



magic moving pixel s.a.

Linux firewall in a corporate environment

Linuxdays Luxembourg • October 25, 2001

Werner Eiden (werner.eiden@mmp.lu)



Linux Firewall

Introduction



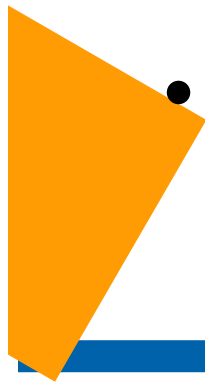
Firewall Classifications

- **Packet Filtering Systems** (static, dynamic)
Flow control of IP data packets between networks
- **Application Gateways**
IP application level proxy services
- **Combining systems to new architectures**
 - Screened hosts
 - Screened Subnet
 - ...



What does Linux provide?

- Linux Kernel based firewalling features
 - netfilter/iptables based stateful packet filtering
 - Policy routing
 - Other Kernel options, parameters, modules, patches
- Application based firewalling features
 - Application proxies running on Linux (e.g. HTTP, DNS, SMTP...)
- Linux Firewall is basically associated with its packet filtering facilities: **ipchains** and **iptables**.



Linux Firewall (R)evolution

kernel module	configuration tool
ipfw	ipfwadm (kernel 1.1.x, 2.0.x) - 1994
ipchains	ipchains (kernel 2.2x) - 1998
netfilter	iptables (kernel 2.4.x) - 1999



Linux Firewall

netfilter / iptables
Stateful Packet Filtering engine



Netfilter/iptables Feature Highlights

- **Packet Filtering with Stateful Inspection**
- **Full NAT Support**
 - Source NAT & Destination NAT
- **Packet Mangling**
 - Manipulate Type of Service (TOS) value
- **Improved Logging**
 - Log messages & Log limits
- **Extensible**
 - the kernel & the iptables tool can be extended



What is Netfilter / iptables?

- **netfilter** is the packet filtering engine in the Linux kernel (as a set of kernel modules or compiled into it).
- **iptables** tells netfilter what packets to filter. It manipulates rules in the kernel's packet filtering table(s).
- iptables is available as command line interface command.



iptables: Rule Components

- **Table:** filter, nat, mangle
- **Chain:** Built-in or user defined
- **Rule Specification with Match Extension(s) and Target (Extension)**
- Structure:
iptables [**<table>**] <chain-spec> [**<rule-spec>**]
- Example:
iptables **-t filter** -A INPUT **-p icmp -j DROP**

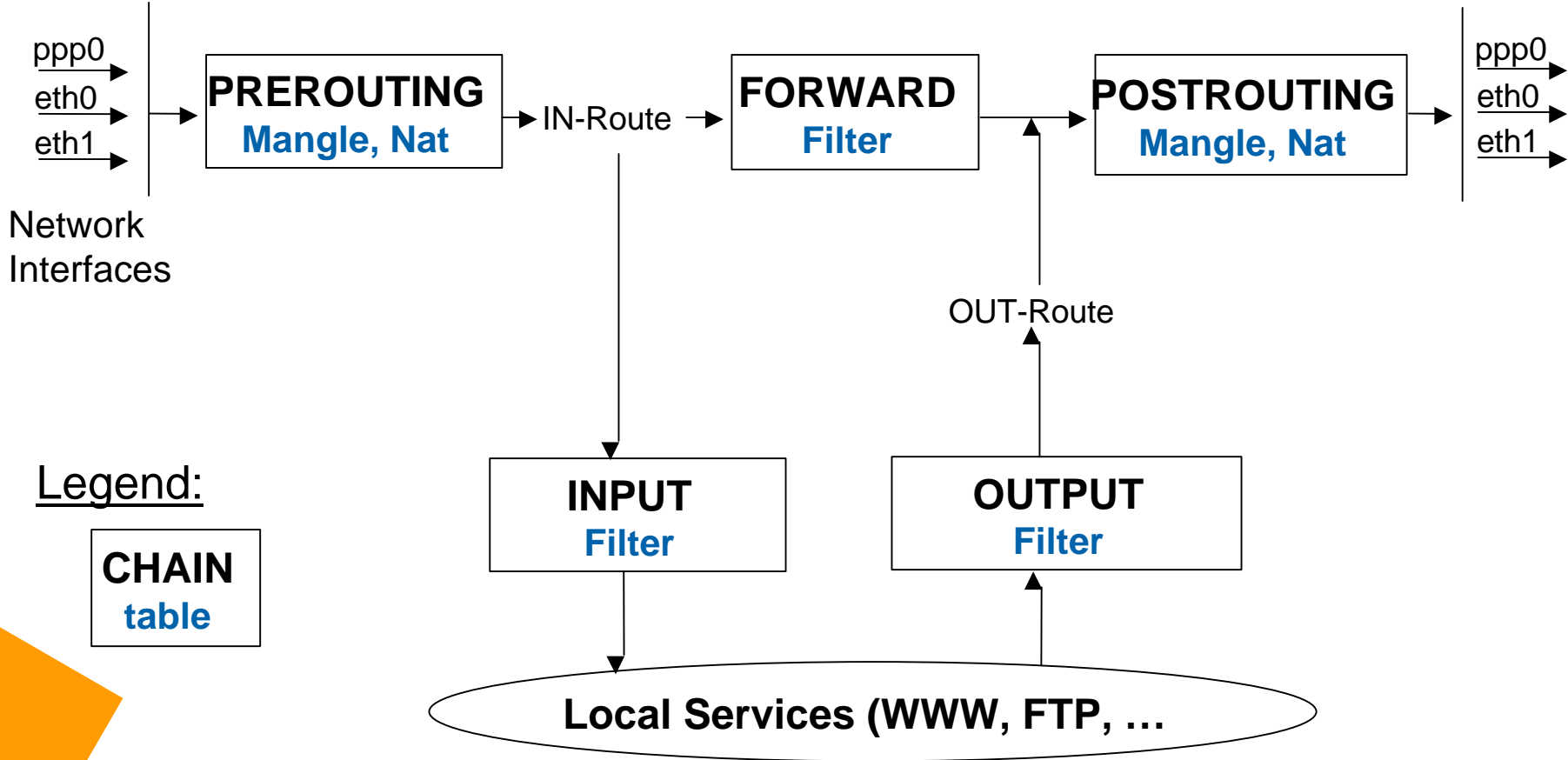


iptables : Tables & Chains

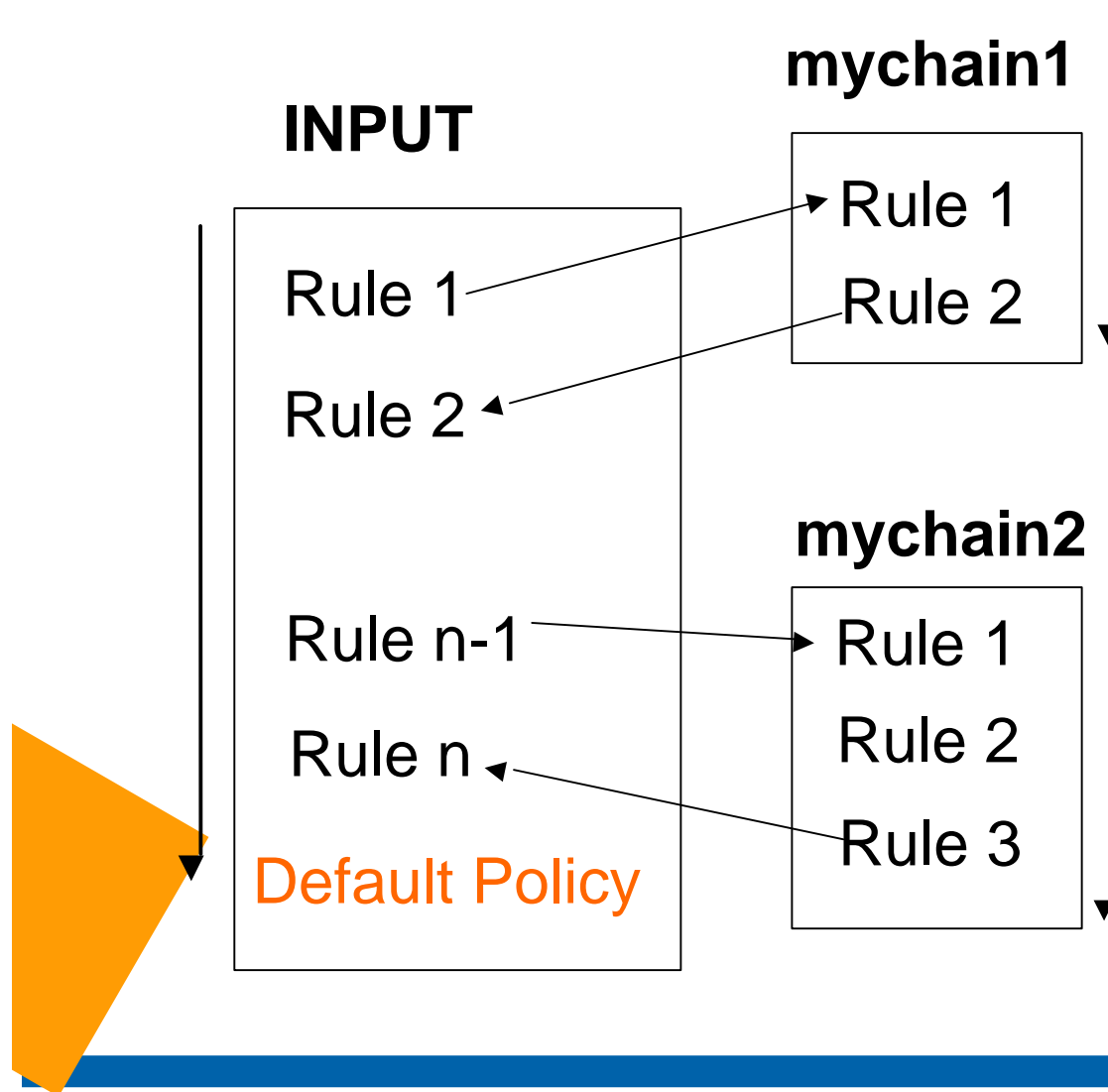
- Rules are organized in tables and chains
- **Tables:**
 - **Filter:** all rules related to packet filtering
 - **Nat:** all rules related to Network Address Translation
 - **Mangle:** all rules for specialized packet alteration
 - Default table: Filter
- **Table chains:**
 - **Filter:** INPUT, OUTPUT, FORWARD, user-defined
 - **Nat:** PREROUTING, POSTROUTING
 - **Mangle:** PREROUTING, POSTROUTING



iptables packet flow chart



Iptables: chains



Rule Sequence!

Chains can be

- created
- flushed
- deleted

Built-in chains do have a default policy

Rules can be

- inserted/appended
- deleted


into/from chains

iptables

Rule Specifications & Match Extensions



iptables: Rule Specifications/Matches

- Apply Rule Specifications & **Match Extensions**
 - Rule Specification Parameters:
 - --protocol, -p (tcp, udp, icmp, all – or protocol number)
 - --source, -s (ip address/mask)
 - --destination, -d (ip address/mask)
 - --in-interface, -i, --out-interface, -o (name)
 - --fragment, -f
 - --jump, -j (target)
 - Match Extensions exists for:
tcp, udp, icmp, multiport, mac, limit, mark, owner, state, unclean, tos
-
- 

Match Extension: tcp

- Match Extension:
-p tcp --sport [!] port[:port] --dport [!] port[:port]
--tcp-flags [!] mask comp [!] --syn
--tcp-option [!] number
- TCP-Flags:
SYN, ACK, FIN, RST, URG, PSH, ALL, NONE
Match if TCP flags are as specified.
- Syn flag
Match if the **SYN** flag is set and ACK, FIN and RST flags unset.



Match Extension: tcp

- Example: Only accept smtp packets to mailserv

```
# iptables -A INPUT -p tcp --dport smtp  
--jump ACCEPT
```

- Example: Only accept packets that have the SYN flag set and all the others unset:

```
# iptables -A INPUT -p tcp --tcp-flags ALL SYN  
--jump ACCEPT
```



Match Extension udp, icmp

- Match Extension: **udp**
 - p udp --source-port [!] port[:port]
 - destination-port [!] port[:port]
- Match Extension: **icmp**
 - p icmp --icmp-type [!] <type-name>



Match Extension: multiport

- Match Extension:
 - `--source-port port[,port]`
 - `--destination-port port[,port]`
 - `--port port[,port]`
- Only in conjunction with `-p tcp` or `-p udp`
- Allows to specify up to 15 ports in a comma separated list
- `--port`: matches is both the source and destination ports are equal to each other and are one of the given ports



Match Extension: mac

- Match extension:
`-m mac --mac-source [!] <mac-address>`
- Used for matching incoming packet's source Ethernet (MAC) address
- Applies to packets traversing the PREROUTING, FORWARD and INPUT chain
- Example: Drop all packets from a specific host located in the same LAN



```
# iptables -A INPUT -m mac --mac-source  
00:60:08:91:DF:A2 --jump DROP
```

Match Extension: limit

- Match Extension:
-m limit --limit rate ('/<second>' | '/<minute>' | '/hour' | '/day') --limit-burst <number>
- Restrict the rate of matches to a given number of times per second / minute / hour or day.
Default: 3/hour
- Limit Burst indicates the maximum burst before the limit kicks in. **Default: 5**




Match Extension: limit

- Example: Rate limited logging
combine match limit with LOG target

```
# iptables -A INPUT -p tcp --dport ftp -m  
limit --limit 10/hour --limit-burst 20 -j LOG
```

Logs the first 20 matches. After this it will take 6 minutes till the next match will be logged.

The burst limit will be recharged after $20 * 6$ minutes if no further packet matches during the next 120 minutes.



Match Extension: limit

Rate Limiting & Denial of Service Attacks

- Syn-Flood Protection

```
# iptables -A FORWARD -p tcp --dport 21 --syn -m  
limit --limit 3/s -j ACCEPT
```

- Furtive port scanner


```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,  
FIN,RST RST -m limit --limit 1/s -j ACCEPT
```

- Ping of death

```
# iptables -A FORWARD -p icmp --icmp-type echo-  
request -m limit --limit 1/s -j ACCEPT
```



Match Extension: state

- Also known as: **Stateful Inspection**
 - Associate a packet with a particular connection
 - Kernel maintains a connection state table
 - Timeout parameter are compiled into the kernel
 - Connection tracking defragments all packets
 - Match extension:
-m state --state <connection-state>
 - Connection States:
NEW, INVALID, ESTABLISHED, RELATED
- 
-
-

Stateful Inspection: Example 1

- **Protect LAN from Internet**

Only forward incoming packets from the external interface that belong to an existing connection.

```
# iptables -A FORWARD --in-interface eth0 -m state  
-state ESTABLISHED, RELATED -j ACCEPT
```

- **eth0**: Interface connected to Internet
 - Rule applies to all protocols (tcp, udp, icmp)
- 
-

Stateful Inspection: Example 2

- **UDP connection tracking**
- Problem: No TCP Sequence numbers, thus "stateless".
- Requires a rule that matches NEW out-bound connections to make an entry in the state table

```
# iptables -A INPUT -p udp -m state --state  
ESTABLISHED -j ACCEPT
```

```
# iptables -A OUTPUT -p udp -m state --state NEW,  
ESTABLISHED -j ACCEPT
```



Stateful Inspection: Example 3

- **ICMP connection tracking**

Only 4 types of icmp that can be categorized as NEW or ESTABLISHED:

- Echo request / reply
- Timestamp request / reply
- Information request / reply
- Address mask request / reply



```
# iptables -A OUTPUT -p icmp -m state --state NEW,  
ESTABLISHED, RELATED -j ACCEPT
```

```
# iptables -A INPUT -p udp -m state --state  
ESTABLISHED, RELATED -j ACCEPT
```

iptables

Rule Targets & Target Extensions



Targets and Target Extensions


- Structure: **--jump <target> [<target-option>]**
- Built-in targets:
ACCEPT, DROP, QUEUE, RETURN
- Target modules (extensions):
REJECT, MASQUERADE, SNAT, DNAT, MIRROR, REDIRECT, MARK, TOS, LOG
- **Terminal targets** terminate a chain
- User-defined chains as a target name
- Target option example:
--jump REJECT --reject-with icmp-host-unreachable

Filtering & Policy Rules via Targets

- Basic Packet Filtering uses:
ACCEPT, DROP, REJECT
- Packets are not modified by these targets.
- ACCEPT & DROP are used to define a chain's **default policy** if no rule in the chain matches at all. A default policy is defined by Policy Rule.
- Example: **# iptables -P FORWARD DROP**



NAT via Target Extensions

- Network Address Translation (NAT)
Target Types:
 - **SNAT & MASQUERADING**
 - **DNAT & REDIRECT**
 - Targets only valid in `nat` table
 - **SNAT & MASQUERADING** is done in the **POSTROUTING** chain
 - **DNAT & REDIRECT** is done in the **PREROUTING** chain
- 
-

NAT: SNAT & MASQUERADING

SNAT: Source NAT

- Modifies the source IP address of all outgoing packets using a specified address or address range, including a port or port range.
- Used if only static IP addresses are involved (e.g. Internet link with leased line).

MASQUERADING is a special form of SNAT

- Should only be used for dial-up connections (dynamically assigned IP addresses).
- 
-
-

NAT: SNAT & MASQUERADING

SNAT Examples

- `iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 172.16.1.1`
- `iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 172.16.1.1 - 172.16.1.5`
- `iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to 172.16.1.1 :1-1023`

MASQUERADE Example:



```
iptables -t nat -A POSTROUTING -o ppp0  
-j MASQUERADE
```

NAT: DNAT & REDIRECT

DNAT: Destination NAT

- Modifies the destination IP address of all incoming packets using a specified address or address range, including a port or port range.
- Support of multiple Servers ("port forwarding")
- Load Sharing (see: <http://linuxvirtualserver.org/>)
- Transparent Proxying

REDIRECT is a special form of DNAT

- Redirects incoming packets to the IP address of the incoming interface.
-
-

NAT: DNAT & REDIRECT

- **DNAT Examples**


```
iptables -t nat -A PREROUTING -i eth0 -j DNAT  
--to 192.168.1.5
```

```
iptables -t nat -A PREROUTING -i eth0 -j DNAT  
--to 192.168.1.5 - 192.168.1.10
```

```
iptables -t nat -A PREROUTING -p tcp --dport  
80 -i eth0 -j DNAT --to 192.168.1.5:8080
```

- **REDIRECT Example**

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --  
dport 80 -j REDIRECT --to-port 3128
```



Logging via Target Extension

- LOG Target :
**-j LOG --log-level <level> --log-prefix <string>
--log-tcp-sequence --log-tcp-options --log-ip-options**
- Log Level: 'debug', 'info', 'notice', 'warning', 'err', 'crit', 'alert' and 'emerg'
- Log Prefix: This string is sent at the start of the log message to have a unique identifier
- Example:
**iptables -A FORWARD -p tcp --dport telnet
-m state --state NEW -j LOG --log-level notice**



Packet Mangling with Targets

- Ability to change/modify packets **before** they are routed and **after** routing has been decided.
- Thus Mangling is done in the PREROUTING and POSTROUTING chain.
- Targets only valid in **mangle** table. Targets are:
 - **Mark** (mark a packet)
`--jump MARK --set-mark <mark>`
 - **TOS** (set type of service)
`--jump TOS --set-tos <tos-value>`



Linux Firewall: Kernel Parameters

- Turn on IP Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
- Turn on anti spoofing protection
**for f in /proc/sys/net/conf/*/rp_filter
do echo 1 > \$f; done**
- others



Linux Firewall

Firewall Management



Organizing your Firewall

- Set-up your Linux box
- Minimize and harden the system
- Define your Security Policy
- Create a file that contains all iptables commands (firewall rules) as shell script
- Run the shell script during system start-up
- Monitor Log Files
- Fine Tune Security Policy
- Maintain System Security



Supporting Tools

- **Rule Policy Editor**

Many freeware implementations but none of them provides full feature support for iptables.

Test:

- IPMenu (<http://users.pandora.be/stes/ipmenu.html>)
- Knetfilter (<http://expansa.sns.it/knetfilter/>)
- Firestarter (<http://firestarter.sourceforge.net/>)
- MonMotha's:
(<http://monmotha.mplug.org/firewall/index.php>)

and give it a chance.



Supporting Tools

- **Log file Analysis tools:**

Automate log analysis and alarms

Test:

- **Swatch:** <http://www.stanford.edu/~atkins/swatch>
generic analysis tool
- **Logcheck:** <http://www.psionic.com/abacus/logcheck/>
generic analysis tool
- **fwlogwatch:**
<http://www.kyb.unistuttgart.de/boris/software.shtml>
support for ipchains & iptables



Supporting Tools

- **Documentation**
 - Linux 2.4 Packet Filtering HOWTO
 - Linux 2.4 NAT HOWTO
 - Iptables 'man' page
 - <http://www.linuxguruz.org/iptables/>
 - ...
 - Much documentation is available for ipchains based Linux firewalls
 - Documentation of firewall concepts based on iptables lacks completeness



Challenge the puzzle!

Linux Firewall

The right firewall for you?



Linux Firewall

- Pro
 - Firewall features can compete with commercial firewalls
 - iptables is just the core of a complex firewall architecture
 - Extendable by additional applications (e.g. VPN)
 - Stable, Reliable, Scalable
 - Independent Support and Development
 - Fast bug fixing
 - Cost effective in the means of hard- and software



Linux Firewall

- **Contra**
 - Good or excellent system knowledge is required
 - Not an out-of-the-box solution
 - Still lacking well developed documentation (will be solved in the future)
 - Lack of integrated administration and management tools
 - PuzzleWare ;-)



Linux Firewall

Questions?

Discussion?

