Linux Networking: 
IP Packet Filter Firewalling

David Morgan

Firewall types

- Packet filter
- Proxy server
Linux “Netfilter” Firewallsing

- Packet filter, not proxy
- Centerpiece command: iptables
- Starting point: packet structure details

IP packet structure

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
<th>Protocol Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP’s Data Payload</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© David Morgan 2003,2004
Payload types - subprotocols

<table>
<thead>
<tr>
<th>Src</th>
<th>Dest</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>UDP (17) packet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Src</th>
<th>Dest</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TCP (6) packet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Src</th>
<th>Dest</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ICMP (1) packet</td>
</tr>
</tbody>
</table>

… and others

UDP packet structure

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UDP’s Data Payload</td>
</tr>
</tbody>
</table>

© David Morgan 2003,2004
UDP/IP packet structure

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
</tr>
<tr>
<td></td>
<td>UDP’s Data Payload</td>
</tr>
</tbody>
</table>

IP’s payload is a UDP packet

Address? Port?

- **Address number**
  - a **Machine** designator
  - identifies one among multiple machines on a network
    - 198.186.203.55 identifies linux.com on internet
    - 198.137.241.43 identifies whitehouse.gov

- **Port number**
  - a **Task** designator
  - identifies one among multiple tasks in a machine
    - 80 identifies web server running on linux.com
    - 22 identifies secure shell server running on linux.com
**Address? Port?**

Two address-port pairs uniquely define a process to process “conversation” across a network.

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
</tr>
</tbody>
</table>

E.g., when President Bush browses `linux.com`:

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>198.137.241.43</td>
<td>198.186.203.55</td>
</tr>
<tr>
<td>(whitehouse.gov)</td>
<td>(linux.com)</td>
</tr>
<tr>
<td>62102</td>
<td>80</td>
</tr>
</tbody>
</table>

**TCP packet structure**

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence #</td>
<td>Acknowledgment</td>
</tr>
<tr>
<td>TCP’s Data Payload</td>
<td></td>
</tr>
</tbody>
</table>

© David Morgan 2003,2004
TCP/IP packet structure

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
</tr>
<tr>
<td>Sequence #</td>
<td>Acknowledgment</td>
</tr>
</tbody>
</table>

IP’s payload is a TCP packet

TCP’s Data Payload

ICMP packet structure

<table>
<thead>
<tr>
<th>ICMP-type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>header of subject/wayward IP packet or other</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICMP-type dependent payload</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© David Morgan 2003,2004
ICMP/IP packet structure

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP-type</td>
<td>Code</td>
</tr>
<tr>
<td></td>
<td>Checksum</td>
</tr>
</tbody>
</table>

header of subject/wayward IP packet
or other
ICMP-type dependent payload

IP’s payload is an ICMP packet

UDP packet example

Used for internet name service inquiries

```
[root@EMACH1 /root]# ping www.acme.com
```

63370 53

“www.acme.com’s IP address please”
TCP packet example

Used for webserver-to-webbrowser traffic

<table>
<thead>
<tr>
<th>80</th>
<th>1050</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence #</td>
<td>Acknowledgment</td>
</tr>
</tbody>
</table>

website text & images

ICMP packet example

Used for echo (ping) requests…

```
[root@EMACH1 /root]# ping www.acme.com
```

<table>
<thead>
<tr>
<th>8</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>echo request – “are you there?”</td>
<td></td>
</tr>
</tbody>
</table>
ICMP packet example

… and for echo (ping) relies.

<table>
<thead>
<tr>
<th>0</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

echo reply – “yes I’m here”

Firewall = ruleset

- An in-memory datastructure by whose elements packets that appear at interfaces are evaluated
- A corresponding series of commands, each invocation of which populates the table with a single element
- Elements are called “rules”
Firewall - iptables

- low level - iptables command (to compose individual rule)
- middle level - Ziegler firewall (a particular ruleset)
- high level – firewall.local (custom extensions to firewall)

Iptables organization

- Tables (have chains)
  - filter table
  - nat table

- Chains (contain rules)
  - filter
    - INPUT chain
    - OUTPUT
    - FORWARD
  - nat
    - PREROUTING chain
    - POSTROUTING
An Individual Rule

- Condition - Examines and qualifies a packet
- Action - Operates on the packet if it qualifies

What a Rule says

- “If a packet’s header looks like this, then here’s what to do with the packet”
- “looks like this” e.g.
  - goes to a certain (range of) address(es) or
  - uses the telnet port, 23 or
  - is an ICMP packet
- “what to do” e.g.
  - pass it
  - discard it
iptables -t filter -A OUTPUT -o eth1 -p tcp --sport 23 --dport 1024:65535 -s 192.168.4.0/24 -d 0.0.0.0/0 –j ACCEPT

- Table for this rule
- Rule action
  - A add rule to chain/list
  - D delete rule from chain/list
  - P default policy for chain/list
- Rule chain/list (tables contain chains)
  - INPUT
  - OUTPUT
  - FORWARD

- Packet qualifiers
  - By interface and direction
  - protocol
  - source port number(s)
  - destination port number(s)
  - source address (range)
  - destination address (range)

- Packet disposition
  - ACCEPT
  - DROP
  - REJECT
  - SNAT
  - DNAT

What a Chain is

- ordered checklist of regulatory rules
  - Multiple rules, for packets with particular characteristics
  - Single rule for default (catch-all) policy

- operation
  - Packet tested against rules in succession
    - First matching rule determines “what to do” to packet
  - If packet matches no rule
    - Chain’s default policy determines “what to do” to packet
Multiple chains

- Input chain
  - When arriving at an interface, do we let a packet come in?
- Output chain
  - When departing from an interface, do we let a packet go out?
- Forwarding chain
  - When traversing this machine to another, do we let a packet pass between interfaces?

Filter traversal by packets

incoming routing decision FORWARD outgoing

INPUT local process
OUTPUT local process
A 4-rule filtering firewall

```
ipables -t filter -A INPUT -i eth1 -p tcp --sport 1024:65535 --dport 23 -s 0.0.0.0/0 -d 192.168.4.1/32 –j ACCEPT
ipables -t filter -A OUTPUT -o eth1 -p tcp --sport 23 --dport 1024:65535 -s 192.168.4.1/32 -d 0.0.0.0/0 –j ACCEPT
ipables -t filter -P INPUT DROP
ipables -t filter -P OUTPUT DROP
```

Executed in chronological sequence as shown, resultant 2-chain firewall permits telnet access between this machine 192.168.4.1 and others via eth1. And nothing else.

Priority of chronology = priority of effect

```
ipables -t filter -A INPUT -i eth1 -p tcp --sport 1024:65535 --dport 23 -s 64.1.1.1/32 -d 192.168.4.1/32 –j DROP
ipables -t filter -A INPUT -i eth1 -p tcp --sport 1024:65535 --dport 23 -s 0.0.0.0/0 -d 192.168.4.1/32 –j ACCEPT
ipables -t filter -A OUTPUT -o eth1 -p tcp --sport 23 --dport 1024:65535 -s 192.168.4.1/32 -d 0.0.0.0/0 –j ACCEPT
ipables -t filter -P INPUT DROP
ipables -t filter -P OUTPUT DROP
```

… EXCEPT no telnet from machine 64.1.1.1, because first rule eclipses second since it preceded it. (Second not reached, never applied.)
nat table: rules that alter packet

• Masquerading
  
  iptables -t nat -A POSTROUTING
  -o eth1 -s 10.0.0.0/8
  -j SNAT --to 216.83.185.193

• Pinholing (port forwarding)
  
  iptables -t nat -A PREROUTING
  -i eth1 -d 216.83.185.193/32 -p tcp --dport 5631
  -j DNAT --to 216.83.185.193

Firewall ruleset philosophies

• Optimistic/lax
  – set everything open
  – apply selective closures

• Pessimistic/strict
  – set everything closed
  – apply selective openings
Set everything closed

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

Firewall persistence

- firewall is memory-resident
- volatile across reboot
- re-create at bootime by init script containing
  - individual iptables commands (e.g., Zeigler), or
  - iptables-restore and iptables-save (e.g., RedHat)
Init script basics

- UNIX has a conventional method to uniformly start/stop services
- one script per service in /etc/rc.d/init.d
- scripts accept parameters start, stop, or restart
- if firewall’s script is:
  /etc/rc.d/init.d/firewall
- start it with:
  /etc/rc.d/init.d/firewall start, or service firewall start

Zeigler-style firewall

- Zeigler firewall in /etc/rc.d/init.d/firewall
- configuration file in /etc/firewall/firewall.conf
- firewall script full of individual iptables invocations (i.e., rules)
- config files set variables for conditional evaluation in main firewall script
Zeigler Firewall
- files & organization

Config file
/etc/firewall/firewall.conf.iptables

- ping = ok
- telnet = not ok
- ftp = ok

Main firewall
/etc/rc.d/init.d/firewall

#!/bin/sh
#
# firewall start/stop script
# read the config file for selections
. /etc/firewall/firewall.conf.iptables
# call local customizations file
. /etc/firewall/firewall.local.iptables

if ping ok
  rule to allow ping
if telnet ok
  rule to allow telnet
if ftp ok
  rule to allow ftp

Customizations file
/etc/firewall/firewall.local.iptables

let managers ssh
allow intra-traffic on our DMZ

Customizations, executed early, have priority

RedHat-default-style firewall

- firewall stored whole in /etc/sysconfig/iptables
- utility pair: iptables-restore, iptables-save
- use their own (ascii) format
- installtime high/medium/low security write different /etc/sysconfig/iptables
- init script /etc/rc.d/init.d/iptables iptables iptables- restores from /etc/sysconfig/iptables
- modified firewall can be stored for future by service iptables save

© David Morgan 2003,2004
RedHat Firewall
- files & organization

Executables
/sbin/iptables-save
/sbin/iptables-restore

Init script
/etc/rc.d/init.d/iptables

#!/bin/sh
#
# firewall   start/stop script
...
...
...
start()
iptables-save < /etc/sysconfig/iptables
stop()
restart()
save()
iptables-save > /etc/sysconfig/iptables

Rules-storage file
/etc/sysconfig/iptables
(in iptables-save/restore format)

*filter
INPUT ACCEPT [0:0]
FORWARD ACCEPT [0:0]
OUTPUT ACCEPT [0:0]
-NH Lokkit-0-50-INPUT - [0:0]
-A INPUT -j RH-Lokkit-0-50-INPUT
-A RH-Lokkit-0-50-INPUT -i lo -j ACCEPT
etc

Please see …

http://www.iptables.org/

Linux Firewalls, 2nd edition, Robert Zeigler,
New Riders, 2002

http://linux-firewall-tools.com/ (Zeigler’s site)

http://www.malibyte.net/iptables/scripts/fwscripts.html (3rd-party derivative of Ziegler fw)