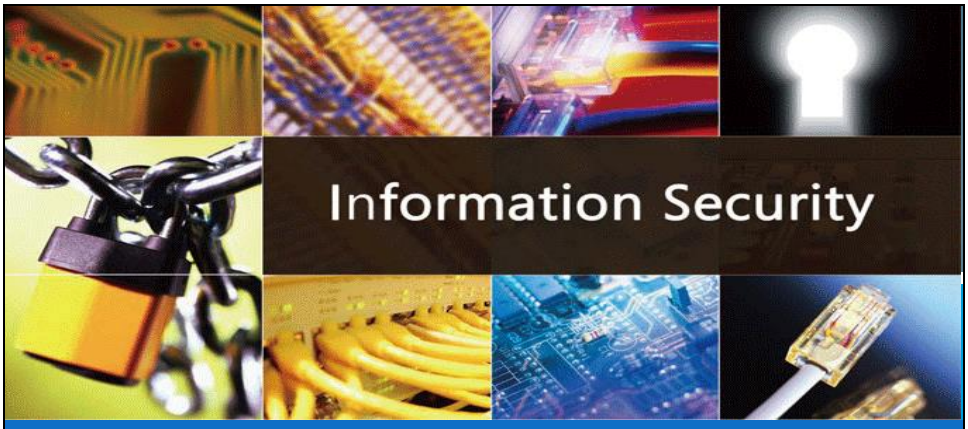


Chapter 10:
LAB - Firewall for Linux - IPTable

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Contents

Introduction

- Firewall for Linux:
  - Netfilter and iptables are building blocks of a framework inside the Linux 2.4.x and 2.6.x kernel.
  - This framework enables
    - packet filtering,
    - network address [and port] translation (NA[P]T) and
    - other packet mangling.
- Version
  - Ipfwadm : Linux kernel 2.0.34
  - Ipchains : Linux kernel 2.2.*
  - Iptables : Linux kernel 2.4.*

Characteristic of Iptables

- Stateful packet inspection.
  - The firewall keeps track of each connection passing through it,
  - This is an important feature in the support of active FTP and VoIP.
- Filtering packets based on a MAC address IPv4 / IPv6
  - Very important in WLAN's and similar enviroments.
- Filtering packets based the values of the flags in the TCP header
  - Helpful in preventing attacks using malformed packets and in restricting access.
- Network address translation and Port translating NAT/NAPT
  - Building DMZ and more flexible NAT enviroments to increase security.
- Source and stateful routing and failover functions
  - Route traffic more efficiant and faster than regular IP routers.

Characteristic of Iptables

- System logging of network activities
  Provides the option of adjusting the level of detail of the reporting
- A rate limiting feature
  Helps to block some types of denial of service (DoS) attacks.
- Packet manipulation (mangling) like altering the TOS/DSCP/ECN bits of the IP header
  Mark and classify packets dependent on rules. First step in QoS.

Download And Install The Iptables Package

- Most Linux already have iptables
- Download from:
  http://www.netfilter.org/downloads.html
- Documentation:
  http://www.netfilter.org/documentation/index.html
- Install from sources or rpm:
  # rpm –ivh iptables-1.2.9-1.0.i386.rpm
  # tar xvfz iptables-1.2.9.tar.gz; ./configure ; make ; make install
- Modules to add functionallity to IPtables:
  Variour proxy modules, for example ftp and h323
  Modules must be loaded into kernel
  # modprobe module
  # insmod module
- Patch-o-Matic (updated and modules)
  http://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/

How To Start iptables

- You can start, stop, and restart iptables after booting by using the commands:
  - Starting IP tables: *service iptables start*
  - Stopping IP tables: *service iptables stop*
  - Restaring IP tables: *service iptables restart*
  - Checking IP tables status (rulechains): *service iptables status*
- To get `iptables` configured to start at boot, use the `chkconfig` command: *chkconfig iptables on*
- iptables itself is a command which we will see soon.
- To show all current rule chains: *`iptables --list`*
- To drop all current rule chains: *`iptables --flush`*

Packet Processing In iptables

- All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing
- Three builtin tables (queues) for processing:

1. **MANGLE**: manipulate QoS bits in TCP header

2. **FILTER**: packet filtering, has three builtin chains (your firewall policy rules)
   - **Forward chain**: filters packets to servers protected by firewall
   - **Input chain**: filters packets destinated for the firewall
   - **Output chain**: filters packets orginating from the firewall

3. **NAT**: network adress translation, has two builtin chains
   - **Pre-routing**: NAT packets when destination address need changes
   - **Post-routing**: NAT packets when source address need changes

Processing For Packets Routed By The Firewall 1/2

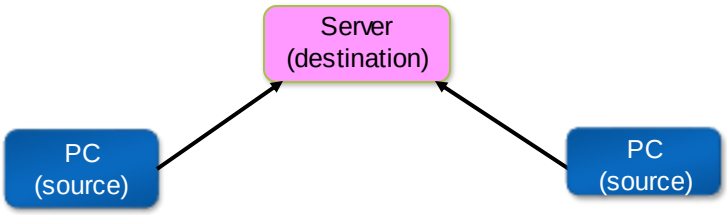
Queue Type	Queue Function	Packet transformation chain in Queue	Chain Function
Filter	Packet filtering	FORWARD	Filters packets to servers accessible by another NIC on the firewall.
		INPUT	Filters packets destined to the firewall.
		OUTPUT	Filters packets originating from the firewall
Nat	Network Address Translation	PREROUTING	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as **destination NAT** or **DNAT**.

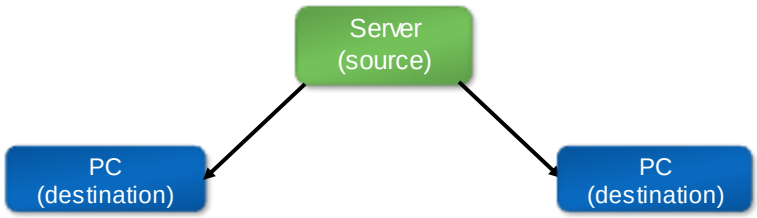
Processing For Packets Routed By The Firewall 2/2

		POSTROUTING	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as **source NAT**, or **SNAT**.
		OUTPUT	Network address translation for packets generated by the firewall. (Rarely used in SOHO environments)
Mangle	TCP header modification	PREROUTING POSTROUTING OUTPUT INPUT FORWARD	Modification of the TCP packet quality of service bits before routing occurs (Rarely used in SOHO environments)

Filter table

Input chain: filters packets destinated for the firewall

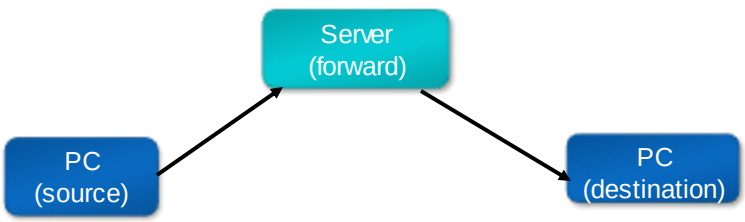




Output chain: filters packets orginating from the firewall

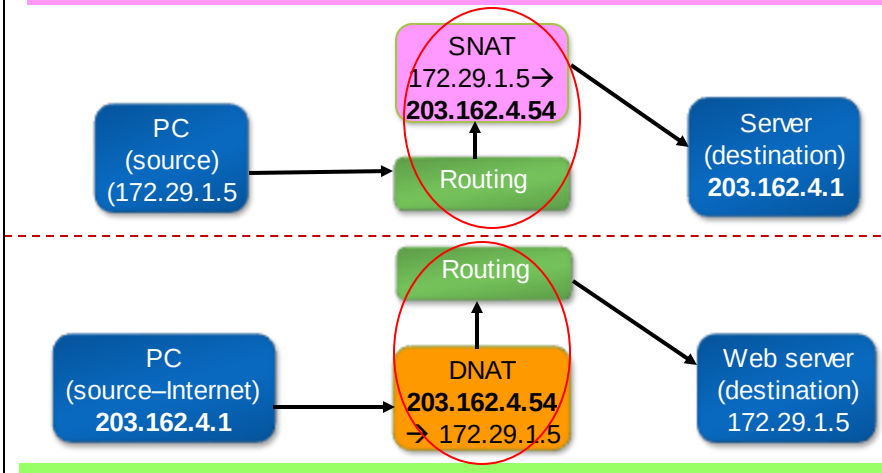
Filter table

Forward chain: filters packets to servers protected by firewall



NAT Table

Post-routing (NAT OUT): NAT packets when source address need changes



Pre-routing (NAT IN): NAT packets when destination address need changes

Targets And Jumps 1/2

- Each firewall rule inspects each IP packet and then tries to identify it as the target of some sort of operation. Once a target is identified, the packet needs to jump over to it for further processing
- ACCEPT
  - `iptables` stops further processing.
  - The packet is handed over to the end application or the operating system for processing
- DROP
  - `iptables` stops further processing.
  - The packet is blocked.
- REJECT
  - Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked
    --reject-with qualifierQualifier is an ICMP message

Targets And Jumps 2/2

- LOG
  - The packet information is sent to the syslog daemon for logging.
  - `iptables` continues processing with the next rule in the table.
  - You can't log and drop at the same time ->use two rules.

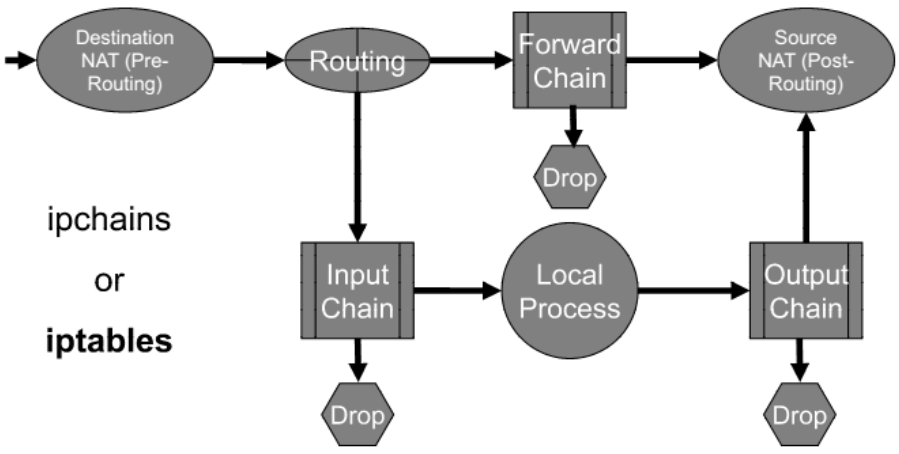
  --log-prefix "reason"
- SNAT
  - Used to do source network address translation rewriting the source IP address of the packet
  - The source IP address is user defined

  --to-source <address>[-<address>][:<port>-<port>]
- DNAT
  - Used to do destination network address translation. ie. rewriting the destination IP address of the packet

  --to-destination ipaddress
- MASQUERADE
  - Used to do Source Network Address Translation.
  - By default the source IP address is the same as that used by the firewall's interface

  [--to-ports <port>[-<port>]]

Packet Processing In iptables



Commands

- Create new chain – `iptables -N chain_name`
- Erase all rules in chain – `iptables -F chain_name`
- Remove empty chain – `iptables -X chain_name`
- Set chain policy –
  `iptables -P chain_name target`
- Managing rules in a chain

  add: `iptables -A chain_name rule_spec`
  delete: `iptables -D chain_name rule_num`
  insert: `iptables -I chain_name [rule_num] rule_spec`

Important Iptables Command Switch Operations 1/2

iptables command Switch	Description
`-t <table>`	If you don't specify a table, then the `filter` table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
`-j <target>`	Jump to the specified target chain when the packet matches the current rule.
`-A`	Append rule to end of a chain
`-F`	Flush. Deletes all the rules in the selected table
`-p <protocol-type>`	Match protocol. Types include, `icmp`, `tcp`, `udp`, and `all`

Important Iptables Command Switch Operations 2/2

-s <ip-address>	Match source IP address
-d <ip-address>	Match destination IP address
-i <interface-name>	Match "input" interface on which the packet enters.
-o <interface-name>	Match "output" interface on which the packet exits

- We try to define a rule that will accept all packages on interface eth0 that uses TCP and has destination address 192.168.1.1.
- We first define the MATCH criterias:
  - Use default filter table (absense of –t)
  - Append a rule to end of INPUT chain (-A INPUT)
  - Match on source address can be any 0/0 address (-s 0/0)
  - Input interface used is eth0 (-i eth0)
  - Match on destination address 192.168.1.1 (-d 192.168.1.1)
  - Match Protocol TCP (-p TCP)
  - If all matches is fulfilled, then jump to ACCEPT chain. (-j ACCEPT)
- iptables -AINPUT -s 0/0 -i eth0 -d 192.168.1.1 -p TCP -j ACCEPT

Common TCP and UDP Match Criteria

Switch	Description
-p tcp --sport <*port*>	TCP source port Can be a single value or a range in the format: *start-port-number:end-port-number*
-p tcp --dport <port>	TCP destination port Can be a single value or a range in the format: *starting-port:ending-port*
-p tcp --syn	Used to identify a new TCP connection request ! --syn means, not a new connection request
-p udp --sport <*port*>	UDP source port Can be a single value or a range in the format: *starting-port:ending-port*

Common ICMP (Ping) Match Criteria

Matches used with --icmp-type	Description
`--icmp-type <type>`	The most commonly used types are echo-reply and echo-request

- Allow ping request and reply
  - `iptables` is being configured to allow the firewall to send ICMP echo-requests (pings) and in turn, accept the expected ICMP echo-replies.

  iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

  iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
- Put limit on ping to prevent flood pings

  iptables -A INPUT -p icmp --icmp-type echo-request \
  -m limit --limit 1/s -i eth0 -j ACCEPT

Defense for SYN flood attacks

- –m limit sets maximum number of SYN packets
  - `iptables` is being configured to allow the firewall to accept maxim 5 TCP/SYN packeds per second on interface eth0.

  iptables -A INPUT -p tcp --syn -m limit --limit 5/s -i eth0 -j ACCEPT

  - If more than 5 SYN packets per second, the packets are dropped.
  - If source/destination sence dropped packets, it will resend three times
  - If drops continue after 3 reset packets, source will reduce packet speed.

Common Extended Match Criteria 1/2

Switch	Description
`-m multiport --sport <port, port>`	A variety of TCP/UDP source ports separated by commas. Unlike when `-m` isn't used, they do not have to be within a range.
`-m multiport --dport <port, port>`	A variety of TCP/UDP destination ports separated by commas. Unlike when `-m` isn't used, they do not have to be within a range.
`-m multiport --ports <port, port>`	A variety of TCP/UDP ports separated by commas. Source and destination ports are assumed to be the same and they do not have to be within a range.
`-m --state <state>`	The most frequently tested states are: **ESTABLISHED**: The packet is part of a connection that has seen packets in both directions **NEW**: The packet is the start of a new connection **RELATED**: The packet is starting a new secondary connection. This is a common feature of such protocols such as an FTP data transfer, or an ICMP error. **INVALID**: The packet couldn't be identified. Could be due to insufficient system resources, or ICMP errors that don't match an existing data flow.

Common Extended Match Criteria 2/2

- Allow both port 80 and 443 for the webserver on inside:

```
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP \
    --sport 1024:65535 -m multiport --dport 80,443 -j ACCEPT
```

- The return traffic from webbserver is allowed, but only of sessions are opened:

```
iptables -A FORWARD -d 0/0 -o eth0 -s 192.168.1.58 -i eth1 -p TCP \
    -m state --state ESTABLISHED -j ACCEPT
```

- If sessions are used, you can reduce an attack called half open
  - Half open is known to consume server all free sockets (tcp stack memory) and is senced as a denial of service attack, but it is not.
  - Sessions are usally waiting 3 minutes.

Using User Defined Chains

- Define fast input queue:
  iptables -A INPUT -i eth0 -d 206.229.110.2 -j fast-input-queue
- Define fast output queue:
  iptables -A OUTPUT -o eth0 -s 206.229.110.2 -j fast-output-queue
- Use defined queues and define two icmp queue's:

  iptables -A fast-input-queue -p icmp -j icmp-queue-in
  iptables -A fast-output-queue -p icmp -j icmp-queue-out
- Finally we use the queue's to define a two rules:

  iptables -A icmp-queue-out -p icmp --icmp-type echo-request \
  -m state --state NEW -j ACCEPT
  iptables -A icmp-queue-in -p icmp --icmp-type echo-reply -j
  ACCEPT

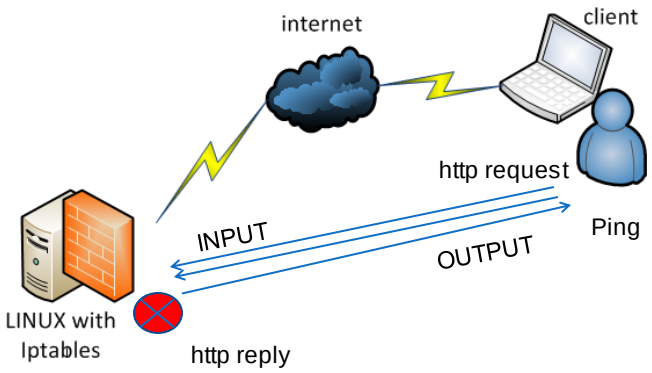
Saving Your iptables Scripts

- RedHat based distributions:
  /etc/sysconfig/iptables
- Other distributions uses:
  There is no specific favourite place, one is:
  /etc/rc.d/rc.firewall
  And maby this is the most common is:
  /etc/init.d/rc.firewall
- RedHat/Fedora's iptables Rule Generator:
  lokkit
- There are three iptable commands:
  iptables (The kernel insert rule command)
  iptables-save > rc.firewall.backup
  iptables-restore < rc.firewall.backup
- In RedHat/Fedora you can also:
  service iptables save

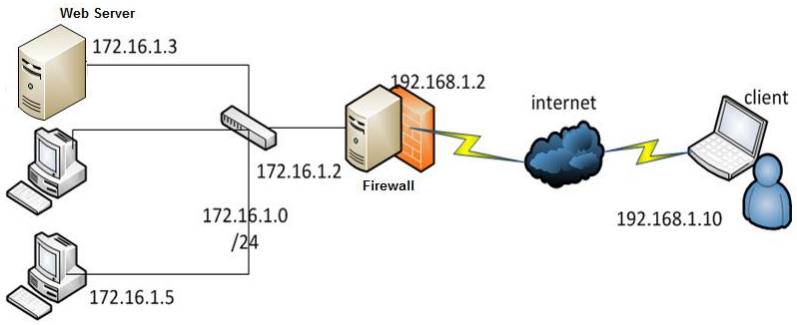
FIREWALL - IPTable

1. Cài đặt Firewall IPTable: (theo mô hình tham khảo sau)
   - FILTER: (Hình 1)
     - Cho phép/ cấm các giao thức ICMP, HTTP:
       - đi vào LAN: INPUT
       - từ mạng LAN ra: OUTPUT
   - NAT IN (PREROUTING) : (Hình 2)
     - Cho phép client bên ngoài sử dụng web ở bên trong mạng nội bộ
   - NAT OUT (POSROUTING) : (Hình 3)
     - Cho phép client bên trong mạng nội bộ sử dụng Internet (chia sẻ dung chung Internet)

IPTable - Filter (Hình 1)



IPTable – NAT IN (Hình 2)



IPTable – NATOUT (Hình 3)