Offensive technologies - Internetworking exercise

Claudio Facchinetti [claudio.facchinetti@studenti.unitn.it]

September 22, 2021

Abstract

In this document it is presented the work done in order to solve the "Internetworking" exercise on DeterLab.

1 The network structure

In order to solve the exercise it was needed first to understand the decided network structure and write the script to make it set the network interfaces accordingly.
The network addresses were assigned with the logic that the ISRouter divides the network into two chunks: the north chuck, composed by the NWWorkstaion1 and the NWRouter, and the south chunk, composed by the the SWWorkstation1 and the SWRouter. The north segment has the following IP addresses: the private addresses belong to the network 10.0.1.0/29 and the public ones belong to the network 1.0.0.0/29. The south segment has the following IP addresses: the private addresses belong to the network 10.0.2.0/29 and the public ones belong to the network 2.0.0.0/29.

2 The script

In order to solve the exercise a script has been created: this script retrieves wiring data, generates configuration scripts for each node and executes exploiting the fact that the home folder of DeterLab user portal is shared with every node in the laboratory.
In order to use it, simply open it, substitute the experiment and project names with your owns, copy it into your DeterLab user portal in a directory named `internetworking` inside your home folder and execute it. In order to execute it you could have to make it executable by running `chmod +x internetworking_setup.sh`.

2.1 Retrieving and parsing connections

The script grabs information about the data using the script provided by DeterLab and then parses it generating environment variables representing the north and south interface name of each device.
The names of the environment variables do follow a particoular naming convention: $<N|S>W<R|W>\_<S|N>IF$. The first choice indicates whether the device is in the north (N) or in the south (S) segment; the second choice indicates whether the device is a router (R) and the final choice indicates if the interface is the northway (N) or the southway (S) one. The only exception is the ISRouter whose variables have the form $ISR\_<S|N>IF$, where the only choice has the same meaning of the third one in other variables.
This is done thanks to piped invocation of the grep and sed command; in particoular this are the lines of code that do the magic.

```
/share/shared/Internetworking/showcabling Facchinetti-Inetwor offtech
| sed 's/ <- is "wired" to -> /;/' | grep -v -e '^[[:space:]]*$'
> wiring.txt

cat wiring.txt | grep NWworkstation1
| sed 's/workstation1 /W_SIF=/'
| sed 's/router /R_NIF=/' >> nics.sh
```

```
cat wiring.txt | grep NWrouter | grep ISrouter
| sed 's/NWrouter /NWR_SIF=/'
| sed 's/ISrouter /ISR_NIF=/' >> nics.sh

cat wiring.txt | grep SWrouter | grep ISrouter
| sed 's/ISrouter /ISR_SIF=/'
| sed 's/SWrouter /SWR_NIF=/' >> nics.sh

cat wiring.txt | grep SWworkstation1
| sed 's/workstation1 /W_NIF=/'
| sed 's/router /R_SIF=/' >> nics.sh
```

2.2 Configuring interfaces and routes

After generating the environment variables for each node the script generates one configuration script for every node in the network. In the script for a single node the script configures the interfaces and also the routes, using the address assignment logic described before.
As an example it is reported the interfaces configuration lines for one workstation and one router.

Code for NWworkstation1

```
ifconfig ${NWW_SIF} 10.0.1.2 netmask ${NETMASK}

route add -net 10.0.2.0 netmask ${NETMASK} gw 10.0.1.1 ${NWW_SIF}
route add -net 1.0.0.0 netmask ${NETMASK} gw 10.0.1.1 ${NWW_SIF}
route add -net 2.0.0.0 netmask ${NETMASK} gw 10.0.1.1 ${NWW_SIF}
```

Code for NWrouter

```
ifconfig ${NWR_NIF} 10.0.1.1 netmask ${NETMASK}
ifconfig ${NWR_SIF} 1.0.0.1 netmask ${NETMASK}

route add -net 2.0.0.0 netmask ${NETMASK} gw 1.0.0.2 ${NWR_SIF}
route add -net 10.0.2.0 netmask ${NETMASK} gw 1.0.0.2 ${NWR_SIF}
```

3 Simulating the Internet

3.1 Dropping packages

In order to simulate the Internet the ISRouter has been configured to drop any packet from and to private network addresses; this has been by appending to the ISRouter configuration script (`namely "setup_isr.sh"`) firewall rules via the iptable command.
Here it is an extract of the ISRouter configuration script showing the added rules.

```
iptables -I FORWARD -d 192.168.0.0/16 -j DROP
iptables -I FORWARD -d 172.16.0.0/12 -j DROP
iptables -I FORWARD -d 10.0.0.0/8 -j DROP
iptables -I FORWARD -s 192.168.0.0/16 -j DROP
iptables -I FORWARD -s 172.16.0.0/12 -j DROP
iptables -I FORWARD -s 10.0.0.0/8 -j DROP
```

3.2 NATting addresses

The ISRouter it is now configured to drop packets from and to private addresses, so we now need to setup address NATting to make sure that the NWworkstation1 is able to access the website hosted on SWworkstation1. To do so we need to configure SWRouter and NWRouter to perform address

NATting whenever the receive a packet from the local network directed to the outside network.
Here it is the code that inserts the iptables rules to perform NATting for both SWRouter and NWRouter; both these snippets are included in the hosts configuration scripts generated by the main script.

Code for SWRouter

```
iptables -t nat -A POSTROUTING -o ${SWR_NIF}
-s 10.0.2.0/29 -j SNAT --to 2.0.0.1
```

Code for NWRouter

```
iptables -t nat -A POSTROUTING -o
${NWR_SIF} -s 10.0.1.0/29 -j SNAT --to 1.0.0.1
```

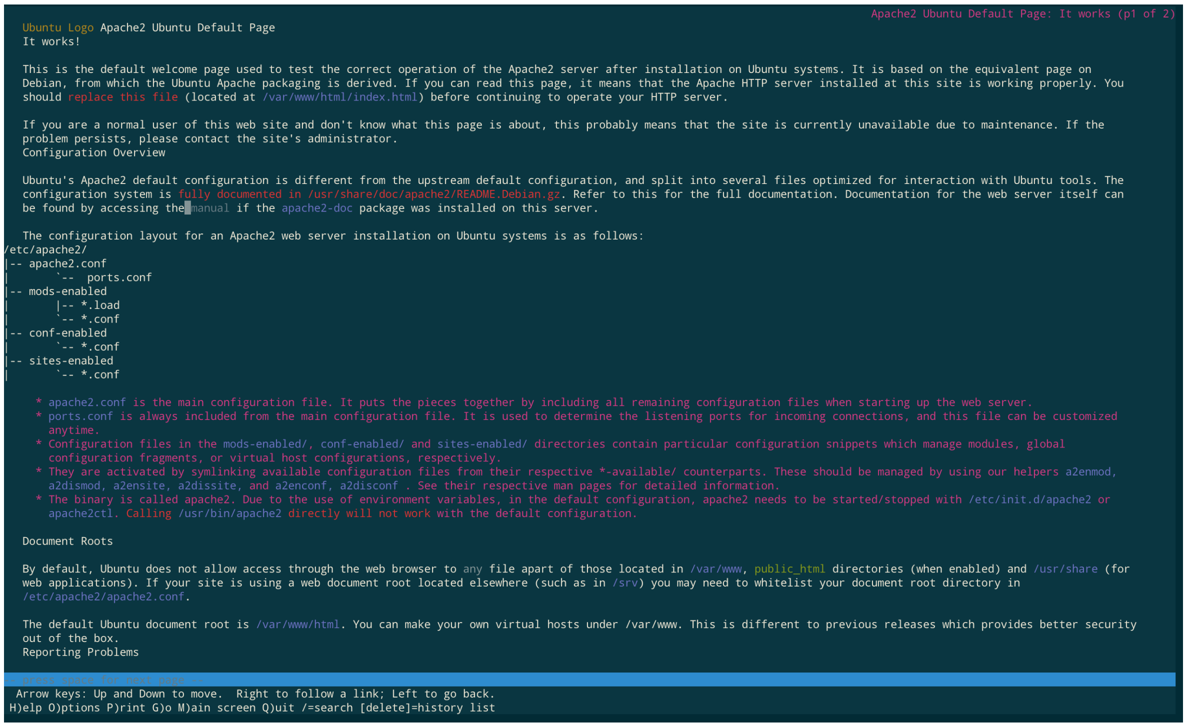
3.3 Port forwarding

The NWworkstation1 cannot directly contact the SWworkstation1 due to the private addressing filtering, therefore it must contact the SWRouter which will then forward the request to the SWworkstation1.
To do this the script used to configure the SWRouter (namely `setup_swr.sh`) instructs IPTable to forward every request arriving on port 80 (HTTP default) to the SWworkstation1.
Here it is the line doing it

```
iptables -t nat -A PREROUTING -i ${SWR_NIF}
-d 2.0.0.1/32 -p tcp --dport 80 -j DNAT --to 10.0.2.2
```

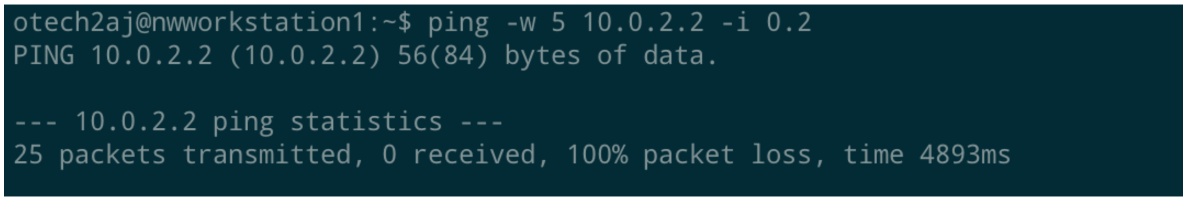
4 Checking the result

In this section the results of the tests are shown.



```
                                                        Apache2 Ubuntu Default Page: It works (p1 of 2)
   Ubuntu Logo Apache2 Ubuntu Default Page
   It works!

   This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on
   Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You
   should replace this file (located at /var/www/html/index.html) before continuing to operate your HTTP server.

   If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the
   problem persists, please contact the site's administrator.
   Configuration Overview

   Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The
   configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz. Refer to this for the full documentation. Documentation for the web server itself can
   be found by accessing the manual if the apache2-doc package was installed on this server.

   The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf

     * apache2.conf is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
     * ports.conf is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized
       anytime.
     * Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/ directories contain particular configuration snippets which manage modules, global
       configuration fragments, or virtual host configurations, respectively.
     * They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers a2enmod,
       a2dismod, a2ensite, a2dissite, and a2enconf, a2disconf . See their respective man pages for detailed information.
     * The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with /etc/init.d/apache2 or
       apache2ctl. Calling /usr/bin/apache2 directly will not work with the default configuration.

   Document Roots

   By default, Ubuntu does not allow access through the web browser to any file apart of those located in /var/www, public_html directories (when enabled) and /usr/share (for
   web applications). If your site is using a web document root located elsewhere (such as in /srv) you may need to whitelist your document root directory in
   /etc/apache2/apache2.conf.

   The default Ubuntu document root is /var/www/html. You can make your own virtual hosts under /var/www. This is different to previous releases which provides better security
   out of the box.
   Reporting Problems

-- press space for next page --
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Figure 1: Result of `lynx 2.0.0.1` on NWworkstation1: the webpage is returned.



```
otech2aj@nwworkstation1:~$ ping -w 5 10.0.2.2 -i 0.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.

--- 10.0.2.2 ping statistics ---
25 packets transmitted, 0 received, 100% packet loss, time 4893ms
```

Figure 2: Result of `ping 10.0.2.2` on NWworkstation1: the packets do not reach the destination, due to the ISRouter iptable entries.