic connection tracking module
- The generic conntrack module is less specific
- It is possible to write your own conntrack modules
- Certain protocols are "complex"
  - Requires extra modules called "conntrack helpers"
  - Examples are FTP, IRC (DCC), AH/ESP and ntalk

Iptables - Basic functionalities - Stateful firewalling (cont.)

Userland states

- NEW
  - All new connections
  - Includes Non SYN TCP packets
- ESTABLISHED
  - All connections that has seen traffic in both directions
- RELATED
  - All connections/packets related to other connections
  - Examples: ICMP errors, FTP-Data, DCC
- INVALID
  - Certain invalid packets depending on states
  - E.g. FIN/ACK when no FIN was sent

Iptables - Basic functionalities - NAT

NAT - Network Address Translation

The science of switching Source or Destination Addresses

Two types of NAT in Linux 2.4

- Netfilter NAT
- Fast NAT

Prohibited in IPv6
Next to a must in IPv4

Usages

- Making a LAN look as if it came from a single source (the firewall)
- Creating separate servers with a single IP

Iptables - Basic functionalities - NAT (cont.)

Netfilter NAT

- DNAT - Destination Network Address Translation
- SNAT - Source Network Address Translation
- Requires Connection tracking to keep states and expectations

Iptables - Basic functionalities - Packet Mangling

Mangling packets going through the firewall

Gives you the ability to a multitude of possibilities.

Example usages

- Strip all IP options
- Change TOS values
- Change TTL values
- Strip ECN values
- Clamp MSS to PMTU
- Mark packets within kernel
- Mark connections within kernel

Iptables - What iptables is not

Not a proxy solution
- Very common misconception
- Use squid instead

Not a packet data filtering solution
- Very closely related to the above problem
- Use squid and snort for this kind of usage

A complete firewall
- Lacks several features, which should always reside in userspace
  - A good NIDS (snort)
  - A filtering proxy solution (squid)

Iptables - What this means in reality

A framework for filtering connections

- Via the filter table
- Powerful and flexible

A framework for accounting

- Via the filter table
- Using the built in packet and byte counters

A simple way to do Network Address Translation

- Good flexibility
- Possible to use even for complex protocols

Ability to mangle packets

- Extremely powerful
- Useful for all sorts of situations

Packet traversal

Table of Content

- Introduction
- Tables
- How they hook together
- Traversal of a single chain

Packet traversal - Introduction

How a packet traverses the inside of the kernel

- Extremely important to understand
- Horrible mistakes possible

3 basic tables

- filter (default)
- nat
- mangle

Each table contains a number of chains
Userspecified chains may be specified in a table

The main chains may then call the userspecified chains

Packet traversal - Tables

Filter table

- Used for filtering
- Contains 3 chains
  - INPUT
  - OUTPUT
  - FORWARD
- Certain targets may not be used here
  - NAT targets
  - Mangle targets
  - Filtering targets works perfectly

Packet traversal - Tables

Nat table

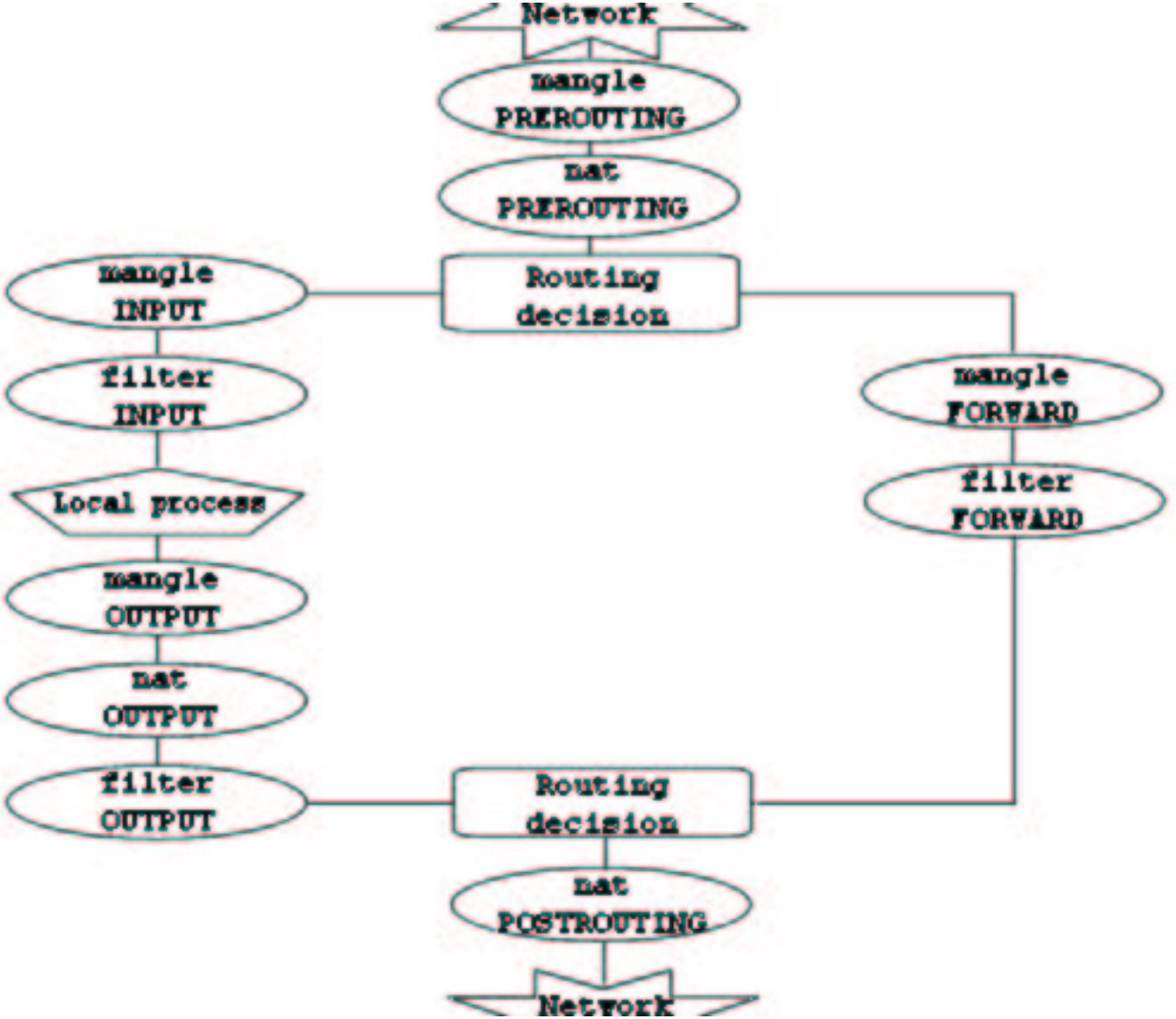
- Used for Network Address Translation
- Only the first packet of a connection hits this table
  - Subsequent packets in the connection has the same action taken
  - Avoid pure filtering in this chain!
- Contains 3 chains
  - PREROUTING
  - POSTROUTING
  - OUTPUT

Packet traversal - Tables

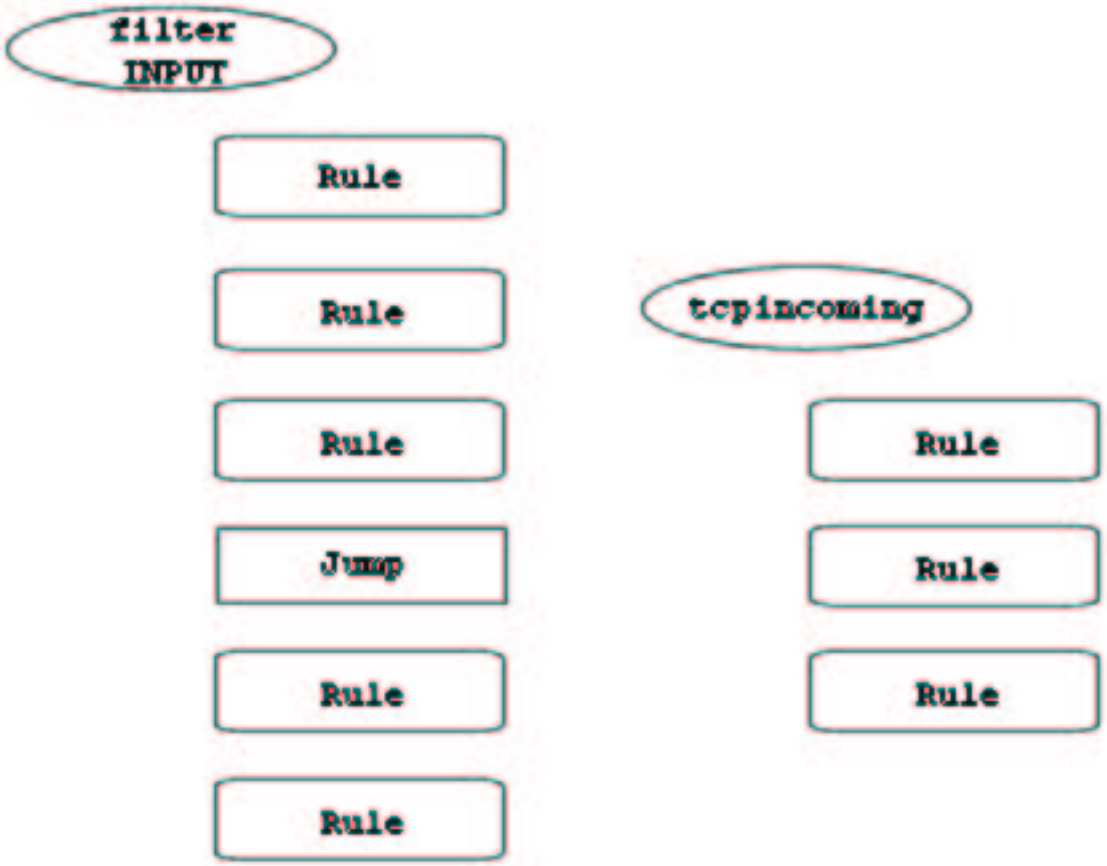
Mangle table

- Used for mangling packets
- Only the first packet in a connection hits this table
  - Same as for the nat table
- Contains 3 or 5 chains
  - PREROUTING
  - POSTROUTING
  - OUTPUT
  - INPUT (with mangle5hooks patch or new kernel)
  - FORWARD (same here)

Packet traversal - How they hook together



Packet traversal - Traversal of a single chain



Iptables syntax

Table of Content

Iptables syntax - The basic iptables syntax

iptables [command] [options] <matches> <target>

- Commands:
  - append, insert, replace, delete, list, policy, etc.
- Options:
  - verbose, line numbers, exact, etc.
- Matches:
  - dport, dst, sport, src, states, TCP options, owner, etc.
- Targets:
  - ACCEPT, DROP, REJECT, SNAT, DNAT, TOS, LOG, etc.

Iptables syntax - A few matches

Protocol
-p, --protocol [!] [protocol]

- tcp, udp, icmp or all
- Numeric value
- /etc/protocols

Destination IP & Port
-d, --destination [!] address[/mask]

- Destination address
- Resolvable (/etc/resolve.conf)

--dport, --destination-port [!] port[:port]

- Destination port
- Numeric or resolvable (/etc/services)
- Port range

Iptables syntax - A few matches (cont.)

Source IP & Port

-s, --source [!] address[/mask]

- Source address
- Resolvable (/etc/resolve.conf)

--sport, --source-port [!] port[:port]

- Source port
- Numeric or resolvable (/etc/services)
- Port range

Iptables syntax - A few matches (cont.)

Incoming and Outgoing interface

-i, --in-interface [!] interface

- Input interface
- + mask

-o, --out-interface [!] interface

- Output interface
- + mask

Iptables syntax - Some targets

ACCEPT

- Accepts the packet
- Ends further processing of the specific chain
- Ends processing of all previous chains
- Except other main chains and tables

DROP

- Drops the packet
- No reply
- Ends all further processing

Iptables syntax - Some targets (cont.)

REJECT
- Drops packet
- Returns a reply
  - User specified reply
  - Calculated reply
  - TCP-RST or ICMP errors
- Ends all further processing

RETURN
- Returns from a chain to the calling chain

Iptables syntax - ... and a few simple rules

iptables -A INPUT -p tcp -m state --state NEW ! --syn -j REJECT --reject-with tcp-reset

iptables -A INPUT -p tcp --dport 80:1024 -j DROP

iptables -A FORWARD -p tcp --dport 22:113 -j DROP

iptables -A FORWARD -p tcp --dport ftp-data:ftp -j DROP

iptables -A OUTPUT -p tcp -o eth0 -j ACCEPT

iptables -A OUTPUT -p tcp -o lo -j ACCEPT

iptables -P OUTPUT DROP

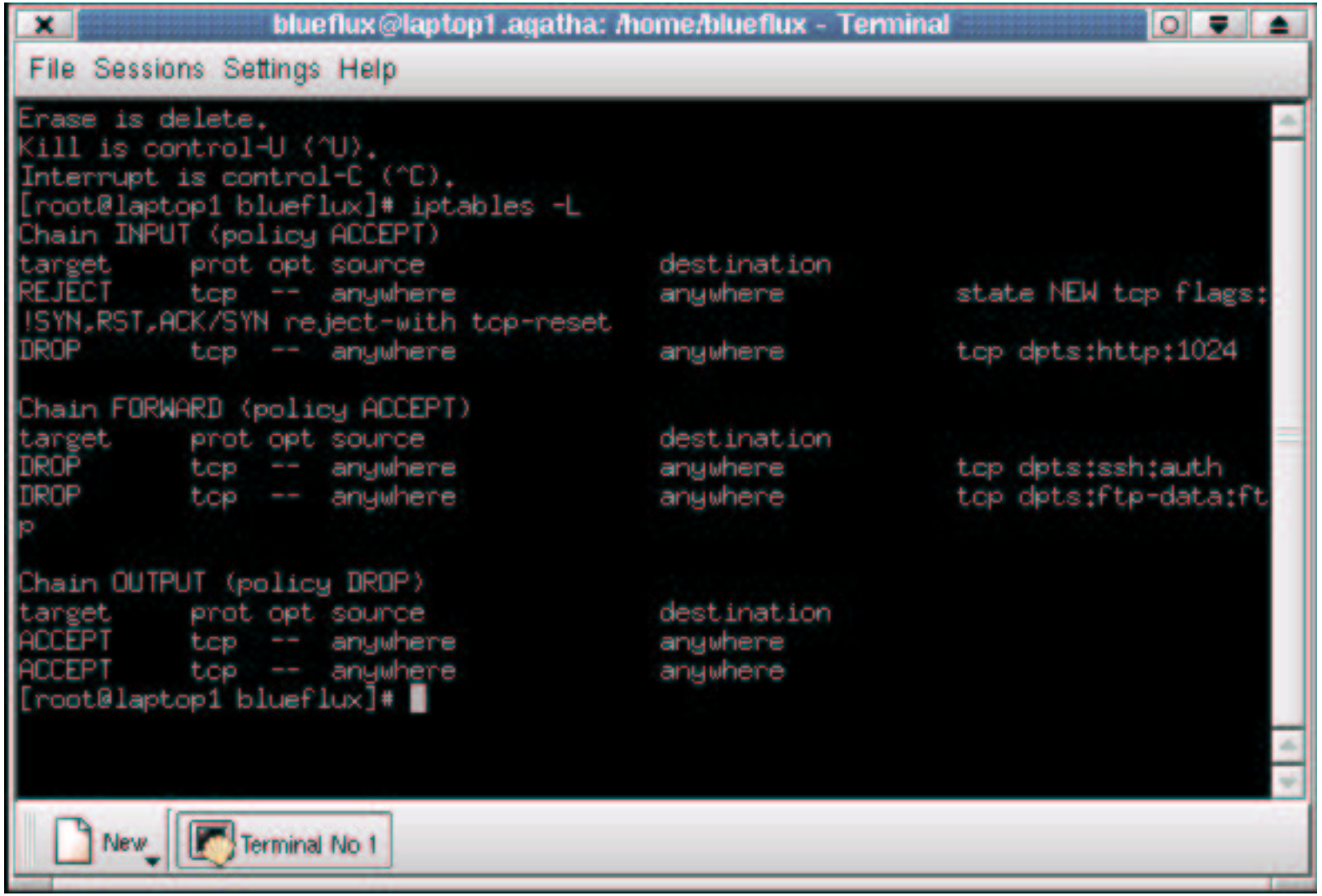
Iptables syntax - Listing the rules

-L, --list [chain]
- Lists ruleset in a table

-n, --numeric
- Turns off name resolutions

-v, --verbose
- Verbose output

Iptables syntax - Listing the rules



Iptables syntax - Flushing the ruleset

-F, --flush [chain]

- Flushes (erases) all rules in a chain
- Or a table

iptables -F INPUT

iptables -F

Iptables syntax - Creating & Deleting user-created chains

-N, --new chain

- Creates a user-specified chain
- There must be no target with that name previously

-X, --delete-chain [chain]

- Deletes a user-created chain
- No rules may reference the chain
- Can delete all user-created chains in a table

Iptables syntax - Creating & Deleting user-created chains (cont.)

Creating...
iptables -t filter -N badtcppackets

and Deleting a chain
iptables -t filter -X badtcppackets

and Deleting all user-created chains
iptables -t filter -X

A simple example ruleset

Table of Content

A simple example ruleset - The goals

The firewall
- Will act as its own firewall
- Incoming:
  - ICMP Echo request & reply
  - Identd requests
  - HTTP requests
- Outgoing:
  - Everything generated by the host
  - Except "nonet" group

And a LAN
- From Internet to LAN
  - Related traffic
  - Established traffic
- From LAN to Internet
  - Everything

A simple example ruleset - The technical details

Firewall
- LAN on eth0
- LAN IP 192.168.1.1
- Internet on eth1
- Internet IP 10.0.0.1/32

LAN
- IP range 192.168.1.0/24

A simple example ruleset - The POSTROUTING chain

We need SNAT to let our LAN out on the Internet
Without this, the Internet don’t know where to route the packets

```
iptables -t nat -A POSTROUTING -i eth0 -o eth1 -j SNAT \
--to-source 10.0.0.1
```

A simple example ruleset - The INPUT chain

Need to allow all incoming traffic specified in goals
Need to allow return traffic for everything we send
Default to DROP

```
iptables -P INPUT DROP
iptables -A INPUT -p tcp --dport 113 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

A simple example ruleset - The OUTPUT chain

Accept everything except the nonet group to leave

```
iptables -A OUTPUT -m owner --gid-owner nonet -j DROP
```

A simple example ruleset - The FORWARD chain

Everything from LAN to Internet
ICMP replies, related and Established traffic from Internet to LAN

```
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -m state \
--state ESTABLISHED,RELATED -j ACCEPT
```

A simple example ruleset - And the complete ruleset

```
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT

iptables -t nat -A POSTROUTING -i eth0 -o eth1 \
-j SNAT --to-source 10.0.0.1

iptables -A INPUT -p tcp --dport 113 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED \
-j ACCEPT

iptables -A OUTPUT -m owner --gid-owner nonet -j DROP

iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -m state \
--state ESTABLISHED,RELATED -j ACCEPT

echo 1 > /proc/sys/net/ipv4/ip_forward
```

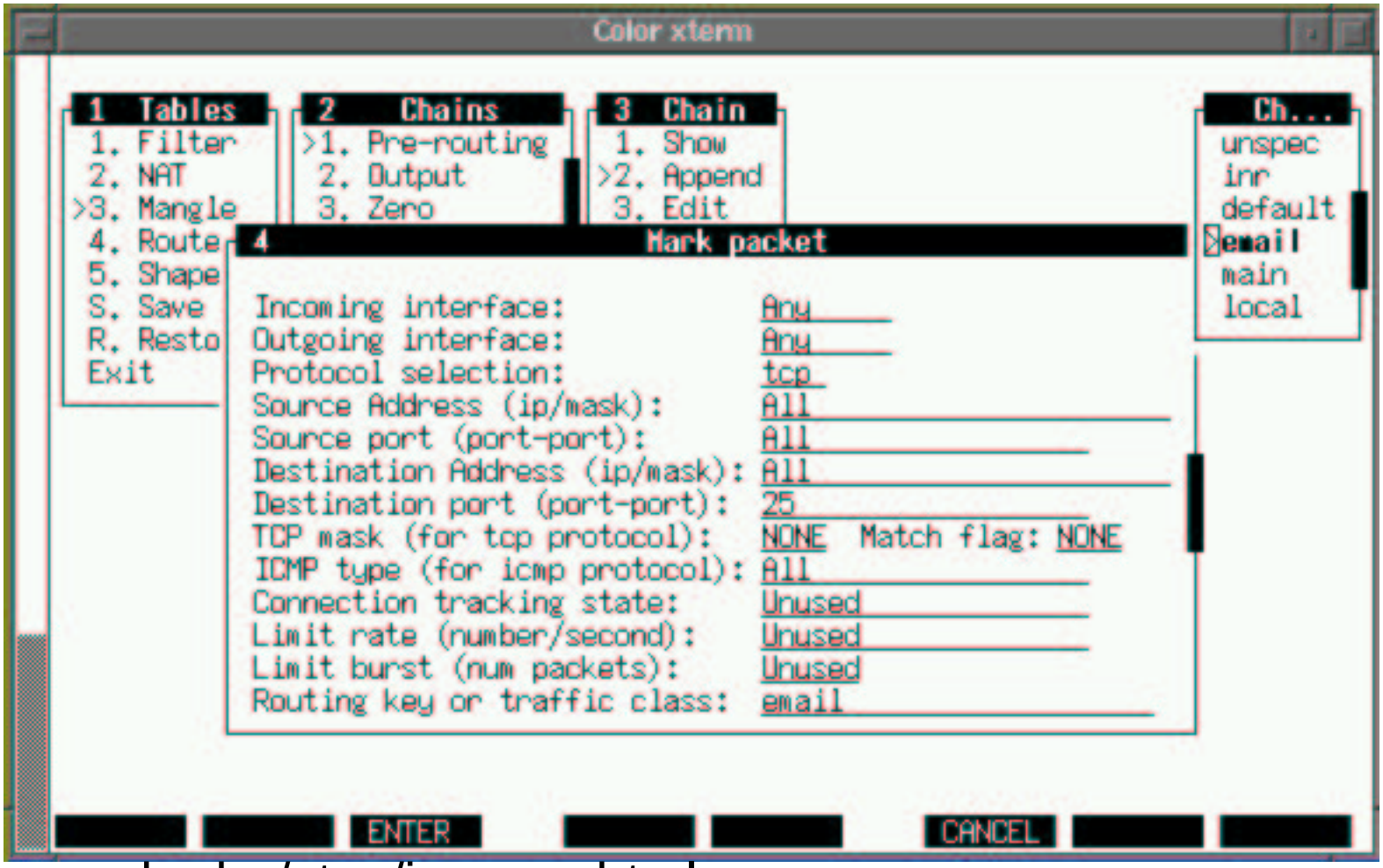
Final notes

Table of Content

- Graphical User Interfaces
- Other resources

Final notes - Graphical User Interfaces

IP Menu



http://users.pandor.be/stes/ipmenu.html

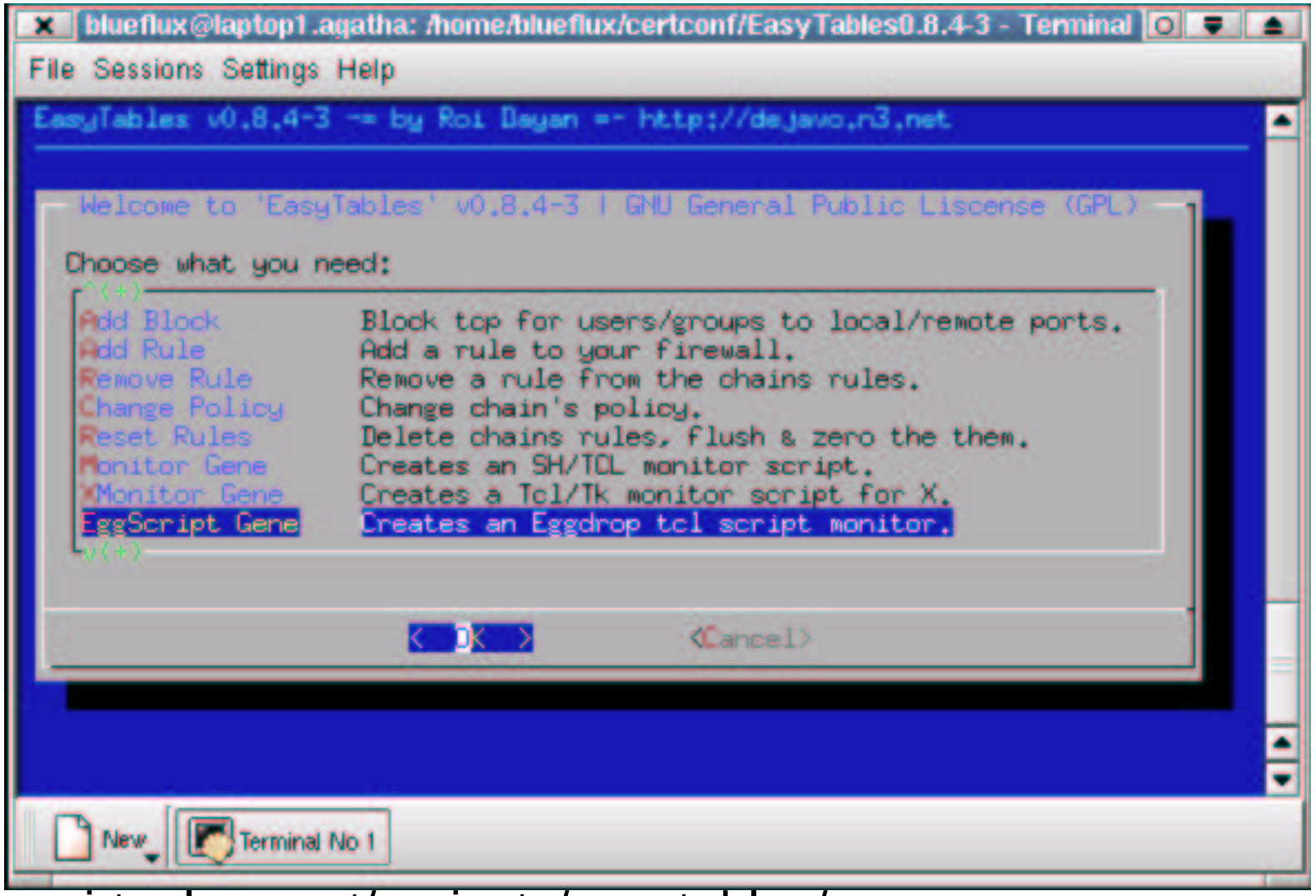
Final notes - Graphical User Interfaces

fwbuilder

http://www.fwbuilder.org

Final notes - Graphical User Interfaces

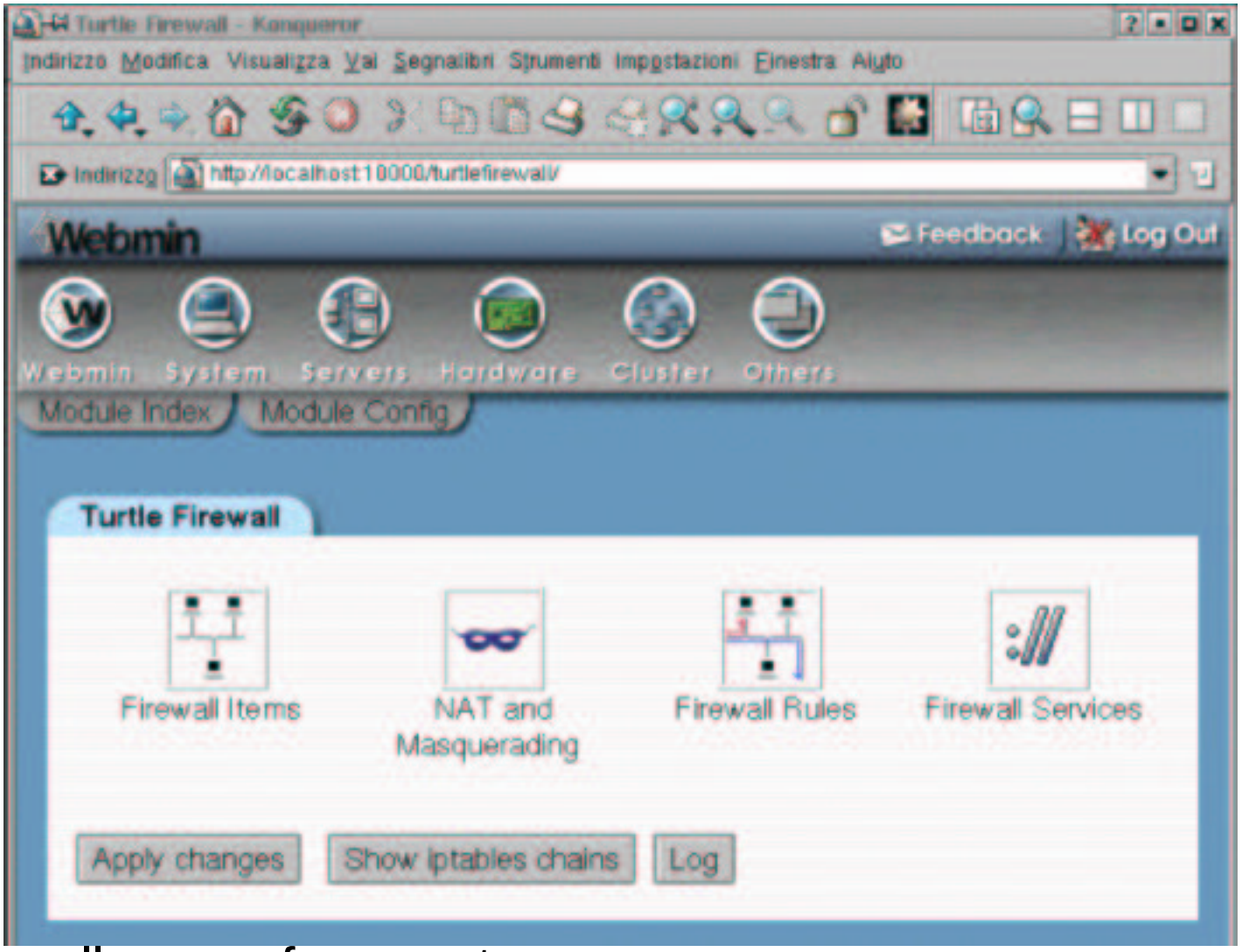
EasyTables



http://dejavo.virtualave.net/projects/easytables/

Final notes - Graphical User Interfaces

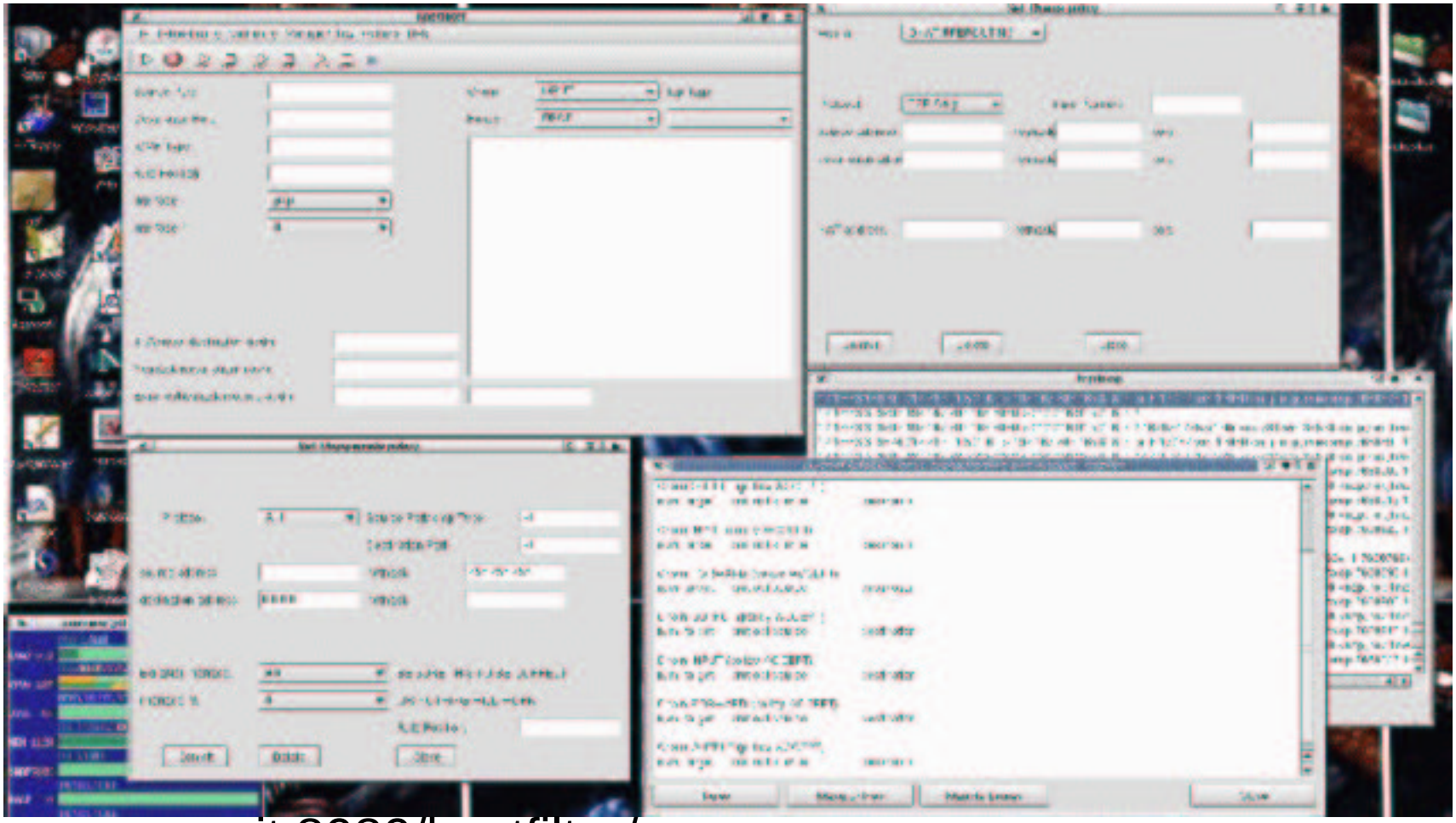
Turtle Firewall



http://turtlefirewall.sourceforge.net

Final notes - Graphical User Interfaces

KNetfilter



http://expansa.sns.it:8080/knetfilter/

Final notes - Other resources

http://www.netfilter.org

http://iptables-tutorial.haringstad.com

http://www.linuxguruz.org/iptables/

http://www.islandsoft.net/veerapen.html

http://www.lartc.org

http://www.docum.org