Netfilter / IPtables

Stateful packet filter firewalling with Linux

Antony Stone

Antony.Stone@Open.Source.IT
Netfilter / IPtables

- Quick review of TCP/IP networking & firewalls
- Netfilter & IPtables components
- How packets pass through the system
- Netfilter matches & targets
- Standard security policy
- Network Address Translation + problems
- New & interesting netfilter matches & targets
- What can go wrong / debugging
Review of TCP/IP & Firewalls

- HTTP requests and responses
- Packaged into TCP packet, with TCP header
  - Source & destination port numbers
  - TCP flags
  - Sequence & acknowledgement numbers
- TCP packaged into IP packet with IP header
  - Source & destination IP addresses
- IP packets travel across the Internet
- Routed by destination address
Review of TCP/IP & Firewalls

• Early Internet - everyone trusted - no firewalls
• Public access - firewalls restrict:
  • External access to internal resources
  • Internal access to external services
  • Internal access to sensitive data
    • Keep the engineers out of the personnel database
• Basic principle:
  • Firewalls are routers which can say “no”.
• Firewall rules based on organisation's security policy
Types of firewalls

- Packet filters vs. proxy firewalls
  - Packet filters look at IP addresses, TCP/UDP port numbers - header information only
  - Proxies look at IP addresses, TCP/UDP port numbers, plus content of datastream
- Stateful vs. non-stateful
  - Stateful packet filters understand 'connections'
  - Reply packets can be handled securely
  - Rulesets are simpler and easier to understand
Netfilter & iptables

• **Netfilter** is the kernel component which processes the packets

• **IPtables** is the userspace application which manages the ruleset

• **Netfilter terminology:**
  - **Chains** - eg: INPUT, FORWARD, OUTPUT
  - **Tables** - eg: filter, nat, mangle, raw
  - **Rule matches** - eg: protocol, address, port etc.
  - **Rule targets** - eg: ACCEPT, REJECT, LOG etc.
Netfilter chains & tables

- **PREROUTING** chain
  - all packets entering an interface (eg: eth, lo, ppp...)

- **INPUT** chain
  - all packets addressed to the firewall

- **FORWARD** chain
  - all packets being routed through the firewall

- **OUTPUT** chain
  - all packets generated from the firewall

- **POSTROUTING** chain
  - all packets leaving an interface (eg: eth, lo, ppp...)
Netfilter chains & tables

• filter table
  • Filtering operations :-)  
  • ACCEPT, REJECT, DROP  
  • Also LOG  
• nat table  
  • Network Address Translation  
  • SNAT, DNAT, MASQUERADE  
  • Also ACCEPT can be useful for exceptions
Netfilter chains & tables

- mangle table
  - Packet (header) mangling
  - Change TTL
  - Change TOS / DSCP
  - Set MARKs
  - Change routing (interfaces, gateway)

- raw table
  - access to packets before connection tracking
Path of packets

PREROUTING -> R -> FORWARD -> R -> POSTROUTING

INPUT

OUTPUT
Path of packets

In to THIS system

PREROUTING

R

FORWARD

R

OUTPUT

From THIS system

INPUT
Path of packets

PREROUTING -> R (Routed through this system) -> FORWARD -> R

INPUT

OUTPUT

POSTROUTING
Path of packets - even more detail

- PREROUTING chain
  - raw --> mangle --> nat
- POSTROUTING chain
  - mangle --> nat
- INPUT & FORWARD chains
  - mangle --> filter
- OUTPUT chain
  - raw --> mangle --> nat --> filter
Netfilter rule matches

• “Match” means “which packets does this rule apply to?”
  - -p tcp - all TCP packets
  - -d a.b.c.d/n - destination address = a.b.c.d/n
  - --dport x - destination port number = x
  - --length - number of bytes in packet
  - --mac-source - MAC address of sending device
  - -i, -o - input / output interface for packet
Netfilter rule targets

- “Target” means “what happens to the packets which match?”
  - ACCEPT - packet is accepted
  - DROP - packet is dropped / discarded
  - DNAT - destination address is changed
  - LOG - packet is logged to syslog (processing continues)
  - REJECT - packet is dropped, reject returned
  - MARK - mark a packet, useful in later processing
  - MIRROR - reverse source & destination :-)

UKUUG Leeds 2004  Netfilter / IPtables  Antony Stone
User-defined chains

- User-defined chains can be created in addition to the five built-in chains
  - `iptables -N mychain`
  - `iptables -A INPUT -p tcp --dport 22 -j mychain`
  - `iptables -A mychain -s 192.168.0.10 -j LOG`
  - `iptables -A mychain -j ACCEPT`
- RETURN target returns from user-defined chain to the calling chain (useful for exceptions)
Standard security policy

- Everything is blocked, except that which is explicitly allowed
  - Default DROP policy on filter tables
  - *(NEVER* set default DROP on nat or mangle!)*
  - Individual rules allow packets which are wanted
  - LOG packets which get blocked?
Example ruleset 1

iptables -P INPUT DROP
iptables -A INPUT -i eth1 -p tcp --dport 22 -j ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -j ACCEPT
Stateful filtering

• What does this mean?
  -m state --state ESTABLISHED,RELATED

• ESTABLISHED matches any packets with source/destination addresses/ports matching an entry in the connection tracking table
  • Source/destination match forward & reverse
  • Conntrack table entries are automatically created when a packet is ACCEPTed
Stateful filtering

• RELATED matches packets which netfilter identifies as being related to an entry in the conntrack table
  
  • FTP data channel is RELATED to the control channel
  
  • ICMP responses (eg: host unreachable, TTL exceeded) are RELATED to the packets they're in response to
Network Address Translation

- **SNAT / MASQUERADE**
  - Changes the source address of packets leaving a network - usually so that the reply packets can get back again

- **DNAT**
  - Changes the destination address of packets so that they go to a different machine than they were originally addressed to
Network Address Translation

- **SNAT / MASQUERADE**
  - Usually used to 'hide' a network of machines using private (RFC1918) internal addresses behind one or more publicly routable IP addresses

- **DNAT**
  - Usually used to provide publicly-accessible services from machines on a privately-addressed network
Network Address Translation

- Some people regard NAT as evil - because it breaks protocols such as FTP, H.323

- Some people regard protocols such as FTP, H.323 as evil - because they embed IP addresses and port numbers in application layer communications

- NAT also breaks IPsec transport mode (AH), which has a checksum involving the addresses
Example ruleset 2

iptables -P INPUT DROP
iptables -A INPUT -i eth1 -p tcp --dport 22 -j ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A POSTROUTING -t nat -o eth0 -j MASQUERADE
Network Address Translation

- “I have DNAT working fine from the Internet to a machine on my network, but why can't clients on my network access its public IP address?”
  - Request goes through firewall (NAT)
  - Reply goes directly across network (no NAT)
  - Client sends to a.b.c.d, gets reply from w.x.y.z

- I would be very happy if nobody ever asked this question again on the netfilter mailing list!
More netfilter matches & targets

- Recent versions of netfilter (currently 1.2.11) have introduced many interesting and less-often used (less-often explained?) matches and targets
- No longer just packet header information
- Also netfilter internal information
  - eg: MARK, CONNMARK, rate limits, helpers
- Also external packet characteristics
  - eg: owner, route, time, random matches
Interesting new rule matches

- **addrtype**
  - UNICAST, LOCAL, BROADCAST, ANYCAST, MULTICAST, BLACKHOLE, UNREACHABLE, PROHIBIT, THROW, NAT, XRESOLVE

- **condition**
  - checks content of /
    proc/net/ipt_condition/filename

- **connmark**
  - matches packets in “marked” connections
  - like the “mark” match, but applies to replies too
Interesting new rule matches

- **conntrack**
  - allows detailed matching of packet against connection tracking table data:
    - original source/destination address
    - reply source/destination address
    - internal conntrack state (EXPECTED, SEEN_REPLY, ASSURED etc)
    - expiry time remaining

- **dstlimit**
  - allows rate limiting per IP address
  - like the “limit” match, but per IP
Interesting new rule matches

- **helper**
  - matches packets according to a particular connection tracking helper module (eg: FTP, IRC)
- **owner**
  - for locally-generated packets, match for the process which generated the packet:
    - UID, GID, PID, SID, command name
- **physdev**
  - allows matching of interfaces when bridging
Interesting new rule targets

- **BALANCE**
  - DNAT to several addresses using round-robin

- **CLASSIFY**
  - set priority value for classifying packets into CBQ (Class-Based-Queuing) classes
  - CBQ is used for allocating bandwidth pools

- **CLUSTERIP**
  - distributes connections to a cluster of machines sharing IP & MAC addresses
Interesting new rule targets

- **CONNMARK**
  - Assign a numeric “mark” to packets, for later matching, but match on reply packets too

- **NETMAP**
  - Map a range of addresses to a second range of addresses (can be 1:1, can map to a smaller range using a mask) (SNAT & DNAT)

- **NOTRACK**
  - Disables connection tracking for selected packets (good for avoiding DoS attacks)
Interesting new rule targets

- **ROUTE**
  - Changes routing information about a packet
    - input interface name
    - output interface name
    - next hop gateway address

- **TCPMSS**
  - Control Maximum Segment Size of TCP packets
    (usually to match the Maximum Transmission Unit of a particular link)

- **TTL**
  - Change the Time To Live value of a packet
Extensions to netfilter

- Patch-o-matic
- Various experimental, unofficial or esoteric extensions to netfilter
- Applies patches to netfilter (in the kernel source code) and iptables (userspace application) - need to recompile both
- Currently still stabilising after being adapted to kernel 2.6 (as well as kernel 2.4)
Extensions to netfilter

- OSF
  - Operating System Fingerprinting
  - Adapted from BSD pf code
- PSD
  - Port Scan Detection
- TARPIT
  - Accepts incoming TCP connections, causing the remote system to get stuck in a 12-24 minute timeout, without allowing connection closure
Extensions to netfilter

• XOR
  • Simplistic encryption of TCP / UDP packet contents using XOR operation

• COMMENT
  • Allows comments to be added to netfilter rules

• connbytes
  • Matches against number of bytes transferred

• CuSeeMe
  • NAT helper for CuSeeMe protocol
Extensions to netfilter

- drop table (and DROPPED chain)
  - Adds a new table for packets which are being dropped, enabling them to be logged
- goto
  - Alternative to jump, returns to parent chain instead of “this” chain
- QUAKE3
  - Adds conntracking & nat support for Quake III
Conntrack technical details

- Connection tracking table
  - ~300 bytes of RAM needed per conntrack entry
  - Default conntrack table size =
    - RAM (Mbytes) x 64  (min 128, max 65536)
    - eg: 256Mbyte machine: 16384 connections
  - This allocates 2% of system RAM for conntrack
  - Dedicated firewall has not much use for most of the remaining 98% RAM
  - Manually adjust:
    `/proc/sys/net/ipv4/netfilter/ip_conntrack_max`
Conntrack technical details

- Connection tracking table can fill up!
  - No more new connections will be accepted
- Common causes:
  - SYN flood (DoS attack)
  - Worm-infected PC on internal network
- Solution:
  - Add rule to block offending IP (or unplug PC)
  - Increase conntrack table size
  - Wait for old connections to timeout
Conntrack technical details

• Connection tracking is entirely based on:
  • Source & destination IP addresses
  • Source & destination TCP/UDP ports

• Connection tracking does **not** use:
  • TCP sequence / acknowledgement numbers

• `/proc/net/ip_conntrack` lists current entries
  • useful first indication of a worm on your network
Firewall debugging

- Client cannot connect when firewall ruleset is in place; client can connect with no ruleset

- How to debug?
  - ACCEPT packets which are wanted
  - DROP packets which are known and unwanted
  - LOG packets which get this far
  - DROP remainder using default policy

- `iptables -L -nvx`
  - Shows packet & byte counters for each rule
Traps for the unwary

- “iptables -L” does not list all the rules
  - The filter table is assumed by default
  - If you want the nat or mangle tables, you must specify them:
    - `iptables -L -t nat`
- DNAT is not working
  - Ensure that the FORWARD rule allows the new (translated) address, not the original address
- LOG logs to the console, not /var/log/messages
  - Use “-j LOG --log-level=6”
  - And check /etc/syslogd.conf
Traps for the unwary

• DNAT sends packets to my server, but nobody can connect
  • check return route from server - must go through firewall for reverse NAT

• Passive FTP works fine, but not active FTP
  • When doing NAT, active FTP requires the FTP NAT helper module loaded, or compiled into kernel
  • Looks for the FTP “PORT” command in the datastream and adds a RELATED conntrack table entry
Traps for the unwary

• LOG in the nat table records almost no packets
  • Only the first packet of a connection goes through the rules in the nat tables - all subsequent packets (both ways) are processed automagically in the background

• DNAT works fine for packets routed through the firewall, but not for packets originating on the firewall machine itself
  • PREROUTING is only for packets entering the machine
  • The OUTPUT chain has a nat table for DNATting locally-generated packets
Netfilter tricks

- Rules do not have to have a target
  
  "iptables -A FORWARD -p tcp --dport 22" is a perfectly valid rule

- Useful for packet counting

- ! can be used to mean "anything except..."

  iptables -A FORWARD -p tcp --dport ! 22 -j LOG

- Will LOG all packets except SSH (TCP 22)
Netfilter tricks

- How to handle two (or more) exceptions?
  - User-defined chain
    - `iptables -N mychain`
    - `iptables -A mychain -d a.b.c.d -j RETURN`
    - `iptables -A mychain -d w.x.y.z -j RETURN`
    - `iptables -A mychain -j LOG`
  - User-defined chains can also have nat and mangle table rules
    - eg: “SNAT all packets except from these three IP addresses”
Networking words of wisdom

90% of all networking problems are routing problems.

9 of the remaining 10% are routing problems, but in the other direction.

The final 1% might be something else, but check the routing anyway.