Master 1 Cryptis Computer Science

Report Reseaux Advance Projet

Realized by:
DO Duy Huy Hoang
NGUYEN Thi Mai Phuong

Version du
January 10, 2021

List of Figures

Contents

I. Introduction

Context and Motivation

A captive portal is a web page accessed with a web browser that is displayed to newly connected users of a Wi-Fi or wired network before they are granted broader access to network resources. Captive portals are commonly used to present a landing or log-in page which may require authentication, payment, acceptance of an end-user license agreement, acceptable use policy, survey completion, or other valid credentials that both the host and user agree to to redirect users of a network that provides outbound internet access to a web page that displays the terms of service.

Objectives

In this project, we have done the following:

- Configuring
  - The network
  - DHCP Server
  - DNS service
  - Firewall
  - TCP Server

II. Configuring network

In this section, we present our network structure which is described in the below figure. We create two machine hA and hB which are connected to a switch mS. mS connect to our VM which act as a router with firewall.

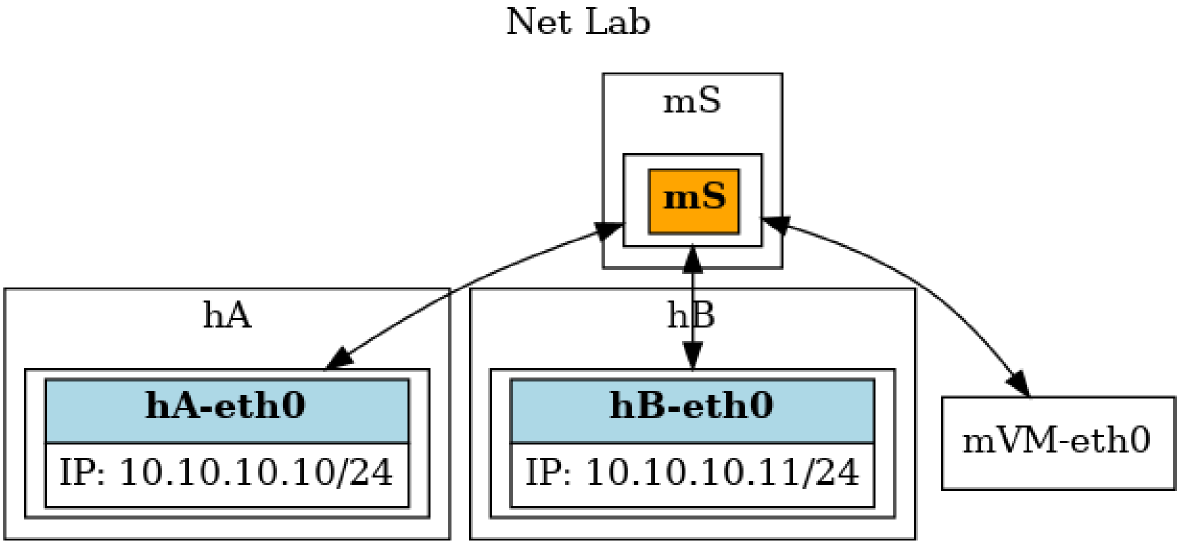


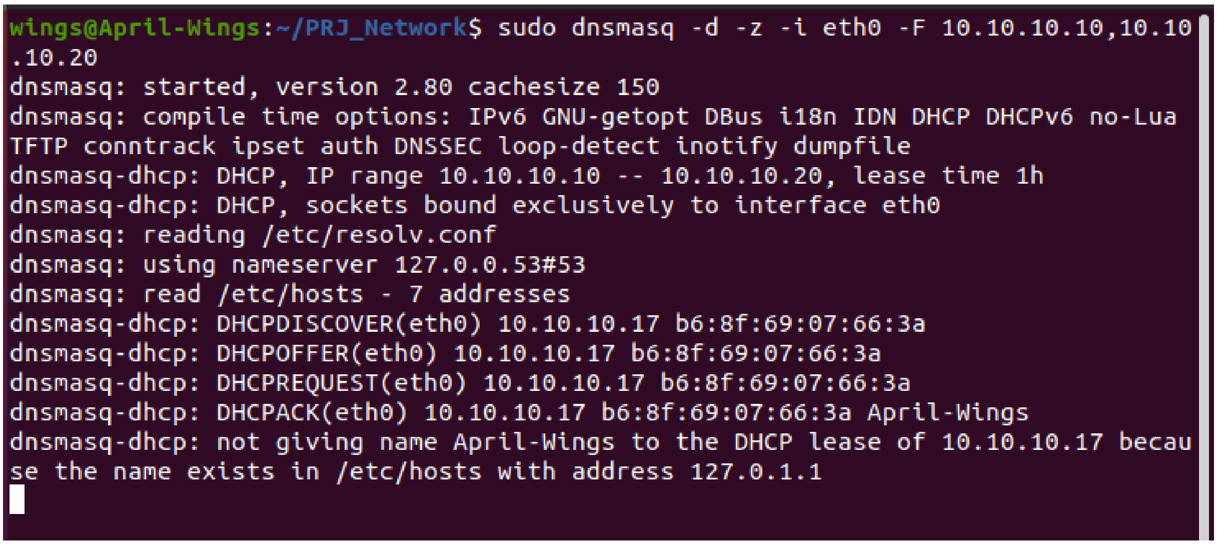
Figure 1: Network Architecture

III. Configuring DHCP Server and DNS Service

In order to ensure the configuration by DHCP with netns connected in the private network 10.10.10.0/24. We use `dnsmasq` which provides a DNS server suitable for resource constrained routers and firewalls.

To setup dnsmasq for DHCP, we execute the following command for binding our VM to `meth0` interface.

```
sudo dnsmasq -d -z -i meth0 -F 10.10.10.10,10.10.10.20
```



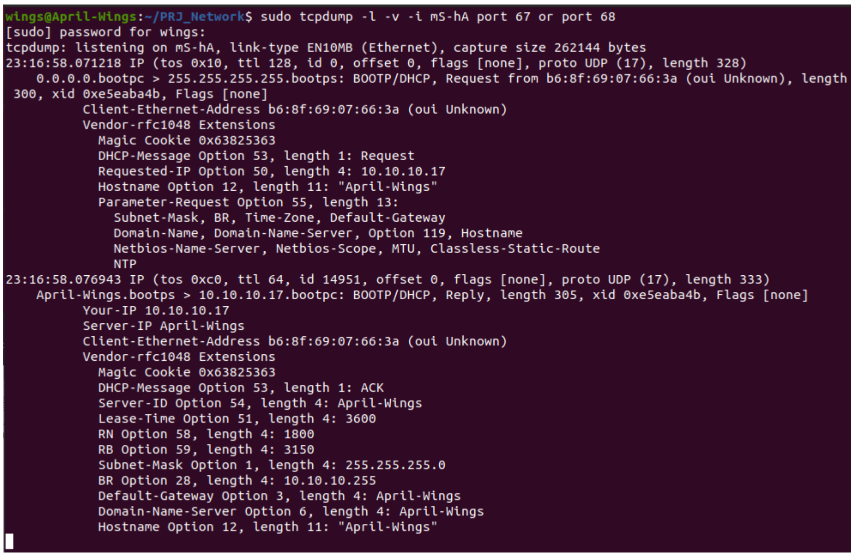
```
wings@April-Wings:~/PRJ_Network$ sudo dnsmasq -d -z -i eth0 -F 10.10.10.10,10.10
.10.20
dnsmasq: started, version 2.80 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6 no-Lua
TFTP conntrack ipset auth DNSSEC loop-detect inotify dumpfile
dnsmasq-dhcp: DHCP, IP range 10.10.10.10 -- 10.10.10.20, lease time 1h
dnsmasq-dhcp: DHCP, sockets bound exclusively to interface eth0
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 127.0.0.53#53
dnsmasq: read /etc/hosts - 7 addresses
dnsmasq-dhcp: DHCPDISCOVER(eth0) 10.10.10.17 b6:8f:69:07:66:3a
dnsmasq-dhcp: DHCPOFFER(eth0) 10.10.10.17 b6:8f:69:07:66:3a
dnsmasq-dhcp: DHCPREQUEST(eth0) 10.10.10.17 b6:8f:69:07:66:3a
dnsmasq-dhcp: DHCPACK(eth0) 10.10.10.17 b6:8f:69:07:66:3a April-Wings
dnsmasq-dhcp: not giving name April-Wings to the DHCP lease of 10.10.10.17 becau
se the name exists in /etc/hosts with address 127.0.1.1
```

Figure 2: Execute dnsmasq for DHCP

After that, we might want to request a dynamic IP for `hA-eth0` and `hB-eth0` by using these command:

```
sudo ip netns exec hA dhclient -d hA-eth0
sudo ip netns exec hB dhclient -d hB-eth0
```

For tracing the packet exchanges, we use tcpdump on the DHCP server for port 67 or 68, the result is shown below



```
wings@April-Wings:~/PRJ_Network$ sudo tcpdump -l -v -i mS-hA port 67 or port 68
[sudo] password for wings:
tcpdump: listening on mS-hA, link-type EN10MB (Ethernet), capture size 262144 bytes
23:16:58.071218 IP (tos 0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)
    0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from b6:8f:69:07:66:3a (oui Unknown), length
 300, xid 0xe5eaba4b, Flags [none]
          Client-Ethernet-Address b6:8f:69:07:66:3a (oui Unknown)
          Vendor-rfc1048 Extensions
            Magic Cookie 0x63825363
            DHCP-Message Option 53, length 1: Request
            Requested-IP Option 50, length 4: 10.10.10.17
            Hostname Option 12, length 11: "April-Wings"
            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
23:16:58.076943 IP (tos 0xc0, ttl 64, id 14951, offset 0, flags [none], proto UDP (17), length 333)
    April-Wings.bootps > 10.10.10.17.bootpc: BOOTP/DHCP, Reply, length 305, xid 0xe5eaba4b, Flags [none]
          Your-IP 10.10.10.17
          Server-IP April-Wings
          Client-Ethernet-Address b6:8f:69:07:66:3a (oui Unknown)
          Vendor-rfc1048 Extensions
            Magic Cookie 0x63825363
            DHCP-Message Option 53, length 1: ACK
            Server-ID Option 54, length 4: April-Wings
            Lease-Time Option 51, length 4: 3600
            RN Option 58, length 4: 1800
            RB Option 59, length 4: 3150
            Subnet-Mask Option 1, length 4: 255.255.255.0
            BR Option 28, length 4: 10.10.10.255
            Default-Gateway Option 3, length 4: April-Wings
            Domain-Name-Server Option 6, length 4: April-Wings
            Hostname Option 12, length 11: "April-Wings"
```

Figure 3: A trace of the packet exchanges on the DHCP server

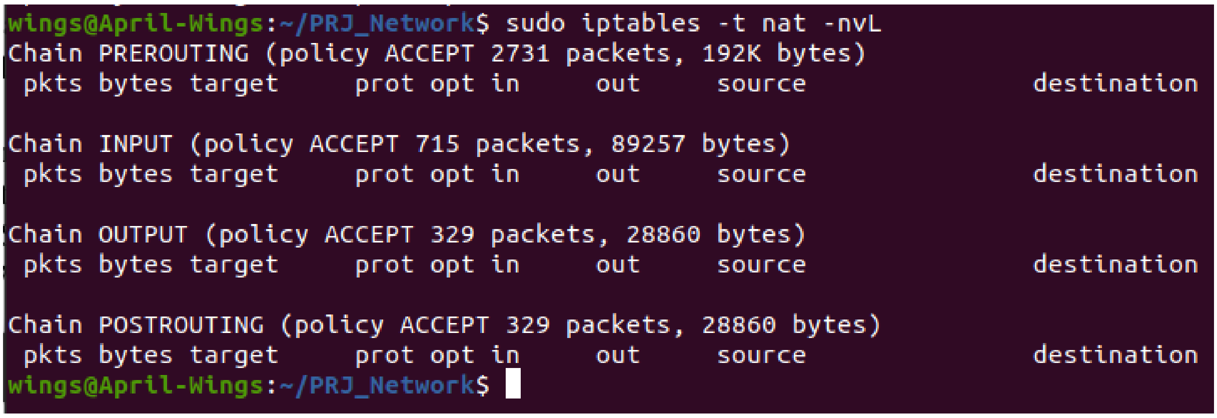
III. Configuring Firewall

For configuring the firewall, we set some rules in NAT and FILTER table.

```
sudo iptables -I FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
sudo iptables -t nat -A POSTROUTING -s 10.10.10.0/24 -j MASQUERADE (private
    network)
sudo iptables -A FORWARD -s 10.10.10.0/24 -p tcp --dport 53 -j ACCEPT
sudo iptables -A FORWARD -s 10.10.10.0/24 -p udp --dport 53 -j ACCEPT
```

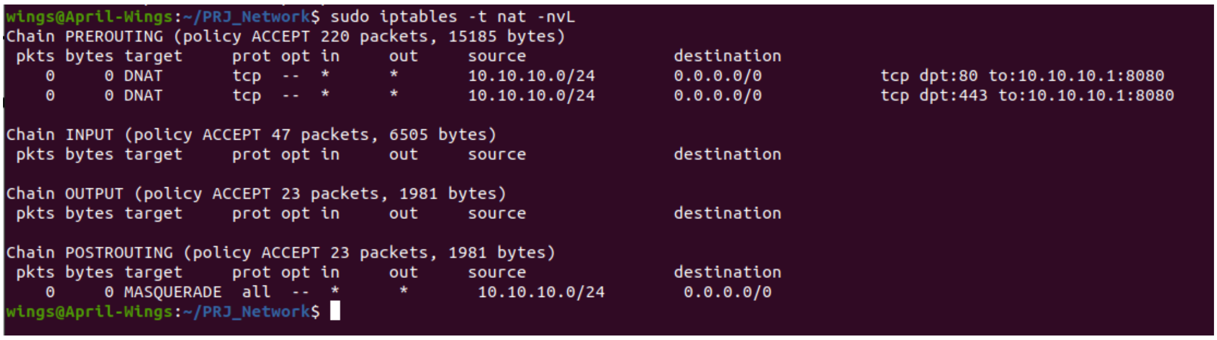
For redirecting traffic from the private network to the destination of the Web:

```
sudo iptables -t nat -A PREROUTING -s 10.10.10.0/24 -p tcp --dport 80 -j DNAT
    --to-destination 10.10.10.1:8080
sudo iptables -t nat -A PREROUTING -s 10.10.10.0/24 -p tcp --dport 443 -j DNAT
    --to-destination 10.10.10.1:8080
```



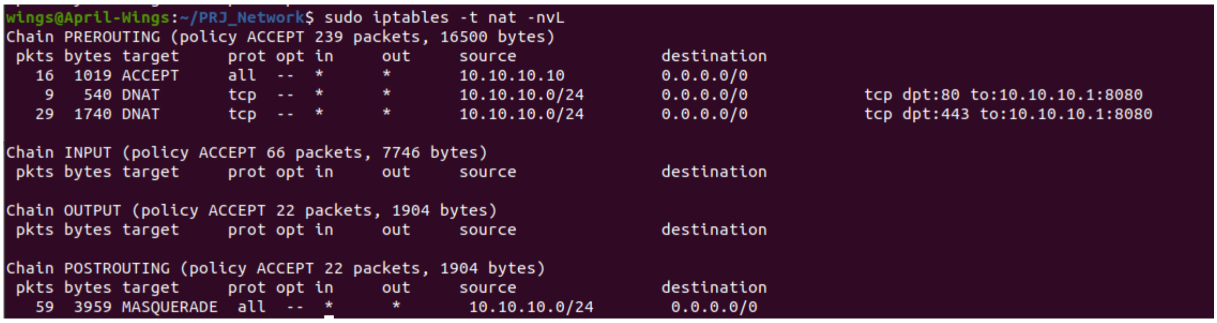
```
wings@April-Wings:~/PRJ_Network$ sudo iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 2731 packets, 192K bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 715 packets, 89257 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 329 packets, 28860 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 329 packets, 28860 bytes)
 pkts bytes target     prot opt in     out     source               destination
wings@April-Wings:~/PRJ_Network$
```

Figure 4: Nat tables before config



```
wings@April-Wings:~/PRJ_Network$ sudo iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 220 packets, 15185 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DNAT       tcp  --  *      *       10.10.10.0/24        0.0.0.0/0            tcp dpt:80 to:10.10.10.1:8080
    0     0 DNAT       tcp  --  *      *       10.10.10.0/24        0.0.0.0/0            tcp dpt:443 to:10.10.10.1:8080

Chain INPUT (policy ACCEPT 47 packets, 6505 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 23 packets, 1981 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 23 packets, 1981 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      *       10.10.10.0/24        0.0.0.0/0
wings@April-Wings:~/PRJ_Network$
```

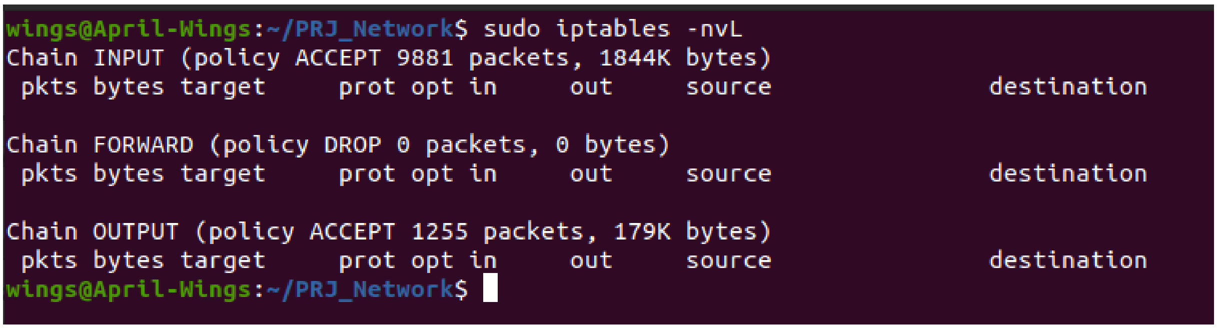
Figure 5: Nat tables after config



```
wings@April-Wings:~/PRJ_Network$ sudo iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 239 packets, 16500 bytes)
 pkts bytes target     prot opt in     out     source               destination
   16  1019 ACCEPT     all  --  *      *       10.10.10.10          0.0.0.0/0
    9   540 DNAT       tcp  --  *      *       10.10.10.0/24        0.0.0.0/0            tcp dpt:80 to:10.10.10.1:8080
   29  1740 DNAT       tcp  --  *      *       10.10.10.0/24        0.0.0.0/0            tcp dpt:443 to:10.10.10.1:8080

Chain INPUT (policy ACCEPT 66 packets, 7746 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 22 packets, 1904 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 22 packets, 1904 bytes)
 pkts bytes target     prot opt in     out     source               destination
   59  3959 MASQUERADE  all  --  *      *       10.10.10.0/24        0.0.0.0/0
```

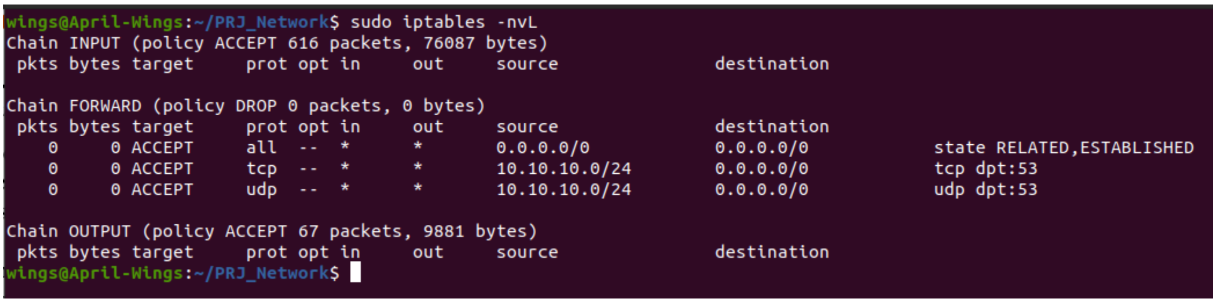
Figure 6: Nat tables after authentication

The result below shows an output of the "iptables -nL" command for the different modified tables as before config, after config and after authentication.



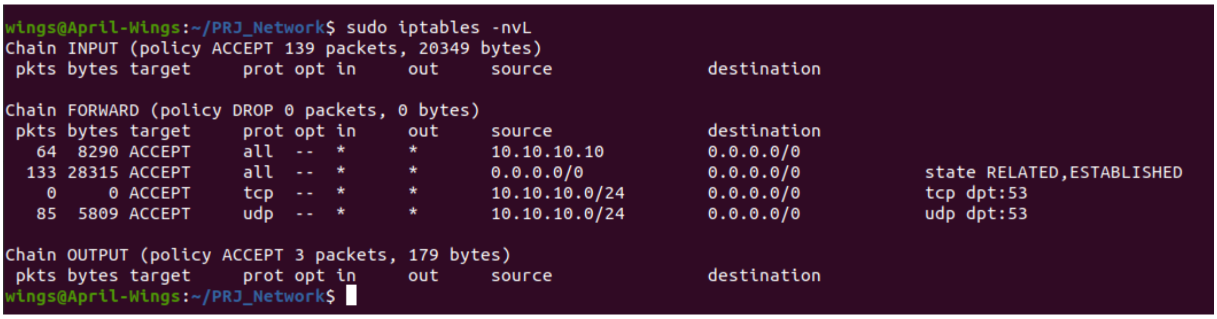
```
wings@April-Wings:~/PRJ_Network$ sudo iptables -nvL
Chain INPUT (policy ACCEPT 9881 packets, 1844K bytes)
 pkts bytes target     prot opt in     out     source               destination

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

Chain OUTPUT (policy ACCEPT 1255 packets, 179K bytes)
 pkts bytes target     prot opt in     out     source               destination
wings@April-Wings:~/PRJ_Network$ █
```

Figure 7: IP Tables before config



```
wings@April-Wings:~/PRJ_Network$ sudo iptables -nvL
Chain INPUT (policy ACCEPT 616 packets, 76087 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
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       10.10.10.0/24        0.0.0.0/0            tcp dpt:53
    0     0 ACCEPT     udp  --  *      *       10.10.10.0/24        0.0.0.0/0            udp dpt:53

Chain OUTPUT (policy ACCEPT 67 packets, 9881 bytes)
 pkts bytes target     prot opt in     out     source               destination
wings@April-Wings:~/PRJ_Network$ █
```

Figure 8: IP Tables after config



```
wings@April-Wings:~/PRJ_Network$ sudo iptables -nvL
Chain INPUT (policy ACCEPT 139 packets, 20349 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
   64  8290 ACCEPT     all  --  *      *       10.10.10.10          0.0.0.0/0
  133 28315 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       10.10.10.0/24        0.0.0.0/0            tcp dpt:53
   85  5809 ACCEPT     udp  --  *      *       10.10.10.0/24        0.0.0.0/0            udp dpt:53

Chain OUTPUT (policy ACCEPT 3 packets, 179 bytes)
 pkts bytes target     prot opt in     out     source               destination
wings@April-Wings:~/PRJ_Network$ 
```

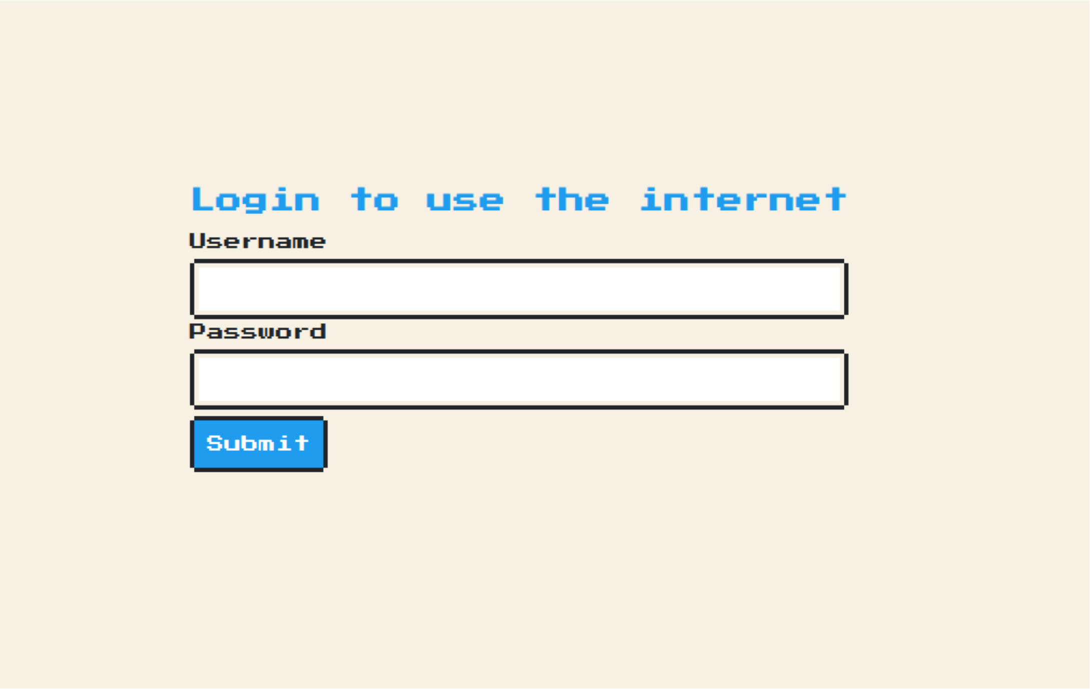
Figure 9: IP Tables after authentication

IV. TCP Server

In this section, we implement a TCP server which serve a login page for client request to access a website in external network. In order to authenticate, we use a perform secure authentication with CAS Unilim server by using `LemonLDAP`

```python
def getCookies(username, password, token):
    cookieProcessor = urllib.request.HTTPCookieProcessor()
    opener = urllib.request.build_opener(cookieProcessor)
    data = urllib.parse.urlencode({'user': username, 'password': password, 'token': token})
    request = urllib.request.Request('https://cas.unilim.fr', bytes(data, encoding='ascii'))
    reponse = opener.open(request)
    cookies = [c for c in cookieProcessor.cookiejar if c.name == 'lemonldap']
    return cookies
```

If the client logins successfully, we will update our firewall with the `remote_IP` and redirect to a successful webpage otherwise, the client will be returned to a login page with a alert message.

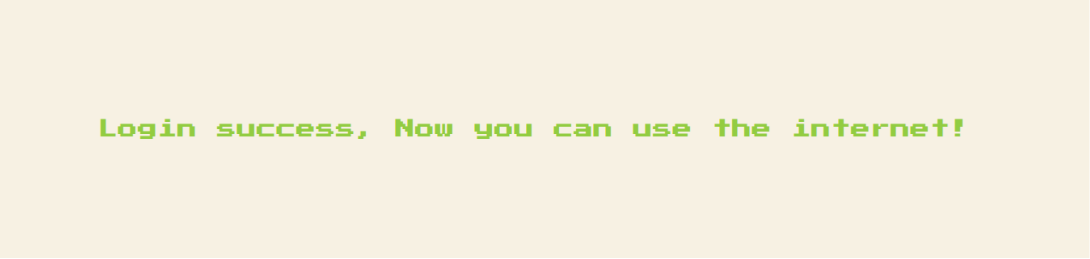


Login to use the internet

Username

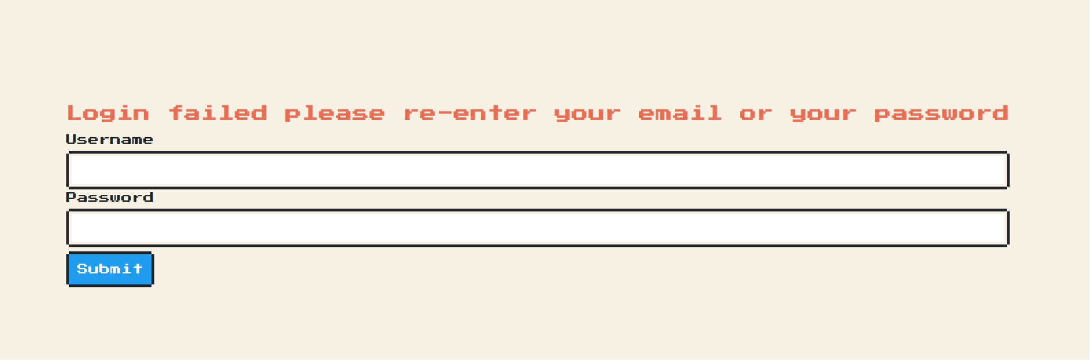
Password

Figure 10: Login page



Login success, Now you can use the internet!

Figure 11: Success page



Login failed please re-enter your email or your password

Username

Password

Figure 12: Failed page

Conclusion and Future works

Conclusion

In this project, we have created a captive portal network with the flowing task:

- Setup and configuring the network as the project description
- Setup a DHCP and DNS server and Configuring Firewall in a precise way to protect the local network.
- Create TCP server

Future works

- Implementing more rules and a better network structure.
- Redirect page to the destination website (instead of successful page).

References

Captive portal Wiki

Captive portal by science direct