BSc in Telecommunications Engineering

TEL3214

Computer Communication Networks

Lecture 11 Firewalls

Eng Diarmuid O'Briain, CEng, CISSP

Department of Electrical and Computer Engineering,
College of Engineering, Design, Art and Technology,
Makerere University

Table of Contents

Illustration Index

1. An introduction to firewalls

A firewall is "*an internetwork gateway that restricts data communication traffic to and from one of the connected networks (the one said to be "inside" the firewall) and thus protects that network's system resources against threats from the other network (the one that is said to be "outside" the firewall).*" RFC 2828 - Internet Security Glossary.

.

2. The digital security problem

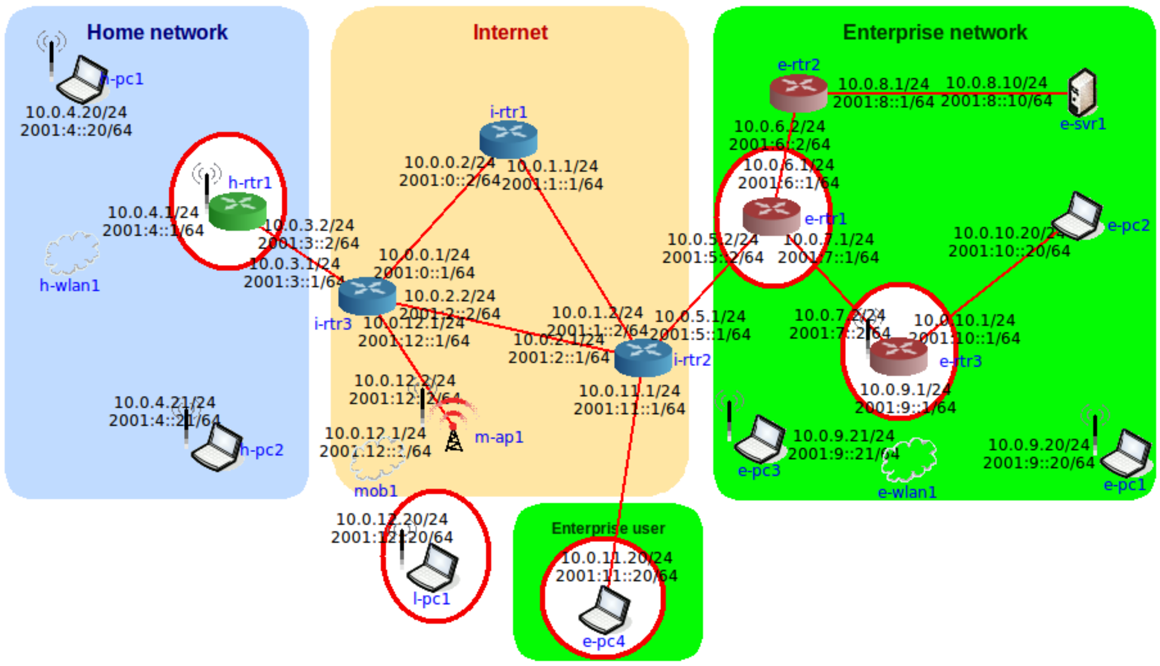


Illustration 1: Traditional attack vectors

The areas marked with red circles show areas that require particular attention. While there are no non vulnerable areas consider the following:

2.1 Home

Placing a firewall at this ingress point of ***h-rtr1*** to the home network makes sense. Obviously the home devices are using WLAN and therefore all devices present a wireless attack vector but assuming adequate controls in terms of encryption and security keys are used the main attack vector is via the home gateway router.

2.2 Enterprise

Now this is a real attack vector, the gateway ***e-rtr1*** to the enterprise, the wireless AP ***e-rtr3*** is also a good attack vector. While I haven't drawn circles around the users in the enterprise they obviously pose a risk particularly if they are wireless users.

The remote worker link ***e-pc4*** is also a risk considering their distance from the physical and IT related security that are implemented at enterprise offices.

Another area of particular concern is the Bring Your Own Device (BYOD) phenomenon within enterprise, how is security catered for in those scenarios. A real headache for the Chief Information Security Officer (CISO) in an organisation.

2.3 Roaming individual

Like the roaming worker, the roaming individual is vulnerable, particularly from physical theft which can lead to digital theft if the thief can gain access. However Personal Area Networks (PAN) like Bluetooth, WiFi and even Near Field Communication (NFC) give good digital attack points.

2.4 Perimeter defence and firewalls

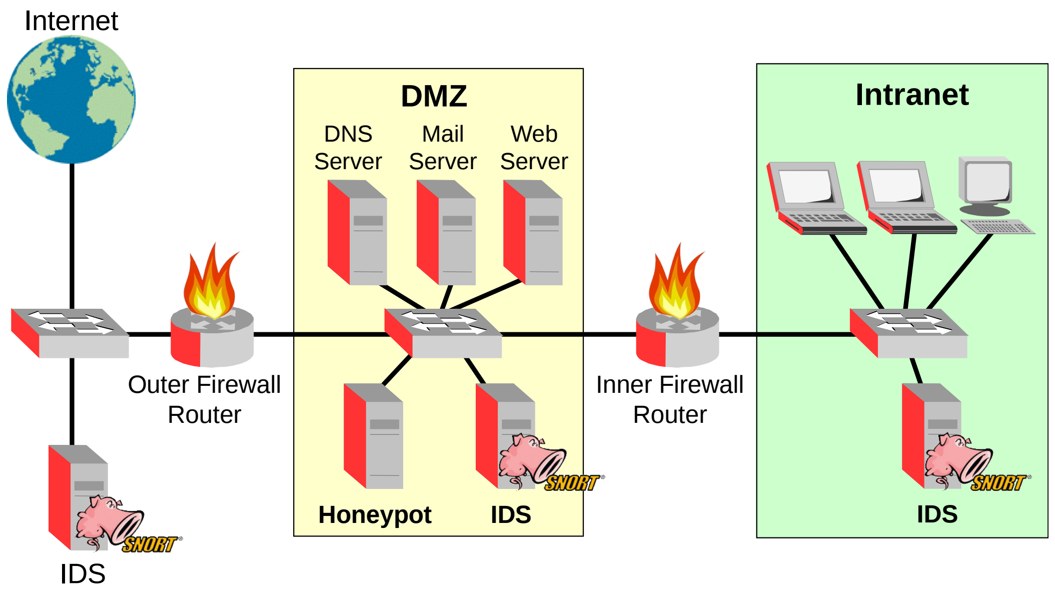


Illustration 2: Perimeter defence and firewalls

Perimeter defence is one method of defending a network from attack. As long as the perimeter is secure, i.e. no rogue devices connecting to the network and a firewall system like that in Illustration 2 is in place to prevent access as well as actively monitor connections in and out with several layers in a defence in depth system gives the network and data security.

2.5 Intrusion prevention and detection - Snort

Snort is a Free and Open Source (FOSS) Network Intrusion Prevention System (NIPS) and Network Intrusion Detection System (NIDS). It has the ability to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks. Snort performs protocol analysis, content searching and matching. These basic services have many purposes including application-aware triggered Quality of Service (QoS), to de-prioritize bulk traffic when latency-sensitive applications are in use.

It is also a tool for the detection of probes or attacks like Operating System fingerprinting attempts, common gateway interface, buffer overflows, server message block probes, and stealth port scans.

Snort can be configured in three main modes:

- **Sniffer** - reads network packets and outputs them on the console.
- **Packet logger** - log packets to disk.
- **NIDS** - Monitor network traffic and analyse it against a rule set. It can also carry out actions based on the rules.

3. Personal Firewall

3.1 Introduction

A personal firewall is an essential consideration on personal computing devices. GNU/Linux provides a very complete kernel based firewall system called Netfilter which will be consider later but its configuration is somewhat complicated for the regular user. It is also designed to deal with through traffic as well as traffic originating or terminating on a personal computing device. For this purpose GNU/Linux has the Uncomplicated Firewall (ufw) tool and a graphical complement called Gnome ufw (Gufw).

3.2 Uncomplicated Firewall (ufw) installation

Ufw is a front-end to iptables, the administration tool for Netfilter IPv4/IPv6 kernel based packet filtering and Network Address Translation (NAT). The goal of ufw is to simplify the configuration of firewall rules on an easy to use interface.

Confirm that ufw and even gufw are installed on the system.

```
$ aptitude show ufw
Package: ufw
State: installed
Automatically installed: no
Version: 0.34~rc-0ubuntu2
Priority: standard
Section: admin
Maintainer: Jamie Strandboge <jamie@ubuntu.com>
Architecture: all
Uncompressed Size: 749 k
Depends: debconf, iptables, ucf, python3:any (>= 3.3.2-2~), python3, debconf
(>=
         0.5) | debconf-2.0, sysv-rc (>= 2.88dsf-24) | file-rc (>= 0.8.16)
Suggests: rsyslog
Description: program for managing a Netfilter firewall
 The Uncomplicated FireWall is a front-end for iptables, to make managing a
 Netfilter firewall easier. It provides a command line interface with syntax
 similar to OpenBSD's Packet Filter. It is particularly well-suited as a
 host-based firewall.
Homepage: https://launchpad.net/ufw
```

```
$ aptitude show gufw
Package: gufw
State: installed
Automatically installed: no
Version: 14.04.2-0ubuntu1.2
Priority: optional
Section: universe/admin
Maintainer: Devid Antonio Filoni <d.filoni@ubuntu.com>
Architecture: all
Uncompressed Size: 2,623 k
Depends: python:any (>= 2.7.1-0ubuntu2), ufw (>= 0.31.1), gir1.2-gtk-3.0,
         policykit-1, gnome-icon-theme-symbolic, python-netifaces,
         gir1.2-webkit-3.0
Description: graphical user interface for ufw
 gufw is an easy and intuitive way to manage your Linux firewall. It supports
 common tasks such as allowing or blocking pre-configured, common p2p, or
 individual port(s), and many others!
Homepage: http://gufw.org/
```

If they are not installed simple install with the following commands.

```
$ sudo aptitude install ufw
$ sudo aptitude install gufw
```

First, obviously, you want to make sure UFW is installed. It should be installed by default in Ubuntu, but if for some reason it's not, you can install the package using aptitude or apt-get using the following commands:

3.3 ufw status

The first thing to do is check the status. If ufw is just installed then it will most likely be inactive.

```
$ ufw status
Status: active
```

3.4 ufw commands

The following are a list of ufw commands. It is also possible to get more information using the **man ufw** command.

```
$ ufw –help

Usage: ufw COMMAND

Commands:
 enable                          enables the firewall
 disable                         disables the firewall
 default ARG                     set default policy
 logging LEVEL                   set logging to LEVEL
 allow ARGS                      add allow rule
 deny ARGS                       add deny rule
 reject ARGS                     add reject rule
 limit ARGS                      add limit rule
 delete RULE|NUM                 delete RULE
 insert NUM RULE                 insert RULE at NUM
 reload                          reload firewall
 reset                           reset firewall
 status                          show firewall status
 status numbered                    show firewall status as numbered list of
RULES
 status verbose                  show verbose firewall status
 show ARG                        show firewall report
 version                         display version information

Application profile commands:
 app list                        list application profiles
 app info PROFILE                show information on PROFILE
 app update PROFILE              update PROFILE
 app default ARG                 set default application policy
```

3.5 ufw configuration

Ensure IPv6 is enabled by checking the ***/etc/default/ufw*** file. It should contain a line like this. If not added it an save the file.

```
$ cat /etc/default/ufw | grep IPV6
IPV6=yes
```

While in that file the following may come to your attention. These are the default input, output and forward policies and are set to either to ACCEPT, DROP, or REJECT. Essentially out of the box ufw drops all incoming and forwarding packets while allowing all packets to leave the device.

```
DEFAULT_INPUT_POLICY="DROP"
DEFAULT_OUTPUT_POLICY="ACCEPT"
DEFAULT_FORWARD_POLICY="DROP"
```

Enable the firewall and check its status as follows:

```
$ sudo ufw enable
Firewall is active and enabled on system startup

$ sudo ufw status
Status: active

$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
```

The defaults in the ***/etc/default/ufw*** file can be edited from the ufw tool without having to edit the file directly. The tool offers the following commands for that.

```
$ sudo ufw default deny incoming
$ sudo ufw default allow outgoing
```

It is possible to be more restrictive and deny outgoing also but this means that it would be necessary to allow individual services access making configuration quite cumbersome. Denying all outgoing connections is achieved with this command:

```
$ sudo ufw default deny outgoing
```

3.6 ufw Allow

The default thus far is to deny all incoming connections. Perhaps we want to allow access for Secure Shell (SSH) and Secure File Transfer Protocol (SFTP). Well to do that is pretty simple.

```
$ sudo ufw allow ssh
Rule added
Rule added (v6)
```

To be more specific it is possible to allow access to say a File Transfer Protocol (FTP) running on the computer but only from another specific computer and access to a webserver only for the subnet at work for example.

```
$ sudo ufw allow proto tcp from 173.34.5.56 to any port 21
```

```
Rule added
$ sudo ufw allow proto tcp from 173.34.5.0/24 to any port 80
Rule added
```

Confirm and enable the new configuration.

```
$ sudo ufw status
Status: active

To                          Action      From
--                          ------      ----
22                          ALLOW       Anywhere
21/tcp                      ALLOW       173.34.5.56
80/tcp                      ALLOW       173.34.5.0/24
22 (v6)                     ALLOW       Anywhere (v6)

$ sudo ufw reload
Firewall reloaded
```

3.7 ufw deny connections

While the SSH protocol is open to all, it may be required to deny specific hosts or subnets.

```
$ sudo ufw deny proto tcp from 189.55.43.0/24 to any port 22
Rule added

$ sudo ufw reload
Firewall reloaded

$ sudo ufw status
Status: active

To                          Action      From
--                          ------      ----
22                          ALLOW       Anywhere
21/tcp                      ALLOW       173.34.5.0/24
80/tcp                      ALLOW       173.34.5.0/24
22/tcp                      DENY        189.55.43.0/24
22 (v6)                     ALLOW       Anywhere (v6)
```

ufw deleting entries

It may become necessary to delete entries. Here is an example removing the need for access to the FTP Server.

```
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 21/tcp                     ALLOW IN    173.34.5.0/24
[ 3] 80/tcp                     ALLOW IN    173.34.5.0/24
[ 4] 22/tcp                     DENY IN     189.55.43.0/24
[ 5] 22 (v6)                    ALLOW IN    Anywhere (v6)


$ sudo ufw delete 2
Deleting:
 allow from 173.34.5.0/24 to any port 21 proto tcp
Proceed with operation (y|n)? y
Rule deleted


$ sudo ufw reload
Firewall reloaded


$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22                         ALLOW IN    Anywhere
80/tcp                     ALLOW IN    173.34.5.0/24
22/tcp                     DENY IN     189.55.43.0/24
22 (v6)                    ALLOW IN    Anywhere (v6)
```

3.8 ufw disable or reset

It is possible to disable the firewall or even reset it to defaults.

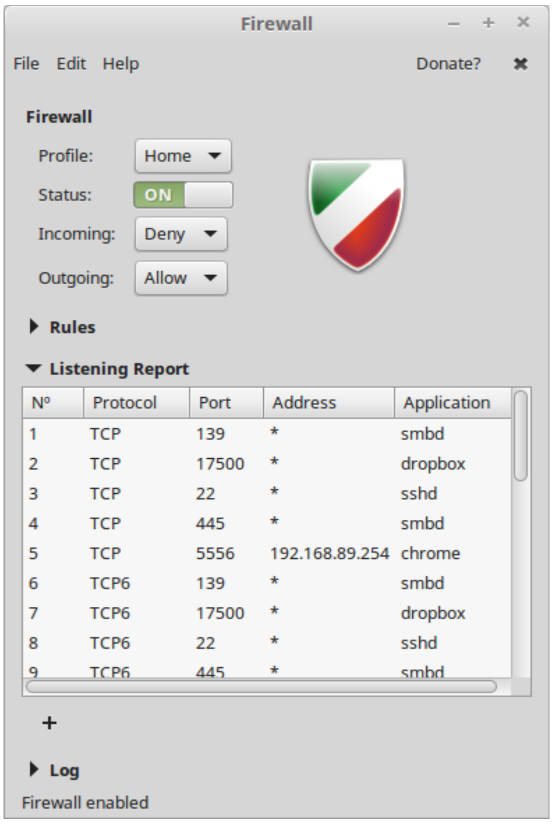
```
$ sudo ufw disable
Firewall stopped and disabled on system startup

$ sudo ufw reset
Resetting all rules to installed defaults. Proceed with operation (y|n)? y
Backing up 'after6.rules' to '/etc/ufw/after6.rules.20160227_083632'
Backing up 'before6.rules' to '/etc/ufw/before6.rules.20160227_083632'
Backing up 'before.rules' to '/etc/ufw/before.rules.20160227_083632'
Backing up 'user.rules' to '/lib/ufw/user.rules.20160227_083632'
Backing up 'user6.rules' to '/lib/ufw/user6.rules.20160227_083632'
Backing up 'after.rules' to '/etc/ufw/after.rules.20160227_083632'

$ sudo ufw status
Status: inactive
```

3.9 Gufw

Illustration 3 Is the **gufw** tool which is a graphical frontend to the **ufw** tool just shown.



Firewall

Profile: Home
Status: ON
Incoming: Deny
Outgoing: Allow

Listening Report

Nº	Protocol	Port	Address	Application
1	TCP	139	*	smbd
2	TCP	17500	*	dropbox
3	TCP	22	*	sshd
4	TCP	445	*	smbd
5	TCP	5556	192.168.89.254	chrome
6	TCP6	139	*	smbd
7	TCP6	17500	*	dropbox
8	TCP6	22	*	sshd
9	TCP6	445	*	smbd

Firewall enabled

Illustration 3: Gufw tool

3.10 Testing ufw

Here are the network mapping of the host 192.168.89.254 and 2a02::250:b6ff:fe6f:df40 before the ufw firewall is applied.

```
$ nmap -p 0-65535 -PN 192.168.89.254

Starting Nmap 6.47 ( http://nmap.org ) at 2016-02-27 05:44 GMT
Nmap scan report for 192.168.89.254
Host is up (0.00012s latency).
Not shown: 65530 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
33129/tcp open  unknown
40055/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.54 seconds
```

```
$ nmap -6 -p 0-65535 -PN 2a02::250:b6ff:fe6f:df40

Starting Nmap 6.47 ( http://nmap.org ) at 2016-02-27 06:04 GMT
Nmap scan report for 2a02::250:b6ff:fe6f:df40
Host is up.
Not shown: 65530 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
33129/tcp open  unknown
40055/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.54 seconds
```

Run ufw denying access to web and FTP traffic. But allowing rcbind and SSH.

```
$ sudo ufw enable
Firewall is active and enabled on system startup

$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip
```

In fact as the default rule is deny it is only necessary to allow **ssh** and **rcbind**.

```
$ sudo ufw allow ssh
Rule added
Rule added (v6)

$ sudo ufw allow proto tcp from any to any port 111
Rule added
Rule added (v6)

$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22                         ALLOW IN    Anywhere
111/tcp                    ALLOW IN    Anywhere
22 (v6)                    ALLOW IN    Anywhere (v6)
111/tcp (v6)               ALLOW IN    Anywhere (v6)

$ sudo ufw reload
Firewall reloaded
```

Check the host using nmap.

```
$ nmap -p 0-65535 -PN 192.168.89.254

Starting Nmap 6.47 ( http://nmap.org ) at 2016-02-27 05:59 GMT
Nmap scan report for 192.168.89.254
Host is up (0.00062s latency).
Not shown: 65534 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
111/tcp open   rpcbind

Nmap done: 1 IP address (1 host up) scanned in 108.31 seconds


$ nmap -6 -p 0-65535 -PN 2a02::250:b6ff:fe6f:df40

Starting Nmap 6.47 ( http://nmap.org ) at 2016-02-27 06:04 GMT
Nmap scan report for 2a02::250:b6ff:fe6f:df40
Host is up.
Not shown: 65534 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
111/tcp open   rpcbind

Nmap done: 1 IP address (1 host up) scanned in 88.53 seconds
```

4. Netfilter/iptables

4.1 Introduction to Netfilter

It is important to limit access to a host to only those services it is offering. GNU/Linux comes with set of hooks inside the kernel that allow kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack. This filter system is called ***Netfilter*** and it is accessed using tools called **i*ptables*** and ***ip6tables***.

4.2 Conntrack

The **conntrack** feature within ***iptables*** and ***ip6tables*** means that ***Netfilter*** can be stateful and packets can be related to tracked connections in four different so called states.

- NEW
- ESTABLISHED
- RELATED
- INVALID

These are accessed with the **--state** option switch.

The ***conntrack-tools*** package is a set tools to allow interaction with the stateful packet inspection for ***iptables*** and ***ip6tables***. The conntrack-tools are the userspace daemon ***conntrackd*** and the command line interface ***conntrack***. Contrack is not automatically installed so check if it is:

```
$ aptitude show conntrack
Package: conntrack
State: installed
Automatically installed: no
Version: 1:1.4.1-1ubuntu1
Priority: optional
Section: universe/net
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Uncompressed Size: 117 k
Depends: libc6 (>= 2.14), libmnl0, libnetfilter-conntrack3, libnfnetlink0
Conflicts: conntrack
Description: Program to modify the conntrack tables
 conntrack is a userspace command line program targeted at system administrators.
It enables them to view and manage the in-kernel connection tracking state table.
Homepage: http://conntrack-tools.netfilter.org/
```

If conntrack : is not installed then add it with the command:

```
$ aptitude install conntrack
```

4.3 netfilter

Netfilter is a framework for packet mangling, outside the normal Berkeley socket interface. It has four parts. Each of IPv4 and IPv6 protocols define "hooks" which are well-defined points in a packet's traversal of that protocol stack. At each of these points, the protocol will call the netfilter framework with the packet and the hook number.

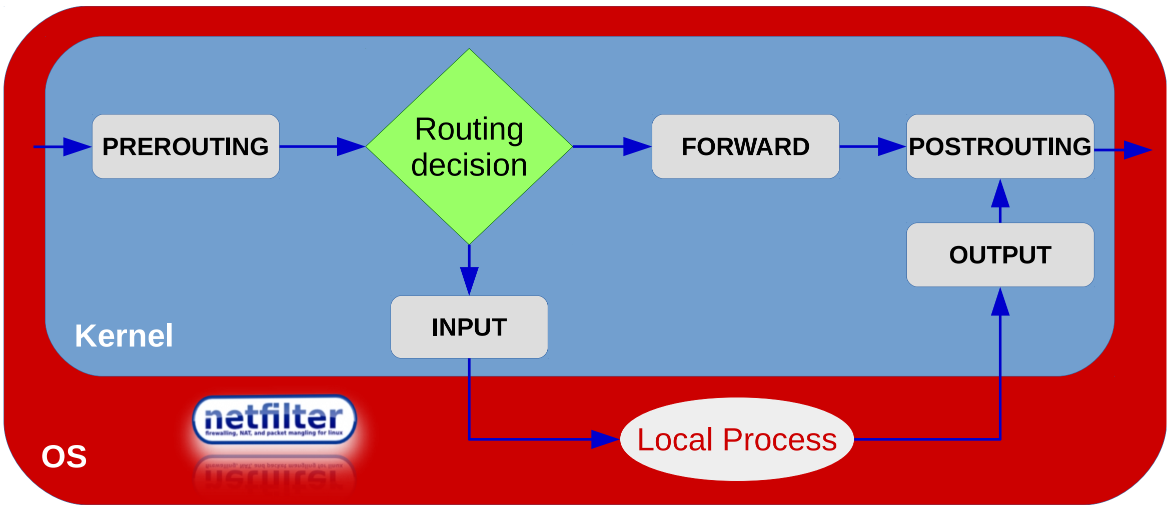


Illustration 4: Netfilter framework

Parts of the OS kernel can register to listen to the different hooks for each protocol. So when a packet is passed to the netfilter framework, it checks to see if anyone has registered for that protocol and hook; if so, they each get a chance to examine (and possibly alter) the packet in order, then discard the packet (NF_DROP), allow it to pass (NF_ACCEPT), tell netfilter to forget about the packet (NF_STOLEN), or ask netfilter to queue the packet for userspace (NF_QUEUE).

The third part is that packets that have been queued are collected (by the ip_queue driver) for sending to userspace; these packets are handled asynchronously.

With netfilter it is possible to:

- Build Internet firewalls based on stateless and stateful packet filtering
- Deploy highly available stateless and stateful firewall clusters
- Use NAT and masquerading for sharing Internet access
- Use NAT to implement transparent proxies
- Work with iproute2 to build sophisticated QoS and policy routers
- Perform packet manipulation (mangling) (altering IP header TOS/DSCP/ECN bits).

4.4 Iptables/ip6tables

Iptables/ip6tables consists of five packet matching tables. The tables are as follows:

4.4.1 Filter

This is the default table. It contains the built-in chains:

- INPUT - for packets destined to local sockets
- FORWARD - for packets being routed through the box
- OUTPUT - for locally generated packets.

4.4.2 NAT

This table is consulted when a packet that creates a new connection is encountered. It consists of three built-ins:

- PREROUTING - for altering packets as soon as they come in
- OUTPUT - for altering locally-generated packets before routing
- POSTROUTING - for altering packets as they are about to go out.

4.4.3 Mangle

This table is used for specialised packet alteration. It has five built-in chains:

- PREROUTING - for altering incoming packets before routing
- OUTPUT - for altering locally-generated packets
- INPUT - for packets coming into the box itself
- FORWARD - for altering packets being routed through the box
- POSTROUTING - for altering packets as they are about to go out.

4.4.4 Raw

This table is used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target. It registers at the netfilter hooks with higher priority and is thus called before ip_conntrack, or any other IP tables. It provides the following built-in chains:

- PREROUTING - for packets arriving via any network interface
- OUTPUT - for packets generated by local processes.

4.4.5 Security

This table is used for Mandatory Access Control (MAC) networking rules, such as those enabled by the SECMARK and CONNSECMARK targets. Mandatory Access Control is implemented by Linux Security Modules such as:

SELinux - The security table is called after the filter table, allowing any Discretionary Access Control (DAC) rules in the filter table to take effect before MAC rules. This table provides the following built-in chains:

- INPUT - for packets coming into the box itself
- OUTPUT - for altering locally-generated packets before routing
- FORWARD - for altering packets being routed through the box

4.5 iptables commands

Flush the existing rules within iptables.

```
$ sudo iptables --flush
```

Flush the existing rules within iptables.

```
$ sudo iptables --list

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Similarly to ufw it is possible to add individual rules like:

```
# iptables --append INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
# iptables --append OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

# iptables --append INPUT --jump DROP
# iptables --append FORWARD --jump DROP
# iptables --append OUTPUT --jump ACCEPT


# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh state
NEW,ESTABLISHED
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp spt:ssh state ESTABLISHED
ACCEPT     all  --  anywhere             anywhere
```

4.6 iptables scripts

It is typical to find ***iptables*** and ***ip6tables*** rules scripted for easy application.

```
#!/bin/sh
# ------------------------------
# Trusted hosts/ranges here:
OK="109.106.96.18 210.22.3.0/24"
OK6="20a2::/64 20b6::55/64"

# flush rules
iptables -F
ip6tables -F

# Allow Ping
iptables -A INPUT -p icmp  -j ACCEPT
ip6tables -A INPUT -p ipv6-icmp  -j ACCEPT

# Allow SSH
iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
ip6tables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT

# Log dropped connections
iptables -N LOGDROP
ip6tables -N LOGDROP

# allow localhost connections to the loopback interface
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT ! -i lo -d 127.0.0.0/8 -j REJECT
ip6tables -A INPUT -i lo -j ACCEPT
ip6tables -A INPUT ! -i lo -d ::1 -j REJECT

# allow connections which are already established
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# allow all outbound connections
iptables -A OUTPUT -j ACCEPT
ip6tables -A OUTPUT -j ACCEPT

# allow all from IPv4 trusted hosts
for Q in $OK; do
    echo "Adding $Q"
    iptables -A INPUT -s $Q -j ACCEPT
done

# allow all from IPv6 trusted hosts
for Q in $OK6; do
    echo "Adding $Q"
    ip6tables -A INPUT -s $Q -j ACCEPT
done

# drop everything else
iptables -A INPUT -j DROP
iptables -A FORWARD -j DROP
ip6tables -A INPUT -j REJECT --reject-with icmp6-port-unreachable
ip6tables -A FORWARD -j REJECT --reject-with icmp6-port-unreachable

# print out lists
iptables -L -v -n
echo "\n--\n"
ip6tables -L -v -n
```

4.7 iptables files

For larger tables it is better to have them applied automatically. To do this ***iptables*** and ***ip6tables*** rules should be stored in two separate files.

- /**etc**/**iptables**/**rules.v4** for IPv4
- /**etc**/**iptables**/**rules.v6** for IPv6

```
$ sudo mkdir /etc/iptables
```

4.8 iptables rules

4.8.1 iptables rules.v4

Edit the ***iptables*** firewall rules files to add IPv4 rules.

```
$ sudo vi /etc/iptables/rules.v4
```

```
*filter

## Filter to allow ssh (3000), smtp (25), Secure IMAP (143),        ##
## email relaying from auth users (587), while all else is blocked. ##

#  Flush tables

-F

#  Allow all loopback (lo0) traffic and drop all traffic to 127/8 that doesn't use lo0

-A INPUT -i lo -j ACCEPT
-A INPUT -d 127.0.0.0/8 -j REJECT

# Allow Incoming SSH on port 3000

-A INPUT -i eth0 -p tcp --dport 3000 -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp --sport 3000 -m state --state ESTABLISHED -j ACCEPT

# Incoming email from other internet domains, addressed to system users, on port 25

-A INPUT -i eth0 -p tcp --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp --sport 25 -m state --state ESTABLISHED -j ACCEPT

# Allow email relaying on port 587

-A OUTPUT -p tcp -d 78.143.141.10 --dport 25 -m state --state NEW -j ACCEPT
-A OUTPUT -p tcp --dport 25 -m state --state NEW -j DROP

# Allow Secure IMAP on port 143

-A INPUT -i eth0 -p tcp --dport 143 -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp --sport 143 -m state --state ESTABLISHED -j ACCEPT

#  Log iptables denied calls
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7

#  Drop all other inbound - default deny unless explicitly allowed policy
-A INPUT -j DROP
-A FORWARD -j DROP

COMMIT
```

4.8.2 ip6tables rules.v6

Edit the ***ip6tables*** firewall rules files to add IPv6 rules.

```
$ sudo vi /etc/iptables/rules.v6
```

```
*filter

## Filter to allow ssh (3000), smtp (25), Secure IMAP (143),        ##
## email relaying from auth users (587), while all else is blocked. ##

#  Flush tables

-F

#  Allow all loopback (lo0) traffic

-A INPUT -i lo -j ACCEPT
-A INPUT -d ::1/128 -j REJECT

# Allow Incoming SSH on port 3000

-A INPUT -i eth0 -p tcp --dport 3000 -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp --sport 3000 -m state --state ESTABLISHED -j ACCEPT

# Incoming email from other internet domains, addressed to system users, on port 25

-A INPUT -i eth0 -p tcp --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp --sport 25 -m state --state ESTABLISHED -j ACCEPT

# Allow email relaying on port 587

-A OUTPUT -p tcp -d 2a02:2158:c:9::10 --dport 25 -m state --state NEW -j ACCEPT
-A OUTPUT -p tcp --dport 25 -m state --state NEW -j DROP

# Allow Secure IMAP on port 143

-A INPUT -i eth0 -p tcp --dport 143 -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp --sport 143 -m state --state ESTABLISHED -j ACCEPT

#  Allow ICMP (ping) - it is not a good idea to block ICMPv6 traffic, IPv6 dependent on
it
-A INPUT -p ipv6-icmp -j ACCEPT

#  Log iptables denied calls
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "ip6tables denied: " --log-level 7

#  Drop all other inbound - default deny unless explicitly allowed policy
#  'Connection refused' message at the other end, masking the firewall's presence
-A INPUT -j REJECT --reject-with icmp6-port-unreachable
-A FORWARD -j REJECT --reject-with icmp6-port-unreachable

COMMIT
```

4.8.3 Load the new rules into Netfilter

Rerun the ***iptables*** and ***ip6tables*** but this time redirecting the new rules in. ***iptables-restore*** and ***ip6tables-restore*** are used to restore IP and IPv6 Tables from data specified on Standard IN (STDIN). I/O redirection in bash is used to read from the files.

```
$ sudo iptables-restore < /etc/iptables/rules.v4
$ sudo ip6tables-restore < /etc/iptables/rules.v6
```

4.8.4 Check the rules have taken

Check the running ***iptables*** and ***ip6tables*** firewall rules to see changes.

```
$ sudo iptables -L

  Chain INPUT (policy ACCEPT)
  target     prot opt source               destination
  ACCEPT     all  --  anywhere             anywhere
  REJECT     all  --  anywhere             127.0.0.0/8          reject-with icmp-port-
unreachable
  ACCEPT     all  --  anywhere             anywhere             state
RELATED,ESTABLISHED
  ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http
  ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https
  ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:ssh
  ACCEPT     icmp --  anywhere             anywhere
  LOG        all  --  anywhere             anywhere             limit: avg 5/min burst
5 LOG level debug prefix "iptables denied: "
  DROP       all  --  anywhere             anywhere

  Chain FORWARD (policy ACCEPT)
  target     prot opt source               destination
  DROP       all  --  anywhere             anywhere

  Chain OUTPUT (policy ACCEPT)
  target     prot opt source               destination
  ACCEPT     all  --  anywhere             anywhere
```

```
$ sudo ip6tables -L

  Chain INPUT (policy ACCEPT)
  target     prot opt     source           destination
  ACCEPT     all          anywhere         anywhere
  ACCEPT     all          anywhere         anywhere             state
RELATED,ESTABLISHED
  ACCEPT     all          anywhere         anywhere             state
RELATED,ESTABLISHED
  ACCEPT     tcp          anywhere         anywhere             state NEW tcp dpt:http
  ACCEPT     tcp          anywhere         anywhere             state NEW tcp dpt:https
  ACCEPT     tcp          anywhere         anywhere             state NEW tcp dpt:ssh
  ACCEPT     ipv6-icmp    anywhere         anywhere
  LOG        all          anywhere         anywhere             limit: avg 5/min burst
5 LOG level debug prefix "ip6tables denied: "
  REJECT     all          anywhere         anywhere             reject-with icmp6-port-
unreachable

  Chain FORWARD (policy ACCEPT)
  target     prot opt     source           destination
  REJECT     all          anywhere         anywhere             reject-with icmp6-port-
unreachable

  Chain OUTPUT (policy ACCEPT)
  target     prot opt     source           destination
```

4.9 Make the firewall persistent

To ensure the firewall rules become active at each time the interface comes up add a ***pre-up*** rule to ***/etc/network/interfaces***.

```
$ cat /etc/network/interfaces

  auto lo
  iface lo inet loopback

  auto eth0
  iface eth0 inet static
     address 78.143.141.78
     netmask 255.255.255.0
     gateway 78.141.143.1
     pre-up iptables-restore < /etc/iptables/rules.v4

  iface eth0 inet6 static
     pre-up modprobe ipv6
     address 2a02:aaaa::20
     netmask 64
     gateway 2a02:aaaa::1
     pre-up ip6tables-restore < /etc/iptables/rules.v6
```

4.10 Testing the new firewall

Test the firewalls on 78.143.141.206 and 2a02:2158:c:9::206.

```
$ nmap -p 0-65535 -PN 78.143.141.206

  Starting Nmap 6.00 ( http://nmap.org ) at 2014-05-26 06:55 IST
  Nmap scan report for 78.143.141.206
  Host is up (0.0026s latency).
  Not shown: 65533 filtered ports
  PORT     STATE  SERVICE
  25/tcp   closed smtp
  143/tcp  closed imap
  3000/tcp closed ppp
  MAC Address: EA:A3:2F:28:CD:9D (Unknown)

  Nmap done: 1 IP address (1 host up) scanned in 104.45 seconds
```

```
$ nmap -6 -p 0-65535 -PN 2a02:2158:c:9::206

  Starting Nmap 6.00 ( http://nmap.org ) at 2014-05-26 07:03 IST
  Nmap scan report for 2a02:2158:c:9::206
  Host is up (0.0015s latency).
  Not shown: 65533 filtered ports
  PORT     STATE  SERVICE
  25/tcp   closed smtp
  143/tcp  closed imap
  3000/tcp closed ppp
  MAC Address: EA:A3:2F:28:CD:9D (Unknown)

  Nmap done: 1 IP address (1 host up) scanned in 145.21 seconds
```

5. Firewall Builder

Firewall Builder (fwbuilder) is a tool that simplifies the firewall policy management for a number of firewall platforms, including Netfilter/iptables, ipfw, PF, Cisco PIX, and others. Firewall Builder simplifies the administrative task of managing the security policy of firewalls efficiently and accurately. When a policy is created it can be compiled and installed on one, or several, firewall devices.

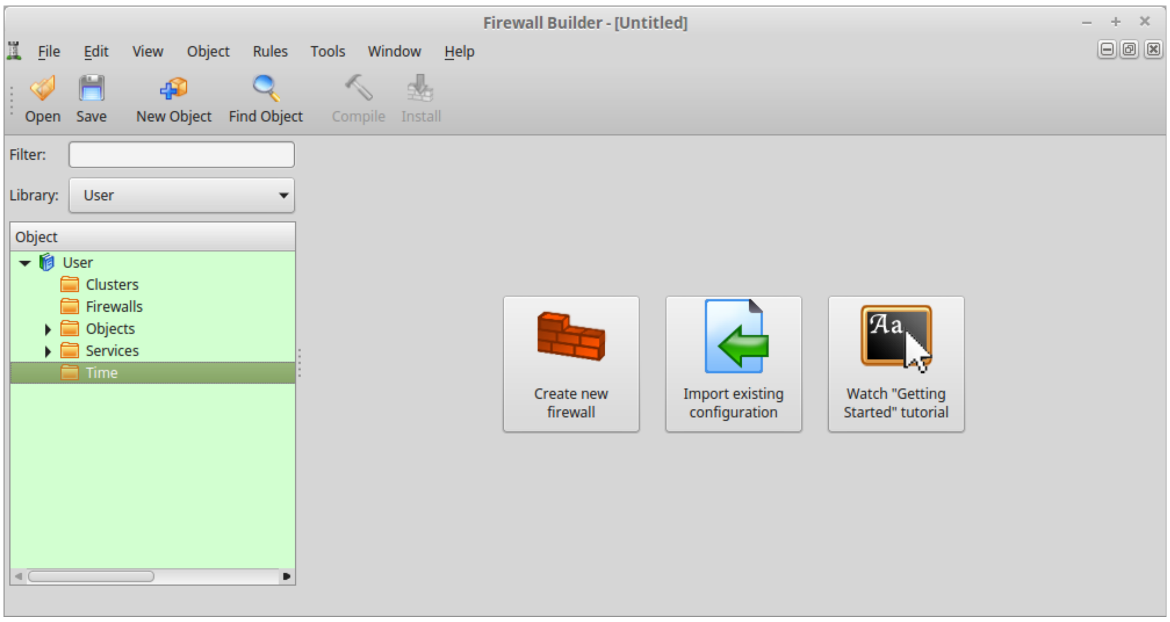


Illustration 5: Firewall builder (fwbuilder)

Fwbuilder works for the following firewalls

- Linux iptables (Netfilter)
- Cisco router Access Control Lists (ACL)
- Cisco ASA/PIX
- Cisco Firewall Service Module (FWSM)
- OpenBSD pf
- FreeBSD ipfw and ipfilter
- HP ProCurve ACL

5.1 Key Concepts

- **Objects**. Firewall Builder is based on the concept of objects. Users create objects like IP networks and IP addresses to represent items that will be used in firewall rules.
- **Libraries**. Objects are stored in libraries. By default Firewall Builder comes with two object libraries. The library called **User** is used to store objects that the user creates. The read-only library called **Standard** contains hundreds of predefined objects like common TCP and UDP services.
- **Compile**. After you create a Policy with firewall rules in Firewall Builder you need to compile the Policy. Compiling converts your rules from the Firewall Builder syntax to the command syntax used by the target firewall platform. Any time you change the rules of a firewall you need to recompile the ruleset.

6. Future of Firewalls

Cloud computing, elastic compute, elastic network and virtualisation are only some of the technologies that are changing how networking is impacting the workplace. It is becoming harder to wall the employees inside a perimeter with a defence in depth system of firewalls, NPDS and NIDSs.

Future firewalls will need to perform real-time network traffic introspection without affecting throughput. That throughput level is increasing all the time and would expect to increase even faster as the elastic network evolves. Expect that firewalls will need to handle throughputs in the range of 10 Gb/s over the next few years.

There are also to many firewall like devices today and multipurpose firewalls incorporating all threat-prevention technologies based in the cloud and offered on a pay per use model are coming. These firewalls are dubbed Virtual Firewalls and a number of vendors offer such solutions today.

Another technology called Runtime Application Self-Protection (RASP) could possible supplant the network firewall function. RASP enables applications to protect themselves by identifying and blocking attacks in real time. RASP leverages the application's intrinsic knowledge of itself to accurately differentiate attacks from legitimate traffic, stopping only malicious traffic.

In the future data analysis will be applied to much larger sets of data which will lead to more sharing of threat information and mitigation techniques and tools to fight cybercrime.

7. Firewall Lab

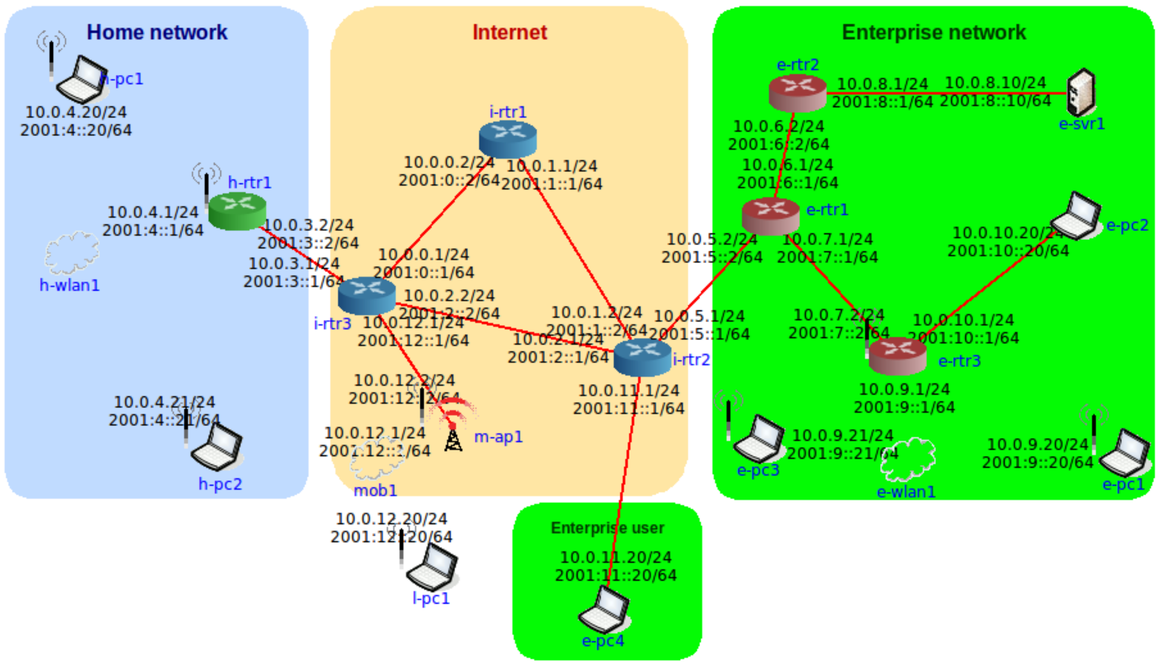


Illustration 6: Firewall Lab

Carry out the following activities on the network file ***TEL3214-Firewall-Example.imn***.

Lone user

- Consider the requirements of the lone user ***l-pc1***. Assuming he is using a GNU/Linux laptop suggest a firewall and a configuration for it.

Home network

- Use ***ufw*** to apply rules on the home router ***h-rtr1***. Permit all traffic to leave the network and only web and SSH in.

Enterprise network

- Use ***iptables*** and ***ip6tables*** to implement a firewall on the webserver ***e-svr1***. Only web traffic is allowed on an open basis and SSH is permitted from networks within the enterprise but not including remote users.

Testing

- Consider the user ***h-pc2*** to be a hobbyist hacker, test your configurations from his computer.

This page is intentionally blank