Report: Réseaux Avancé Projet

Nguyen Viet Sang, Le Tan Nhat Linh
{viet-sang.nguyen, tan-nhat-linh.le}@etu.unilim.fr

Master 1 Cryptis, Faculty of Science and Technique, University of Limoges

1 Introduction

In this report, we presents the work of building a transparent-proxy. We have done almost requirements and our system operates successfully. Our main work in this project includes configuring a network (section 2), setting DHCP server for providing IPs to clients and DNS service (section 3), configuring the firewall (section 4), and writing the TCP server (section 5). In addition to the main work, we also do some extra work which is a different way to implement a TCP server (section 6).

To run the attached codes, it is neccessary to follow the instruction in README.md

2 Configure the network

The structure of our network is shown as figure 1. We create two namespaces h1 and h2 which act as two machine in the network **SW**. Interface `router-eth0` is connected to our VM which is considered as a router.

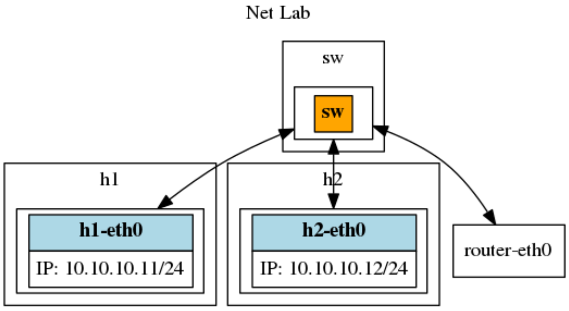
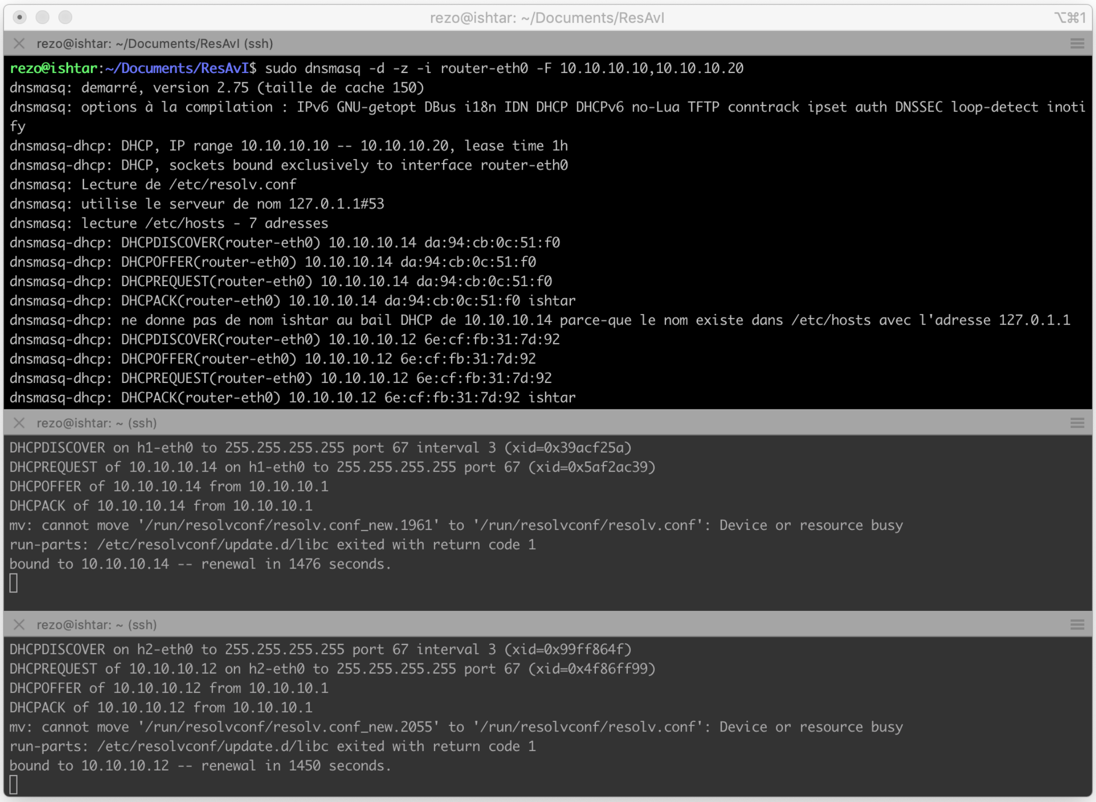


Fig. 1: Network Structure

3 Configure `dnsmasq` for DHCP

In order to apply DHCP, the interface `router-eth0` is firstly assigned the address 10.10.10.1/24 and its route to indicate our local network `ip r add 10.10.10.0/24 dev router-eth0`. To setup `dnsmasq` for DHCP, on the router (the VM), we run the command `dnsmasq -d -z -i router-eth0 -F 10.10.10.10,10.10.10.20` which means running in non-daemon, and binding to interface `router-eth0` (as shown in figure 2). On the client side, we execute `dhclient -d h1-eth0` (on h1, for instance) to request a dynamic IP for the interface h1-eth0 in type of non-daemon (-d).

Figure 3 shows a trace of exchanging packets for the configuration of a container with the server DHCP by using `tcpdump`. We can learn from this figure that DHCP receives a *request* to offer an IP 10.10.10.14 from a client. After checking current resources, DHCP *discovers* that this IP is available at the moment and allocates this IP address to the corresponding client.

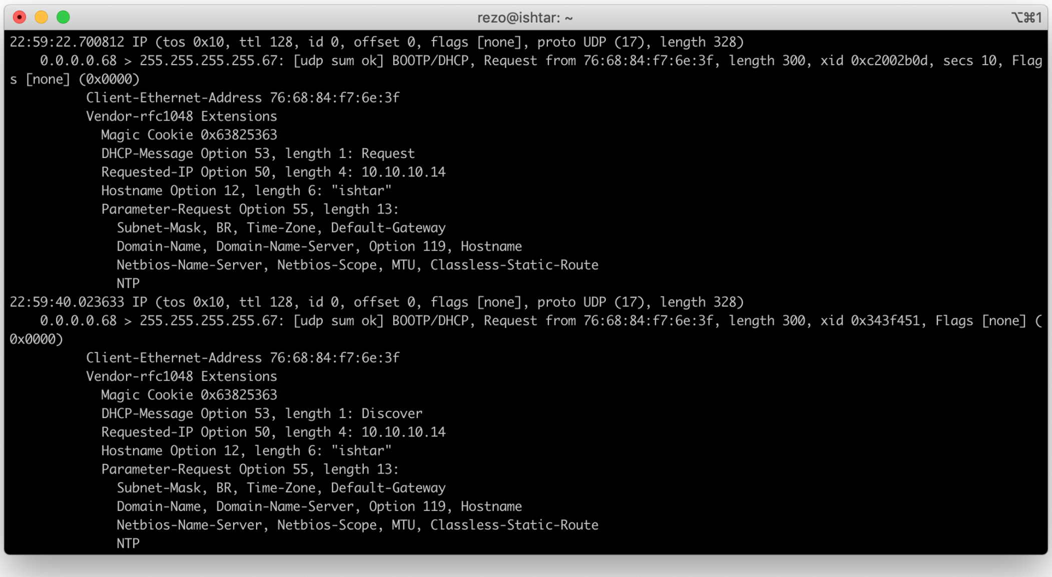

rezo@ishtar: ~/Documents/ResAvI
⌥⌘1
rezo@ishtar: ~/Documents/ResAvI (ssh)
rezo@ishtar: ~ (ssh)
rezo@ishtar: ~ (ssh)

```
rezo@ishtar:~/Documents/ResAvI$ sudo dnsmasq -d -z -i router-eth0 -F 10.10.10.10,10.10.10.20
dnsmasq: demarré, version 2.75 (taille de cache 150)
dnsmasq: options à la compilation : IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6 no-Lua TFTP conntrack ipset auth DNSSEC loop-detect inoti
fy
dnsmasq-dhcp: DHCP, IP range 10.10.10.10 -- 10.10.10.20, lease time 1h
dnsmasq-dhcp: DHCP, sockets bound exclusively to interface router-eth0
dnsmasq: Lecture de /etc/resolv.conf
dnsmasq: utilise le serveur de nom 127.0.1.1#53
dnsmasq: lecture /etc/hosts - 7 adresses
dnsmasq-dhcp: DHCPDISCOVER(router-eth0) 10.10.10.14 da:94:cb:0c:51:f0
dnsmasq-dhcp: DHCPOFFER(router-eth0) 10.10.10.14 da:94:cb:0c:51:f0
dnsmasq-dhcp: DHCPREQUEST(router-eth0) 10.10.10.14 da:94:cb:0c:51:f0
dnsmasq-dhcp: DHCPACK(router-eth0) 10.10.10.14 da:94:cb:0c:51:f0 ishtar
dnsmasq-dhcp: ne donne pas de nom ishtar au bail DHCP de 10.10.10.14 parce-que le nom existe dans /etc/hosts avec l'adresse 127.0.1.1
dnsmasq-dhcp: DHCPDISCOVER(router-eth0) 10.10.10.12 6e:cf:fb:31:7d:92
dnsmasq-dhcp: DHCPOFFER(router-eth0) 10.10.10.12 6e:cf:fb:31:7d:92
dnsmasq-dhcp: DHCPREQUEST(router-eth0) 10.10.10.12 6e:cf:fb:31:7d:92
dnsmasq-dhcp: DHCPACK(router-eth0) 10.10.10.12 6e:cf:fb:31:7d:92 ishtar
```

```
DHCPDISCOVER on h1-eth0 to 255.255.255.255 port 67 interval 3 (xid=0x39acf25a)
DHCPREQUEST of 10.10.10.14 on h1-eth0 to 255.255.255.255 port 67 (xid=0x5af2ac39)
DHCPOFFER of 10.10.10.14 from 10.10.10.1
DHCPACK of 10.10.10.14 from 10.10.10.1
mv: cannot move '/run/resolvconf/resolv.conf_new.1961' to '/run/resolvconf/resolv.conf': Device or resource busy
run-parts: /etc/resolvconf/update.d/libc exited with return code 1
bound to 10.10.10.14 -- renewal in 1476 seconds.
```

```
DHCPDISCOVER on h2-eth0 to 255.255.255.255 port 67 interval 3 (xid=0x99ff864f)
DHCPREQUEST of 10.10.10.12 on h2-eth0 to 255.255.255.255 port 67 (xid=0x4f86ff99)
DHCPOFFER of 10.10.10.12 from 10.10.10.1
DHCPACK of 10.10.10.12 from 10.10.10.1
mv: cannot move '/run/resolvconf/resolv.conf_new.2055' to '/run/resolvconf/resolv.conf': Device or resource busy
run-parts: /etc/resolvconf/update.d/libc exited with return code 1
bound to 10.10.10.12 -- renewal in 1450 seconds.
```

Fig. 2: Execute `dnsmasq` for DHCP



```
22:59:22.700812 IP (tos 0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)
    0.0.0.0.68 > 255.255.255.255.67: [udp sum ok] BOOTP/DHCP, Request from 76:68:84:f7:6e:3f, length 300, xid 0xc2002b0d, secs 10, Flag
s [none] (0x0000)
          Client-Ethernet-Address 76:68:84:f7:6e:3f
          Vendor-rfc1048 Extensions
            Magic Cookie 0x63825363
            DHCP-Message Option 53, length 1: Request
            Requested-IP Option 50, length 4: 10.10.10.14
            Hostname Option 12, length 6: "ishtar"
            Parameter-Request Option 55, length 13:
              Subnet-Mask, BR, Time-Zone, Default-Gateway
              Domain-Name, Domain-Name-Server, Option 119, Hostname
              Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
              NTP
22:59:40.023633 IP (tos 0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)
    0.0.0.0.68 > 255.255.255.255.67: [udp sum ok] BOOTP/DHCP, Request from 76:68:84:f7:6e:3f, length 300, xid 0x343f451, Flags [none] (
0x0000)
          Client-Ethernet-Address 76:68:84:f7:6e:3f
          Vendor-rfc1048 Extensions
            Magic Cookie 0x63825363
            DHCP-Message Option 53, length 1: Discover
            Requested-IP Option 50, length 4: 10.10.10.14
            Hostname Option 12, length 6: "ishtar"
            Parameter-Request Option 55, length 13:
              Subnet-Mask, BR, Time-Zone, Default-Gateway
              Domain-Name, Domain-Name-Server, Option 119, Hostname
              Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
              NTP
```

Fig. 3: A trace of exchange packets with server DHCP by `tcpdump`

4 Configure the firewall

In this section, we present the way to configure the firewall requested in *question c*. Our commands to configure are shown in listing 1.1 and 1.2 below. In details, we set some rules in NAT table and FILTER table.

Listing 1.1 indicates the commands we use in FILTER table. The rule in line 1 blocks by default the traffic from the private network 10.10.10.0/24. We then intercept this network communicate with DNS server by setting the rules in line 2 and 3. Every DNS queries to any server will be redirected to our DNS server in 10.10.10.1 : 53 and will be process by dnsmasq.

```
1 sudo iptables -P FORWARD DROP
2 sudo iptables -t nat -A PREROUTING -s 10.10.10.0/24 -p udp --dport 53 -j
     REDIRECT --to-ports 53
3 sudo iptables -t nat -A PREROUTING -s 10.10.10.0/24 -p tcp --dport 53 -j
     REDIRECT --to-ports 53
```

Listing 1.1: IPtables Rules

For the traffic from private network to the destination of Web, we implement 2 solutions during the development of the project. In the first solution,we run a TCP server at the address 10.10.10.1 and port 8080 which is in state of listening. We then redirect the traffic to this server by the configuration of DNAT shown in listing 1.2. The second solution is using a HTTP+SSL for the login server at 10.10.10.1 : 8080. It will be demonstrate further in the Extra Work 6.

```
1 sudo iptables -t nat -A PREROUTING -s 10.10.10.0/24 -i router-eth0 -p tcp --
     dport 80 -j REDIRECT --to-ports 8080
2 sudo iptables -t nat -A PREROUTING -s 10.10.10.0/24 -i router-eth0 -p tcp --
     dport 443 -j REDIRECT --to-ports 8080
```

Listing 1.2: Rules to redirect Web traffic

5 TCP server with Python

5.1 Modification of firewall before and after authentification

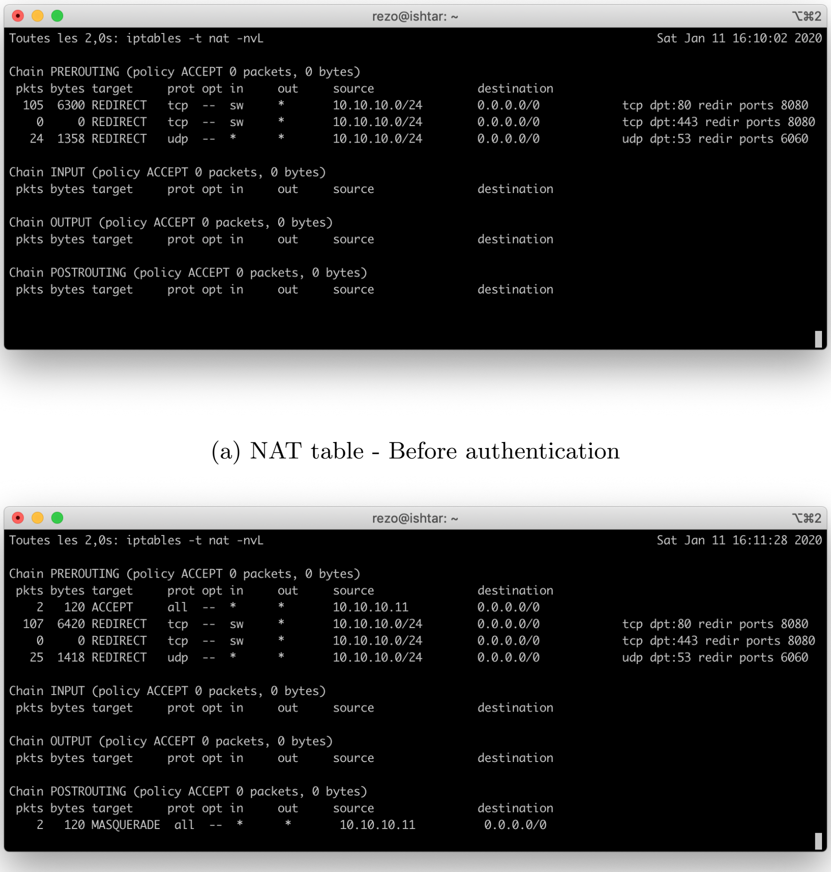
Figures 5 and 4 show a trace of modification of rules in firewall which are captured when using command `watch`. We could see the rules in NAT table and FITLER table before and after authorising the username and password. At the beginning, there are only some rules mentioned in section 4. If the account is successfully authorised, we set the rules to the firewall from TCP server written in Python as shown in listing 1.3. Line 1 is the rule of SNAT in version `MASQUERADE` for handling the traffic to Internet. Line 2 is the route of `PREROUTING` with the type of **INSERT** in order to avoid the packets will be redirected to TCP server again. Line 3 and 4 present the rules of forwarding packets from internal to external and vice versa.

```
1 cmd1 = "sudo iptables -t nat -A POSTROUTING -s {} -j MASQUERADE".format(
     client_address[0])
2 cmd2 = "sudo iptables -t nat -I PREROUTING -s {} -j ACCEPT".format(
     client_address[0])
3 cmd3 = "sudo iptables -A FORWARD -s {} -j ACCEPT".format(client_address[0])
4 cmd4 = "sudo iptables -A FORWARD -d {} -j ACCEPT".format(client_address[0])
```

Listing 1.3: Rules to allow client to access Internet

5.2 Captures of browser

For convenience, instead of using an application such as FireFox, we use the library `links` to access a certain link directly with bash. Figure 6 shows different states of login including a form to input information of authentication (figure 6a), a status when login failed (figure 6b) and another one



```
Toutes les 2,0s: iptables -t nat -nvL                                                  Sat Jan 11 16:10:02 2020

Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  105  6300 REDIRECT   tcp  --  sw     *       10.10.10.0/24        0.0.0.0/0            tcp dpt:80 redir ports 8080
    0     0 REDIRECT   tcp  --  sw     *       10.10.10.0/24        0.0.0.0/0            tcp dpt:443 redir ports 8080
   24  1358 REDIRECT   udp  --  *      *       10.10.10.0/24        0.0.0.0/0            udp dpt:53 redir ports 6060

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

(a) NAT table - Before authentication

```
Toutes les 2,0s: iptables -t nat -nvL                                                  Sat Jan 11 16:11:28 2020

Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   120 ACCEPT     all  --  *      *       10.10.10.11          0.0.0.0/0
  107  6420 REDIRECT   tcp  --  sw     *       10.10.10.0/24        0.0.0.0/0            tcp dpt:80 redir ports 8080
    0     0 REDIRECT   tcp  --  sw     *       10.10.10.0/24        0.0.0.0/0            tcp dpt:443 redir ports 8080
   25  1418 REDIRECT   udp  --  *      *       10.10.10.0/24        0.0.0.0/0            udp dpt:53 redir ports 6060

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   120 MASQUERADE  all  --  *      *       10.10.10.11          0.0.0.0/0
```

(b) NAT table - After authentication

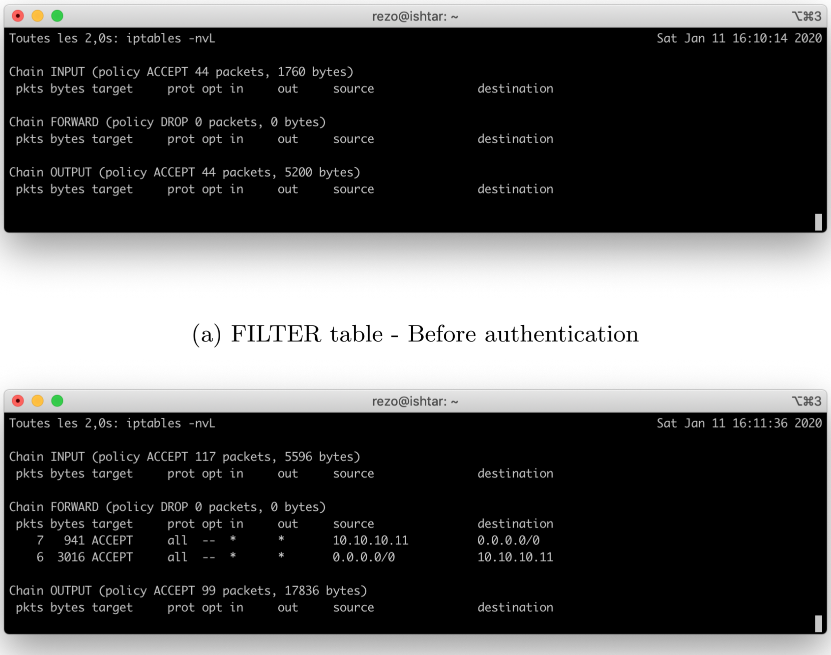
Fig. 4: Configuration of NAT before and after authentication

when login successful (figure 6c). When a client request to access a web page in external network, we redirect the request by DNAT configuration with target `REDIRECT`. The TCP server which is listening at $(10.10.10.1, 8080)$ catches this request, then sends a HTTP response to display the form demanding the client to input username and password. This response is shown in listing 1.4. The information containing username and password from this form is sent back to the TCP server by method `POST`.

After receiving username and password from client, TCP server uses `HTTPCookieProcessor()` provided in the requirement to authorise this account. We check the login whether successfully or not by examining the returned cookies. If this account is refused, TCP server send a HTTP response to inform that the login is failed as shown in figure 6b. Contrarily, when the login is successful, TCP server requests the page from external network and return to the client. From that time, client can exchange with external network.

```python
def get_page_html_auth():
  data = "HTTP/1.1 200 OK\r\n"
  data += "Content-Type: text/html; charset=utf-8\r\n"
  data += "\r\n"
  data += "<html><body><form action=\"\" method=\"POST\">\nUsername: <input type=\"text\" name=\"username\"><br>\n" + \
      "Password: <input type=\"password\" name=\"password\"><br>\n" + \
      "<input type=\"submit\" value=\"Connect\">\n" + \
      "</form></body></html>\r\n\r\n"
  return data.encode()
```

Listing 1.4: Response from TCP server to require authentication



```
Toutes les 2,0s: iptables -nvL                                                  Sat Jan 11 16:10:14 2020

Chain INPUT (policy ACCEPT 44 packets, 1760 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 44 packets, 5200 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

(a) FILTER table - Before authentication

```
Toutes les 2,0s: iptables -nvL                                                  Sat Jan 11 16:11:36 2020

Chain INPUT (policy ACCEPT 117 packets, 5596 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    7   941 ACCEPT     all  --  *      *       10.10.10.11          0.0.0.0/0
    6  3016 ACCEPT     all  --  *      *       0.0.0.0/0            10.10.10.11

Chain OUTPUT (policy ACCEPT 99 packets, 17836 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

(b) FILTER table - After authentication

Fig. 5: Configuration of FILTER before and after authentication

5.3 Capture by `tcpdump` of segment SYN

Figure 7 shows a screenshot of intercepting packets by using `tcpdump`. We run `tcpdump` on the interface 10.10.10.1 of router (above terminal) and on the interface *en*0*sp*3 (below terminal). They correspond to the interfaces which the packets from client meet before and after passing the router. As we can see in the figure, packets from the client have source IP of the host in local network 10.10.10.11 before going through router (above terminal). These packets after passing the router contain the source IP 10.0.2.5 (below terminal) which is the IP of *en*0*sp*3. So the source IPs of packets change when passing the router.

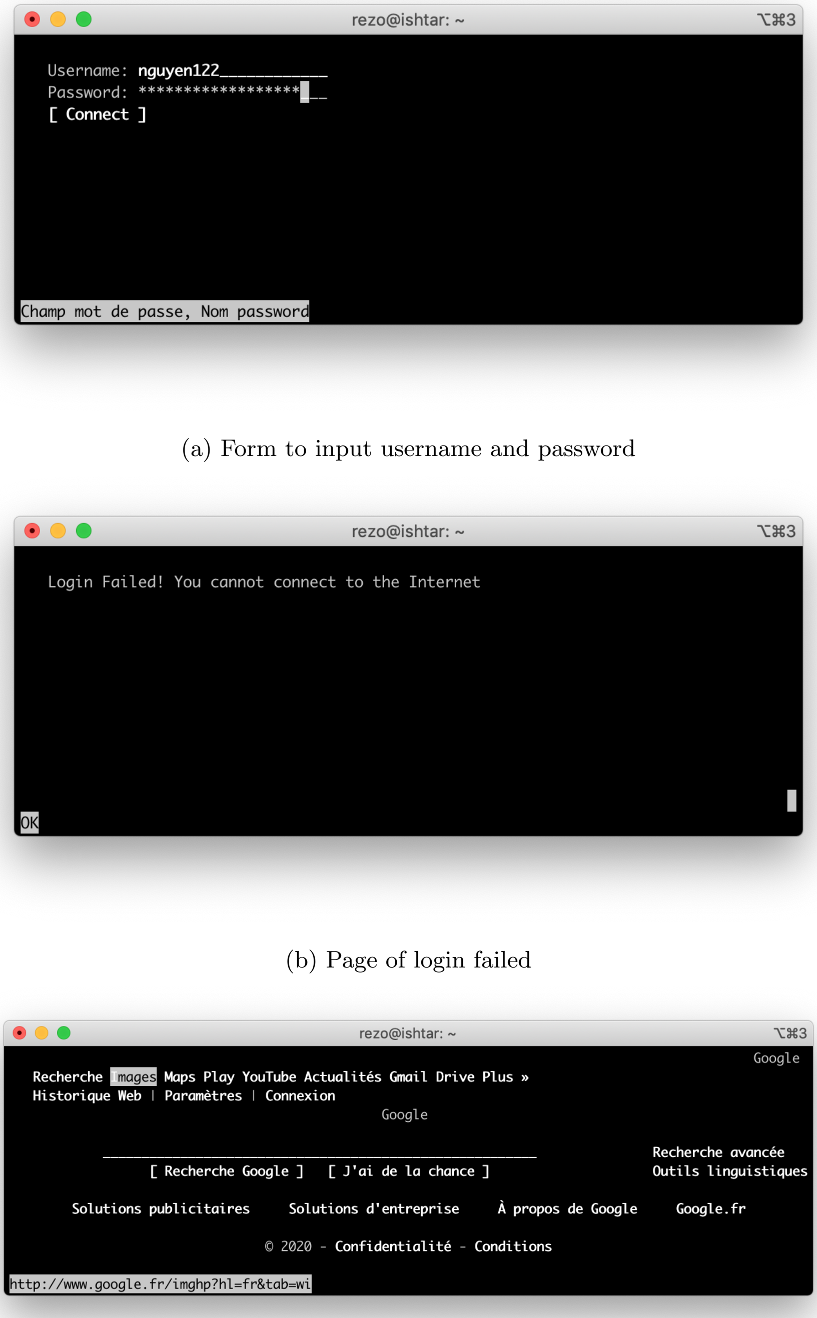
5.4 A trace by `Connection Tracking`

Figure 8 shows a trace by using Connection Tracking. We have already filtered the related logs to show. As we can see, the client 10.10.10.11 have a connection to port 80 of an external web server 216.58.215.36. This connection's state is ESTABLISHED. We also see a UDP connection from this client to DNS server in the next line.

6 Extra work

6.1 Configure `dnsmasq` to return fake DNS

We can configure `dnsmasq` using file `/etc/dnsmasq.conf` with option: `addn-hosts=/etc/dnsmasq.hosts`. Dnsmasq server will use file hosts to return the IP for queries. Inside `/etc/dnsmasq.hosts`, we put `1.1.1.1 www.facebook.com`. In this example, if a query ask for `www.facebook.com`, dnsmasq will return 1.1.1.1 as the ip address of `www.facebook.com`. For testing, as we can see in figure 9, if we excute the command `nslookup www.facebook.com` as the router, the DNS return is real DNS of Facebook. If we execute as namespace h1 or h2, the local dns server is Google and OpenDNS but the result is 1.1.1.1



```
Username: nguyen122___________
Password: *****************__
[ Connect ]

Champ mot de passe, Nom password
```

(a) Form to input username and password

```
Login Failed! You cannot connect to the Internet

OK
```

(b) Page of login failed

```
                                                                      Google
Recherche Images Maps Play YouTube Actualités Gmail Drive Plus »
Historique Web | Paramètres | Connexion
                                    Google

     _____________________________________________          Recherche avancée
         [ Recherche Google ]   [ J'ai de la chance ]       Outils linguistiques

    Solutions publicitaires    Solutions d'entreprise    À propos de Google    Google.fr

                    © 2020 - Confidentialité - Conditions

http://www.google.fr/imghp?hl=fr&tab=wi
```

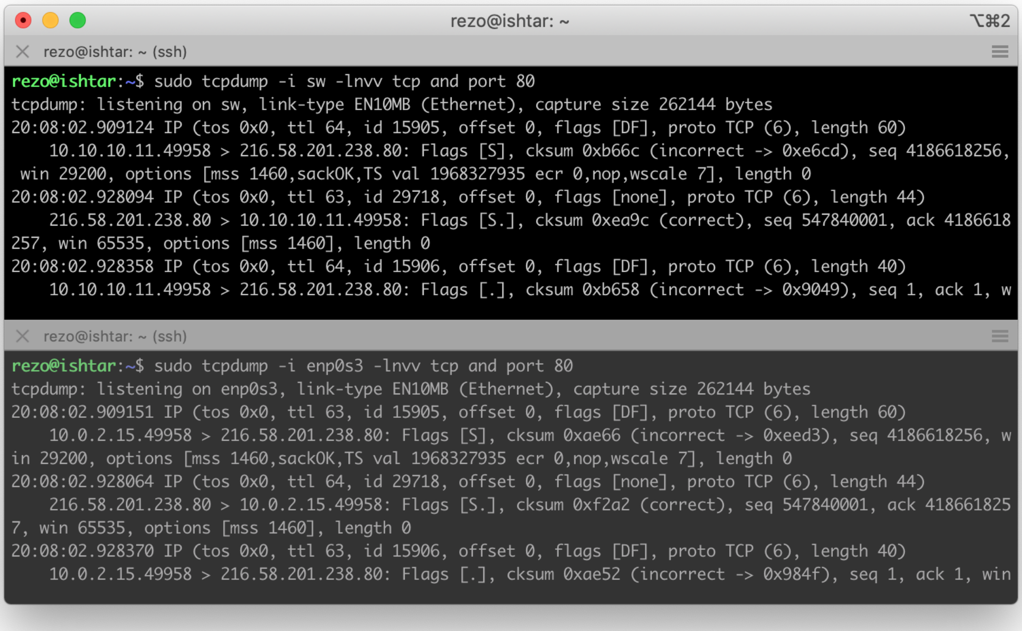
(c) Page of login successful (redirect to the original requested page of client)

Fig. 6: Captures of browser

6.2 HTTPS Login server

This is the second solution for web intercept using HTTP+SSL server. In this solution, we implement a HTTPS login server at the address 10.10.10.1 : 8080 and we create a non-SSL HTTP redirect server at the address 10.10.10.1 : 8181. We have to set different rules (listing 1.5) to DNAT any request to our redirect server. The redirect server will redirect to HTTPS login server.

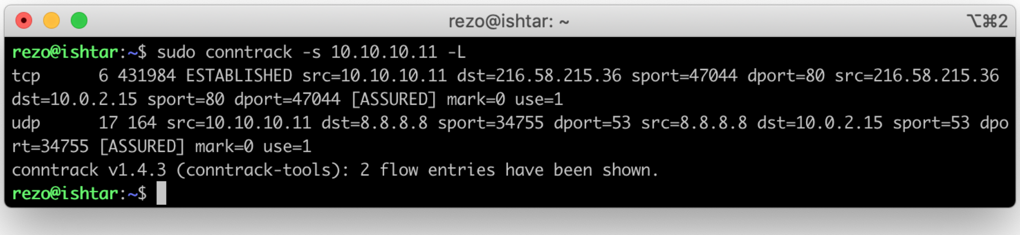
```
iptables -t nat -A PREROUTING -s 10.10.10.0/24 -i router-eth0 -p tcp --dport
    80 -j DNAT --to 10.10.10.1:8181
```



```
rezo@ishtar:~$ sudo tcpdump -i sw -lnvv tcp and port 80
tcpdump: listening on sw, link-type EN10MB (Ethernet), capture size 262144 bytes
20:08:02.909124 IP (tos 0x0, ttl 64, id 15905, offset 0, flags [DF], proto TCP (6), length 60)
    10.10.10.11.49958 > 216.58.201.238.80: Flags [S], cksum 0xb66c (incorrect -> 0xe6cd), seq 4186618256,
 win 29200, options [mss 1460,sackOK,TS val 1968327935 ecr 0,nop,wscale 7], length 0
20:08:02.928094 IP (tos 0x0, ttl 63, id 29718, offset 0, flags [none], proto TCP (6), length 44)
    216.58.201.238.80 > 10.10.10.11.49958: Flags [S.], cksum 0xea9c (correct), seq 547840001, ack 4186618
257, win 65535, options [mss 1460], length 0
20:08:02.928358 IP (tos 0x0, ttl 64, id 15906, offset 0, flags [DF], proto TCP (6), length 40)
    10.10.10.11.49958 > 216.58.201.238.80: Flags [.], cksum 0xb658 (incorrect -> 0x9049), seq 1, ack 1, w
```

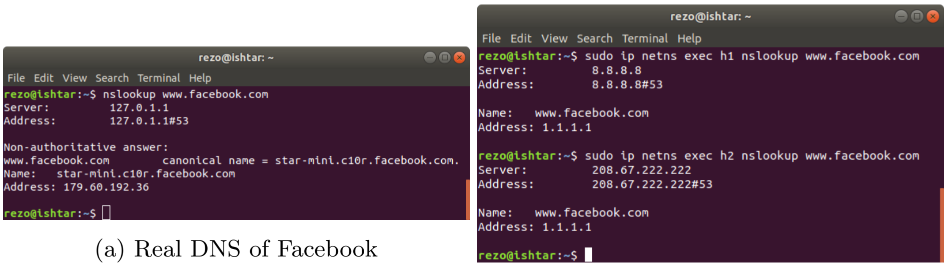
```
rezo@ishtar:~$ sudo tcpdump -i enp0s3 -lnvv tcp and port 80
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
20:08:02.909151 IP (tos 0x0, ttl 63, id 15905, offset 0, flags [DF], proto TCP (6), length 60)
    10.0.2.15.49958 > 216.58.201.238.80: Flags [S], cksum 0xae66 (incorrect -> 0xeed3), seq 4186618256, w
in 29200, options [mss 1460,sackOK,TS val 1968327935 ecr 0,nop,wscale 7], length 0
20:08:02.928064 IP (tos 0x0, ttl 64, id 29718, offset 0, flags [none], proto TCP (6), length 44)
    216.58.201.238.80 > 10.0.2.15.49958: Flags [S.], cksum 0xf2a2 (correct), seq 547840001, ack 418661825
7, win 65535, options [mss 1460], length 0
20:08:02.928370 IP (tos 0x0, ttl 63, id 15906, offset 0, flags [DF], proto TCP (6), length 40)
    10.0.2.15.49958 > 216.58.201.238.80: Flags [.], cksum 0xae52 (incorrect -> 0x984f), seq 1, ack 1, win
```

Fig. 7: Captures using `tcpdump` of segment SYN



```
rezo@ishtar:~$ sudo conntrack -s 10.10.10.11 -L
tcp      6 431984 ESTABLISHED src=10.10.10.11 dst=216.58.215.36 sport=47044 dport=80 src=216.58.215.36
dst=10.0.2.15 sport=80 dport=47044 [ASSURED] mark=0 use=1
udp      17 164 src=10.10.10.11 dst=8.8.8.8 sport=34755 dport=53 src=8.8.8.8 dst=10.0.2.15 sport=53 dpo
rt=34755 [ASSURED] mark=0 use=1
conntrack v1.4.3 (conntrack-tools): 2 flow entries have been shown.
rezo@ishtar:~$ 
```

Fig. 8: A trace by Connection Tracking


rezo@ishtar: ~
File Edit View Search Terminal Help
rezo@ishtar: ~
File Edit View Search Terminal Help

```
rezo@ishtar:~$ nslookup www.facebook.com
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
www.facebook.com        canonical name = star-mini.c10r.facebook.com.
Name:   star-mini.c10r.facebook.com
Address: 179.60.192.36

rezo@ishtar:~$ 
```

(a) Real DNS of Facebook

```
rezo@ishtar:~$ sudo ip netns exec h1 nslookup www.facebook.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Name:   www.facebook.com
Address: 1.1.1.1

rezo@ishtar:~$ sudo ip netns exec h2 nslookup www.facebook.com
Server:         208.67.222.222
Address:        208.67.222.222#53

Name:   www.facebook.com
Address: 1.1.1.1

rezo@ishtar:~$ 
```

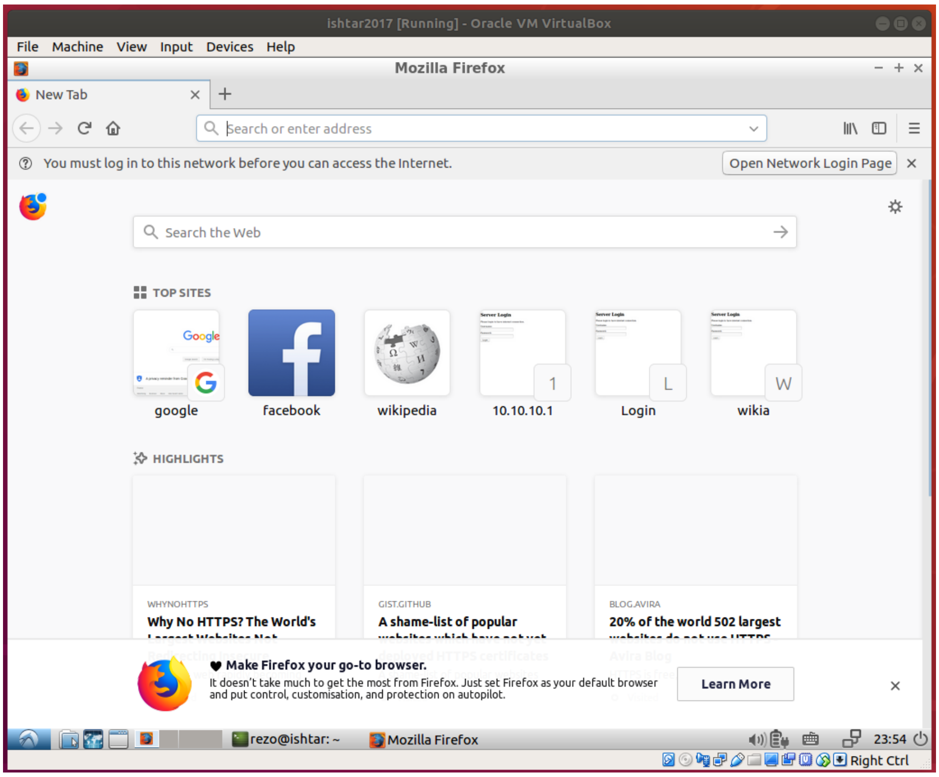
(b) Fake DNS of Facebook

Fig. 9: DNS Intercept using /etc/dnsmasq.hosts

```
iptables -t nat -A PREROUTING -s 10.10.10.0/24 -i router-eth0 -p udp --dport
    443 -j DNAT --to 10.10.10.1:8181
```

Listing 1.5: Rules for HTTPS redirect

In order to overcome some inconvenient and to have better experience when using Firefox, we implemented mechanism that using a redirect server. We start firefox process in h1. In figure 10 The redirect server can be found automatically. After access the redirect server, users will be redirected to a HTTPS login web server. In the login website, users can login to have internet as describe above. For convenience, all the services (including the creation of network structure, dnsmasq for DHCP and DNS Intercept, HTTPS Login web server, HTTP redirect server) will be initiated from `sudo ./extra_work/build_architecture`. All given services and all redundant directories can be deleted with `sudo ./extra_work/build_architecture`.



You must log in to this network before you can access the Internet.

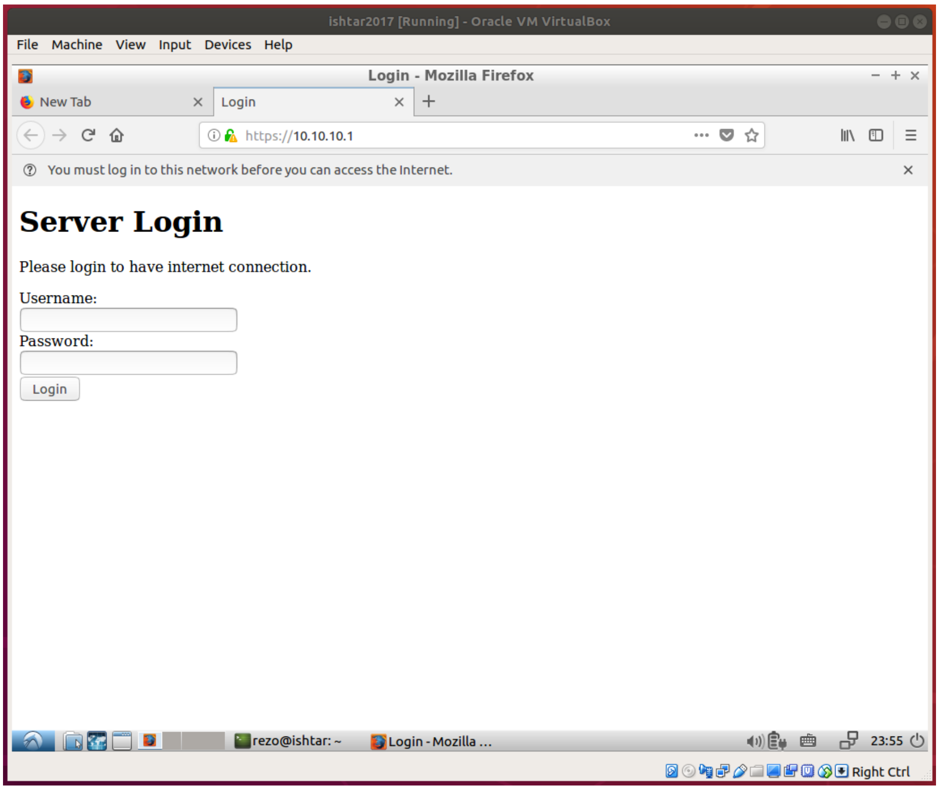
Make Firefox your go-to browser.

It doesn't take much to get the most from Firefox. Just set Firefox as your default browser and put control, customisation, and protection on autopilot.

Fig. 10: Firefox recognize redirect server

7 Conclusion

In this project, we successfully build a network and set `dnsmasq` as well as configure the firewall in a precise way to protect the local network. Besides, we not only complete the given requirements but also raise an another approach to build the TCP server.



You must log in to this network before you can access the Internet.

Server Login

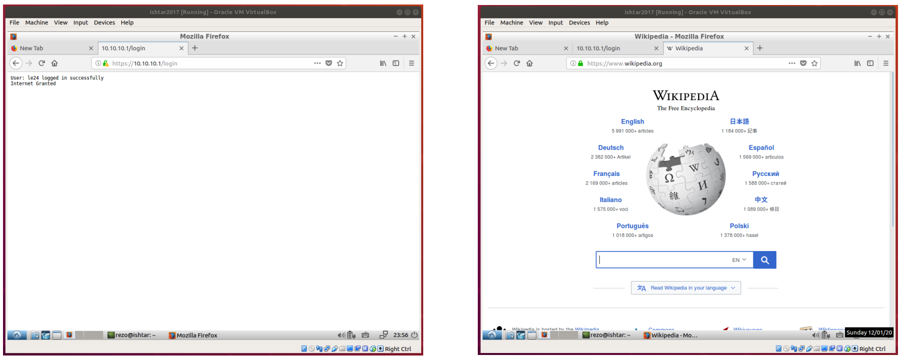
Please login to have internet connection.

Username:

Password:

Login

Fig. 11: Firefox Login web server



```
User: le24 logged in successfully
Internet Granted
```

(a) Firefox Login Success

(b) Firefox Connect to Internet

Fig. 12: Firefox Result