Linux Netfilter (iptables)

Luca Veltri
(mail.to: luca.veltri@unipr.it)

Corso di Sicurezza nelle reti, Reti di telecomunicazioni C, a.a. 2009/2010
http://www.tlc.unipr.it/veltri

Linux packet filter

- Linux kernels have had packet filtering since the 1.1 series
- Packet filtering is implemented by netfilter
- Netfilter is a general framework inside the Linux kernel which other things can plug into (such as the iptables module)
- The tool `iptables` talks to the kernel and tells it what packets to filter
  - `iptables` inserts and deletes rules from the kernel's packet filtering table
  - `iptables` is a replacement for the old `ipfwadm` and `ipchains`

Making rules permanent

- The current firewall setup stored in the kernel is lost on reboot
- There are two ways to restore a firewall setup:
  - `iptables-save/iptables-restore`
    - `iptables-save` and `iptables-restore` scripts save it to, and restore it from a file
  - using initialization scripts
    - The other way is to put the commands required to set up the rules in an initialization script

Simple example of netfilter script

```
# Insert connection-tracking modules (not needed if built into kernel).
inmod ip_conntrack
inmod ip_conntrack_ftp

# Create chain which blocks new connections, except if coming from inside.
iptables -N block
iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
iptables -A block -j DROP

# Jump to that chain from INPUT and FORWARD chains.
iptables -A INPUT -j block
iptables -A FORWARD -j block
```
Netfilter basic chains

- The kernel starts with three built-in lists of rules in the `filter` table
  - These lists are called firewall chains or just chains
  - The three built-in chains are called INPUT, OUTPUT and FORWARD
  - The three chains can't be deleted

![Diagram showing the flow of traffic through chains]

When a packet reaches a chain, that chain is examined to decide the fate of the packet
- If the chain says to DROP the packet, it is killed there, but
- If the chain says to ACCEPT the packet, it continues traversing the diagram

- A chain is a checklist of rules
  - each rule says 'if the packet header looks like this, then here's what to do with the packet'
  - if the rule doesn't match the packet, then the next rule in the chain is consulted
  - finally, if there are no more rules to consult, then the kernel looks at the chain policy to decide what to do
  - in a security-conscious system, this policy usually tells the kernel to DROP the packet

iptables operations

- Operations to manage whole chains:
  - Create a new chain (-N)
  - Delete an empty chain (-X)
  - Change the policy for a built-in chain. (-P)
  - List the rules in a chain (-L)
  - Flush the rules out of a chain (-F)
  - Zero the packet and byte counters on all rules in a chain (-Z)

- Operations to manipulate rules inside a chain:
  - Append a new rule to a chain (-A)
  - Insert a new rule at some position in a chain (-I)
  - Replace a rule at some position in a chain (-R)
  - Delete a rule at some position in a chain, or the first that matches (-D)

Operations on a single rule

- Each rule specifies a set of conditions the packet must meet (matching condition), and what to do if it meets them (‘target’ or action)
- For example
  - to drop all ICMP packets coming from the IP address 127.0.0.1
    - the conditions are that the protocol must be ICMP and that the source address must be 127.0.0.1
    - the target is ‘DROP’
  - to add the rule:
    - `iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP`
  - to test:
    - `PING 127.0.0.1`
  - to delete the rule:
    - `iptables -D INPUT 1` or
    - `iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP`
Filtering specifications

- Specifying Source and Destination IP Addresses
  - source (`-s', `--source' or `--src') and destination (`-d', `--destination' or `--dst') IP addresses can be specified in four ways
    - using the full name, such as 'localhost' or 'www.linuxhq.com'
    - specifying the IP address, such as '127.0.0.1'
    - specifying a group of IP addresses, such as '199.95.207.0/24'
    - or such as '199.95.207.0/255.255.255.0'

- Specifying Inversion
  - many flags can have their arguments preceded by `!' (NOT) to invert (negate) the given matching condition
    - e.g. `-s ! localhost' matches any packet not coming from localhost

- Specifying Protocol
  - protocol can be specified with the `-p' (or `--protocol') flag
  - protocol can be a number or a name ('tcp', 'udp' or 'icmp')

Matching extensions

- TCP, UDP and ICMP protocols automatically offer new matching tests
  - It is possible to specify the new match test on the command line after the `-p' option

- Other extension can be loaded explicitly
  - using the `-m' option followed by the match test
    - e.g. `-m mac --mac-source 45:e4:23:6b:82:a0

TCP/UDP extensions

- --tcp-flags
  - allows the filtering on specific TCP flags
  - used with two parameters (strings of flags)
    - the first string is the mask: a list of flags you want to examine
    - the second string of flags tells which one(s) should be set
  - for example,
    - `iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DROP

- --syn
  - shorthand for `--tcp-flags SYN,RST,ACK SYN'

- --source-port (--sport) and --destination-port (--dport)
  - followed by either a single TCP/UDP port, or a range of ports
  - ranges are two port names separated by a `:'

- --tcp-option
  - followed by a TCP option
Example on TCP extensions

- It is sometimes useful to allow TCP connections in one direction, but not the other.
- The solution is to block only the packets used to request a connection.
- By disallowing only these packets, we can stop attempted connections in their tracks.
- For example, to specify TCP connection attempts from 192.168.1.1:
  ```
  -p TCP -s 192.168.1.1 --syn
  ```

Other match extensions

- `--icmp-type`
  - followed by an icmp type name (e.g. `host-unreachable`), or a numeric type (e.g. `3`), or a numeric type and code separated by a `/` (e.g. `3/3`)
- `-m mac --mac-source` (or `--match mac --mac-source `)
  - followed by an ethernet address in colon-separated hexbyte notation, e.g.
    ```
    --mac-source 00:60:08:91:CC:B7
    ```

The state match

- The `state` extension interprets the connection-tracking analysis (of the `ip_conntrack` module).
- `-m state` allows an additional `--state` option, which is a comma-separated list of states to match.
- These states are:
  - `NEW` — a packet which creates a new connection
  - `ESTABLISHED` — a packet which belongs to an existing connection
  - `RELATED` — a packet which is related to, but not part of, an existing connection (e.g. an ICMP error, or an FTP data connection)
  - `INVALID` — a packet which could not be identified for some reason

Example of `state` match extension:

```
iptables -A FORWARD -i ppp0 -m state ! --state NEW -j DROP
```

This has an effect similar to:

```
iptables -A FORWARD -i ppp0 -p tcp --syn -j DROP
```
Target specifications

- Rule's target is what to do to the packets which match the rule
- There are two very simple built-in targets: DROP and ACCEPT
- There are two types of targets other than the built-in ones:
  - extensions
  - user-defined chains

User-defined chains

- It is possible to create new chains, in addition to the three built-in ones (INPUT, FORWARD, and OUTPUT)
- When a packet matches a rule whose target is a user-defined chain
  - the packet begins traversing the rules in that user-defined chain
  - if that chain doesn’t decide the fate of the packet, then traversal resumes on the next rule in the current chain

New targets

- The other types of targets are new extension targets
- There are several extensions in the default netfilter distribution:
  - LOG
    - this module provides kernel logging of matching packets
  - REJECT
    - has the same effect as `DROP`, except that the sender is sent an ICMP `port unreachable` error message
- There are two special built-in targets: RETURN and QUEUE.
  - RETURN
    - has the same effect of falling off the end of a chain
  - QUEUE
    - is a special target, which queues the packet for userspace processing

Operations on an entire chain

- Creating a New Chain
  - using the `-N` (or `--new-chain`) command
  - e.g. `iptables -N test`
- Deleting a Chain
  - using the `-X` (or `--delete-chain`) command
  - e.g. `iptables -X test`
- Flushing a Chain
  - using the `-F` (or `--flush`) command
  - e.g. `iptables -F FORWARD`
- Listing a Chain
  - using the `-L` (or `--list`) command
    - `-n` (numeric) option prevents `iptables` from lookup the IP addr
    - `-v` options shows you all the details of the rules
Operations on an entire chain

- Setting Policy
  - the policy of the chain determines the fate of the packet
  - only built-in chains (INPUT, OUTPUT and FORWARD) have policies
  - The policy can be either ACCEPT or DROP, for example:
    - e.g. `# iptables -P FORWARD DROP`

The entire netfilter (including NAT)

References

http://www.netfilter.org/documentation/index.html#documentation-howto

http://iptables-tutorial.frozentux.net/

[3] Linux iptables man pages