Name of the Department: Computer Science & Engineering

Semester: Spring 2024

Course Number: CSE - 487

Course Title: Cybersecurity, Law & Ethics

Project Title: Securing a Networked System with Public Key Infrastructure

Group Number: 507

Group ID:

1. 2021-2-60-041 (Fardin Islam)
2. 2021-2-60-008 (Fardin Rahman)
3. 2021-2-60-089 (Kazi Md Nafiz Fuad)

Date of Submission: 05/05/2024

Contents

OS:

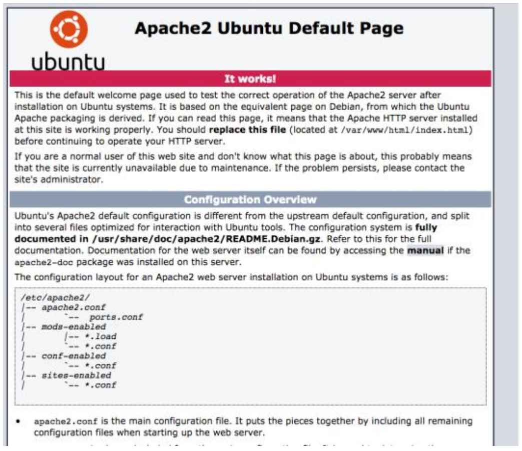
Linux is best. But windows will also do. We are using a Debian based Linux.

Software and Configuration Needed:

1. Bind9 for DNS.
2. OpenSSL for CA infrastructure.
3. Apache2 for Webhosting.
4. Snort for Intrusion Detection.
5. UFW for Firewall.
6. If VM, convert the network to host-only network from virtual machine’s settings.

Configuring Apache2:

1. Install Apache2.
2. Go to browser and type localhost to test the server is running (systemctl status apache2).


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.

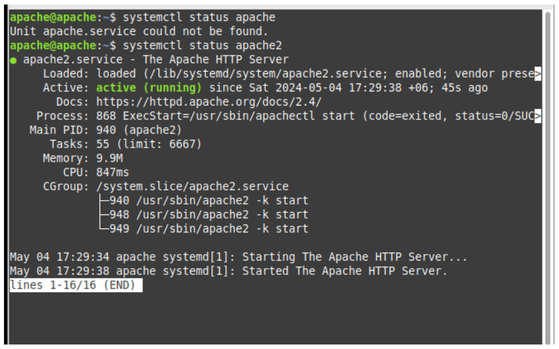
Configuration Overview

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.


```
apache@apache:~$ systemctl status apache
Unit apache.service could not be found.
apache@apache:~$ systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese>
     Active: active (running) since Sat 2024-05-04 17:29:38 +06; 45s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 868 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUC>
   Main PID: 940 (apache2)
      Tasks: 55 (limit: 6667)
     Memory: 9.9M
        CPU: 847ms
     CGroup: /system.slice/apache2.service
             ├─940 /usr/sbin/apache2 -k start
             ├─948 /usr/sbin/apache2 -k start
             └─949 /usr/sbin/apache2 -k start

May 04 17:29:34 apache systemd[1]: Starting The Apache HTTP Server...
May 04 17:29:38 apache systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

3. Change user to super or root. *super su*
4. mkdir /var/www/< name of your domain > example, mkdir /var/www/fardin.home
5. create a index.html for testing. nano /var/www/fardin.home/index.html
6. change directory to /etc/apache2/sites-available. *cd* /etc/apache2/sites – available
7. make a new conf file for fardin.home. cp 000 – default.conf fardin.home.conf

8. Edit the configuration file of your website to your need. I am writing down mine.

```
<VirtualHost *:443>
        SSLEngine on
        ServerAdmin mfardinr@gmail.com
        ServerName fardin.home
        ServerAlias www.fardin.home
        DocumentRoot /var/www/fardin.home
        SSLCertificateFile      "/var/www/fardin.home/generated/chained.crt"
        SSLCertificateKeyFile "/var/www/fardin.home/generated/server.key"
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>


# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

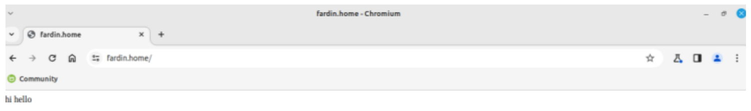
9. Now we have to enable site the new conf file. a2ensite fardin.home.conf

10. Now we have to disable default site (optional). a2dissite 000 – default.conf

11. Now enable SSL module. a2enmod ssl

12. Now we have to restart the apache2 *systemctl restart apache*2

13. Now if we go to the browser and type www.fardin.home we should get our website.



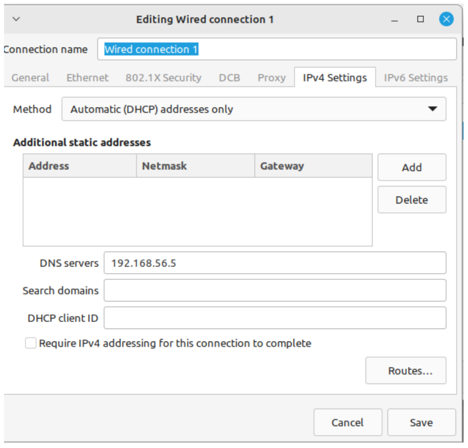
hi hello

Configuring Bind9:

1. Change user to super or root. *super su*
2. Change directory to /etc/bind. *cd* /etc/bind
3. Open named.conf.options with nano. *nano named.conf.options*
4. Configure a forwarder and subnet for the service.

```
acl internal-network{
192.168.56.0/24;
};
options {
        directory "/var/cache/bind";
        allow-query { localhost; internal-network; };
        allow-transfer { localhost; };
        forwarders {1.1.1.1;  8.8.8.8; };


        dnssec-validation auto;
        listen-on-v6 { any; };
};
```

5. Stop the system's DNS. *systemctl stop systemd − resolved*
6. Disable the system's DNS on startup. *systemctl disable systemd − resolved*
7. Add the DNS's IP to the network adapter.



Connection name: Wired connection 1

Method: Automatic (DHCP) addresses only

Additional static addresses

Address	Netmask	Gateway

DNS servers: 192.168.56.5

Search domains:

DHCP client ID:

☐ Require IPv4 addressing for this connection to complete

8. Change the dns to your system's ip. nano /etc/resolve.conf

9. Now open named.conf.local. *nano named.conf.local*

10. Add forward zone for your website.

```
zone "fardin.home" IN{
        type master;
        file "/etc/bind/forward.fardin.home";
        allow-update { any; };
    };
```

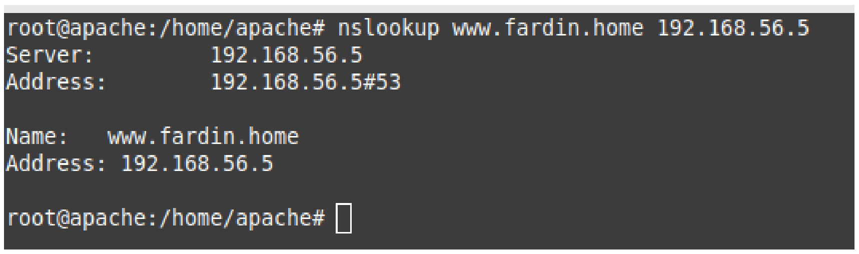
11. Now create the forward file. *nano forward.fardin.home*

```
$TTL    604800
@       IN      SOA     www.fardin.home. mfardin.gmail.com. (
                        2024042400    ; Serial
                         604800             ; Refresh
                          86400             ; Retry
                        2419200             ; Expire
                         604800 )     ; Negative Cache TTL
;
@       IN      NS      www.fardin.home.
www.fardin.home.        IN      A       192.168.56.5
ns1.fardin.home.        IN      A       192.168.56.5
```

12. Now restart the named. *systemctl restart named*

13. Now check the status. If everything okay it should be active. *systemctl status named*

14. Now use nslookup tool to see if the DNS is working. *nslookup www.fardin.home* 192.168.56.5



```
root@apache:/home/apache# nslookup www.fardin.home 192.168.56.5
Server:         192.168.56.5
Address:        192.168.56.5#53

Name:   www.fardin.home
Address: 192.168.56.5

root@apache:/home/apache# 
```

Create Hierarchical CA Infrastructure:

1. Create a ca directory at your desired location. `mkdir ca`
2. Create file structure.
   a. `mkdir -p {root-ca,sub-ca,server}/`
      `{private,certs,index,serial,pem,crl,csr}`
   b. `mkdir generated`
3. Create index files
   a. `touch root-ca/index/index`
   b. `touch sub-ca/index/index`
4. Create serial files
   a. `openssl rand -hex 16 > root-ca/serial/serial`
   b. `openssl rand -hex 16 > sub-ca/serial/serial`
5. Cerate the conf files.
   a. `Nano root-ca/root-ca.conf`

```
#root-ca.conf
[ca]
default_ca    = CA_default


[CA_default]
dir      = root-ca
certs      =  $dir/certs
crl_dir     = $dir/crl
new_certs_dir   = $dir/pem
database   = $dir/index/index
serial     = $dir/serial/serial
RANDFILE   = $dir/private/.rand
private_key   = $dir/private/ca.key
```

```
certificate   = $dir/certs/ca.crt
crlnumber   = $dir/crlnumber
crl    =  $dir/crl/ca.crl
crl_extensions   = crl_ext
default_crl_days    = 30
default_md   = sha256


name_opt   = ca_default
cert_opt   = ca_default
default_days  = 365
preserve   = no
policy    = policy_strict

[ policy_strict ]
countryName   = supplied
stateOrProvinceName  =  supplied
organizationName  = match
organizationalUnitName  =  optional
commonName   =  supplied
emailAddress   =  optional

[ policy_loose}
countryName   = optional
stateOrProvinceName  = optional
localityName   = optional
organizationName  = optional
organizationalUnitName   = optional
commonName   = supplied
emailAddress   = optional
```

```
[ req ]
# Options for the req tool, man req.
default_bits   = 2048
distinguished_name  = req_distinguished_name
string_mask   = utf8only
default_md  =  sha256
# Extension to add when the -x509 option is used.
x509_extensions   = v3_ca

[ req_distinguished_name ]
countryName                    = Country Name (2 letter code)
stateOrProvinceName            = State or Province Name
localityName                   = Locality Name
0.organizationName             = Organization Name
organizationalUnitName         = Organizational Unit Name
commonName                     = Common Name
emailAddress                   = Email Address
countryName_default  = BD
stateOrProvinceName_default = DHK
localityName_default = RAMPURA
0.organizationName_default = ACME LTD
organizationalUnitName_default = ACME ROOT
commonName_default = rootCA

[ v3_ca ]
# Extensions to apply when createing root ca
# Extensions for a typical CA, man x509v3_config
subjectKeyIdentifier  = hash
authorityKeyIdentifier  = keyid:always,issuer
basicConstraints  = critical, CA:true
keyUsage   =  critical,  digitalSignature, cRLSign, keyCertSign
```

```
[ v3_intermediate_ca ]
# Extensions to apply when creating intermediate or sub-ca
# Extensions for a typical intermediate CA, same man as above
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
#pathlen:0 ensures no more sub-ca can be created below an intermediate
basicConstraints = critical, CA:true, pathlen:0
keyUsage   = critical, digitalSignature, cRLSign, keyCertSign



[ server_cert ]
# Extensions for server certificates
basicConstraints = CA:FALSE
nsCertType   = server
nsComment   =  "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage   =  critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

b. $Nano\ sub - ca/sub - ca.conf$

```
#sub-ca.conf

[ca]
default_ca    = CA_default

[CA_default]
dir     = sub-ca
```

```
certs     =  $dir/certs
crl_dir    = $dir/crl
new_certs_dir   = $dir/pem
database   = $dir/index/index
serial    = $dir/serial/serial
RANDFILE   = $dir/private/.rand
private_key   = $dir/private/sub-ca.key
certificate   = $dir/certs/sub-ca.crt
crlnumber    = $dir/crlnumber
crl    =  $dir/crl/ca.crl
crl_extensions    = crl_ext
default_crl_days    = 30
default_md   = sha256
name_opt   = ca_default
cert_opt   = ca_default
default_days  = 365
preserve   = no
policy    = policy_loose

[ policy_strict ]
countryName   = supplied
stateOrProvinceName  =  supplied
organizationName  = match
organizationalUnitName  =  optional
commonName    =  supplied
emailAddress   =  optional

[ policy_loose ]
countryName   = optional
stateOrProvinceName  = optional
localityName   = optional
```

```
organizationName  = optional
organizationalUnitName   = optional
commonName   = supplied
emailAddress   = optional

 [ req ]
# Options for the req tool, man req.
default_bits   = 2048
distinguished_name  = req_distinguished_name
string_mask   = utf8only
default_md  =  sha256
# Extension to add when the -x509 option is used.
x509_extensions   = v3_ca

[ req_distinguished_name ]
countryName                    = Country Name (2 letter code)
stateOrProvinceName            = State or Province Name
localityName                   = Locality Name
0.organizationName             = Organization Name
organizationalUnitName         = Organizational Unit Name
commonName                     = Common Name
emailAddress                   = Email Address
countryName_default  = BD

stateOrProvinceName_default = DHK
localityName_default = RAMPURA
0.organizationName_default = ACME LTD
organizationalUnitName_default = ACME SubCA A
commonName_default = SubCA A

 [ v3_ca ]
```

```
# Extensions to apply when createing root ca
# Extensions for a typical CA, man x509v3_config
subjectKeyIdentifier  = hash
authorityKeyIdentifier  = keyid:always,issuer
basicConstraints  = critical, CA:true
keyUsage   =  critical, digitalSignature, cRLSign, keyCertSign

[ v3_intermediate_ca ]
# Extensions to apply when creating intermediate or sub-ca
# Extensions for a typical intermediate CA, same man as above
subjectKeyIdentifier  = hash
authorityKeyIdentifier  = keyid:always,issuer
#pathlen:0 ensures no more sub-ca can be created below an intermediate
basicConstraints  = critical, CA:true, pathlen:0
keyUsage   = critical, digitalSignature, cRLSign, keyCertSign

[ server_cert ]
# Extensions for server certificates
basicConstraints  = CA:FALSE
nsCertType   = server
nsComment   =  "OpenSSL Generated Server Certificate"
subjectKeyIdentifier  = hash
authorityKeyIdentifier  = keyid,issuer:always
keyUsage   =  critical, digitalSignature, keyEncipherment
extendedKeyUsage  = serverAuth
subjectAltName = @alt_names

[alt_names]
DNS.1 = www.fardin.home
DNS.2 = *.fardin.home
DNS.3 = localhost
```

```
IP.1 = 127.0.0.1
IP.2 = ::1
#besure to change the alt_names
```

c. $Nano\ openssl - san.ssl$

```
[ req ]
default_bits          = 2048
distinguished_name    = req_distinguished_name
req_extensions        = req_ext

[ req_distinguished_name ]
countryName           = Country Name (2 letter code)
stateOrProvinceName   = State or Province Name (full name)
localityName          = Locality Name (eg, city)
organizationName      = Organization Name (eg, company)
commonName            = Common Name (e.g. server FQDN or YOUR name)

# Optionally, specify some defaults.
countryName_default           = BD
stateOrProvinceName_default   = DHK
localityName_default          = Agargaon
0.organizationName_default    = Example Limited
organizationalUnitName_default = IT
emailAddress_default          = reshma@gmail.com

[ req_ext ]
subjectAltName = @alt_names
```

```
[alt_names]
DNS.1 = www.fardin.home
DNS.2 = *.fardin.home
DNS.3 = localhost
IP.1 = 127.0.0.1
IP.2 = ::1
#besure to change the alt_names
```

6. Create keys for ca. For a and b you will have to enter a confidential phrase of your choosing.
    a. $openssl\ genrsa\ -aes256\ -out\ root-ca/private/ca.key\ 2048$
    b. $openssl\ genrsa\ -aes256\ -out\ sub-ca/private/sub-ca.key\ 2048$
    c. $openssl\ genrsa\ -out\ server/private/server.key\ 2048$
7. Create root certificate. Here you can change anything you want like organization name, unite name etc.

$$openssl\ req\ -config\ root-ca/root-ca.conf\ -key\ root-ca/private/ca.key\ -new\ -x509\ -days\ 7305\ -sha256\ -extensions\ v3\_ca\ -out\ root-ca/certs/ca.crt$$

8. Request for sub-ca certificate. Repeat this step **if Multiple CA.** Organization name should match with root ca. Other then that you can change anything you like.

$$openssl\ req\ -config\ sub-ca/sub-ca.conf\ -new\ -key\ sub-ca/private/sub-ca.key\ -sha256\ -out\ sub-ca/csr/sub-ca.csr$$

9. Sign the sub-ca.csr using rootca

```
openssl ca  − config root − ca/root − ca.conf
                − extensions v3_intermediate_ca  − days 365  − notext
                − in sub − ca/csr/sub − ca.csr  − out sub − ca/certs/sub
                − ca.crt
```

10. Generating the server certificate signing request. Be sure to change the openssl-san.cnf file. Or do it form terminal prompt.

```
openssl req  − key server/private/server.key  − new  − sha256
                − out server/csr/server.csr  − config ./openssl − san.cnf
```

11. Sign the server.csr using the sub-ca.

```
openssl ca  − config sub − ca/sub − ca.conf  − extensions server_cert
                − days 365  − notext  − in server/csr/server.csr
                − out server/certs/server.crt
```

12. Create the chained crt.

```
cat server/certs/server.crt sub − ca/certs/sub − ca.crt root
                − ca/certs/ca.crt  >  ./generated/chained.crt
```

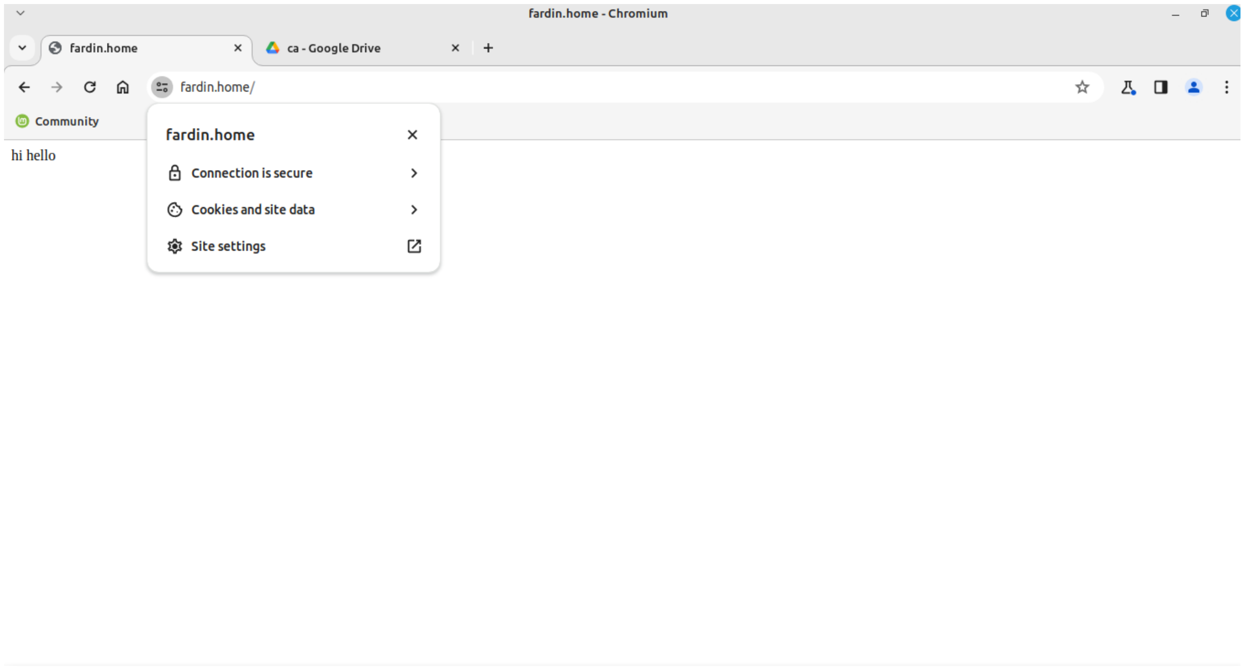
13. Now you need to add three lines in the /etc/apache/sites-available/fardin.home.conf which will point to the certificate file and certificate key of the website.

```
SSLEngine on
SSLCertificateFile      "/var/www/fardin.home/generated/chained.crt"
SSLCertificateKeyFile "/var/www/fardin.home/generated/server.key"
```

14. Now we have to restart the apache2 *systemctl restart apache*2

15. Go to your browser and import the ~/ca/root-ca/certs/ca.crt

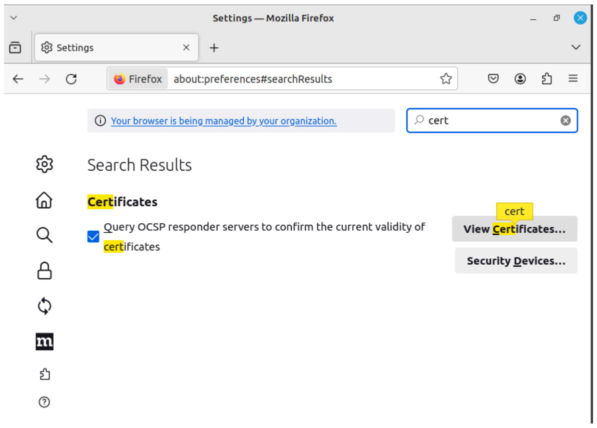
16. Now if we go to https://www.fardin.home we should see that it has a padlock icon.

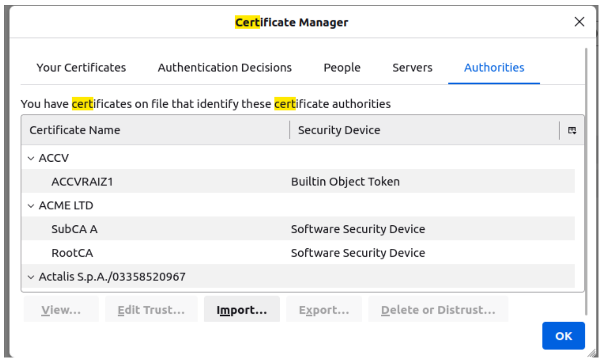


Client Configuration:

1. Turn host-only network in Virtual Machine's network settings.
2. Change the DNS from settings page inside the installed distribution's settings.
3. Change to superuser or root
4. Edit resolve.conf. change the IP address to DNS's IP address. Ex. 192.168.56.5.

   nano /etc/resolve.conf
5. Import the root-ca.crt form ca server.
6. Install it using browser's settings page

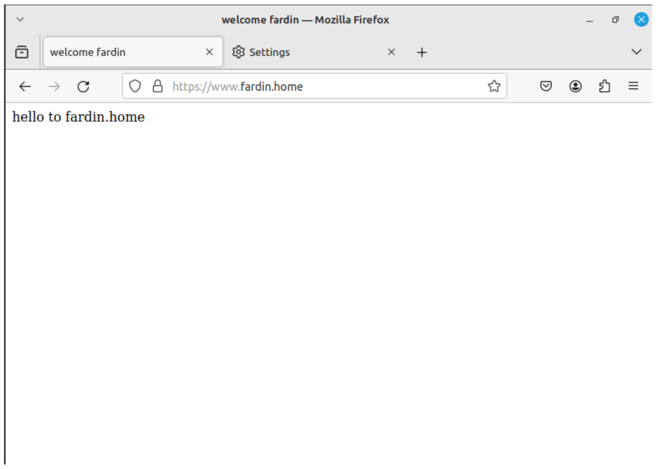

Settings — Mozilla Firefox
Settings
Firefox about:preferences#searchResults
Your browser is being managed by your organization.
cert
Search Results
Certificates
Query OCSP responder servers to confirm the current validity of certificates
cert
View Certificates...
Security Devices...



You have certificates on file that identify these certificate authorities

Certificate Name	Security Device
ACCV	
ACCVRAIZ1	Builtin Object Token
ACME LTD	
SubCA A	Software Security Device
RootCA	Software Security Device
Actalis S.p.A./03358520967	

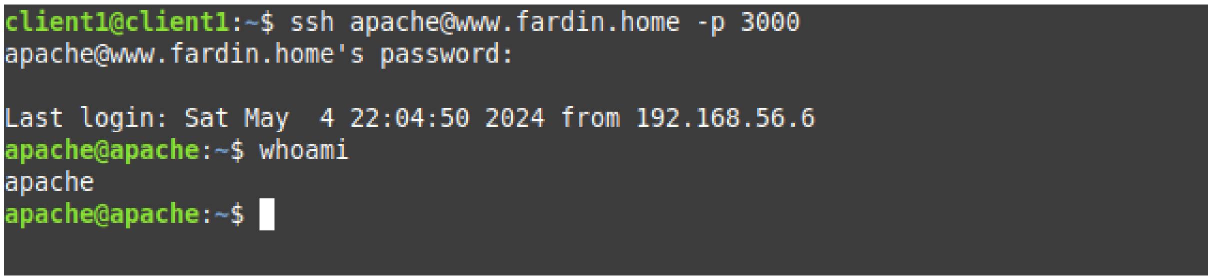
7. Enable https only from browser's settings page
8. Now if you visit https://www.fardin.home you will have a secure connection.



hello to fardin.home

Configure OpenSSH:

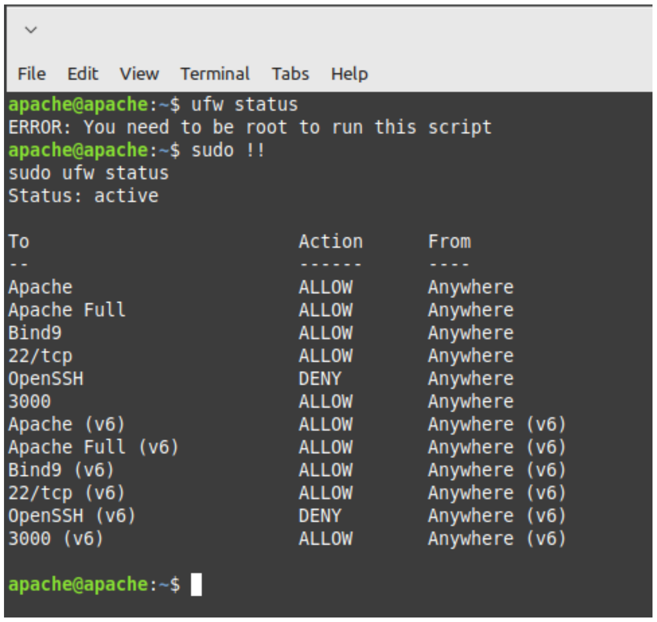
1. Depending on the flavor of Linux you may already have SSH server installed. Check using *systemctl status ssh*
2. Now we have to change the port number on which openSSH will run. *nano /etc/ssh/sshd_config*
3. Find commented out Port 22. Remove the comment and replace it with your choice. Ex Port 3000.
4. Allow the port on UFW (if enable). *sudo ufw allow* 3000
5. Restart OpenSSH. *sudo systemctl restart ssh*
6. Now go to a client pc's terminal. *ssh apache@www.fardin.home −p* 3000



```
client1@client1:~$ ssh apache@www.fardin.home -p 3000
apache@www.fardin.home's password:

Last login: Sat May  4 22:04:50 2024 from 192.168.56.6
apache@apache:~$ whoami
apache
apache@apache:~$
```

Configure UFW:

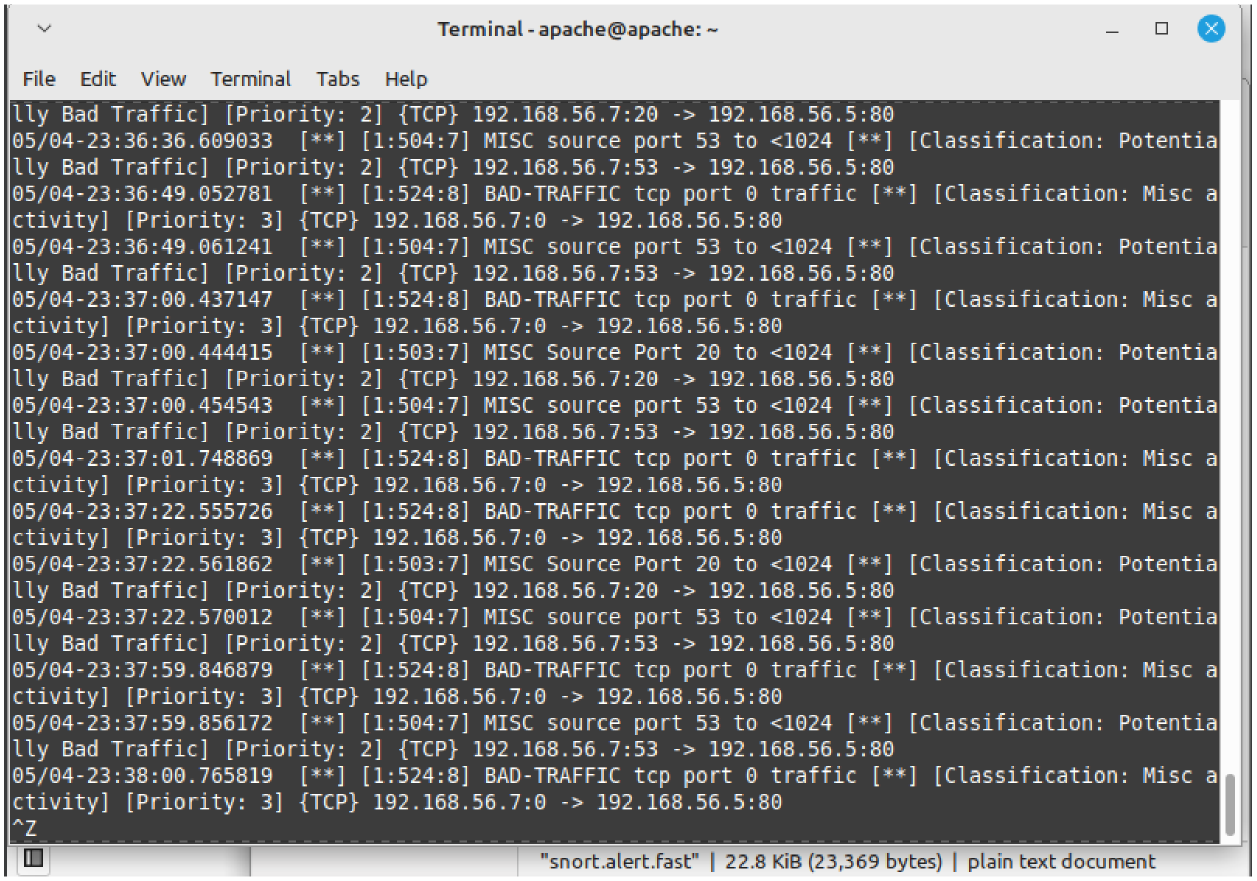
1. Change user to root. *sudo su*
2. Enable UFW. *ufw enable*
3. Deny all ports. *sudo ufw default deny*
4. Allow all the necessary ports ex. 22, 53, 80, 443, 3000. *ufw allow* 22



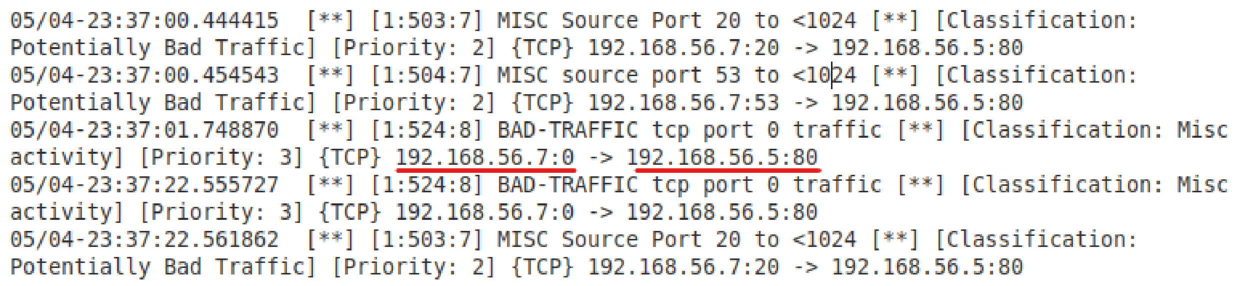
```
apache@apache:~$ ufw status
ERROR: You need to be root to run this script
apache@apache:~$ sudo !!
sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
Apache                     ALLOW       Anywhere
Apache Full                ALLOW       Anywhere
Bind9                      ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
OpenSSH                    DENY        Anywhere
3000                       ALLOW       Anywhere
Apache (v6)                ALLOW       Anywhere (v6)
Apache Full (v6)           ALLOW       Anywhere (v6)
Bind9 (v6)                 ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)
OpenSSH (v6)               DENY        Anywhere (v6)
3000 (v6)                  ALLOW       Anywhere (v6)

apache@apache:~$
```

Configuring Snort:

1. *sudo apt install snort*
2. Change to root user. *sudo su*
3. Go to snort config directory. *cd /etc/snort/*
4. Open snort.conf. *nano snort.conf*
5. Change the home net to your subnet. Ex. *ipvar HOME_NET* 192.168.56/24
6. Run this command *snort −A console −c /etc/snort/snort.conf*
7. If any intrusion is detected it will log the data in */var/log/snort/snort.alert.fast*


Terminal - apache@apache: ~
File Edit View Terminal Tabs Help
"snort.alert.fast" | 22.8 KiB (23,369 bytes) | plain text document

```
lly Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:20 -> 192.168.56.5:80
05/04-23:36:36.609033  [**] [1:504:7] MISC source port 53 to <1024 [**] [Classification: Potentia
lly Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:53 -> 192.168.56.5:80
05/04-23:36:49.052781  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**] [Classification: Misc a
ctivity] [Priority: 3] {TCP} 192.168.56.7:0 -> 192.168.56.5:80
05/04-23:36:49.061241  [**] [1:504:7] MISC source port 53 to <1024 [**] [Classification: Potentia
lly Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:53 -> 192.168.56.5:80
05/04-23:37:00.437147  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**] [Classification: Misc a
ctivity] [Priority: 3] {TCP} 192.168.56.7:0 -> 192.168.56.5:80
05/04-23:37:00.444415  [**] [1:503:7] MISC Source Port 20 to <1024 [**] [Classification: Potentia
lly Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:20 -> 192.168.56.5:80
05/04-23:37:00.454543  [**] [1:504:7] MISC source port 53 to <1024 [**] [Classification: Potentia
lly Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:53 -> 192.168.56.5:80
05/04-23:37:01.748869  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**] [Classification: Misc a
ctivity] [Priority: 3] {TCP} 192.168.56.7:0 -> 192.168.56.5:80
05/04-23:37:22.555726  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**] [Classification: Misc a
ctivity] [Priority: 3] {TCP} 192.168.56.7:0 -> 192.168.56.5:80
05/04-23:37:22.561862  [**] [1:503:7] MISC Source Port 20 to <1024 [**] [Classification: Potentia
lly Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:20 -> 192.168.56.5:80
05/04-23:37:22.570012  [**] [1:504:7] MISC source port 53 to <1024 [**] [Classification: Potentia
lly Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:53 -> 192.168.56.5:80
05/04-23:37:59.846879  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**] [Classification: Misc a
ctivity] [Priority: 3] {TCP} 192.168.56.7:0 -> 192.168.56.5:80
05/04-23:37:59.856172  [**] [1:504:7] MISC source port 53 to <1024 [**] [Classification: Potentia
lly Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:53 -> 192.168.56.5:80
05/04-23:38:00.765819  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**] [Classification: Misc a
ctivity] [Priority: 3] {TCP} 192.168.56.7:0 -> 192.168.56.5:80
^Z
```



```
05/04-23:37:00.444415  [**] [1:503:7] MISC Source Port 20 to <1024 [**] [Classification:
Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:20 -> 192.168.56.5:80
05/04-23:37:00.454543  [**] [1:504:7] MISC source port 53 to <1024 [**] [Classification:
Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:53 -> 192.168.56.5:80
05/04-23:37:01.748870  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**] [Classification: Misc
activity] [Priority: 3] {TCP} 192.168.56.7:0 -> 192.168.56.5:80
05/04-23:37:22.555727  [**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**] [Classification: Misc
activity] [Priority: 3] {TCP} 192.168.56.7:0 -> 192.168.56.5:80
05/04-23:37:22.561862  [**] [1:503:7] MISC Source Port 20 to <1024 [**] [Classification:
Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.56.7:20 -> 192.168.56.5:80
```

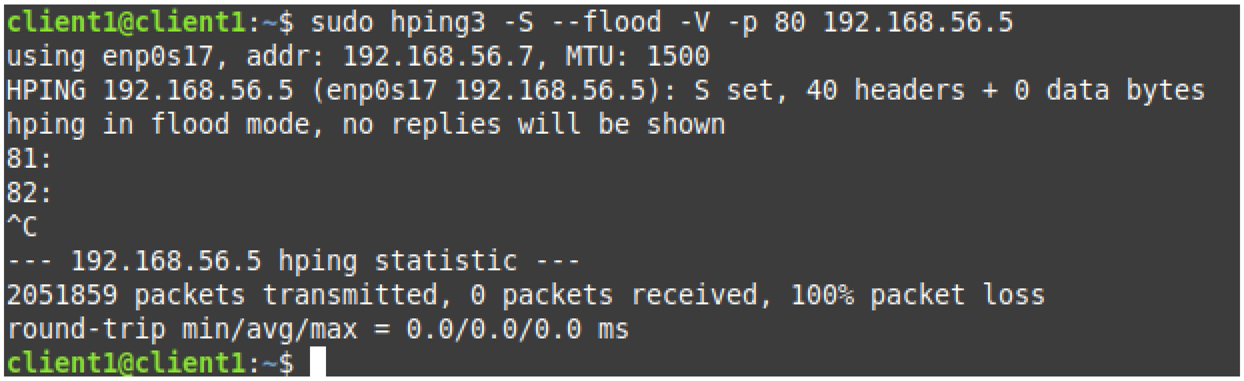
Here 192.168.56.5 is our server. And 192.168.56.7 is the attacker.

Configure Hping3:

1. Install Hping3. $sudo\ apt\ install\ hping3$

2. Run command for SYNFlood attack

   $sudo\ hping3\ - S\ - -flood\ - V - p\ 80\ 192.168.56.5$



```
client1@client1:~$ sudo hping3 -S --flood -V -p 80 192.168.56.5
using enp0s17, addr: 192.168.56.7, MTU: 1500
HPING 192.168.56.5 (enp0s17 192.168.56.5): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
81:
82:
^C
--- 192.168.56.5 hping statistic ---
2051859 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
client1@client1:~$
```